ISSN 2347 - 3983
Volume 8. No. 10, October 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter018102020.pdf
https://doi.org/10.30534/ijeter/2020/018102020

A Cloud-Based Distributed Platform for Secured EPUB EBOOK Contents

Yunus Parvej Faniband[1], Iskandar Ishak[2], Fatimah Sidi[3], Marzanah A. Jabar[4]
[1]King Fahd University of Petroleum & Minerals, Research Institute, Center for Communications and IT Research, Dhahran-31261, Saudi Arabia.
[2]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 UPM, Serdang, Selangor Darul Ehsan, Malaysia, iskandar_i@upm.edu.my
[3]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 UPM, Serdang, Selangor Darul Ehsan, Malaysia
[4]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 UPM, Serdang, Selangor Darul Ehsan, Malaysia

ABSTRACT

Digital content creators who also own their digitally created content Utilizing Digital Rights Management or DRM in order to safeguard and govern the distribution and usage of their digital contents. The features of digital content protection (such as ebook) ranges from content limitation or transfer, and reading permission to authorized reading hardware or Software, to corporate delivery policies involving identification of users and devices for a definite period. Until today, standard for EPUB does not provide a specific DRM standard to protect eBook copyrights. In this paper, we propose a cloud-based EPUB content protection and control platform and describe the proposed ways in protecting EPUB contents through a cloud-based solution for digital content protection and access regulation. We have implemented a prototype with OpenStack open source cloud computing platform enforce access control and efficient in data management with data-at-rest encryption.

Key words: digital publication, distributed platform, epub, ebook

1. INTRODUCTION

Currently, the usage of E-books is becoming more popular and it has become to be as a serious competitor to the classically printed books (Chao et al., 2012) (Bounie et al., 2013). E-books offer more valuable features that are absent in paper-based printed books including such as built-in dictionary, embedded videos and audios, search functionality, and can be read using eReader software or hardware.

Currently, desktop computers, laptops, handheld devices, and mobile phone are able toper form as eReader device (e.g., Kindle reader) due to the availability of appropriate reading software. Among the popular eBook formats are MOBI (Mobipocket or PRC, based on the Open eBook standard), iBook (Apple), LRF (Sony broadband BBeB), AZW (Kindle Format 8), and EPUB. However, security issues over the Internet has threaten to weaken organizations control over information asset that may include digital content (Iskandar Ishak et al., 2013; Sidi et al., 2017), as well as the technique of accessing and querying the digital content (Saad et al., 2014; 2016).

There are a number of formats that support DRM that controls the eBooks by restricting Other devices or apps for readers, or sharing material with other people on eBooks. An example of a DRM is EPUB, The International Digital Publishing Forum(IDPF) developed a free and open eBook standard based on technologies and standards such as Open eBook and XHTML. One or the advantage of EPUB content is, it is flexible to the device screen In which the material is able to adapt its appearance to the device screen of the reader. The presence of digital eBook stores on the Internet makes it easy to download digital books to smart phones. This has resulted into illegal activities such as illegal replication, piracy and transferring or copying digital assets without permission. Therefore, there is a need to provide secure digital intellectual property (Huffman, 2013).Researchers have proposed ways to protect data in mobile or wireless communications such as by using secured data/document (Mohamed et al., 2013; 2016a; 2016b). Hence DRM is an approach to give protection to copyrighted digital content and to control the usage and circulation of these digital assets. DRM is a scheme that content owners can implement in order to protect their content circulation in terms of it being printed, copied, or distributed.

To implement a DRM for the content Submitted in the form of an e-book, such as material based on EPUB, IDPFF, (Bounie et al., 2013) provides no standard to impose DRM of EPUB-based content. However, it provides some commendations Based on the principles of XML protection set by the World Wide Web Consortium, (W3C). IDPF recommends for copyright protection and but there are no standard algorithm for protecting copyrighted digital content. There are issues with respects to the Security of e-book material through copyright, such as EPUB, because a content provider grants user rights in phases or aims to prohibit access to content by an unauthorised user. Because of the XML standard, there are different options to integrate DRM into EPUB.

Some existing EPUB copyright protection systems employ encryption of the Using a single encryption key, the whole '.epub' file or separately encrypting components of an ’.epub’ file using a single encryption key, following the Lightweight Content Protection guidelines (IDPF, 2012) from IDF. Some systems follow non-interoperable DRM schemes that are fixed to specific eBook reader devices or application addition to following restricting fixed to specific eBook reader devices or applications. Digital Watermarking (McGuire & O’Leary, 2011)is another means of protecting the content. Readium is an open-source framework with high-performance protection and distribution techniques for EPUB parsing and provide design standard reference software to apply DRM independent of EPUB.

