International Journal of Computer Networks and Communications Security

VOL. 3, NO. 6, JUNE 2015, 264–270
Available online at: www.ijcncs.org
E-ISSN 2308-9830 (Online) / ISSN 2410-0595 (Print)

Symmetric Key Encryption Using AES-GCM and External Key Derivation for Smart Phones

Ahmad Mohammed Almorabea[1] and Muhammad Ahtisham Aslam[2]

[1, 2] King Abdulaziz University, Department of Information System, Jeddah, Saudi Arabia

E-mail: [1]ahmad@almorabea.net, [2]maaslam@kau.edu.sa

ABSTRACT

Information is one of the most important resources for organizations as well as for individuals. Keeping the information (e.g. credit card information, images that represent organizational internal processes, organizational strategic documents etc.) secured from unauthorized resources is becoming one of the key challenges of the current age, especially when information is being exchanged very frequently over the Internet. For many years people are concern with privacy and security but with this revelation of information and the social networks privacy became real concern. One of the main reasons is that when we share the information (in the form of images or documents) with our known contacts then actually we are also sharing the information with the telecommunication companies and the company that provide the software. In this paper we present a symmetric key encryption based solution for information privacy and security. Our approach is based on Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) to ensure confidentiality and integrity of the encrypted information. We used several techniques regarding the key derivation function to ensure the strength of the key that will be used in the encryption process. Our encryption process makes use of nonce to randomize outputs at different stages of the encryption process and hence maximizing the security measures. We have also developed an application (i.e. Crypto Ghost) which is available as a standalone application as well as information security utility that works at backend of different information exchange applications and social media applications. We also present the security analysis of the encrypted information and the performance of the key derivation functions in limited resources like smart phones.

Keywords: *Security, Cryptography, Symmetric Key, Encryption, AES GCM.*

1 INTRODUCTION

Cryptography is the art of achieving security by encoding messages to make them non-readable [1] in order to protect our privacy from unauthorized parties. We can use cryptography to maximize the security and privacy of the information that is shared or transmitted over insecure network like the Internet. Our daily life activities (e.g. sharing information over social media applications or through emails, buying the products online by using credit card information etc.) are not gaining the potential at a level at which they should. The Main reason behind using cryptography is authentication, secrecy, non-disclaimer, consistency and honesty at any instant of data transfers [2] over the public network. To boost up business transactions and online business activities we have to increase the level of customer satisfaction and protection of confidential information from unauthorized resources.

Information security experts have provided different solutions and approaches to secure the information from unauthorized access. For example, in [3] authors have provided a solution for information security that is based on Data Encryption Standard (DES). Also, an Advanced Encryption Standard (AES) based approach has been presented Authors have provided an image content manipulation to hide information from unauthorized resources. Many applications and solutions have also been developed for information security e.g. the [4] provides a solution for securing the information by hiding the text message as image. Main limitations of these approaches are that they are based on bit or RGB manipulation of

source document with limited key length. Similarly, another solution, based on information hiding is discussed in [5, 6].Major limitation of these approaches is that they are using the bit manipulation to encrypt the file or they are using AES algorithm with Electronic Code Book (ECB) mode and this is not a secure mode to work with because it will encrypt identical message blocks (i.e., the amount of data encrypted in each invocation of the block-cipher) to identical ciphertext blocks. This is a problem because it will reveal if the same messages blocks are encrypted multiple times and even if the same value are encrypted it will give the same ciphertexts.

We address limitations of existing work by combining some existing technique for the key derivation algorithms and the encryption scheme that can encrypt files. Also, our proposed solution makes sure to use the standards and the most applicable methods for securing the data by using Advanced Encryption Standard (AES) for encryption and decryption in Galois Counter Mode (GCM).

The rest of the paper is organized as follows: In section 2 we describe the related work. Our methodology is discussed in Section 3. Architecture and interface of our application (i.e. Crypto Ghost) is described in Section 4. We show the results of our proposed approach and its.

2 RELATED WORK

In this section we describe the related work both from theoretical and application point of view.

A Data Encryption standard (DES) based approach is discussed in [3]. Authors propose the use of DES to encrypt images. The main problem of this approach is that the encrypted file reveals details about the image and RGB values. It's like using AES in Electronic Code Book (ECB) mode beside DES. It is because that DES is not secure anymore with key length 56 bit and could be brute forced as 2^56.Authors propose the use of bit and RGB manipulation to secure the contents of files. This approach efficiently hides information from unauthorized resources by transforming bits and RGB values of images. This approach is good if the target is to secure the information by hiding it but it becomes less useful when target is to save it from illegal access and content reading especially on the public network where DES is supposed to be not very good solution.

