International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 5 (2018) pp. 2569-2581

Secure Use of Cloud Storage of Data on Smartphones using Atomic AES on ARM Architectures

Solomon Babatunde Olaleye [a] and Shri Kant [b]

[a] Department of Computer Science and Engineering, Sharda University, Greater Noida-201306, India. olaleye3@yahoo.com and shri.kant@sharda.ac.in
[b] Research and Technology Development Centre, Sharda University, Greater Noida-201306, India.

Abstract

Security of data is of great importance to smartphones' users today. Cloud computing provides unlimited storage for its users but secure transmission and retrieval of data from the cloud are of great challenge. This work made use of Atomic Advanced Encryption Standard (AES) algorithm to provide security for user data on ARM architectures prior to cloud storage. AES was adopted because of its robust security among the symmetric block ciphers. A mobile application named 'Solo App' was developed and implemented on 15 ARM architectures. The implementation showed that it can effectively protect data by encryption using cloud storage while maintaining minimal overhead cost on limited resources on Android smartphones. The evaluation of this work was done based on compatibility test, encryption time, decryption time and throughput. The study results when compared with existing works were found impressive and the compatibility passing rate is 100% because the application functions well as coded on all the devices.

Keywords: Atomic-AES; Solo App; Cryptography; Cloud Storage; ARM

INTRODUCTION

Smartphones adoption all over the world is increasing daily. This is because from business view, it has become a critical element of business advancement. From personal usage, it has become a means of contacting families and friends from far and near by means of voice, Short Message Service (SMS), email, Facebook, WhatsApp, Tweeter etc. From financial transaction point of view, it is used for online banking, one time pin (OTP) generation, automated teller machine (ATM) blocking etc. They are equally equipped with applications for browsing, note taking, appointment booking, health monitoring, religious information and so on. Due to this multi usage activities, a lot of sensitive data are being generated which need to be protected against attack.

According to Hwang et al. (2008) smartphones is popularly built using advanced reduced instruction set computer machine (ARM) architecture. ARM architecture has evolved over the years. Initially ARMv5 architecture was used in smartphones. Its hardware floating point unit was improved upon for speed which brought next version ARMv7 architecture. The widely used ARMv8 made use of 64 bit architecture. Therefore, smartphones manufacturers today use ARM architecture for smartphones (Vijay and Bansode, 2015).

The objective of this work is to protect users' sensitive data on ARM architectures of smartphones using Atomic AES 128-bit key for encryption and decryption. It makes use of Amazon EC2 for cloud storage. A mobile application named 'Solo App' (Olaleye and Ojha, 2017) was developed and implemented on 15 ARM architectures of smartphones of varied configurations. The results were recorded and compared with existing related works. The study results were found impressive in terms of encryption time and decryption time. Further, there are two major contributions of this work; enhancing storage space of smartphones by using cloud storage and provision for security of users' sensitive data at rest and in transit.

The rest of this research paper is arranged as follows: Section 2 gives the reasons why improving storage space of smartphones is important. It explores the existing security implementation on ARM architectures of smartphones from 2008 to 2016. It presents the Atomic AES algorithm used for the encryption of data prior to cloud storage. Section 3 provides the proposed secure migration of data to cloud and the model for the implementation of the work. Section 4 describes the experimental setup and results of the study in form of graphs and tables while Section 5 discusses the results as observed and in relation to existing works. It summarises the work and gives the concluding comments.

WHY IMPROVING STORAGE?

Balasubramanyam et al. (2009) reported that smartphone storage is a bottleneck for most mobile applications performance. Mendon and Sass (2008) described that nowadays improvement on storage performance is one of the areas of research in phone industry because a sluggish smartphone cannot meet the expectation of the users. Liu et al. (2013) explained that the performance of underlying storage plays important role in ensuring better experience of user in today's smartphone consumer electronics. As the mobile applications performing on them are getting much complex the storage system tends to be a problem to application performance. Similarly, Kim et al. (2011) reported that the performance of storage influences common mobile applications performance such as browsing of web, maps, Facebook and email.

Chun and Maniatis (2009) explained that most smartphones have limited storage and users have to regularly eliminate certain infrequently used applications manually to release more storage space for new application installation. Satyanarayan et al. (2009) also mentioned that when the storage of smartphone is full, warning sign is usually given to user.

In view of the above, the way out is the use of cloud storage. A number of researchers had mentioned this. For instance, Oberheide et al. (2008) reported that to resolve smartphones limitations, a mechanism known as virtual machine is presented with the assistance of cloud computing. The cloud offers storage for application and data of user as well as virtual machines to execute big applications that cannot be accommodated on smartphones due to resource limitations. According to Gupta et al. (2010) offloading is one of the techniques to send computation to another machine which enables smartphones to offload to cloud. Likewise, Jiang et al. (2010) stated that storage and computational offloading support can be taken from device of user to cloud server. Cloud computing technology can help to reduce smartphones battery consumption as well as making provision for adequate storage. While going for storage enhancement using cloud, security is the concern for communication between smartphone and cloud.

Existing security implementation

There are several studies related to existing security implementation on ARM architectures of smartphones. This section presents the review of the existing security works (Table 1) which we came across from the literature, clearly showing the strengths and weaknesses of each from 2008 to 2016.

Table 1: Existing security implementation on ARM architecture 2008 - 2016

Existing security	Authors and year	Benefits/limitations
Fine-grained permissions in Android Applications	(Enck et al., 2008; Fang et al., 2014)	**Benefit**: Users to decide what to install **Limitation**: Users either accept or reject permission
Password, Pattern and PIN	(Tao and Adams, 2008)	**Benefit**: Secure the smartphone **Limitation**: No security for storage data
Virtualization using Xen Hypervisor for Advanced Reduced instruction set computing Machine-Based (ARM)	(Hwang et al., 2008)	**Benefits**: Created virtual machines for smartphones ARM architecture, implemented using Xen hypervisor. **Limitation**: Simulated to provide security through smartphone hardware.
Android security extensions and mobile virtualization techniques	(Rusello et al., 2011; Tarle, 2015)	**Benefits**: Android security extension by defining context application permission at installation, virtualization technique. **Limitation**: It requires source code modification.
Graphical password based hybrid authentication system	(Ray, 2012)	**Benefit**: proposed replacement of textual passwords with symbols **Limitations**: Log-in process is slow, graphical password takes longer time to input.
Encryption file system	(Wang et al., 2012)	**Benefit**: Used AES to encrypt data-at-rest. **Limitations**: No security of data-in-transit? Speed of encryption was high in some files.
Biometric encryption in MCC	(Zhao et al., 2013)	**Benefits**: Proposed BE for MCC, advanced protocol for BE. **Limitations**: Takes more storage to store user biometric information, consumes smartphone battery.
Locking scheme of smart multimedia devices	(Jeong et al., 2014)	**Benefits**: Emphasis on pattern lock, password & PIN for smartphones, secure locking screen using time pattern. **Limitations**: Stored data are not encrypted & loss of device means loss of data.