In this paper we proposed a cloud-based content distribution system that enables securing and storing the eBook content in trusted cloud store. The proposed system will also be integrate Sigil EPUB editor with EncSwift(Bacis et al, 2016) which enforce access control and efficient in data management with data-at-rest encryption for OpenStack object storage.

Table 1: eBook DRM technology providers

DRM Platform	Filetype	Encryption	Electronic Signature
Adobe Content Server	Epub/pdf	AES-128	Private
Amazon	AZW	Private	Private
Apple Fairplay DRM	iBook	AES algorithm with MD5 hashes	MD5
Fasoo	Private Format	SEED AES-128 RSA1024 3DES	RSA1024

2. LITERATURE REVIEW

EPUB format has been explored in detail in a number of literatures (Williams, 2017). Based on the literatures, different tools were used to convert the main publishing format of an open access journal from PDF to EPUB format (Eikebrokk et al., 2014). There are also, a number of implementations to collect related information in EPUB content and it was prototyped using Readium, an open source SDK (Kataoka et al., 2013). Due to the recent popularity of social media, a procedure is proposed to turn Facebook contents into EPUB files, using Facebook SDK and EPUB generation tools (EPUBGen) (Chen et al., 2011). In order to create and read EPUB content, Adobe provides In Design tool along with review functionality integrating the Adobe digital Editions protection technology (see Table 1).

Apple Inc. has created iBooks Author tool (Fenwick et al., 2013) that follows a proprietary Apple file format similar to EPUB format by integrating the Apple FairPlay DRM. Sigil and Calibre are two popular open source EPUB authoring tools. Both Sigil and Calibre provide editing and importing support for EPUB file (EPUB 2.0 and EPUB 3.0 specifications). However, both tools do not provide any built-in content protection mechanisms with encryption.

The work solution proposed in (Eun-Bum et al., 2012) rely on an EPUB multiple key encryption method and has the feature of partial preview of content. This study in (Kim et al., 2013) attempts to explore the technological approaches to licensing methods that support eBook DRM compatibility based on the four Korean EPUB DRM standards. Readium project is an effort to design standard reference software to apply DRM independent of EPUB, still has solves the complexity and compatibility issues in plural DRMs.

The researchers (Kim et al., 2015) made efforts to solve compatibility of Readium SDK. The same group of researchers proposed a method to minimize the complexity of Readium SDK (Kim et al., 2016) in order to provide easy interlock between plural DRM and SDK. Authors of (Chen et al., 2017) propose method to improve the annotation mechanism based on Readium. Table 1 shows the leading enterprise eBook DRM platforms and their associated technologies. (Park et al., 2000) explained the various types of application-level security architecture implementations possible for the DRM content in which there is an implementation of a personal digital rights system for mobile devices (Bhatt et al., 2009).

3. MATERIALS AND METHODS

In this paper, A lightweight framework of content security that includes the module for content creation and the reading framework is proposed. In the proposed system, the proposed system recommends an The file called 'encryption.xml' is an The encryption file contains key details and algorithm information to be created when the EPUB standard is met and the content of the e-book is encrypted. The suggested file is intended to encrypt the contents of the container in the META-INF directoryThe EPUB Open Container Format (OCF) uses XML Encryption to provide an encryption mechanism that enables different algorithms to be used in compliance with W3C 's XML security standards. In order to avoid falsification of content, a file called 'signatures.xml' is also proposed to carry the digital signatures of the container and its contents. Along side the file, a file named ‘rights. xml‘is also proposed to hold information of the DRM. The OCF specification also The use of a particular algorithm and rights-expression language is recommended, but the use of the chosen algorithm for content encryption-signature and rights-control is not controlled.

The proposed framework authoring and content protection system lets the user/content provider create digital content in the EPUB form. The framework enables the user/content provider to protect the EPUB content file by associating the certain control sets with it within the authoring environment and share in the cloud for others users. The framework allows the authorized user to unlock and display the protected content only after enforcing the control set. The design of the proposed system involves designing the following central aspects of the system:

1. An EPUB authoring and reading tool: A Desktop EPUB authoring tool facilitate creating/editing and 8 encrypting the content and upload to the cloud.

2. Cloud based crypto system enable securing and storing the content in trusted cloud store. Prior to describing the design of these aspects for our system, we will consider some of the important security and functional characteristics, and assumptions.