An Advanced Encryption Standard (AES) based approach is described in [4]. Authors propose the use of AES to encrypt the image and then embed some text inside the image after encryption. Authors also propose the use of two keys i.e. one for encryption and decryption and one for data hiding. This is good for encrypting the image with secure algorithm. One major limitation of this approach is that if image is altered after the encryption process, then decryption process can't authenticate it. It means that AES implementation doesn't support integrity of the image and attacker can modify the image and put some external information.

A symmetric key cryptography algorithm for encryption and decryption of files is discussed in [5]. The encryption and decryption method makes use of random key square matrix 65536 instead of 256 elements which is comparatively easy to break by a brute force method. One major limitation of this approach is that it becomes less effective as long as the size of the file to be encrypted increases, as well as with randomization number and number of encryption to be done.

An application based solution is provided in [6]. This application secures the information by hiding text messages in images. Hiding text inside images secures information to some extant but from cryptographic point of view, text that is inside the image is not encrypted. It still can be treated as a plaintext format resulting in extracting information from image.

Similarly in [7] authors propose a solution of information security by locking the images or files. This could be a good solution as long as information needs to be secured on a particular device but is not useful when information needs to be shared and exchanged over the public network like the Internet.

Table 1 provides a comparison of existing work based on different parameters and security measures. We also highlighted in this table that up to what extent we address limitations of existing work by providing support for larger key size (i.e. 256 bit), integrity and larger key space.

Table 1: Comparison of existing approaches and our proposed work based on certain security measures/factors

	Encryption type	Key Size	Provide Integrity	Large Key Space
Image Crypto	No Encryption	No Key	No	No
Hide Image	AES	128	No	Yes
Image Crypt	DES	65	No	No
Secure Gallery	AES	192	No	Yes
Crypto Ghost	AES	256	Yes	Yes

3 METHODOLOGY

In this section we describe our methodology to secure and protect information. The basic idea behind our methodology consists of the utilization of email address and password of the user, which we consider as master password. This master password is used to derive the key which is used in encryption and decryption process. In the subsequent sub-sections we describe detail process of key derivation, file encryption & decryption and then we describe that how the *Advanced Encryption Standard* (*AES)* is applied in the encryption and decryption process together with nonce to achieve the more secure encrypted files.

3.1 Key Derivation

The key derivation is an important part of our methodology. The main part of key derivation is the user password. If user password is random and has a large key space, most of attacks will be harder to accomplish, like brute force attack. Our key derivation process makes use 256 bits key that is derived based on email address and password that is entered by the user. We applied a restriction on user password of being at least 10 characters long. It is because we want to make the user give a password that is not easy to guess and in the same time computer can't get the permutation of different characters by brute force. This input email address and password is injected inside *BLAKE2* [8] (it's a one way hash function and this will produce 512 bits output).*BLAKE2*operates on 64-bit words and returns a 64-byte hash value and this output is injected into *Scrypt*[9].This is a password-based key derivation function and is dedicated to derive passwords, typically use repeated invocations of a cryptographic hash to increase the time required to perform brute force attacks on stored password digests. The *Scrypt* is invoked using following parameters:

$$N = 2^{14}, r = 8, \ p = 1, L = 32$$

Where ***N*** is General work factor, iteration count , ***r*** is block size in use for underlying hash; fine-tunes the relative memory-cost, ***p*** is parallelization factor; fine-tunes the relative CPU-cost and ***L*** is the length of output produced by *HMAC_SHA256()*.

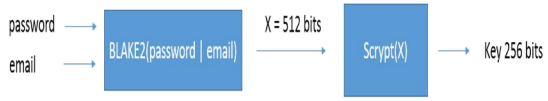


Fig. 1. Key Derivation process

The figure 1 shows the key derivation process which start with email and password as input and results in 256 bits private key which is used in the encryption process (as discussed in the next section).

