# A Study of Incremental Cryptography for Security Schemes in Mobile Cloud Computing Environments

Abdul Nasir Khan [1,*], M.L. Mat Kiah [1], Sajjad A. Madani [2], Atta ur Rehman Khan [1], Samee U. Khan[3]

Faculty of Computer Science and IT[1]
University of Malaya
Kuala Lumpur, Malaysia
*anasir@siswa.um.edu.my

Department of Computer Science[2]
COMSATS Institute of Information Technology
Abbottabad, Pakistan

Department of Electrical and Computer Engineering[3]
North Dakota State University
USA

***Abstract*— While using the cloud storage services on resource constraint mobile device, the mobile user needs to ensure the confidentiality of the critical data before uploading on the cloud storage. The resource limitation of mobile devices restricts mobile users for executing complex security operations using computational power of mobile devices. To make security schemes suitable for mobile devices, large volume of existing security schemes execute complex security operations remotely on cloud or trusted third party. Alternatively, few of the existing security schemes focus on the reduction of the computational complexity of the cryptographic algorithms. Keeping in view the resource limitation of mobile devices, this paper, introduces an incremental cryptographic version of the existing security schemes, such as encryption-based scheme, coding-based scheme, and sharing-based scheme, for improving the block(s) modification operations in term of resource utilization on mobile device. The experimental results show significant improvement in resource utilization on mobile device while performing block insertion, deletion, and modification operations as compared to the original version of the aforementioned schemes.**

***Keywords—cloud computing; mobile cloud computing; security; privacy***

## I. INTRODUCTION

An emerging computing paradigm offers vast range of IT services to cloud subscribers [1, 2]. The cloud subscribers must pay-per-use basis of the cloud resources [3]. The offered services provide the support of elasticity [4] for meeting the fluctuating computational requirements of the cloud subscriber. The cloud subscriber can upgrade and downgrade the cloud services on fly according to the desirable computational requirements at any given time. Likewise, the accessibility of cloud elastic services on resource constraint mobile devices laid foundation for a new computing paradigm, called mobile cloud computing [5, 6]. The main objectives of the mobile cloud computing are to increase the processing/storage capabilities of the mobile device and reduce the energy consumption while executing the computationally intensive jobs. To achieve the aforementioned objectives, mobile users offload processing intensive and storage demanding portion of mobile application from resource constraint mobile device to resource enriched cloud. The offloading of the processing intensive and storage demanding portion(s) of mobile application enhances the capabilities of mobile devices in term of processing, storage, and battery [7, 8]. The wireless technologies, such as Wi-Fi, Wi-Max, 3G, 4G, or Satellite Internet connectivity, can be used for communication between mobile users and cloud services provider.

In spite a large range of services offered by cloud, most of the business organizations are not interested to adopt such services due to risks associated with security and privacy. The execution of the cloud subscribers' jobs in a trusted mode and assurance of the confidentiality, integrity, authenticity [9, 10] of uploaded data on cloud may increase the number of potential cloud's subscribers. The research organizations and academia are continuously identifying and resolving the security and privacy issues in cloud environment. However, there are still a few grey areas that must be addressed, such as perfect isolation of virtual machines on cloud server [11]. Due to the inherited nature of mobile cloud computing, the security threats of cloud computing are also shifted in mobile cloud computing with the additional limitations of resource constraint mobile devices. Keeping in view the resource limitation of mobile devices, there is a need of light-weight security schemes. There are two approaches adopted in literature for keeping the security operations light-weight [1]. In the first approach, the processing intensive security operations are offloaded on the cloud in a trusted mode. The second approach reduces the computational complexity of cryptographic operations and executes entire security operations on mobile device.

By considering the resource limitation of mobile devices, this paper introduces the incremental version of encryption-based scheme, coding-based scheme, and sharing-based scheme that provides confidentiality and integrity services to mobile users for the file(s) stored on the cloud storage. If the data owner wants to update the encrypted file uploaded on the cloud storage, after modification the existing schemes encrypt and upload entire file on cloud storage without considering the updated portion of file. If there is a minor change in the uploaded file, even then mobile user must repeat the entire encryption and uploading process that needs to be improved. The incremental version of the aforementioned schemes improves the file modification operations using the block

insertion, deletion, and modification operations. The proposed schemes require comparatively more processing while performing initial encryption and uploading due to the involvement of extra file management and cryptographic operations. However, in the long run, this process will improve the file modification operations in terms of turnaround time, communication overhead, and energy consumption on the mobile devices.

