IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 4 Issue 2, February 2017
ISSN (Online) 2348 – 7968 | Impact Factor (2016) – 5.264
www.ijiset.com

Compact Software Implementation of AES on Atomic Smartphones Architecture

Solomon Babatunde Olaleye[1] and Shrikant Ojha[2]

[1]Research Scholar, Department of Computer Science & Engineering, Sharda University, Greater Noida, 201306, India
[2]Dean Research, Research and Technology Development Centre, Sharda University, Greater Noida, 201306, India

Abstract

The smartphones have enhanced the way of life of present day generation in terms of businesses, communications, Internet browsing and so on. The hardware architecture of smartphones is becoming compact year by year due to reduction in the number of gates used, latency cycles and data paths. However, security plays a major role in order to enjoy optimally the functionalities of smartphones. This study therefore design and implement a compact software using AES-128 bit. Existing AES attacks based on records were studied and guided the application development. The work was evaluated in terms of encryption time, decryption time and throughput. The results were recorded and compared with earlier results. Our results are found to be faster and can be used to provide enhanced security on smartphones using cloud.

Keywords: *Sensitive Data, Solo App, Cloud server, Smartphones Security, AES Implementation.*

1. Introduction

Modern day technology has greatly improved human ways of life in terms of voice, text or image communications, Internet surfing, record keeping and easy retriever out of numerous benefits of technology advancement. However, these benefits come with their own challenges. There are adversaries with modified intension can eavesdrop the communications sent between user A and user B. In order to solve this problem encryption techniques are used to disguise the intruder in the network. Encryption has become an outstandingly significant means of securing messages (data) sent over communication networks [1].

This research work presented a compact software implementation of AES on atomic smartphone architecture using java + xml on Android studio 2.2.3. We used Amazon EC2 (Elastic Compute Cloud) for cloud server and PhP for creating web services. Atomic AES is a hardware compact architecture for implementation of AES encryption and decryption proposed by Banik et al [2]. The circuit was built on Moradi et al [3] proposed circuit of Eurocrypt 2011. The design goal was solely reduction in circuit size by reducing number of gates, reduction in number of clock cycles and data paths. To achieve this, they used a holistic approach that optimizes the total design and not individual components. Hence, they achieved an implementation that requires just 2400 GE and needs 226 clock cycles which was 23% smaller than any previously published implementations. The rest of the paper is organised as follows: section 2 discusses a compact implementation of AES dual encryption and decryption. Section 3 identified published records of different attacks against AES which guided this work. Section 4 shows the proposed improved algorithm design. In addition, section 5 explains a compact implementation of the improved algorithm. Results, discussion and comparison with other work are presented in section 6 while the research paper concluded in section 7.

2. Compact Implementation of AES Dual Encryption/Decryption

For a compact architecture implementation of AES encryption and decryption, Banik et al [2] investigated whether the basic circuit of Moradi et al [3] can be modified or slightly adjusted to provide dual functionalities of encryption and decryption while ensuring that the hardware overhead is kept low. Earlier works with three smallest known circuits that perform the dual functionalities of both encryption and decryption are;

1. Grain of Sand implementation at 3400 GE by Feldhofer et al [4].
2. 8-bit serial implementation at 4037 GE by Mathew et al [5].
3. 32-bit serial implementation at 5400 GE by Satoh et al [6].

However, Banik et al [2] presented an atomic-AES with 8-bit serial architecture that performs the dual functionalities of encryption and decryption which has a circuit size of around 2645 GE and latency of 226 cycles. This is significant improvement ever recorded.

2.1 AES Operations Sequence

The AES-128 bits operations sequence for encryption and decryption are not the same. For encryption, they are as listed below;

1. Add whitening key
2. For 1 to 9 Rounds
   - Substitution layer
   - Shiftrows
   - Mixcolumn
   - Add roundkey
3. For 10th Round
   - Substitution layer
   - Shiftrows
   - Add roundkey

For the decryption operations, the operations sequence is the inverse of encryption, i.e.

- Inverse Shiftrows
- Inverse Mixcolumn
- Add roundkey
- Inverse S-box

But Banik et al proposed the following order for decryption because the order is exactly the mirror inverse of the AES encryption and that no swapping of the Add roundkey and the Inverse Mixcolumn is needed.

- Inverse Mixcolumn
- Inverse Shiftrows
- Inverse S-box + Add roundkey

2.2 8-bit Serial AES Architecture

The 8-bit compact AES architecture of [3] and [2] are as depicted in figure 1 and figure 2 respectively.