Combine cryptographic algorithms for SMS security	(Mandavkar et al., 2014)	**Benefits**: Security for SMS, combined cryptographic algorithms used **Limitations**: Security for SMS only, No reference to cloud storage
Secure data sharing in cloud using multi-authority attribute based encryption	(Divya and Sadhasivam, 2014)	**Benefits**: Proposed progressive elliptic curve cryptography for multiple encryptions, different keys for encryption but single key for decryption. **Limitations**: Not implemented in real life scenario, multiple encryptions tend to be slow.
Backup and restore techniques	(Nayadkar and Parne, 2014; Nayadkar, 2015)	**Benefit**: Proposed scheduled backup daily, weekly or monthly. **Limitations**: Proposed and not implemented
Full disk encryption	(Gotzfried and Muller, 2014; Muller and Freiling, 2015)	**Benefit**: All data on storage are encrypted. **Limitations**: Takes time, consumes battery, consumes storage space.
Lightweight encryption and secure protocol	(Zegers et al., 2015)	**Benefits**: Uses lightweight encryption i.e. AES with 144-bit keys and securing data before cloud. **Limitations**: AES Encryption round is 7, the work was simulated, uses Android framework APIs
Secure MCC in cyber security	(Vittapu et al., 2015)	**Benefits**: Used RSA algorithm for encryption and decryption, proposed cloud intrusion detection system. **Limitations**: Not implemented in real life scenario, user has limited control over cloud security and privacy.
Partial encryption on smartphone and cloud	(Kaur et al., 2015)	**Benefits**: Partial encryption on smartphones using DES, partial encryption on cloud, final (full) encryption stored on smartphone. **Limitations**: It is slow, it consumes smartphone limited resources, and it takes more storage space.
Cloud-based system and Modern Encryption Standard (MES)-II algorithm	(Agrawal and Kulurkar, 2016)	**Benefit**: proposed the use of MES II for encryption to cloud. **Limitations**: Not implemented in real life scenario, proposed encryption for text/pdf files only
Secure local storage of smartphone data	(Poonguzhali et al., 2016)	**Benefits**: Secured user data on device only, used Password Based Encryption Standard (PBES). **Limitations**: Loss of device means loss of data, No cloud storage approach, tested on 3 devices.

For the efficient and secure implementation of our work, the existing AES-128 bits algorithm which was publicly tested and approved by the National Institute of Standards and Technology (NIST) for protecting sensitive data was studied and modified to achieve the atomic AES. The atomic AES algorithm was implemented using a mobile application named 'Solo App', PhP (Lurig, 2008) scripting language was used to develop Representational State Transfer (REST) Application Program Interface (API) (Richardson and Ruby, 2007) for web services, and Amazon EC2 was used for cloud services.

Advanced Encryption Standard (AES)

AES was developed by Daemen Joan and Rijmen Vincent in 2000. It is a block cipher of 128 bits. It has three key lengths which are 128, 192 and 256 bits. The different keys are generated using 10, 12 and 14 rounds. Table 2 shows the summary of the key length and numbers of rounds (FIPS 197, 2001; Saini and Bangar, 2014; Rihan et al., 2015; Alsharani and Walker, 2015)

Table 2: AES key lengths and number of rounds

Algorithm	Key length (bits)	Block size (bits)	No of round
AES- 128	128	128	10
AES- 192	192	128	12
AES- 256	256	128	14

For the existing AES encryption, the order of encryption is given as *SubBytes*, *ShiftRows*, *MixColumns* and *Round key*. To achieve atomic AES, the order is modified without compromising security but with enhanced speed of encryption and decryption. This is fully explained and the pseudocode given in the next section.

Atomic AES

Atomic AES was first mentioned by Banik et al., 2016 with 8-bit serial architecture that can carry out the dual functionalities of encryption and decryption for enhanced speed. It has a circuit size of around 2645 Gate Equivalent and latency of 226 cycles. It is an improvement on existing AES. Atomic AES implementation on ARM architectures can secure data more efficiently and faster as proposed in this work in terms of encryption time, decryption time and throughput.

The four operations that are carried out for atomic AES encryption are discussed below and depicted in Fig. 1;

1. **Shift Rows**: Here, the 1st row will remain as it is, 2nd row will be shifted by 1 byte to the left, 3rd row will be shifted by 2 bytes to the left and 4th row will be shifted by 3 bytes to the left (FIPS 197, 2001).
2. **Mix Columns**: Here, each byte of the column in the two dimensional array will be mapped to another value which is a function of the 4 bytes in the column. All multiplications are carried out based on GF (2^8) arithmetic while all additions are XOR operations. The values of GF (2^8) are in binary polynomials where GF is a Galois Field by a given matrix. Both shift rows and mix columns are used to cause diffusion in AES (FIPS 197, 2001).
3. **Add Round Key**: In AES, add round key processing is devised as a stream cipher where the 128-bit of state are bitwise XOR with four 32-bit words of extended key resulting from key expansion. It is an aspect of AES operations that includes using key to guarantee security (FIPS 197, 2001). In the atomic AES the 128-bit key is used for securing user data on smartphones
4. **Sub Bytes**: This means substitution bytes. It is simply a two dimensional array of bytes which is a 16 by 16 matrix. It consists of byte values known as S-box. The S-box is used for this stage task which contains the outcome of the substitution and permutation for all possible 8 bit values (FIPS 197, 2001). This stage was introduced to cause confusion and make atomic AES difficult to break as it is in existing AES.