The rest of the paper is organized as follows. Section II presents the existing security schemes proposed for MCC. In section III, we define the incremental version of security schemes. The results of the comprehensive empirical analysis with some critical remarks are presented in section IV. We conclude our work in section V.

## II. EXISTING SECCURITY SCHMES FOR MOBILE CLOUD COMPUTING

To increase the processing capability of mobile devices, most of the existing literature focuses on the offloading of computationally intensive jobs on the cloud. However, offloading and execution of jobs on cloud should be in a trusted mode. The security issues that arise due to offloading of portion(s) of an application on cloud are identified and addressed in [12], [13], [14], [15], [16], [17], [18], and [19]. To increase the storage capacity, mobile users may utilize cloud storage services. The loss of the physical control on the uploaded files laid foundation for new security issues that are identified and covered in [20], [21], [22], [23], [24], [25], [26], and [27]. Due to space limitation, we cannot include the details description of aforementioned security schemes. However, interested reader may refer to [1] for detail description, shortcoming, and strength of the aforementioned schemes.

This paper only covers the incremental version of **(a)** encryption based scheme, **(b)** coding based scheme, and **(c)** sharing based scheme presented in [21]. We have selected these schemes due to the following reasons:

- Focus of the abovementioned schemes is on the reduction of the computational complexity of the cryptographic algorithms for providing confidentiality and integrity services.
- Entire security operations are executed on the mobile device which helps to identify the performance improvement of incremental version of the aforementioned schemes on mobile device

The notations used in this paper are presented in Table I.

**Table I:** Notations and abbreviations

| No | Notation | Abbreviation |
|---|---|---|
| 1) | FN | File Name |
| 2) | FS | File Size |
| 3) | PwD | Password |
| 4) | EK | Encryption Key |
| 5) | IK | Integrity Key |
| 6) | HMAC | Hash-based Message Authentication Code |
| 7) | MAC | Message Authentication Code |
| 8) | \|\| | String Concatenation |
| 9) | EF | Encrypted File |
| 10) | $\oplus$ | Exclusive-OR |
| 12) | F | Original File |
| 13) | DF | Decrypted File |
| 14) | A | Coding Vector |
| 15) | SC | Secrecy Code |
| 16) | AR | Accumulative Results |
| 17) | RS | Random Shares |

### A. Encryption-based Scheme (EnS)

EnS uses standard cryptography functions for providing confidentiality and integrity services to mobile users in cloud environment. When a mobile user wants to upload file(s) on cloud server(s), mobile user provides a password which is transformed into encryption and integrity keys using the following procedure:

$$EK = H(PwD \,||\, FN \,||\, FS) \quad (1)$$

$$IK = H(FN \,||\, PwD \,||\, FS) \quad (2)$$

where $\boldsymbol{H}$ refers to the standard hash function and || denotes the concatenation operation. To achieve confidentiality, files are encrypted using standard cryptography function and encryption key as:

$$EF = Encrypt_{EK}(F) \quad (3)$$

To achieve integrity, message authentication code is generated using the cryptographic hash function and integrity key as shown below:

$$MAC = HMAC_{IK}(F) \quad (4)$$

Thereafter, the mobile user uploads the encrypted file together with the H(FN) and message authentication code on the cloud storage. To keep the file more secure, mobile user only saves file name in local file table and deletes the encryption key, integrity key, and original file from the mobile local storage.