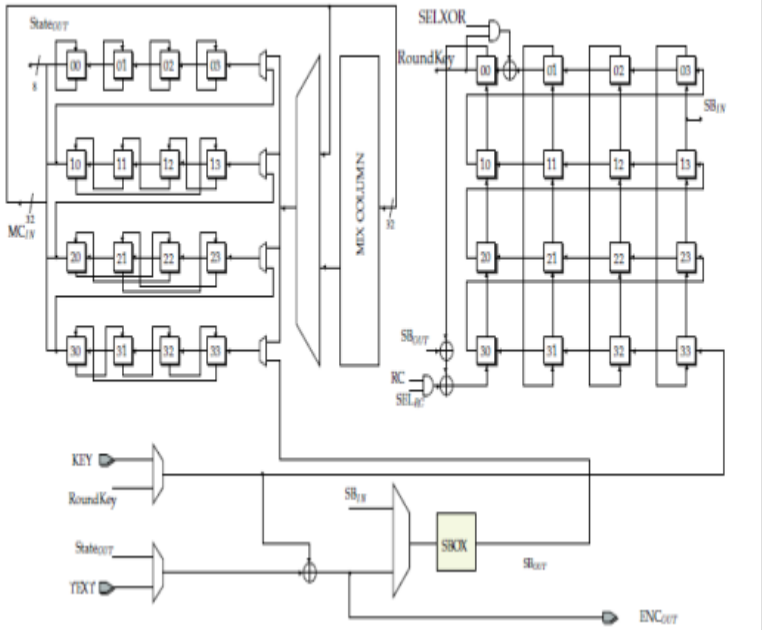


Fig. 1: 8-bit Serial Architecture [3]

The architecture depicted in figure 1 was studied and modified by Banik et al [2] to achieve the atomic-AES. To achieve the architecture of figure 2 the following logic components were added over the basic circuit of figure 1.

a. 2 extra 8-bit multiplexers in data path of the state.
b. 3 extra 8-bit XOR gates in data path of the key
c. 24 extra AND gates in data path of the key
d. 1 extra 8-bit multiplexer, 1 extra 8-bit XOR gate, 16 extra AND gates during addition of state-key.
e. Other extra logic used are in the S-box and its inverse, Mixcolumn and its inverse, and Round constants and their inverses.

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 4 Issue 2, February 2017
ISSN (Online) 2348 – 7968 | Impact Factor (2016) – 5.264
www.ijiset.com

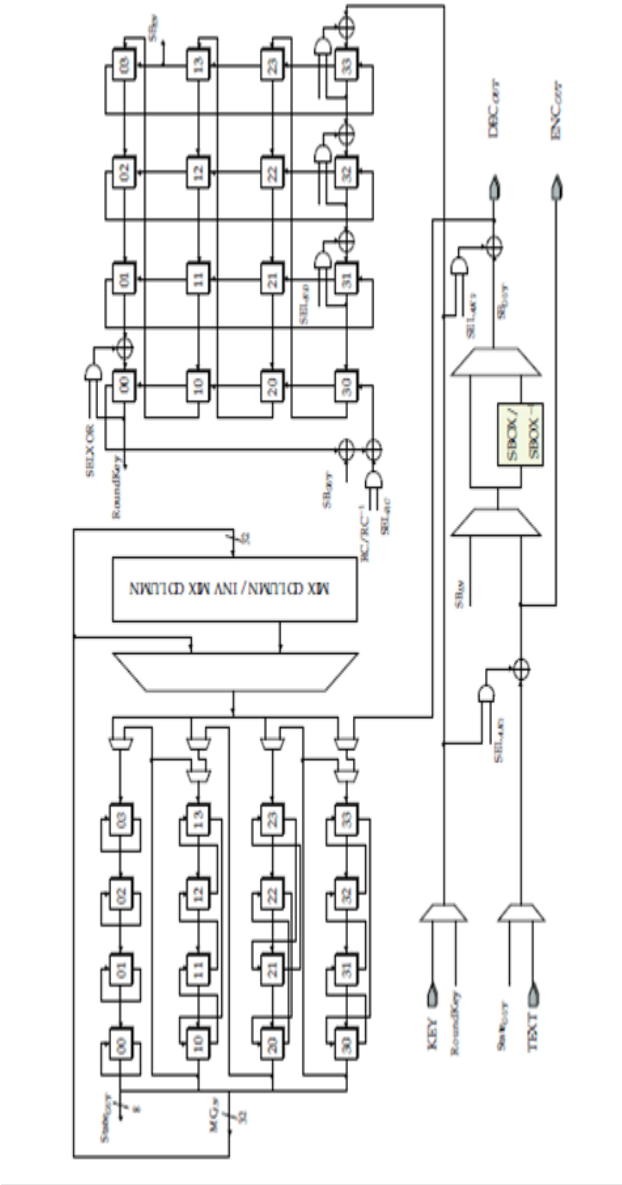


Fig. 2: Atomic-AES 8-bit Encryption and Decryption Architecture [2]

SN	Architecture	Type	Area GE	Latency (Clock Cycles)	Energy (nJ)	TP Max (Mbps)
1	Modular AES [26]	ED	10,799	64	-	241
2	8-bit serial [3]	E	2400	226	8.4	-
3	Grain of sand [4]	ED	3400	1032/1165	46.4/52.4	9.9/8.8
4	8-bit serial [5]	ED	4037	336/216	3.9/2.5	432/671
5	32-bit serial [6]	ED	5400	54/54	-	311
6	Atomic-AES [2]	ED	2645	226/226	3.3/2.2	94.4/57.8