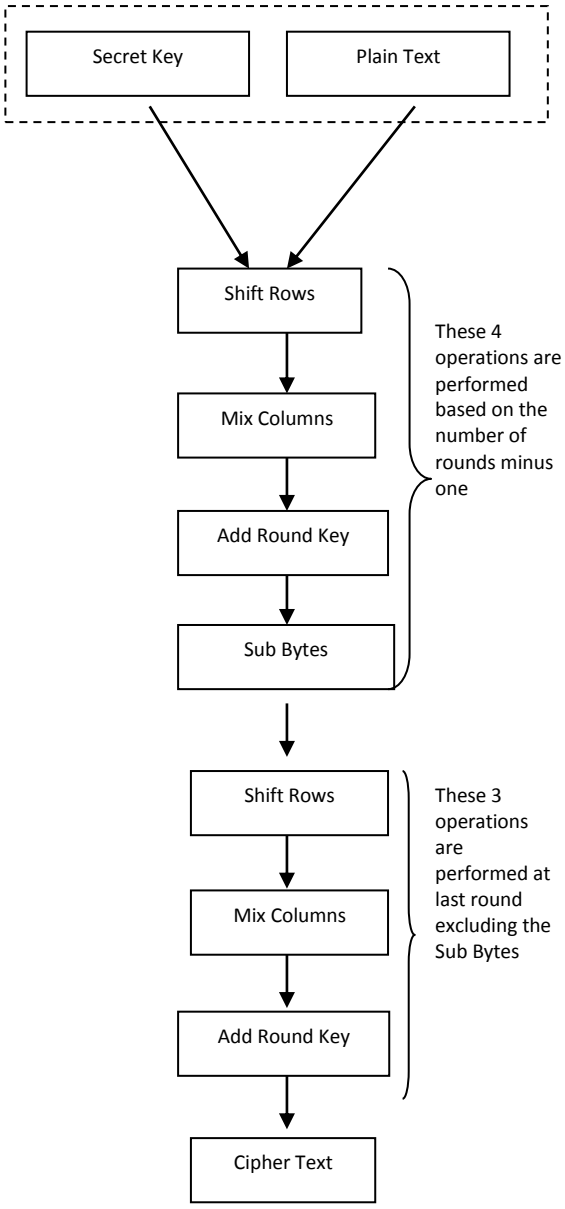

Figure 1. Atomic AES encryption

The pseudocode for the atomic AES encryption is given below;

Pseudocode 1. Atomic AES encryption

```
1:  Encryption (byte in[4*Nb], byte out[4*Nb], word
      w[Nb*(Nr+1)])
2:  begin
3:  byte state[4,Nb]
4:  state = in
5:    AddRoundKey(state, w[0, Nb-1])

6:  for round = 1 step 1 to Nr–1
7:     ShiftRows(state)
8:     MixColumns(state)
9:    AddRoundKey(state, w[round*Nb,  (round+1)*Nb-1])
10:   SubBytes
11: end for
12: ShiftRows(state)
13: MixColumns(state)
14: AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
15: out = state
16: end
```

Here, Nr is the number of round, Nb is the number of block and w is the word used for key schedule which is an array of 4 bytes and state is the intermediate encryption result which is in form of array of bytes.

For the atomic AES decryption operation, it is the inversed of the encryption operation. The steps are given below;

1. Inverse Shift Rows
2. Inverse Mix Columns
3. Add Round Key
4. Inverse Sub Bytes

The pseudocode for the atomic AES key expansion remains the same with the existing AES (FIPS 197, 2001) as given below;

Pseudocode 2. AES key expansion

```
1:  KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
2:  begin
3:  word temp
4:  i = 0
5:  while (i < Nk)
6:      w[i] = word(key[4*i], key[4*i+1], key[4*i+2],
      key[4*i+3])
7:      i = i+1
8:  end while
9:  i = Nk
10: while (i < Nb * (Nr+1)]
11:     temp = w[i-1]
12:     if (i mod Nk = 0)
13:        temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
14:     else if (Nk > 6 and i mod Nk = 4)
15:        temp = SubWord(temp)
16:     end if
17:    w[i] = w[i-Nk] xor temp
18:    i = i + 1
19:  end while
20: end
```

Here, the AES encryption algorithm takes the encryption key k and performs a key expansion process to generate a key. A total of Nb (Nr + 1) words is generated such that an initial set of Nb words are required while each round (Nr) requires Nb words of key data. The resultant key expansion consists of an array of 4 byte words, represented by w_i where $0 \leq i < Nb$ (Nr + 1). Nk is the number of 32-bit words consists of the encryption key and Nr is the number of rounds that is a function of Nk and Nb.

PROPOSED SECURE MIGRATION OF DATA TO CLOUD

The atomic AES algorithm was used to provide enhanced security for user sensitive data on ARM architectures using cloud. The proposed secure migration of data to cloud algorithm (Fig. 2) authenticates users with 6 digits PIN which is an improvement over existing and commonly uses 4 digits PIN. Every digit can be entered in 10 different ways i.e. 0-9. For 4 digits PIN there are 10^4 possible trials while for 6 digits PIN there are 10^6 possible trials. The 6 digits PIN is more secured and is encrypted before cloud storage. Further, user can select file to encrypt to server and can as well select file to decrypt from the server whenever needed.

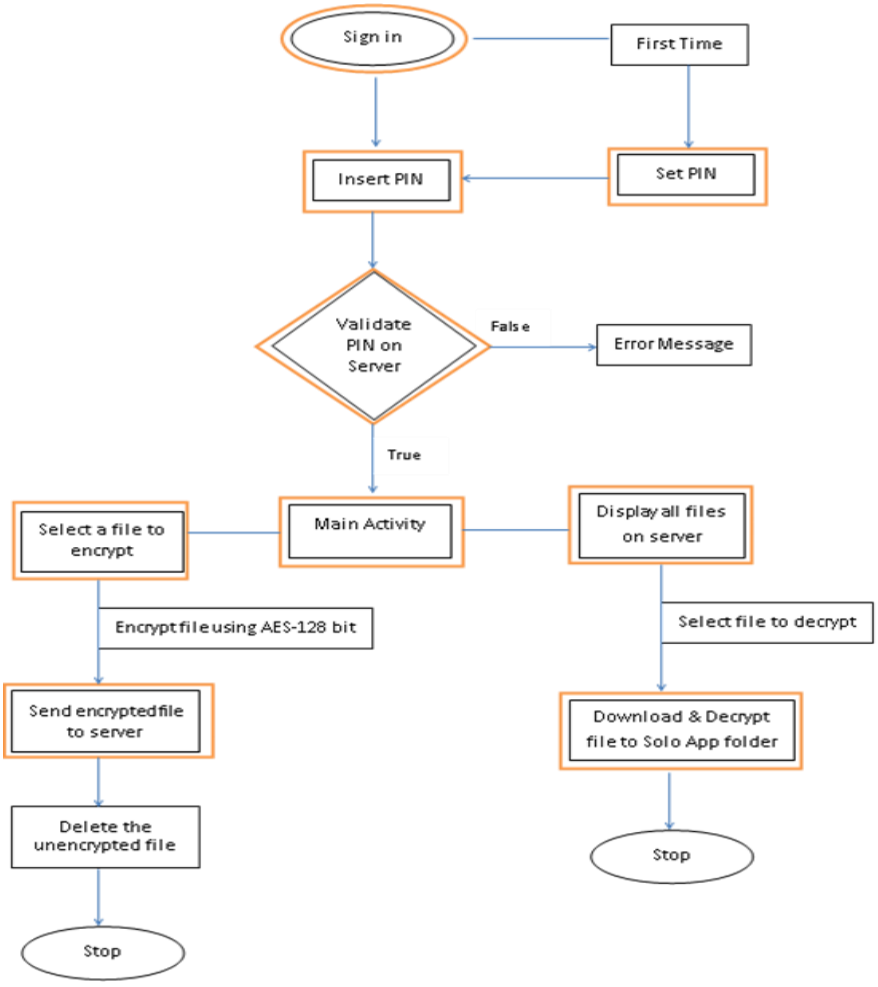

Figure 2. Proposed secure migration of data to cloud (Olaleye and Ojha, 2017)