For downloading a file, a mobile user sends downloading request along with the H(FN) to the cloud service provider. The cloud service provider searches for the file on the basis of received H(FN) and delivers the corresponding encrypted file along with the message authentication code to the requesting entity. To decrypt a file, mobile user has to provide a password which is transformed into encryption and integrity keys using the similar procedure discussed in (1) and (2). The generated keys are used to decrypt and verify the integrity of the uploaded file as shown in (5) and (6).

$$DF = Decrypt_{EK}(EF) \quad (5)$$

$$MAC = HMAC_{IK}(DF) \quad (6)$$

The downloaded message authentication code is compared with the newly calculated message authentication code to confirm the integrity of the downloaded file. Similarity of message authentication codes confirms the integrity of the uploaded file.

### B. Coding-based Scheme (CoS)

CoS uses matrix multiplication and cryptographic hash function for providing confidentiality and integrity services to mobile users in cloud environment. For uploading, the mobile user divides the file into ‘d’ parts, each part is represented in the form of matrix having ‘t’ rows of size ‘n’ bits. The mobile user must provide a password to generate the coding vector as shown in (7). The coding vector is generated by performing recursive hash function on concatenation of password, file name, and file size.

$$\alpha_i = H^i(PwD \,||\, FN \,||\, FS)\ where\ 1 \le i \le t,$$
$$H^1(x) = H(x),\ H^i(x) = H\left(H^{i-1}(x)\right), 2 \le i \le t \quad (7)$$

The coding vector is generated on the basis of password that is only known to the owner of the file; therefore the coding vector is used as an encryption key and multiplied with the matrix form of each part of file to achieve the confidentiality as expressed in the following equation.

$$SC[j] = \left(\sum_{i=1}^{t} \alpha_i * F[i][j]\right), 1 \le j \le d\ and\ 1 \le i \le t \quad (8)$$

'F[i][j]' is the representation of j[th] part of file. To achieve integrity, the mobile user produces the integrity key by applying standard hash functions on the concatenation of the coding vector elements. The cryptographic hash function along with the integrity key is used to generate the message authentication code as expressed in the following equations.

$$IK = H(\alpha_1 || \alpha_2 || \alpha_3 \ldots || \alpha_t) \quad (9)$$
$$MAC = HMAC(F, IK) \quad (10)$$

Thereafter, the generated secrecy codes along with the corresponding H(FN+j) and message authentication code is uploaded on the cloud storage. The secrecy codes can be uploaded on a single or multiple cloud servers. For security reasons, the mobile user only saves the file name in the local file table and deletes the coding vector, integrity key, and original file from the local storage of mobile.

While downloading, the mobile user sends the downloading requests for each of the secrecy code along with H(FN+j). The cloud service provider searches for the secrecy code on the basis of received H(FN+j) and sends the corresponding secrecy code along with message authentication code to the requesting entity. To decrypt a file, the mobile user provides a password that is transformed into the coding vector and the integrity key using the same procedure discussed in (7) and (9). The downloaded secrecy codes are multiplied with inverse of a coding vector for producing the original file as expressed in the following equation.

$$F[i][j] = (\alpha_i^{-1} * SC[j]), 1 \le i \le t\ and\ 1 \le j \le d \quad (11)$$

To verify the integrity of the uploaded file, the message authentication code is generated on the decrypted file and the newly generated integrity key. Similarity of the downloaded and recalculated message authentication codes confirms the integrity of the downloaded file.

### C. Sharing-based Scheme (ShS)

ShS uses a similar procedure for integrity verification as discussed in EnS. Confidentiality is achieved using simple exclusive-OR operations, we will only limit our discussion on confidentiality services for ShS in this section.

For uploading a file, the mobile user generates (d – 1) random shares/files, each having size equal to the original file. Confidentiality is achieved in two phases. In the first phase, all (d - 1) random shares are exclusive-OR with each other to produce the accumulative result. In the second phase, the accumulative result is further exclusive-OR with the original file to produce d[th] share as shown in the following equation.

$$AR = (\oplus_{i=1}^{d-1} RS[i]) \quad (12)$$
$$RS[d] = AR \oplus F \quad (13)$$

All 'd' shares along with the H(FN+j) and message authentication code are uploaded on the cloud storage. The shares can be stored on a single or multiple cloud server(s).