Table 1: Performance Comparison of Atomic – AES with Previous AES Architecture

Table 1 showed a clear comparison among the earlier works on compact AES architectures and atomic-AES architecture, where E stands for Encryption and ED means Encryption/Decryption. The major difference in atomic-AES over others are the reduction in the number of gates, clock cycles and data paths. Atomic-AES has the minimum 2645 GE for encryption/decryption with latency cycle of 226. The architecture is beneficial in lightweight creation of block cipher modes that involve access to both the encryption and decryption blocks.

3. AES Attacks

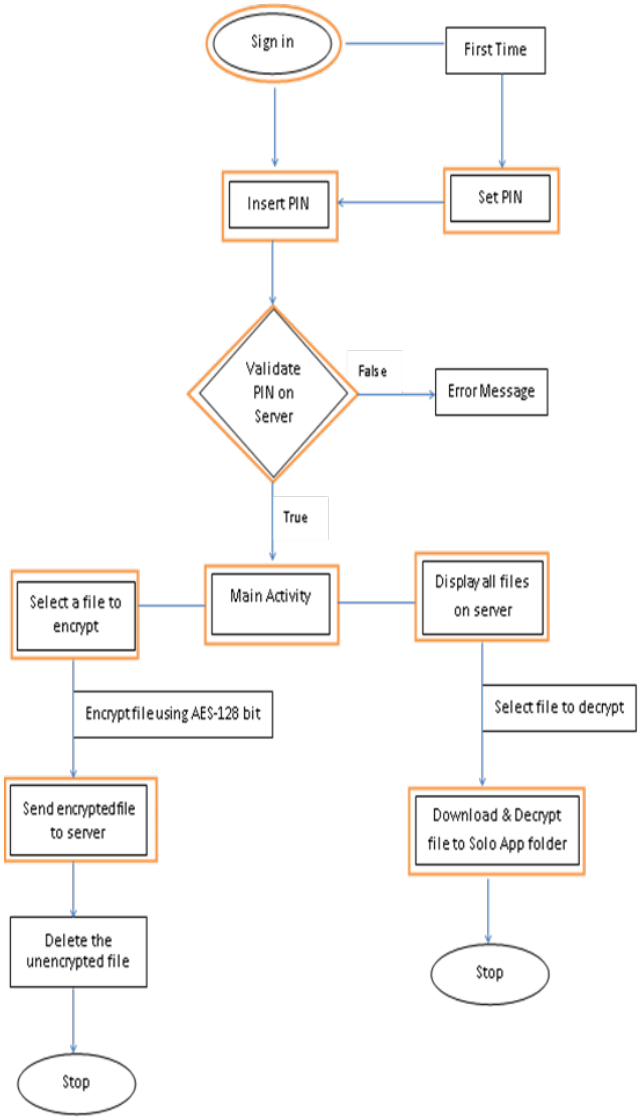
The AES can be implemented in hardware and software to encrypt sensitive data [7]. AES security was thoroughly analysed by the National Security Agency (NSA) which declared in 2003 that AES has no known vulnerability and that the longer-key variants of AES-192 and AES-256 can be used to protect top secret of US government data [12]. The design of AES could resist any modern cryptanalysis. However, many authors had published different attacks against the reduced rounds of AES. Some of these attacks are side channel attacks [8-11], key recovery attack [13-17], Square attack [18], differential power analysis attack [19], and Biclique attack [20-21].

These AES attacks were studied and guided the compact software implementation of AES on smartphones architecture. However, work is on-going trying these attacks on the implementation of our improved algorithm.

Fig. 5: Design of improved algorithm for security of smartphones data

4. Proposed Improved Algorithm

The AES algorithm was studied and improved upon for its software implementation on smartphones architecture to provide enhanced security for user sensitive data using cloud as shown in figure 5. The improved algorithm authenticates users with six digits PIN which is an improvement over existing and commonly uses four digits PIN. Further user can select file to encrypt to server and can as well file to decrypt from the server whenever needed.



4.1 The algorithm steps

Step 1. Sign in – Sign in/ Register using Google API Client and save user's data to server (Get username, user Email and user Unique Id for registering the user on server.)
Step 2. Set PIN – Ask user to set a PIN if the user is registering for the first time.
Step 3. Insert PIN – Insert your selected PIN to continue.
Step 4. Main Activity – Choose whether to encrypt or decrypt files.
Step 5. Open a file chooser Intent to fetch the file from device.
Step 6. Create a random key and encrypt file (using AES-128).
Step 7. Upload the file and random Key to server using REST API.
Step 8. Delete the unencrypted file from storage.
Step 9. To decrypt display all encrypted files on server
Step 10. Select the file to decrypt
Step 11. Download file and Decrypt using random key got from server.
Step 12. Save the decrypted file to Downloads / SoloApp directory.