Description of the proposed secure migration of data

A mobile application was developed using Android studio 2.2.3 written in java and xml for the design of user interface. Representational State Transfer (REST) services API (Richardson and Ruby, 2007) was used and written in PhP (Lurig, 2008) which is a general purpose scripting language suitable for server side web development. Amazon Elastic Compute Cloud (EC2) server was used for cloud storage and database. To guide against attack, the encryption key is randomly generated.

The developed application is named 'Solo App' and can be downloaded from Google Play Store. The Google Play Store is an official store for all Android applications approved by Google where safe applications can be installed and use on all Android devices. The icon of Solo App has been registered with Google Play Store and is as shown in Fig. 3. It can be located on the Play Store by typing 'solomon babatunde'. To sign-in to use the Solo App, Google API client (GitHub, 2017) for sign-in authorization was used to fetch user data to the server. Moreover, in case the user forgets his/her sign-in 6 digits PIN an OTP message for PIN recovery will be sent to the user gmail account on request. There is facility to change the 6 digits PIN at will as shown in Fig. 4 and Fig.5 for change PIN interface 1 and change PIN interface 2 respectively. While it is possible for user to change his/her PIN, this does not in any way affect the encryption key. When user file is encrypted and sent to cloud storage, the unencrypted file is deleted and overwritten by random bits in order to avoid retriever of the file from local storage by any other means. This enhances the security of data on smartphones. The user files stored in the cloud can as well be retrieved when needed by providing accurate user credential. The credential is validated and the file is decrypted back into a created new folder named 'SoloApp folder' on smartphone's storage for easy access.

Figure 3. Solo App

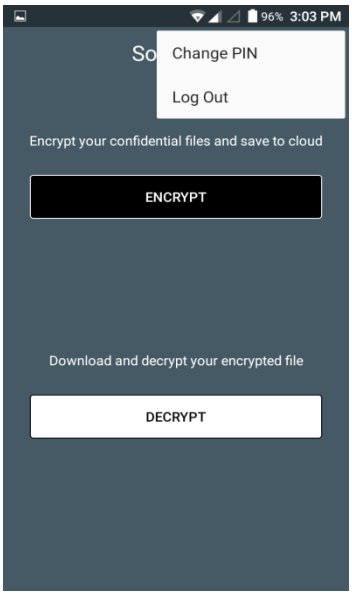

Figure 4. Change PIN interface 1

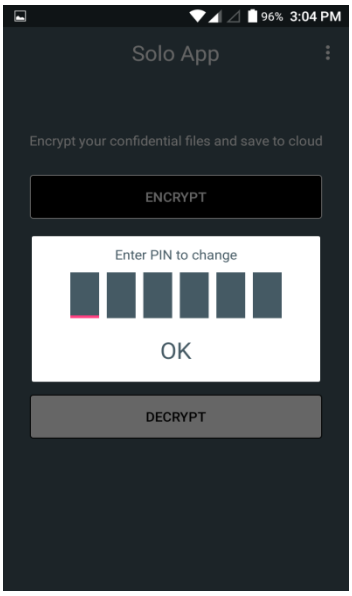

Figure 5. Change PIN interface 2

Proposed model for the implementation

The proposed model for the implementation of the research work is as shown in Fig. 6. Smartphones users are represented as U_1, U_2….U_n, locations of users are represented as L_1, L_2…..L_n and smartphones of users are represented as SP1, SP2, SP3….SPn. Smartphones users irrespective of their locations can download 'Solo App' application into their smartphones from Play Store. Users' data are stored into Amazon EC2 where there are infrastructure and services such as server, compute, storage, load balancer, security etc. However, before these services can be utilized there must be Internet connectivity either in the form of Wi-Fi or cellular data.

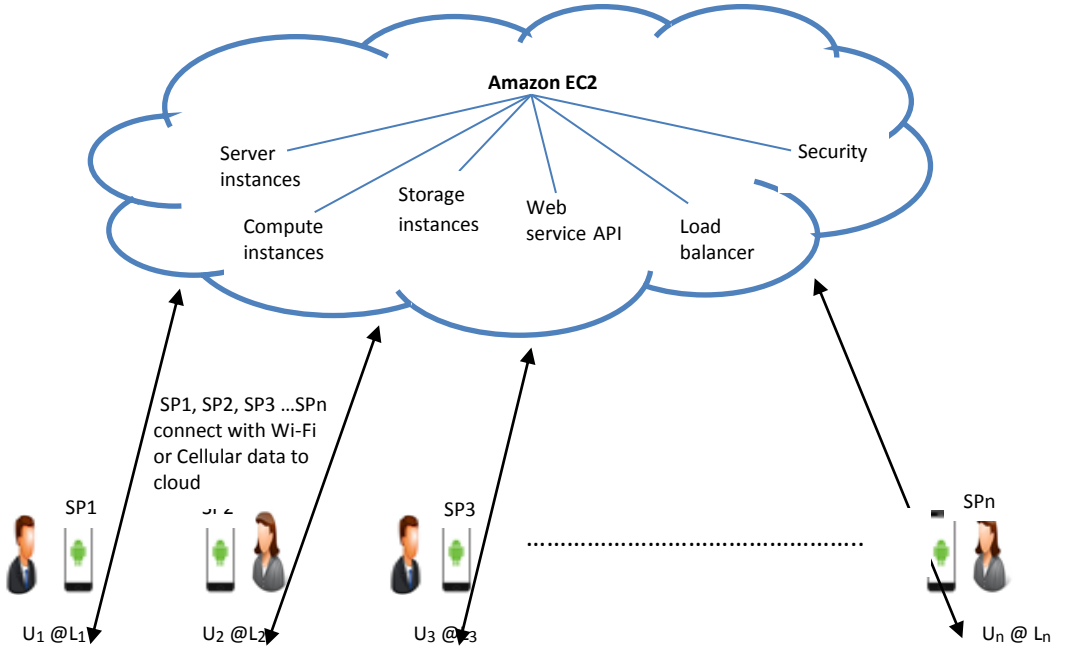

Figure 6. Proposed model for the implementation of Solo App

EXPERIMENTAL SETUP

ARM architectures used for the testing

Fifteen ARM architectures were used for the study with varied configurations as shown in Table 3.