While downloading, the mobile user sends the downloading requests for each share along with H(FN+j). The cloud service provider searches for the share on the basis of received H(FN+j) and sends the corresponding share along with the message authentication code to the requesting entity. To decode a file, the mobile user performs exclusive-OR operations on received 'd' shares as expressed in the following equation.

$$F = (\oplus_{i=1}^{d} RS[i]) \quad (14)$$

The aforementioned security schemes provide confidentiality and integrity of the mobile user's files/data stored on distrusted cloud servers. However, when mobile user wants to change the contents of the uploaded file in each scheme, the mobile user encrypts the whole file and uploads on the cloud server after modification. If there is a small modification in the downloaded file, even than mobile user has to encrypt and upload the entire file on the cloud server that consumes more resources on the mobile device. Keeping in view the resources limitation of mobile devices, this paper introduces incremental version of the aforementioned schemes to improve the resource utilization of mobile device while performing the file modification operations. Moreover, the impact of the incremental version on turnaround time and energy consumption during initial encryption process is also analyzed.

## III. Incremental Version of Security Schemes for Mobile Cloud Computing Environments

For improving the file modification operations in term of turnaround time and energy consumption, the original file is divided into equal-size 'd' blocks of the 'n' bits, so the following condition must be satisfied:

$$FS\ \%\ d = 0 \quad (15)$$

In practice, to guarantee this symmetric file defragmentation, usually some extra bits must be padded at the end of the file. To achieve confidentiality, the mobile user provides a password that is transformed into encryption and integrity keys using the same process discussed in (1) and (2). Each block of the file is encoded separately and the final encrypted file is generated by performing the concatenation operation as expressed in the following equations.

$$C_j = Encrypt_{EK}(F_j), Where\ 1 \le j \le d \quad (16)$$
$$EF = C_1 || C_2 || C_3 || C_4 || \ldots\ldots || C_d \quad (17)$$

The message authentication code is generated separately for each block of the file using cryptographic hash function. The final message authentication code is generated by performing cryptographic hash function on the concatenation of the individual file blocks' message authentication code as expressed in the following equations.

$$MAC_{F_j} = HMAC_{IK}(F_j), Where\ 1 \le j \le d \quad (18)$$
$$MAC = HMAC_{IK}\ (MAC_{F_1} || MAC_{F_2} || \ldots\ldots || MAC_{F_d}) \quad (19)$$

Thereafter, the mobile user uploads the encrypted file together with H(FN), blocks' message authentication code,

and final message authentication code. For security reasons, mobile user only stores file name and value of 'd' in the local file table and deletes security keys, codes, and original file. The value of 'd' is used to perform block insertion, deletion, and modification operations later on. The overall system model is shown in Fig. 1.

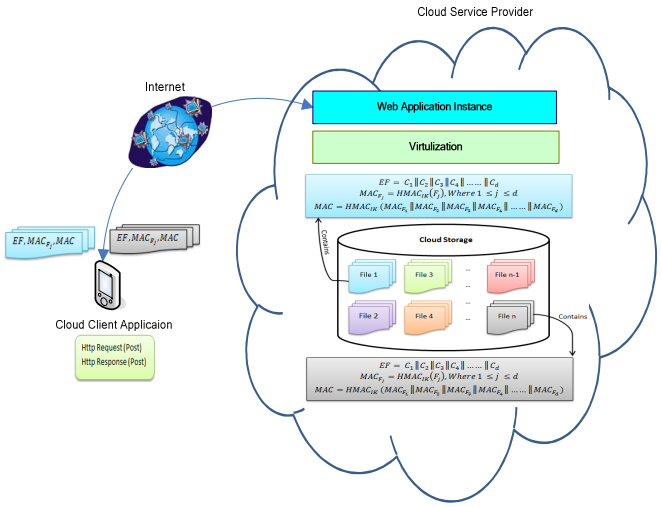


**Figure 1:** Overall system model for the proposed scheme

For downloading a file, the mobile user sends the downloading request along with the H(FN) to the cloud service provider. The cloud service provider delivers the encrypted file along with the corresponding message authentication code. To decrypt the downloaded file, the mobile user provides a password that is transformed into encryption key and integrity key. The downloaded file is divided into 'd' blocks as discussed above. The blocks of the file are decrypted and concatenated appropriately with each other to produce the original file as expressed in the following equations.