5. Compact Implementation of the Improved Algorithm

Mobile application was developed using Android studio 2.2.3 written in java + xml. The application was named Solo App and AES-128 bit was used for encryption and decryption. Representational state transfer (REST) services API [22] was used for distributing data. And it was written in PhP [23]. It 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 AES attacks, the encryption key is randomly generated in java using class SecretKeySpec and KeyGenerator class for the AES algorithm. Two databases were created, first is to store the randomly generated secret keys and the second to store the ciphertext using Amazon server.

To sign-in to use the Solo App which is an improved version of an earlier version accepted for publication in Indian Journal of Science and Technology [24]. Google

API client [25] for sign-in authentication was used to fetch user data to the server. This is because every Android phone user has a google mail account. Moreover, in case the user forgets his/her sign-in PIN a message for PIN recovery will be sent to the user gmail account. Figure 6 and figure 7 show the Solo App interface for sign-in and PIN set respectively for new user. User has to set a PIN for the first time and such PIN will be used as a means of security for subsequent sign-in. Please note here that the application cannot work without Internet connectivity either by using Internet Service Provider's network (ISP) or Wireless Fidelity (Wi-Fi).

Fig. 6: Google Sign-in Interface

Fig. 7: SET PIN Interface

The Solo App was tested in real life on ten different Android devices of various configurations as listed in (table 4).

To carry out the test, files of different sizes and types were used as shown in table 5.

Table 5: File Sizes and File Types Used

SN	FILE SIZES (KB)	FILE TYPE
1	0.441	TXT
2	1.2	JSON
3	6.81	PHP
4	13.2	HTML
5	44.1	DOC
6	155	IMAGE
7	229	DOC
8	296	PDF
9	3660	DOC
10	4760	PDF

Figure 8 below shows the major activity interface for encryption and decryption of any file size and file type on Android smartphones storage. Once a user presses the ENCRYPT option he/she will be able to browse and select any file type to encrypt. The encryption is done and the encrypted file is stored on Amazon server while the unencrypted file on smartphone's storage is deleted immediately to create space. Subsequently if the encrypted file is needed, the user will press the DECRYPT option, list of all files already encrypted by the user on server will be displayed for the user to select the file to decrypt. The encrypted files on server maintain the same name with the unencrypted files on smartphone's storage for easy identification of files. The encrypted files on server when chosen will be decrypted into SoloApp folder on smartphones for ease of location.

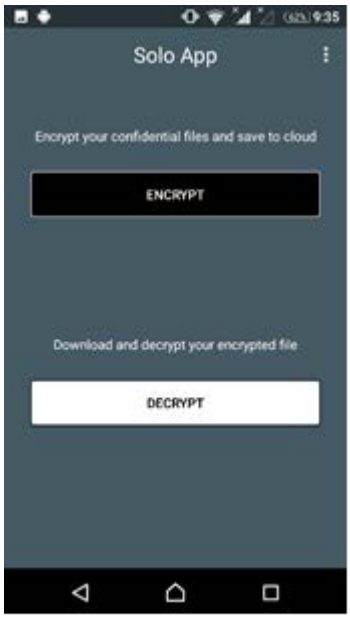


Fig. 8: User interface for encryption and decryption

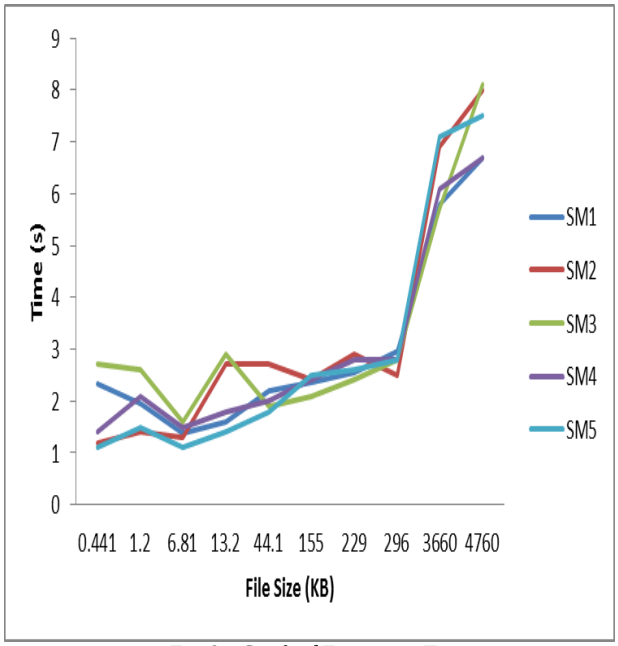


Fig. 9a: Graph of Encryption Time

6. Results and Discussion

In this section we present the results of our work. The encryption and decryption processes were carried out using Wi-Fi and were repeated three types and the average time in seconds were recorded. (Table 6) shows the results of the encryption time on the ten smartphones used for the study while (table 7) shows the decryption time. The graph of the encryption time against the file sizes is as shown in SM1 to SM5 in figure 9a and SM6 to SM10 in figure 9b. Similarly, SM1 to SM5 in figure 10a and SM6 to SM10 in figure 10b show the graph of the decryption time against the file sizes. The graph results show that as the file size increases the time for encryption and decryption also increase.