3.2 Encryption Process

Our encryption process makes use of *Advanced Encryption Standard* (*AES)* algorithm in *GCM* [10, 11, and 12] *Mode* with 256 bits key length. In addition to *AES*, our encryption process makes use of nonce [13] with 16 bytes length. The input file is encrypted together with 256 bits key generated in the first step (as discussed in the Section 3.1) and *AES* (as show in the Figure 2). The joint venture of the *AES* and nonce ensures the confidentiality and authenticity (integrity) of the encrypted file. An interesting feature of our encryption process is unique value of nonce for each encryption process i.e. nonce value will be different for every encryption process and it will be unique for every input file/image, and this nonce will be on the first 16 bytes in the input file.

$$c \leftarrow E(K, M, N)$$

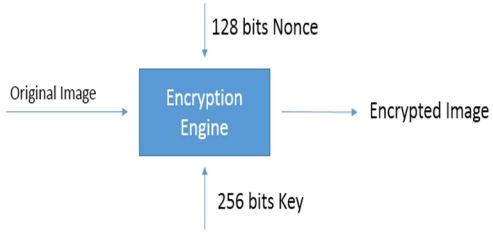


Fig. 2. Encryption process

Our encryption process maximizes the security measures by producing different result for the same input image that is if we encrypt the same image (original image) twice it will not give the same result i.e.

$$c \leftarrow D(K, M, N) \neq c1 \leftarrow D(K, M, N)$$

This feature improves the information integrity and security from unauthorized resources e.g. attackers. It is because that if attackers intercept the same image with two encryption files, they will not be able to understand that if they are same image and this is because the nonce that we using is random and it will not repeat value twice. Figure 3

(a) and (b) show sample of the two different *HEX* values generated for the same input file.

	0	1	2	3	4	5	6	7
00000000	E1	14	F8	D6	04	2B	74	0B
0000001A	4B	37	44	0B	4A	BD	F8	2C
00000034	EF	0B	DA	74	17	FD	DB	9D
0000004E	BA	B9	CC	D9	AA	D7	48	BC
00000068	96	90	7B	02	C6	5E	39	B9
00000082	A2	40	31	1B	AE	1B	F3	FE
0000009C	00	02	63	36	FB	2C	80	CF
000000B6	00	A5	A3	50	3D	F8	04	EB
000000D0	AB	ED	5A	62	22	3C	1C	97
000000EA	4F	7F	2D	C2	89	02	C5	FF

Fig. 3. (a)

	0	1	2	3	4	5	6	7
00000000	79	0A	1E	96	C5	1A	84	65
0000001A	9E	02	FA	0A	61	22	C1	85
00000034	6E	5F	79	75	C6	BD	25	8B
0000004E	00	8D	A5	B2	AA	BB	9E	13
00000068	11	66	85	07	96	06	20	04
00000082	28	BA	FF	95	C8	52	CF	6C
0000009C	D8	8C	56	DF	87	96	90	B1
000000B6	37	13	40	6E	42	BD	80	FF
000000D0	D1	04	F5	54	47	14	A1	F0
000000EA	DB	8D	43	41	D4	46	23	42

Fig. 3. (b)

Fig. 3. Sample of different HEX values generated from the same input file.

The encryption process encrypts the input file and makes it ready to be shared with other party. At the end of this process an input file (e.g. image file) becomes unreadable and could not be recognized by related applications as a valid file, until and unless it is decrypted and original information is recovered back. In the next section we describe our decryption process to recover the actual information back.

3.3 Decryption Process

Here, we describe the decryption process in order to decrypt the encrypted image by the receiver. In the decryption process, first of all we extract the nonce from the encrypted file. Once, nonce is extracted from the encrypted file, the key is injected in to the file in order for the decryption function to decrypt the input file. If the input key is correct, file will decrypted successfully (as show in the Figure 4).If there is any temper in the data (even a single byte) the decryption function will detect that and it will not decrypt the input file. Also, if the file has some new bytes that attached to it after the encryption process has been done the decryption function will detect that also and the file will not be decrypted.

$$m := D(K, C, N)$$

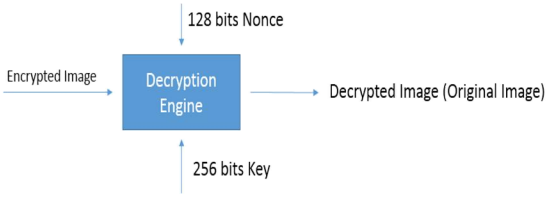


Fig. 4. Decryption process

4 ARCHITECTURE AND USER INTERFACE

In this section we describe the architecture and user interface of the application that we developed by using above discussed methodology.