$$F_j = Decrypt_{EK}(C_j), (1 \leq j \leq d) \quad (20)$$

$$F = F_1||F_2||F_3||F_4|| \dots \dots ||F_d \quad (21)$$

To verify the integrity of the download file, mobile user generates the message authentication code using the procedure discussed in (22) and (23).

$$MAC_{F_j} = HMAC_{IK}(F_j), (1 \leq j \leq d) \quad (22)$$

$$MAC = HMAC_{IK}\,(MAC_{F_1}||MAC_{F_2}|| \dots \dots ||MAC_{F_d}) \quad (23)$$

Same value of downloaded message authentication code and recalculated message authentication code confirms the integrity of the downloaded file.

The incremental version of each scheme performs integrity verification using similar process. However, confidentiality is achieved through different processes for each scheme. For the incremental version of EnS, the standard cryptography functions are used to encrypt and decrypt the file's chunks. For the experimental setup, DES (encrypts/decrypts 64 bits data block using 56 bits key) and SHA-1 are used to provide confidentiality and integrity services to mobile users. In the incremental version of CoS, coding vector matrix is generated using the process discussed in (7). The generated coding vector matrix is multiplied with the file's chunks to get the secrecy code. The incremental version of ShS uses exclusive-OR based secret sharing for encoding and decoding the file's chunks.

The incremental version of EnS, CoS, and ShS consume more resources for encryption and decryption on the mobile device due to generation of the extra message authentication codes and file management overhead. However, the proposed encryption process makes the block insertion, deletion, and modification operations efficient in term of turnaround time and energy consumption on mobile device. In this paper, we assume that the mobile user modified only whole block in a file. When the mobile user wants to modify a block in the file, the mobile user downloads the corresponding file and decrypts using the aforementioned procedure. Thereafter, the mobile user modifies block(s), only encrypts the updated block, and generates the corresponding message authentication code(s). The modified encrypted block information is delivered to the cloud service provider for file updating on the basis of the owner request. The file modification request contains the fields shown in Fig. 2.

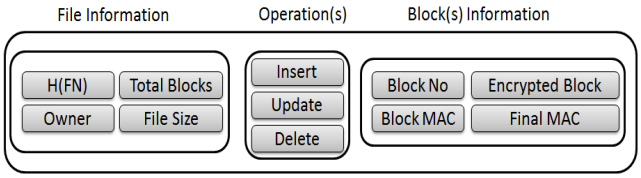


**Figure 2:** Block modification request

If file modification request contains an insert operation, the block number field contains information about the location where mobile user wants to insert the block. The cloud service provider inserts the block data to the respective position along with the message authentication code. Due to change in the file contents, the cloud service provider also updates the final message authentication code on the basis of received information. Moreover, the mobile user increments the value of 'd' in the local file table for the corresponding file record. If the file modification request contains block deletion operation, the block number field contains information about the location where mobile user wants to delete the block. The cloud service provider deletes the block and updates the final message authentication code. In case of delete operation, the block data and block MAC fields contain a null value. Additionally, the mobile user decreases the value of 'd' by one in the local file table for the corresponding file record. If the received operation on cloud side is an update operation, the cloud service provider updates the corresponding block data, block's message authentication code, and final message authentication code with new received values. Furthermore, value of the 'd' remains unchanged.

## IV. Results and Discussion

The aforementioned security schemes are implemented in the BlackBerry JDE 7.0.0. The implemented cloud client applications are deployed on the BlackBerry 9900 smartphone. The cloud client application sends the http requests to communicate with the hosted web instant on the

cloud. The hosted web instant receives the http requests and provides the storage and processing services to the requesting entity. The dataset that is used for experimental setup to evaluate the turnaround time and energy consumption is given in Table II.