Table 3: ARM architectures used for the testing

Notation	Name of device	Ram	Storage	Android OS	ARM cortex-A architecture
SP1	Samsung Galaxy J7 Prime	3GB	32GB	Marshmallow v6.0.1	Octa-core 1.6 GHz
SP2	Samsung Galaxy J5	2GB	16GB	Marshmallow v6.0.1	Quad-core 1.2 GHz
SP3	Samsung Galaxy J2 Pro	2GB	16GB	Marshmallow v6.0.1	Quad-core 1.5 GHz
SP4	HTC 10	4GB	32GB	Marshmallow v6.0.1	Quad-core 2x1.6 GHz kryo
SP5	HTC One M9	3GB	32GB	v5.0 Lollipop	Octa-core 4x1.5 GHz
SP6	Motorola Moto G5 Plus	4GB	64GB	v7.0 Nougat	Octa-core 2.0 GHz
SP7	Motorola Moto M	4GB	64GB	Marshmallow v6.0.1	Octa-core 2.2 GHz
SP8	Sony Xperia XZ	3GB	32GB	Marshmallow v6.0.1	Quad-core 2x2.15 GHz
SP9	Sony Xperia XA Ultra	3GB	16GB	Marshmallow v6.0.1	Octa-core 4x2.0 GHz
SP10	Lenovo K6 Note	4GB	32GB	Marshmallow 6.0.1	Octa-core 1.4 GHz
SP11	Lenovo P2	4GB	32GB	Marshmallow v6.0.1	Octa-core 2.0 GHz
SP12	Xiaomi Redmi Note 3	3GB	32GB	Marshmallow v6.0.1	Hexa-core 4x1.4 GHz
SP13	Xiaomi mi 5	3GB	32GB	Marshmallow v6.0.1	Quad-core 2x1.8 GHz kryo
SP14	Asus Zenfone 3 max	3GB	32GB	Marshmallow v6.0.1	Quad-core 1.25 GHz
SP15	Asus Zenfone 3 max	4GB	32GB	Marshmallow v6.0.1	Octa-core 1.4 GHz

Results

The results of the studied as tested on 15 ARM Cortex-A architectures are presented below. Different 5 file sizes were used on the testing devices to provide the same bench mark for comparison of performance. The graphs of the encryption/decryption time against the file sizes for SP1-SP5 are depicted in Fig. 7, Fig. 8, Fig. 9, Fig. 10 and Fig. 11 respectively while the graph of encryption throughput is shown in Fig. 12.