The proposed algorithm protects smartphones sensitive data in transit as well at rest in the cloud because the files are encrypted and stored in cloud storage. Hence, this study has overcome a study carried out by Wang et al [27] on filesystem encryption on Android smartphones that protect only data kept on internal or external storage but not data over network. Furthermore, when comparing the speed of encryption and decryption, the times it took to encrypt and decrypt on various smartphones are faster when compared with the work of [28]. Their encryption of different file sizes is as shown in table 8.

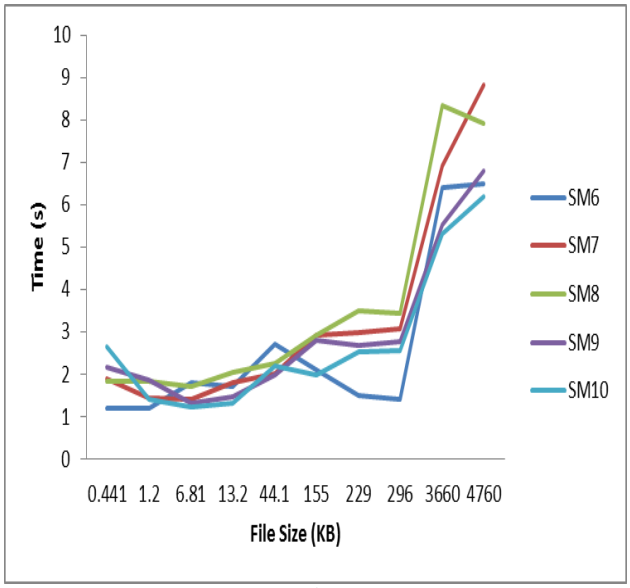


Fig. 9b: Graph of Encryption Time

9	1715000	38
10	3156000	72

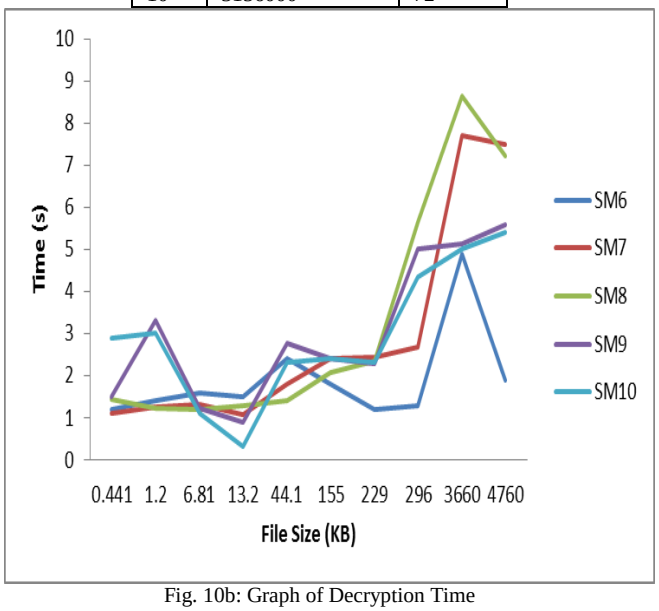


Fig. 10b: Graph of Decryption Time

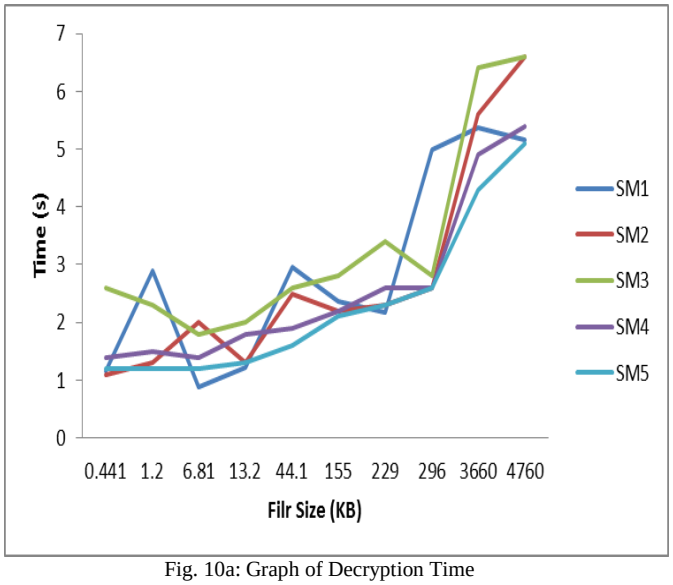


Fig. 10a: Graph of Decryption Time

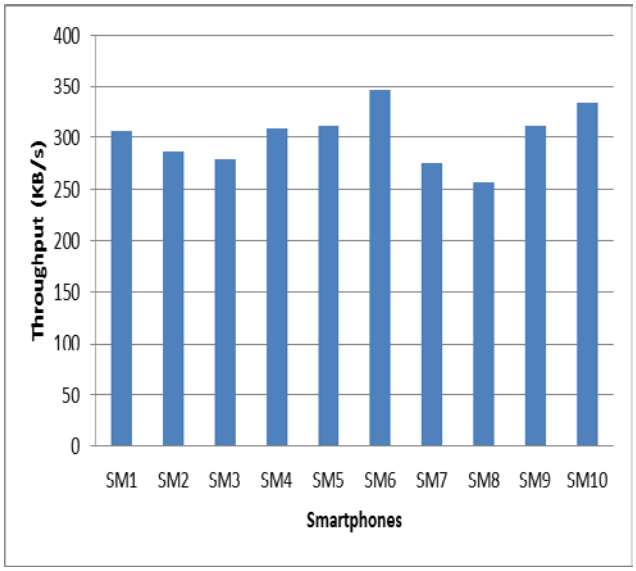


Fig. 11: Graph of Encryption Throughput

Table 8: AES Encryption [28]

SN	File Size (Bytes)	Time (s)
1	72000	2
2	154000	4
3	203000	4
4	351000	8
5	476000	11
6	589000	13
7	718000	16
8	1222000	28

The throughput of each of the smartphones encryption is computed and presented in (table 9) while the graph is as shown in figure 11. The high throughput values indicate that the software implementation of Atomic AES consumes less computing power on smartphones. The variations in

throughput values are due to the differences in device configurations used for the study.

7. Conclusion

This study proposed an improved security algorithm for protecting user sensitive data on smartphones. The objective of the paper is to improve and implement a compact AES-128 bit algorithm in a such a way that it will provide the desired security and enhanced the constraints resources on smartphones using cloud server. The results were recorded in terms of encryption time, decryption time and throughput as performance metrics. When compared with earlier work, our work was found to be reasonably fast in terms of encryption and decryption time with minimal overhead consumption of already limited smartphones computing resources. Further storage was greatly enhanced because the