**Table II:** The experimental dataset for performance Evaluation

| No. | File Size | Total Files |
|---|---|---|
| 1) | 1048576 Bytes | 50 |
| 2) | 1572864 Bytes | 50 |
| 3) | 2097152 Bytes | 50 |
| 4) | 2621440 Bytes | 50 |

The performance of the security schemes are evaluated on the basis of turnaround time and energy consumption. To evaluate the energy consumption and turnaround time on the blackberry smartphone, DeviceInfo API and System class method currentTimeMillis() are used. Each experiment is performed ten times and average results are presented in the graphs.

### *A. Turnaround Time and Energy Consumption during Uploading and Downloading*

In the case of uploading, the turnaround time includes file reading time, encryption time, and uploading time. In the case of downloading, the turnaround time is a time required to send downloading request, time required to download file, and decryption time. Fig. 3 and Fig. 4 report that the incremental version of the existing schemes requires more time to complete uploading and downloading requests due to the involvement of extra message authentication codes, file partitioning, and reassembling overhead. Fig. 5 and Fig. 6 show energy consumption during uploading and downloading, respectively.

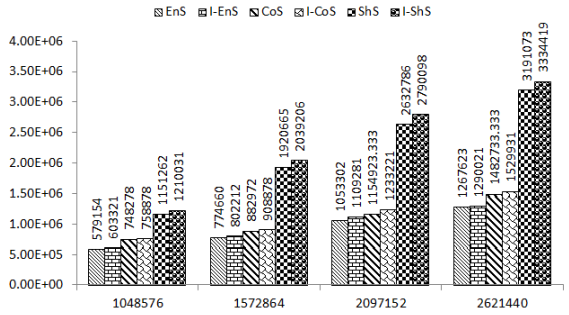


**Figure 3:** Turnaround time during uploading

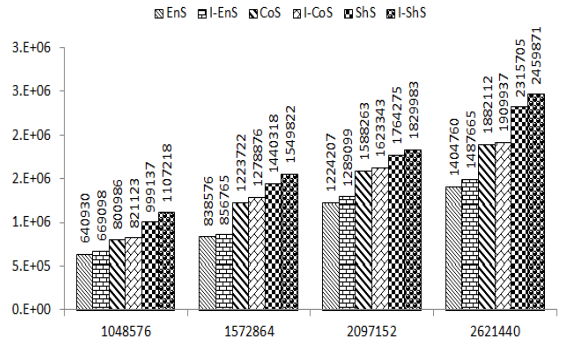


**Figure 4:** Turnaround time during downloading

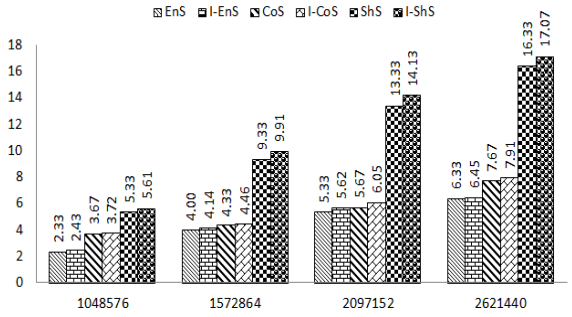


**Figure 5:** Energy consumption during uploading

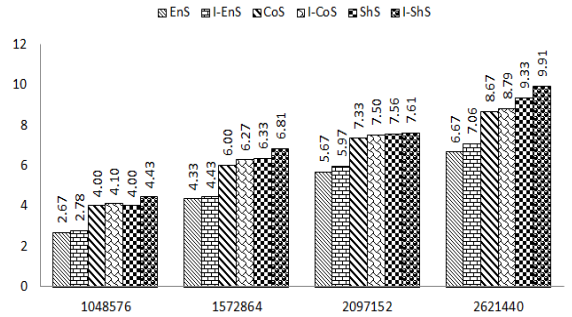


**Figure 6:** Energy consumption during downloading

### *B. Turnaround Time and Energy Consumption during Block Modification Operation*

The turnaround time and energy consumption are evaluated by performing a single block modification operation. In the EnS, CoS, and ShS, the whole file is encrypted and uploaded on the cloud server that consumes considerable amount of resources on the mobile device. However, the incremental version of the aforementioned schemes only encrypts and uploads the modified block(s) on the cloud server. The encryption and uploading of the modified block improves the turnaround time and energy consumption on the mobile device as expressed in Fig. 7 and Fig. 8, respectively.