4.1 Architecture

We have developed an application (i.e. Crypto Ghost[1]) based on the methodology described in the previous section. The architecture of the Crypto Ghost is especially designed for devices with limited resources (e.g. smart phone devices). The Crypto Ghost architecture consists of three main modules (i.e. key derivation, AES-GCM encryption and AES-GCM decryption) (as shown in Figure 5).

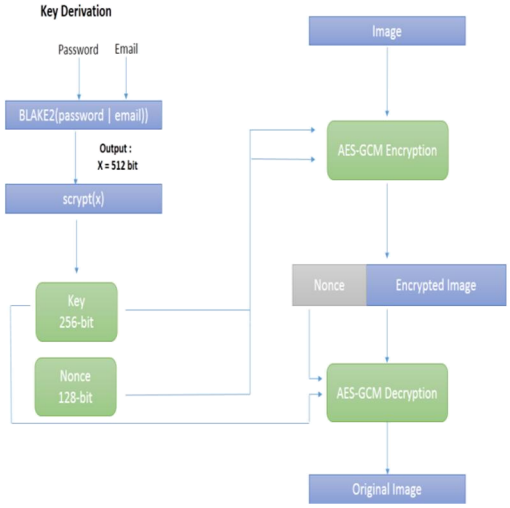


Fig. 5. Architecture of the Crypto Ghost.

[1] http://cryptoghost.com

In the first step user enters his password and email address in order for the system to derive the private key. This is injected in *BLAKE2* hash function. Let's assume that password is "CryptoGhost" and the email address is "crypto@ghost.com" so the hash value that is generated by the application is:

9A13F3B7E810DF3FF5551D3CD3D353113F43E
D02B6BE261E701C546D8E1EACEAD5B9F9E5
A34DBBC75964916FC8D7BF70CF988B22744A6
45C31C0EC7762C161F9

When this hash value is injected inside *Scrypt* function we get the following private key:

$s0$e0801$nSNmWIgoEUYLXkJO9zCyyA==$RI
8wGQBD4/txgJutXWdV3YsBc54HziqQAmWImrr
zUAk=

Note here, that if we use the same information (i.e. same email and password as above) to generate the private key, then every time application will generate different key (as discussed in Section 3.1).Now we have the private key (let's denoted by ***K***), this key is injected inside the encryption module to encrypt the data and this encryption module takes key ***k*** , message ***w***here $\boldsymbol{m} \in \boldsymbol{M}$ and nonce ***n*** as input and generates *ciphertext* **c** and it will be random every time even if the same information encrypted twice.

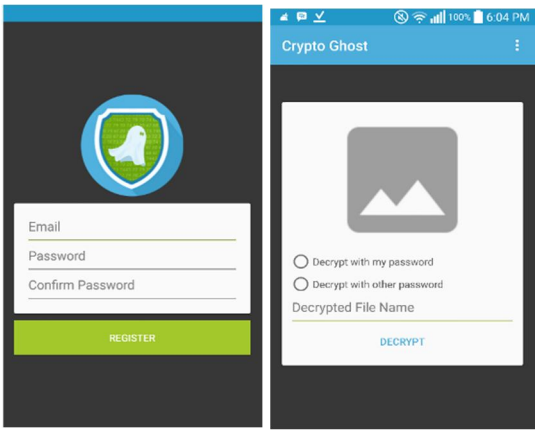
$$c \leftarrow E(k, m, n)$$

The architecture of our application shows that the decryption module takes 3 input parameters i.e. key ***k***, *ciphertext* ***c*** and nonce ***n*** in order to decrypt the input file and recover the original message ***m*.**

$$m := D(k, c, n)$$

4.2 User Interface

User interface of our application consists of very easy to use steps so that it could be used even for normal users who have no cryptographic background. The Figure 6(a) shows the registration screen where user can register the device by entering his email and password and derive the private key for the application. The Figure6(b) shows the decryption screen where user can select the encrypted image/file and then choose the suitable option and then press "Decrypt" to recover the original image back.



(a) (b)

Fig. 6. Screen shots of Crypto Ghost application.

5 RESULTS

In this section we describe results of our approach and we also do the performance analysis of our work especially in the limited resources environment.

Let suppose Figure 7 (a) is the input file (i.e. image file) to be encrypted. If we see the input image file in the *Gray Scale* and *Segmented Graph* (i.e. Figure 7), we can notice the blank at the top of the *Gray Scale, Segmented Graph*. This is just a redundant values it's not effecting the image itself, it's just a random values that we add to show that in the encryption process it will be gone and the attacker can't know anything from the encrypted data. So the attacker can't recognize if there is any redundant data nor he will see random bytes with no redundancy.