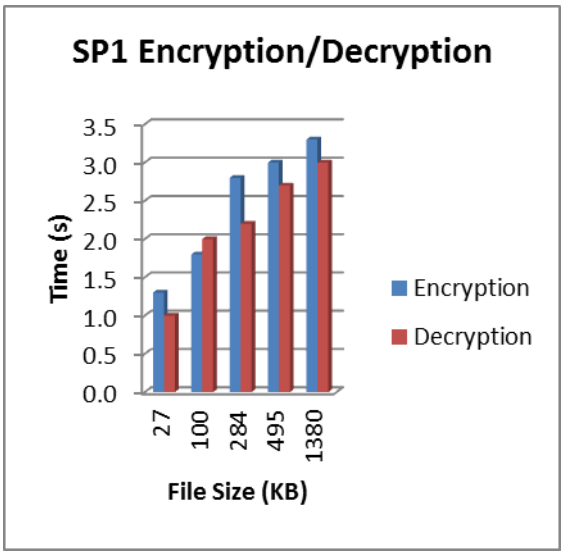

Figure 7. KB/s for encryption/decryption time on SP1

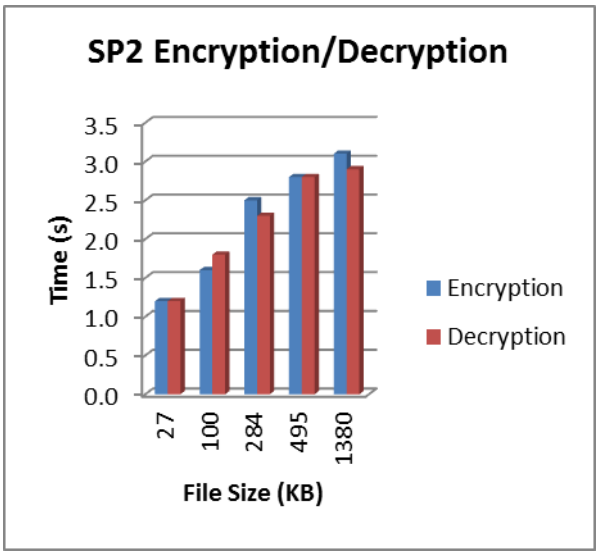

Figure 8. KB/s for encryption/decryption time on SP2

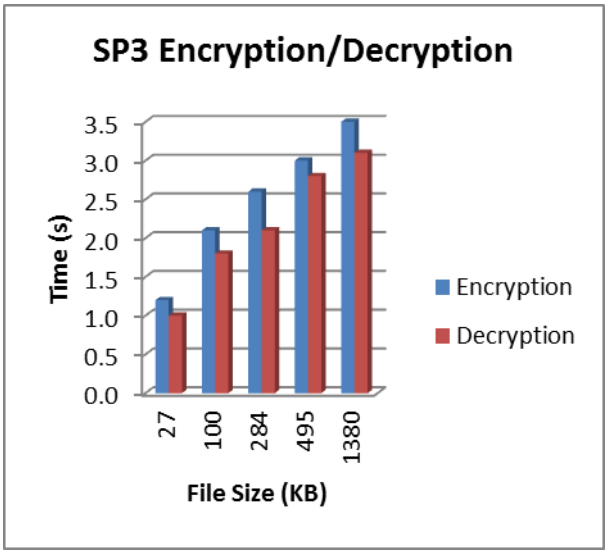

Figure 9. KB/s for encryption/decryption time on SP3

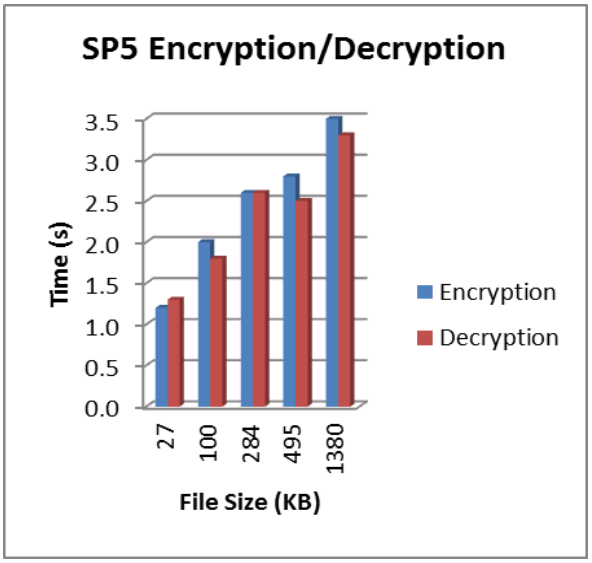

Figure 11. KB/s for encryption/decryption time on SP5

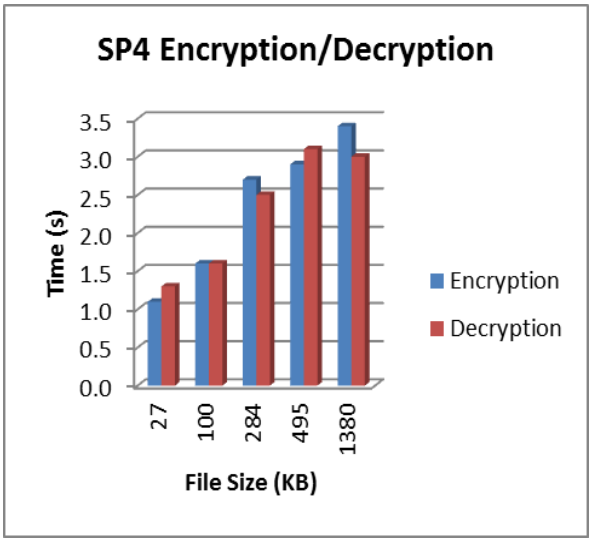

Figure 10. KB/s for encryption/decryption time on SP4

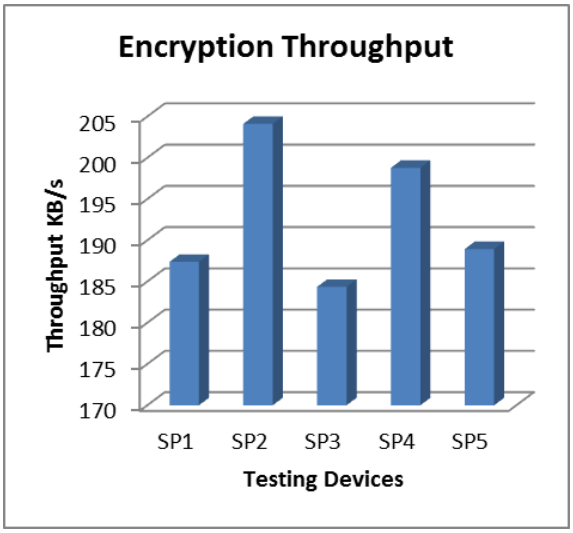

Figure 12. Atomic AES encryption throughput on SP1-SP5

Table 4: Comparison of encryption and decryption time (SP1-SP5)

FILE	SP1		SP2		SP3		SP4		SP5	
(KB)	E(s)	D(s)	E(s)	D(s)	E(s)	D(s)	E(s)	D(s)	E(s)	D(s)
27	1.30	1.00	1.20	1.20	1.20	1.00	1.10	1.30	1.20	1.30
100	1.80	2.00	1.60	1.80	2.10	1.80	1.60	1.60	2.00	1.80
284	2.80	2.20	2.50	2.30	2.60	2.10	2.70	2.50	2.60	2.60
495	3.00	2.70	2.80	2.80	3.00	2.80	2.90	3.10	2.80	2.50
1380	3.30	3.00	3.10	2.90	3.50	3.10	3.40	3.00	3.50	3.30
2286	12.20	11.90	11.20	11.60	12.40	10.80	11.70	11.50	12.10	11.50
TP (KB/s)	187.38	192.10	204.11	197.07	184.35	211.67	195.38	198.78	188.93	198.78

Table 5: Comparison of encryption and decryption time (SP6-SP10)

FILE	SP6		SP7		SP8		SP9		SP10	
(KB)	E(s)	D(s)	E(s)	D(s)	E(s)	D(s)	E(s)	D(s)	E(s)	D(s)
27	1.20	1.00	1.10	1.30	1.30	1.00	1.20	1.00	1.30	1.20
100	1.80	2.10	2.20	1.80	2.10	2.20	1.80	1.70	1.90	2.10
284	2.50	2.10	2.70	2.20	2.40	2.00	2.60	2.40	2.70	2.30
495	2.90	2.60	3.00	2.80	2.80	2.60	3.00	2.70	2.90	2.20
1380	3.10	2.80	3.40	3.10	3.40	3.00	3.40	3.00	3.40	2.80
2286	11.50	10.60	12.40	11.20	12.00	10.80	12.00	10.80	12.20	10.60
TP (KB/s)	198.78	215.66	184.35	204.11	190.50	211.67	190.50	211.67	187.38	215.66

Table 6: Comparison of encryption and decryption time (SP11-SP15)

FILE	SP11		SP12		SP13		SP14		SP15	
(KB)	E(s)	D(s)	E(s)	D(s)	E(s)	D(s)	E(s)	D(s)	E(s)	D(s)
27	1.40	1.40	1.20	1.00	1.50	1.20	1.60	1.30	1.50	1.10
100	1.70	1.50	1.50	1.50	1.80	2.00	2.20	2.00	1.90	1.80
284	2.50	2.30	2.10	1.50	2.70	2.50	2.70	2.30	3.00	2.70
495	2.90	2.50	2.60	2.40	2.80	2.80	3.00	2.70	2.80	3.00
1380	3.30	2.70	2.90	2.30	3.20	3.00	3.40	2.60	3.20	2.80
2286	11.80	10.40	10.30	8.70	12.00	11.50	12.90	10.90	12.40	11.40
TP (KB/s)	193.73	219.81	221.94	262.76	190.50	198.78	177.21	209.72	184.35	200.53