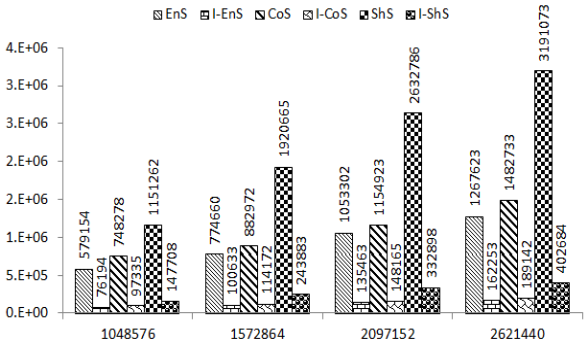


**Figure 7:** Turnaround time during block modification

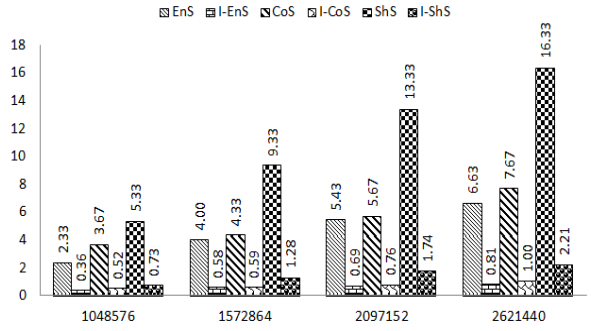


**Figure 8:** Energy consumption during block modification

## V. CONCLUSION

In this paper, the incremental version of the EnS, CoS, and ShS are compared with the original versions on the basis of turnaround time and energy consumption. During the initial encryption and uploading, the incremental version of the proposed scheme consumed more resources on mobile device as compared to the original versions due to the involvement of extra file management overhead. However, the incremental versions show significant improvement in performance while performing the block(s) modification operation. In some scenario, the incremental versions consume more resources as compared to original versions, such as the owner of the file wants to change all the file blocks. In such scenario, the mobile user has to encrypt and generate the message authentication code for all blocks. However, the possibility of occurrence of such case is negligible.

## ACKNOWLEDGMENT

The authors would like to acknowledge the financial support of the Bright Sparks Program at University of Malaya, Malaysia.

## REFERENCES

[1] A. N. Khan, M. L. M. Kiah, S. U. Khan, and S. A. Madani, "Towards Secure Mobile Cloud Computing: A Survey," Future Generation Computer Systems, vol. 29, pp. 1278-1299, July 2013.

[2] M. Armbrust *et al.*, "Above the Clouds: A Berkeley View of Cloud Computing", Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb. 2009.

[3] Pay-as-You-Go with Cloud Computing, http://technology.inc.com/2008/05/01/pay-as-you-go-with-cloud-computing/, access data: 05 October, 2012.

[4] S. Das, D. Agrawal, and A. E. Abbadi, "ElasTraS: An Elastic Transactional Data Store in the Cloud," proc. of the 2009 conference on Hot topics in cloud (USENIX '09), June 2009.

[5] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches," Wireless Communication and Mobile Computing, in press, 2011, doi: 10.1002/wcm.1203.

[6] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile Cloud Computing: A Survey," Future Generation Computer Systems, vol. 29, pp. 84–106, January 2013.

[7] K. Kumar and Y. H. Lu, "Cloud Computing For Mobile Users: Can Offloading Computation Save Energy?," IEEE Journal Computer, vol. 43, pp. 51-56, April 2010.

[8] E. Lagerspetz and S. Tarkoma, "Mobile Search and the Cloud: The Benefits of Offloading," IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 117–122, March 2011.[9] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," Journal of Internet Services and Applications, vol. 1, pp. 7-18, May 2010.

[10] S. Subashini and V. Kavitha, "A Survey on Security Issues in Service Delivery Models of Cloud Computing," Journal of Network and Computer Applications, vol. 34, pp. 1-11, January 2011, doi:http://dx.doi.org/10.1016/j.jnca.2010.07.006.