lifted encrypted files are stored using cloud infrastructures while the former unencrypted files are deleted out of smartphone's storage to create more spaces. The Solo App developed has been tested and stored in Google Play Store for use. For future work feedback from users of the application will be used to improve the work.

References

[1] I. Azmi, S. Zulhuda and S. Jarot, "Data Breach On The Critical Information Infrastructures: Lessons From the Wikileaks, Cyber Security, Cyber Warfare And Digital Forensic (Cyber Sec)", International Conference, 2012, pp. 306 – 311.

[2] S. Banik, A. Bogdanov and F. Regazzoni, "Atomic-AES: A Compact Implementation of the AES Encryption/Decryption Core", In O. Dunkelman and S. K. Sanadhya (Eds): INDOCRYPT 2016, LNCS 10095, 2016, pp. 173 – 190. DOI:10.1007/978-3-319-49890-4-10.

[3] A. Moradi, A. Poschmann, S. Ling, C. Paar and H. Wang, "Pushing the Limits: A Very Compact and a Threshold Implementation of AES", In Eurocrypt 2011, LNCS, Vol. 6632, 2011, pp. 69 – 88.

[4] M. Feldhofer, J. Workerstorfer and V. Rijmen, "AES Implementation on a Grain of Sand", IEEE Proceedings of Information Security, Vol. 152, No. 1, 2005, pp. 13 – 20.

[5] S. Mathew, S. Satpathy, V. Suresh, M. Anders, H. Kaul, A. Agarwal, S. Hsu, G. Chen and R. K. Krishnamurthy, 340mV-1.1V, 289 Gbps/W, 2090-gate NanoAES Hardware Accelerator with Area-Optimized Encrypt/Decrypt GF $(2^4)^2$ Polynomials in 22 nM tri-gate CMOS", IEEE Journal of Solid-State Circuits, Vol. 50, 2015, pp. 1048 – 1058.

[6] A. Satoh, S. Morioka, K. Takano and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization", Asiacrypt 2001, LNCS, Vol. 2248, 2001, pp. 239 – 254.

[7] M. Rouse, "Advanced Encryption Standard (AES)", www.searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard. Accessed 9th January, 2017.

[8] K. Merrit, "Differential Power Analysis Attacks on AES", Cryptography II, VCSG-706, May 2012, pp. 1-17, https://people.rit.edu/kjm5923/DPA_attacks_on_AES_pdf.

[9] G. Irazoqui, M. S. Inci, T. Eisenbarth and B. Sunar, "Wait a Minute! A Fast, Cross-VM Attack on AES", Cryptology ePrint Archive, Report 2014/435, 2014. https://eprint.iacr.org/2014/435.pdf.