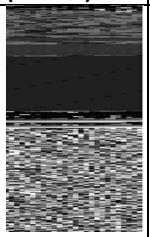
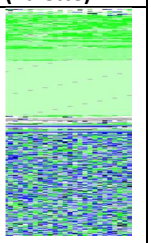
Original Image	Gray Scale (Palette)	Segmented (Palette)
		

*Fig. 7. A sample original image and its Gray Scale and Segmented Graph (File Size: 300 KB - Dimensions: 1222 * 917)*

5.1 Encrypted Image

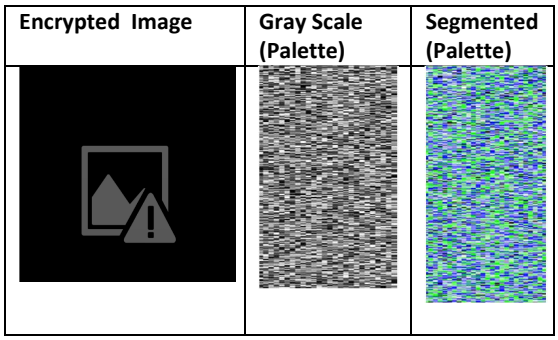


*Fig. 8. Encrypted image and its Gray Scale and Segmented Graph (File Size: 143 KB - Dimensions: 1222 * 917)*

The Figure 8 shows the encrypted image which being encrypted cannot be recognized by any Image Viewer application. Even the *Gray Scale* is not providing information about any redundant pixels and if an attacker sees the image he will not get any idea about the original image. Another feature is that real size of this image was 300KB but we added a function to reduce the size of the image so it will be encrypted smoothly with limited resources devices.

Figure 9 show overall results of our application. The Figure 7 shows the actual image with size 300kb, the Figure 8 show the encrypted image that cannot be recognized by any image viewer application, and then Figure 9 shows the final decrypted image. The image is encrypted as well as it size is reduced to limited extent so that to make the image secure as well as light weight for sharing over network with high traffic. This feature also saves the image encryption, decryption and sharing time when used in limited resources environment with minimum impact on the actual image quality.

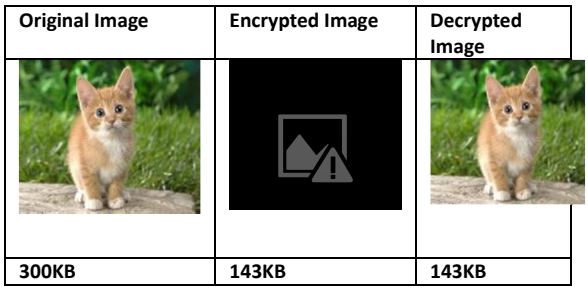


Fig. 9. Overall results of encryption process.

5.2 Key Derivation – Performance

Working with android platform is always critical because we are dealing with limited resources not like a normal computer. We always kept this point in mind so that we can efficiently and effectively address the performance issues. The Table 2 show performance results of our approach and its implementation: in the *CPU Cost*. It is general work factor for *Scrypt* function. *CPU – RAM Size* are the specification of the device that has been tested on the *Key Size* of the key length that used in the algorithm. *Heap Size-Byte* size is how much it takes to from the heap and *Time* is how much time this operation take on this device. It can be observer from the performance analysis data (as shown in the Table 2) that our proposed approach works very fine and in a time efficient way even for heavy files and devices with limited resources

Table 2: Performance analysis of our approach in limited resource environment.

CPU Cost	CPU	RAM Size	Key Size	Heap Size	Byte Size	Time
2^10	1.4GHz	1 GB	256 bits	10.148 MB	1048592 bytes	0.884 Sec
2^12	1.4GHz	1 GB	256 bits	13.147 MB	4194320 bytes	2.211 Sec
2^14	1.4GHz	1 GB	256 bits	25.151 MB	16777232 bytes	8.365 Sec
2^10	2.5GHz	3 GB	256 bits	10.148 MB	1048592 bytes	0.247 Sec
2^12	2.5GHz	3 GB	256 bits	13.147 MB	4194320 bytes	0.818 Sec
2^14	2.5GHz	3 GB	256 bits	25.151 MB	16777232 bytes	3.684 Sec

6 CONCLUSION

Cryptography is used to accomplish confidentiality and integrity of the information. Many approaches have been proposed so far that work on servers or PCs. In this paper we presented our approach (as a combination of multiple security measures) that works very well for desktop users as well as for users of devices with limited resources (e.g. Smart phones users). Files encrypted by using our approach ensure the confidentiality and integrity and this is the two of the pillars of the security triangle. Our approach uses standardized algorithms that have been fully documented and reviewed by the security community, our approach uses private key settings and we will try to make this application work on Public key infrastructure for the next version as future work.