[11] N. Santos, K. P. Gummadi, and R. Rodrigues, Towards Trusted Cloud Computing, in: Proc. Hot Topics in Cloud Computing (HotCloud '09), San Diego, CA, June 2009.

[12] X. Zhang, J. Schiffman, S. Gibbs, A. Kunjithapatham , and S. Jeong, "Securing Elastic Applications on Mobile Devices for Cloud Computing," Proc. ACM Workshop on Cloud Computing Security (CCSW '09), pp. 127-134, November 2009.

[13] S. Xiao and W. Gong, "Mobility Can Help: Protect User Identity with Dynamic Credential," Proc. 11th Int. Conference on Mobile Data Management (MDM '10), pp. 378-380, May 2011.

[14] S. Wang and X. S. Wang, "In-Device Spatial Cloaking for Mobile User Privacy Assisted by the Cloud," Proc. 11th Int. Conference on Mobile Data Management (MDM '10), pp. 381-386, May 2010.

[15] R. Chow *et al.*, "Authentication in the Clouds: A Framework and its Application to Mobile Users," Proc. ACM Cloud Computing Security Workshop (CCSW '10), pp. 1-6, October 2010.

[16] D. Huan, X. Zhang, M. Kang, and J. Luo, "MobiCloud: Building Secure Cloud Framework for Mobile Computing and Communication," Proc. 5th IEEE Int. Symposium on Service Oriented System Engineering (SOSE '10), pp. 27-34, June 2010.

[17] D. Huang, Z. Zhou, L. Xu, T. Xing and Y. Zhong, "Secure Data Processing Framework for Mobile Cloud Computing," Proc. IEEE INFOCOM Workshop on Cloud Computing (INFOCOM '11), pp. 614-618, April 2011.

[18] Y. J. Chen and L. C. Wang, "A Security Framework of Group Location-Based Mobile Applications in Cloud Computing," Proc. Int. Conference on Parallel Processing Workshops (ICPPW '11), pp. 184-190, September 2011.

[19] I. Bilogrevica *et al.*, "Meetings Through the Cloud: Privacy-preserving Scheduling on Mobile Devices," Journal of Systems and Software, vol. 84, pp. 1910-1927, November 2011.

[20] W. Jia, H. Zhu, Z. Cao, L. Wei, and X. Lin, "SDSM: A Secure Data Service Mechanism in Mobile Cloud Computing," Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS '11), pp. 1060-1065, April 2011.

[21] W. Ren, L. Yu, R. Gao, and F. Xiong, "Lightweight and Compromise Resilient Storage Outsourcing with Distributed Secure Accessibility in Mobile Cloud Computing," Journal of Tsinghua Science and Technology, vol. 16, pp.520-528, October 2011.

[22] W. Itani, A. Kayssi, and A. Chehab, "Energy-efficient Incremental Integrity for Securing Storage in Mobile Cloud Computing," Proc. Int. Conference on Energy Aware Computing (ICEAC '10), pp. 1-2, December 2010.

[23] S.C. Hsueh, J.Y. Lin, and M.Y. Lin, "Secure Cloud Storage for Conventional Data Archive of Smart Phones," Proc. 15th IEEE Int. Symposium on Consumer Electronics (ISCE '11), pp. 156-161, June 2011.

[24] J. Yang, H. Wang, J. Wang, C. Tan, and D. Yu1, "Provable Data Possession of Resource Constrained Mobile Devices in Cloud Computing," Journal of Networks, vol. 6, pp. 1033-1040, July 2011.

[25] Z. Zhou and D. Huang, "Efficient and Secure Data Storage Operations for Mobile Cloud Computing," IACR Cryptology ePrint Archive, 2011.

[26] G. Ateniese, K. Fu, M. Green, S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," ACM Transactions on Information and System Security, vol. 9, pp. 1-30, Febraury 2005.

[27] P. K. Tysowski and M. A. Hasan, "Re-Encryption-Based Key Management Towards Secure and Scalable Mobile Applications in Clouds," IACR Cryptology ePrint Archive 2011, vol. 668, 2011.