[10] D. Gullasch, E. Bangerter and S. Krenn, "Cache Games-bringing Access-based Cache Attacks on AES to Practice", IEEE Symposium on Security and Privacy 2011, pp. 490 – 505.

[11] Y. Yarom and K. E. Falkner, "Flush + Reload: A High Resolution, Low Noise, I_3 Cache Side-Channel Attack", IACR Cryptology ePrint Archive 2013, 2013.

[12] National Security Agency (NSA), National Policy on the Use of AES to Protect National Security Systems and National Security Information, June 2003, http://www.cnss.gov/Assets/pdf/cnssp_15_fs.pdf

[13] A. Biryukov, D. Khovratovich and I. Nikolic, "Distinguisher and Related-key Attack on the Full AES-256", In Crypto 2009, LNCS, Springer, 2009.

[14] A. Biryukov and D. Khovratovich, "Related Key Cryptanalysis of the Full AES-192 and AES-256", 2009. http://eprint.iacr.org/2009/317.pdf.

[15] J. Kim, S. Hong and B. Preneel, "Related-key Rectangle Attacks on Reduced AES-192 and AES-256", In FSE 2007, Vol. 4593, LNCS, Springer, 2007, pp. 225 -241.

[16] E. Biham, O. Dunkelman and N. Keller, "Related Key Boomerang and Rectangle Attacks", In Eurocrypt 2005, Vol. 3494, LNCS, Springer 2005, pp. 507 – 525.

[17] A. Biryukov, O. Dunkelman, N. Keller, D. Khovratovich and A. Shamir, "Key Recovery Attacks of Practical Complexity on AES Variants with up to 10 Rounds", Annual International Conference on the Theory and Applications of Cryptographic Techniques, Eurocrypt 2010: Advances in cryptology, 2010, pp. 299 – 319.

[18] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner and D. Whiting, "Improved Cryptanalysis of Rijndael, Fast Software Encryption", FSE 2000, LNCS, Vol. 1978, Springer, 2000, pp. 213 – 230.

[19] K. Smith, "Methodologies for Power Analysis Attacks on Hardware Implementations of AES", Master's Thesis, Rochester Institute of Technology, 2009.

[20] D. Khovratovich, C. Rechberger and A. Savelieva, "Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family", Available at https://eprint.iacr.org/2011/286.pdf, 2011.

[21] A. Bogdanov, D. Khovratovich and C. Rechberger, "Biclique Analysis of the full AES", Cryptology ePrint Archive – IACR, available at https://eprint-iacr.org/2011/449.pdf.

[22] L. Richardson and S. Ruby, RESTful Web Services, USA: O'Relly Media Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2007.

[23] M. Lurig, PHP Reference: Beginner to Intermediate PHP5, ONLINE: First Edition, ISBN: 978-1-4357-1590-5, 2008. www.phpreferencebook.com. Accessed 12th January, 2017.

[24] S. B. Olaleye, I. Ranjan and S. Ojha, "SoloEncrypt: A Smartphone Storage Enhancement Security Model for Securing Users Sensitive Data", Indian Journal of Science and Technology (Accepted).

[25] GitHub, Inc., Samples of using the Google API, Client Library for Java, https://developers.google.com/api-client-library/java/apis/. Accessed 5th January, 2017.

[26] S. Mangard, M. Aigner and S. Dominikus, "A Highly Regular and Scalable AES Hardware Architecture", IEEE Transactions on Computers, Vol. 52, No. 4, April 2003, pp. 483 – 491.

[27] Z. Wang, R. Murmuria and A. Stavrou, "Implementing and Optimizing an Encryption Filesystem on Android", 2012 IEEE 13th International Conference on Mobile Data Management (MDM), 23 – 26 July 2012.

[28] C. J. Ezeofor and A. G. Ulasi, "Analysis of Network Data Encryption & Decryption Techniques in Communication Systems", International Journal of Innovative Research in Science, Engineering and Technology, Vol. 13, Issue 12, December 2014, pp. 17797 – 17807.

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 4 Issue 2, February 2017
ISSN (Online) 2348 – 7968 | Impact Factor (2016) – 5.264
www.ijiset.com

Appendix

Tables

Table 4: Android Devices used for the Test

Sn	Name of Device	RAM	Storage	OS	Processor
SM1	YUPHORIA(7.1)	2GB	16GB	Nougat 7.1	Quad Core 1.2 GHz
SM2	LYF 7s	2GB	16GB	Marshmallow 6.0	Quad Core 1.3 GHz
SM3	Coolpad Note 3	3GB	16GB	Marshmallow 6.0	Octa Core 1.3 GHz
SM4	Moto G4 Plus	2GB	32GB	Marshmallow 6.1	Octa Core
SM5	Redmi 2	2GB	16GB	Kitkat 4.4	Quad Core 1.2 GHz
SM6	YUPHORIA(5.0)	2GB	16GB	Lollipop 5.0	Quad Core 1.2 GHz
SM7	MediaCom PhonePAd Duo s531	1GB	16GB	Kitkat 4.4	Octa Core 1.4 GHz
SM8	Samsung GT-P5200 (TAB)	1GB	16GB	Kitkat 4.4	Dual-core 1.6 GHz
SM9	Xperia Z5	4GB	64GB	Nougat 7.0	Quad Core 1.4 GHz
SM10	Oneplus 2	3GB	32GB	MarshMallow 6.0.1	Quad Core 1.2 GHz