DISCUSSION

The motivation behind this work is to design and implement an efficient and secure encryption algorithm on smartphones which at present use ARM architecture. Our focus is to secure user data by encryption before storing on cloud. User data is first encrypted on user device before it is sent to the cloud storage where there is availability of unlimited storage. With the present efficient and secure implementation of atomic AES on smartphones ARM architecture, the confidentiality, integrity and availability of user data are guaranteed. For compatibility test of the developed mobile application, we installed it on the 15 ARM architectures testing devices used for the study. The compatibility passing rate is 100% because the application functions well as coded on all the devices. For precautionary measures to get most accurate results; all background processing were closed on the devices. The encryption time and decryption time of all the devices were measured in seconds. Table 4 shows the encryption/decryption time for SP1-SP5, Table 5 shows the encryption/decryption time for SP6 -SP10 and Table 6 shows the encryption/decryption time for SP11-SP15. The variations in results are due to the increase in the file sizes used as well as variations in the hardware configurations. This is because as the file size increases the encryption time increases as recorded in Tables 4, 5 and 6. For instance Table 6, 27KB, 100KB, 284KB, 495KB, 1380KB, 2286KB file sizes took 1.40s, 1.70s, 2.50s, 2.90s, 3.30s and 11.80s to encrypt respectively for SP11. Exceptions are shown in some, for example in Table 6 for SP12, where 495KB file size took 2.40s to decrypt while 1380KB file size took 2.30s to decrypt, this is due to Wi-Fi speed fluctuations. Fig. 12 shows the AES encryption throughput of smartphones SP1 – SP5. The throughput is measured in KB/s and determines the power consumption of each of the devices used for the test. This is because as the encryption time and decryption time are reduced, the throughput increases which signifies speed and less power consumption.

Our proposed work and implementation are better over existing security of (Poonguzhali et al., 2016) where user data is secured only on device such that loss of device means loss of data. This is overcome by providing users to store their data in the cloud and can be downloaded whenever needed. We keep track of all the user encrypted data in the cloud such that the list is made available to the user having verified his/her credentials when needed. Agrawal and Kulurkar (2016) proposed encryption for text and pdf files on smartphones, our

implementation can encrypt and decrypt any time of files on smartphones to cloud. Further Zegers et al. (2015) of AES implementation for securing data on smartphone reduced number of round to 7 and increased the encryption key to 144-bit, reduced round of AES is vulnerable to square attack (Ferguson et al., 2000). In our work, we implemented AES at full round our speed of encryption and decryption are better, because of our compact software implementation on ARM architecture. This is in line with (Eisenbarth et al., 2012) who stated that the performance of software implementation is dependent on coding style. Full disk encryption (Gotzfried and Muller, 2014; Muller and Freiling, 2015) takes time and consumes battery but our work allows user to select file for encryption. Ray (2012) proposes graphical password for login, the process is slow instead we proposed 6 digits PIN for login.

Additionally, we developed our API using REST; once encryption is done to cloud, the unencrypted file is deleted from storage and the space overwritten with random bits to prevent recovery. When data is decrypted on user smartphone, it can easily be located in 'SoloApp' folder created by the developed application. There are provisions for PIN change, one time password in case user forgets his/her PIN to assist user regains access to his/her files. This work has been able to enhance storage space of smartphone using cloud and security for user data at rest and in transit.

CONCLUSION

In this work, we have implemented atomic AES on ARM architectures in order to provide security for user data by adopting cloud storage. A mobile application was developed which can function effectively on any ARM architecture of smartphones. We emphasised the importance of securing user sensitive data on mobile devices in this present time against threats. The performance metrics used for analysing the efficient implementation of the work on devices used for the experiment are encryption time, decryption time and throughput. The study results were in line with existing similar studies and more enhanced in terms of implementation and speed. The study made good use of available constraint resources on smartphones and gives better throughput in relation to battery consumption. The study was able to achieve this because of the way the coding was done. In future, it will be of interest to explore the use of asymmetric cryptography to protect user data on ARM architectures.

ACKNOWLEDGEMENT

This research work was sponsored by Tertiary Education Trust Fund, Nigeria and partly supported by Sharda University, Greater Noida, India.

REFERENCES

[1] Agrawal, D. D. and Kulurkar, P. A cloud-based system for enhancing security of android devices using modern encryption standard-ii algorithm, International Journal of Innovations and Advancement in Computer Science, 2016; 5(4),pp. 60-66.

[2] Alsharani, A. M. and Walker, S. New approach in symmetric block cipher security using a new cubical technique, International Journal of Computer Science and Information Technology (IJCSIT), 2015; 7(1), pp. 69-75.

[3] Balasubramanian, N., Balasubramanian, A. and Venkataramani, A. Energy consumption in mobile phones: a measurement study and implications for network applications. In IMC '09 Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, ACM IMC, Chicago, 2009; pp. 280-293.

[4] Banik, S., Bogdanov, A. and Regazzoni, F. Atomic-AES: A compact implementation of the AES encryption/decryption core, In Dunkelman O. and Sanadhya S. K. (Eds): INDOCRYPT 2016, LNCS 10095, 2016; pp. 173 – 190. doi:10.1007/978-3-319-49890-4-10.

[5] Chun, B. G. and Maniatis, P. Augmented smartphone applications through clone cloud execution. In Journal of Hot Topics on Operating Systems (HotOS), 9, 2009; pp. 8 - 11.

[6] Divya, K. and Sadhasivam, N. Secure data sharing in cloud environment using multi-authority attribute based encryption, International Journal of Innovative Research on Computer and Communication Engineering, 2(1), 2014; pp.24-29.

[7] Eisenbarth, T., Gong, Z., Guneysu, T., Heyse, S., Indesteege, S., Kerckhof, S., Koeune, F., Nad, T., Plos, T., Regazzoni, F., Standaert, F., Oldenzeel, L. Compact implementation and performance evaluation of block ciphers in ATtiny devices. Progress in Cryptology-AFRICACRYPT 2012, Lecture notes in computer science, 7374, 2012; pp. 172-87.

[8] Enck, W., Ongtang, M. and McDaniel, P. Mitigating android software misuse before it happens, Networking and Security Research Center, Technical Report, The Pennsylvania State University, USA, 2008; pp. 1-16.

[9] Fang, Z., Han, W. and Li, Y. Permission based Android security: Issues and countermeasures, ELSEVIER Computer & Security, 2014; pp. 1-14..http://dx.doi.org/10.1016/j.cose.2014.02.007

[10] Ferguson N., Kelsey J., Lucks S., Schneier B., Stay M., Wagner D. and Whiting D., Improved cryptanalysis of Rijndael, fast software encryption, LNCS, Springer, 2000, pp. 213- 230.

[11] FIPS 197. Specification for the Advanced Encryption Standard (AES), Federal Information Processing Standards Publications by National Institute of Standards and Technology (NIST), 2001; pp. 1-51. https://doi.org/10.6028/NIST.FIPS.197.

[12] GitHub. Samples of using the Google API, Client Library for Java, 2017. Available at: https:// developers.google.com/api-client-library/java/ apis/.