7 REFERENCES

[1] Ayushi , "A Symmetric Key Cryptographic Algorithm", International Journal of Computer Applications(0975 - 8887), Volume 1 – No. 15, 2010,

[2] Kaladharan N," Unique Key Using Encryption and Decryption of Image", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 3, Issue 10 , October 2014

[3] J. Singh, K. Lata and J. Ashraf," Image Encryption & Decryption with Symmetric Key Cryptography using MATLAB", International Journal of Current Engineering and Technology, Vol. 5, No. 1, pages, 448-451, ISSN 2347 – 5161, Feb, 2015.

[4] T. Arumuga Maria Devi, Sabitha.S, . "Symmetric Key Cryptography on Images in AES Algorithm and Hiding Data Losslessly", International Journal of Modern Engineering Research (IJMER), Vol.2, 1951-1954, ISSN: 2249-6645, July-Aug. 2012

[5] D. Chatterjee, J. Nath, Joyshree, S. Dasgupta and A. Nath, "A New Symmetric Key Cryptography Algorithm Using Extended MSA Method: DJSA Symmetric Key Algorithm", Proceedings of the 2011 International Conference on Communication Systems and Network Technologies, ISBN: 978-0-7695-4437-3, pages: 89-94, 2011.

[6] Image Crypto: An application to hide text inside images [available online on Google play] https://play.google.com/store/apps/details?id=pl.kchdev. Image.crypto , October, 2014

[7] Hide File: An application to lock images and make it invisible [available online on Googleplay] https://play.google.com/store/apps/details?id=com.tonado.boli.chenghaicys.wenjianyincang, August, 2012

[8] Jean-Philippe Aumasson1 , Samuel Neves2 , Zooko Wilcox-O'Hearn3 , and Christian Winnerlein4, "[BLAKE2: simpler, smaller, fast as MD5]" , 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings, ISBN 978-3-642-38980-1, Pages 119-135, Jan 2013

[9] COLIN PERCIVAL ,"STRONGER KEY DERIVATION VIA SEQUENTIAL MEMORY-HARD FUNCTIONS", the technical BSD conference (BSDCan '09), Ottawa, Canada, May 2009

[10] Christoforus Juan Benvenuto ,"Galois Field in Cryptography", University of Washington, May 2012.

[11] David A. McGrew and John Viega2."[AES-GCM]", National Institute of Standards and Technology, Jan 2014.

[12] Morris Dworkin, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", National Institute of Standards and Technology, November 2007

[13] Phillip Rogaway, [Nonce-Based Symmetric Encryption], 11th International Workshop, FSE 2004, Delhi, India,. Revised Papers,ISBN 978-3-540-25937-4 ,Pages 348-358,Publisher Springer Berlin Heidelberg, February 5-7, 2004.

AUTHOR PROFILES:

Ahmad Mohammed Almorabea is student and researcher in Information Security and Cryptography research group at Department of Information System, King Abdulaziz University, Jeddah, Saudi Arabia. Ahmad has interest in information security, especially in cryptography and its algorithms. Ahmad also has passed a cryptography course from Stanford University. He also has been developing applications for android operating system and already has 5 applications in the market.

Muhammad Ahtisham Aslam is currently working as Assistant Professor at Department of Information System, King Abdulaziz University, Jeddah, Saudi Arabia. He has been working as Senior Staff Researcher at Artificial Intelligence Lab, Knowledge Technology Cluster, Malaysian Institute of Microelectronics Systems (MIMOS), Kuala Lumpur, Malaysia. He also has been working as Assistant Professor at COMSATS Institute of Information Technology, Pakistan. He did his PhD from University of Leipzig, Germany. He has several conference and journal publications in the area of semantic web, web services, knowledge extraction and knowledge engineering. His research interests are semantic web, semantic web services, web 2.0, social semantic network and knowledge engineering.