Table 6: Encryption Time

SOLO APP TEST- ENCRYPTION TIME										
FILE (KB)	SM1 (s)	SM2 (s)	SM3 (s)	SM4 (s)	SM5 (s)	SM6 (s)	SM7 (s)	SM8 (s)	SM9 (s)	SM10 (s)
0.441	2.33	1.2	2.7	1.4	1.1	1.2	1.89	1.84	2.16	2.65
1.2	1.96	1.4	2.6	2.1	1.5	1.2	1.43	1.82	1.87	1.42
6.81	1.37	1.3	1.6	1.5	1.1	1.8	1.4	1.7	1.33	1.24
13.2	1.61	2.7	2.9	1.8	1.4	1.7	1.81	2.06	1.48	1.32
44.1	2.19	2.7	1.9	2.0	1.8	2.7	2.02	2.25	2.0	2.2
155	2.37	2.4	2.1	2.4	2.5	2.1	2.91	2.93	2.8	2.0
229	2.54	2.9	2.4	2.8	2.6	1.5	2.97	3.49	2.68	2.52
296	2.96	2.5	2.8	2.8	2.8	1.4	3.09	3.43	2.78	2.56
3660	5.79	6.9	5.7	6.1	7.1	6.4	6.91	8.33	5.54	5.32
4760	6.7	8.0	8.1	6.7	7.5	6.5	8.82	7.93	6.8	6.2

Table 7: Decryption Time

SOLO APP TEST - DECRYPTION TIME										
FILE (KB)	SM1 (s)	SM2 (s)	SM3 (s)	SM4 (s)	SM5 (s)	SM6 (s)	SM7 (s)	SM8 (s)	SM9 (s)	SM10 (s)
0.441	1.15	1.1	2.6	1.4	1.2	1.2	1.1	1.44	1.51	2.9
1.2	2.89	1.3	2.3	1.5	1.2	1.4	1.27	1.23	3.31	3.0
6.81	0.89	2.0	1.8	1.4	1.2	1.6	1.32	1.2	1.23	1.11
13.2	1.22	1.3	2.0	1.8	1.3	1.5	1.09	1.28	0.89	0.32
44.1	2.95	2.5	2.6	1.9	1.6	2.4	1.79	1.41	2.78	2.32
155	2.37	2.2	2.8	2.2	2.1	1.8	2.42	2.07	2.4	2.4
229	2.17	2.3	3.4	2.6	2.3	1.2	2.44	2.34	2.29	2.32
296	4.99	2.6	2.8	2.6	2.6	1.3	2.67	5.64	5.02	4.34
3660	5.38	5.6	6.4	4.9	4.3	4.9	7.71	8.64	5.14	5.02
4760	5.16	6.6	6.6	5.4	5.1	1.9	7.5	7.21	5.58	5.42

Table 9: Encryption Throughput

SOLO APP TEST - ENCRYPTION THROUGHPUT										
FILE (KB)	SM1 (s)	SM2 (s)	SM3 (s)	SM4 (s)	SM5 (s)	SM6 (s)	SM7 (s)	SM8 (s)	SM9 (s)	SM10 (s)
0.441	2.33	1.2	2.7	1.4	1.1	1.2	1.89	1.84	2.16	2.65
1.2	1.96	1.4	2.6	2.1	1.5	1.2	1.43	1.82	1.87	1.42
6.81	1.37	1.3	1.6	1.5	1.1	1.8	1.4	1.7	1.33	1.24
13.2	1.61	2.7	2.9	1.8	1.4	1.7	1.81	2.06	1.48	1.32
44.1	2.19	2.7	1.9	2.0	1.8	2.7	2.02	2.25	2.0	2.2
155	2.37	2.4	2.1	2.4	2.5	2.1	2.91	2.93	2.8	2.0
229	2.54	2.9	2.4	2.8	2.6	1.5	2.97	3.49	2.68	2.52
296	2.96	2.5	2.8	2.8	2.8	1.4	3.09	3.43	2.78	2.56
3660	5.79	6.9	5.7	6.1	7.1	6.4	6.91	8.33	5.54	5.32
4760	6.7	8.0	8.1	6.7	7.5	6.5	8.82	7.93	6.8	6.2
9165.751	29.82	32	32.8	29.6	29.4	26.5	33.25	35.78	29.44	27.43
TP (KB/S)	307.3692	286.4297	279.4436	309.6538	311.7602	345.8774	275.6617	256.1697	311.3367	334.1506