(Accessed: 5 January 2017).

[13] Gotzfried, J. and Muller, T. Analysing android's full disk encryption feature, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 2014; 5(1), pp. 84-100.

[14] Gupta, A., Rodriguez A. and Preston, K. DebianRunner: running desktop applications on Android smartphones. Technical Report, University of Illinois, USA, 2010.

[15] Hwang, J. Y., Suh, S. B., Heo, S. K., Park, C. J., Ryu, J. M., Park, S. Y. and Kim, C. R. Xen on ARM-Based Secure mobile phones, Proceedings of IEEE 5th Consumer Communication and Networking Conference, 2008; pp. 257-261.

[16] Jeong, Y. S., Kim, H. W. and Park, J. H. An effective locking scheme of smart multimedia devices with convenience and enhanced security, Multimed Tools Appl. Springer Science, 2014; pp. 1-13.

[17] Jiang, X., Wang, X. and Xu, D. Stealthy malware detection and monitoring through vmm-based out-of-the-box semantic view reconstruction. ACM Transactions on Information Systems Security, 2010; 13:12:1–12:28.

[18] Kaur, G., Mahajan, E. N. and Arora, M. Enhancing mobile phone data privacy using the cloud, International Journal of Emerging Research in Management & Technology, 2015; 4(7), pp. 200-203.

[19] Kim, H., Agrawal, N. and Ungureanu, C. Examining storage performance on mobile devices, Proceedings of the 3rd Association of Computing Machinery (ACM), Symposium on Operating Systems Principles (SOSP) and Workshop on Networking, Systems, and Applications on Mobile Handhelds, New York, USA, 2011.

[20] Liu, W., Hu, Y., Rong, H. and Li, R. Optimizing file system performance for android-based consumer electronics by an experimental method, Journal of Convergence Information Technology (JCIT), 2013; 8(17), pp. 50-57.

[21] Lurig, M. PHP reference: beginner to intermediate PHP5, ONLINE: First Edition, 2008. ISBN: 978-1-4357-1590-5. www.phpreferencebook. com. Accessed 12th January, 2017.

[22] Mandavkar, P., Patil, G., Shetty, C. and Parkar, V. SMS security for android mobile using combine cryptographic algorithms, International Journal of Advanced Research in Computer and Communication Engineering, 2014; 3(4), pp. 6221-6225.

[23] Mendon, A. A. and Sass, R. A hardware file system implementation for high-speed secondary storage, in reconfigurable computing and FPGAS. IEEE International Conference on ReConFig'08, 2008; pp. 283-288.

[24] Muller, T. and Freiling, F. C. A systematic assessment of the security of full disk encryption, IEEE Transactions on Dependable and Secure Computing, 2015; 12(5), pp. 491-503.

[25] Nayadkar, P. P. and Parne, B. L. A survey on different backup and restore techniques used in mobile devices, International Journal of Computer Science and Information Technologies, 2014; 5(6), pp. 8236-8238.

[26] Nayadkar, P. P. Automatic and secured backup and restore technique in android, IEEE Conference on Innovations in Information, Embedded and Communication Systems, March 2015; pp. 1-4.

[27] Oberheide, J., Veeraraghavan, K., Cooke, E., Flinn, J. and Jahanian, F. Virtualized in-cloud security services for mobile devices. In Proceedings of the First Workshop on Virtualization in Mobile Computing, MobiVirt '08, 2008; pp. 31–35.

[28] Olaleye, S. B. and Ojha, S. K. Compact software implementation of AES on atomic smartphones architecture, International Journal of Innovative Science, Engineering and Technology, 2017; 4(2), pp. 102-112.

[29] Poonguzhali, P., Dhanokar, P., Chaithanya, M. K. and Patil, M. U. Secure storage of data on android based devices, International Journal of Engineering and Technology, 2016; 8(3), pp. 177-182.

[30] Ray, P. P. Ray's scheme: graphical password based hybrid authentication system for smart hand held devices, International Journal of Computer Trends and Technology, 2012; 3(2), pp. 235-241.

[31] Richardson, L. and Ruby, S. RESTful web services, USA: O'Relly Media Inc., 2007.

[32] Rihan, S. D., Khalid, A. and Osman, S. E. F. A performance comparison of encryption algorithms AES and DES. International Journal of Engineering Research and Technology (IJERT), 2015; 4(12), pp. 151-154.

[33] Rusello, G., Crispo, B., Fernandes, E. and Zhauniarovich, Y. YAASE: yet another android security extension, Proceedings of IEEE 3rd International Conference on Social Computing and Privacy, Security, Risk and Trust, 2011; pp.1033-1040.

[34] Saini, V. and Bangar, P. Design and implementation of advanced encryption standard algorithm-128 using verilog, International Journal of Engineering and Advantage Technology (IJEAT), 3(5), 2014; pp. 265-268.

[35] Satyanarayanan, M., Bahl, V., Caceres, R. and Davies, N. The case for vm-based cloudlets in mobile computing. IEEE Pervasive Computing, Special Issue on VM, 2009; 8(4), pp. 2-11.

[36] Tao, H. and Adams, C. Pass-Go: a proposal to improve the usability of graphical passwords, International Journal of Network Security, 2008; 7(2), pp. 273-292.

[37] Tarle, P. R. Comparative study of smartphone security techniques, International Journal of Emerging

Technology and Advanced Engineering, 2015; 5(2), pp. 240-244.

[38] Vijay, J. V. and Bansode, B. ARM processor architecture, evolution and applications, International Journal of Science, Engineering and Technology Research, 2015; 4(10), pp. 3385-3387.

[39] Vittapu, M. S., Sunkari, V., Abate, A. Y. and Sreenivas, N. A proposed solution to secure mcc uprising issue and challenges in the domain of cyber security, Open Journal of Mobile Computing and Cloud Computing, 2015; 2(1), pp. 19-33.

[40] Wang, Z., Murmuria, R. and Stavrou, A. Implementing and optimizing an encryption filesystem on android, Mobile Data Management (MDM), July 2012.

[41] Zegers, W., Chang, S., Park, Y. and Gao, Y. J. A lightweight encryption and secure protocol for smartphone cloud, published in Service-Oriented Engineering (SOSE), IEEE Conference Publications, doi:10.1109/sose.2015.47, 2015; pp. 259-266.

[42] Zhao, K., Jin, H., Zou, D., Chen, G. and Dai, W. Feasibility of deploying biometric encryption in mobile cloud computing, Proceedings of 8th ChinaGrid Annual Conference, 2013; pp. 28-33. doi:10.1109/ChinaGrid 2013.10.