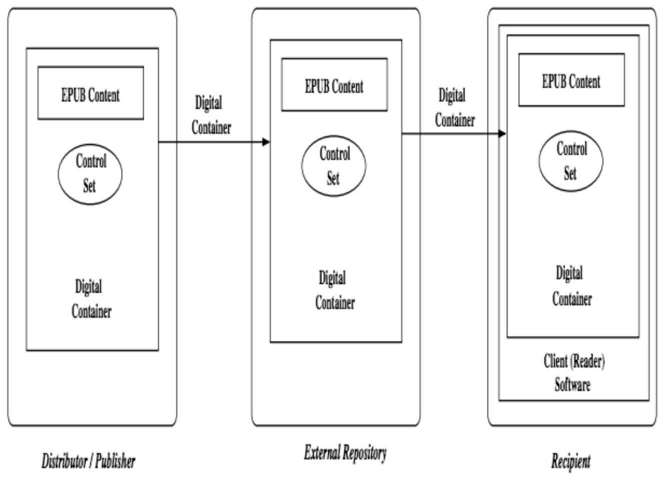


Figure 1: General security architecture with embedded control and external repository

3.1 Proposed System Features

In the proposed system, the EPUB archive is contained in a digital container (C1). The control set serve as a mechanism for the group of access rights and usage rules to manage the recipient's access and usage of the EPUB archive. The digital container (DC) (Sibbert et al., 1995) (Kaplan et al., 1996) is a primary feature of use-control technologies. The high level block diagram of this security architecture with embedded control and external repository is shown in Figure 1. DC can be used to prevent tampering and utilized as an electronic envelope to secure digital content and to regulate usage by integrating with cryptographic mechanisms.

The digital information with EPUB content is wrapped within a DC and sent to the repository server (open stack swift). Recipient can be restricted from storing the DC on their storage (C6).The architecture enables the content provider to locally store the content and also revoke a previously granted access (C2) through the policy updates. The control set in this system can restrict storage of non-encrypted EPUB content on the recipient's non-volatile storage (RAM). The recipient has to explicitly store the EPUB digital container for further access. But the control set can restrict the opening of EPUB digital container by third recipient if it is re-published to them (C3).

3.2 Securing Content using Cloud-based Crypto System

Content encryption is a standard technique to secure the content in DRM. As per the requirement C2, the publisher must to able to specify who can unlock and get the content rendered. For example if client 'A' publish some protected content to recipient 'B', it must be desired that content can be unlocked in B only and 16 no other recipient should play the content. The Publisher needs to share the EPUB content and used external cloud service to store data in OpenStack Swift object storage. The Publisher would like to share the content in these Swift objects, selectively to other users. The architecture of OpenStack Swift object storage stores data into accounts, containers, and objects hierarchically. Hence by giving access to each of the created Containers, the Publisher can share grant access to content all of the objects in the Container. The proposed system uses OpenStack Swift for the object storage, where the data is divided hierarchically into accounts, containers, and objects. The method of selective and policy based encryption (Section 3.2.1)is employed for enforcing access control by the cloud provider and over-encryption are applied over the data. As in typical client-server architecture, the system encrypts the data (swift objects) before transferring them to the cloud store and decrypt data when fetched from the cloud store. The system executes crypto graphic operations in different layers with encryption keys handled at both, the server side as well as the client side.

3.2.1 Selective and Policy Based Encryption

Initially, a Data Owner or Publisher creates a container (a Book in publishing sense) and define a DEK. Architecture of OpenStack Swift store data hierarchically into accounts, containers and objects. In order to control the access right properly, it is mandatory for the Publisher to retain the access control list architecture and make cure each object in the container is subjected to same access control list. This is done by Publisher orthe Data Owner through encryption of all the objects (each file in eBook archive) in a particular container with the DEK of that container. Policy-based encryption enforces only authorized users have permission for accessing the objects in each container.

In order to achieve this, a key is required for authorized users in which they have knowledge about this key or can derive from this key. The key is used to encrypt each object in the container to make sure the same group of users has the access to digital Book content. Meanwhile at the encryption layer in the cloud side, the DEK is encrypted using the Master key of the Data Owner or the publisher. The OpenStack swift provide access control list (ACL) to enable access policy to the objects in the container. For each of the user permitted to access objects in the container, the DEK is encrypted with the combination of user RSA public key and signed by the content owner or Publisher with the user's signature private key. The access mechanism of a User for eBook content works as follows. When there is a request to retrieve a particular content (object), it is required to get the attribute of the DEK that is used to encrypt that object.

In the beginning, Object descriptor reads this DEK identifier and allows access to the respective KEK and further determine corresponding DEK. In case of symmetric KEK it is determined by user own Master Key or the user RSA private encryption key for asymmetric KEK.

Table 2: Definition of different keys used in over-encryption approach in Openstack Swift

Key	Description
Master Keys	The Master Key belongs to the client side. The systems identify a particular user with two pairs of public and private keys and a symmetric Master Key.
RSA key pairs	One combination is utilized for encryption (RSA key pair) and one combination is needed during signing messages (RSA signature combination)
Data Encryption Keys (DEKs)	The Encryption Layer use a DEK is a symmetric key for encrypting or decrypting an swift object
Key Encryption Keys (KEKs)	KEK refers to the stored encrypted form of DEKs

3.2.2 Encryption using Swift (EncSwift)

The systems identify a particular user With a symmetrical Master Key and two public and private key combinations (See Table 2). While the Master Key is kept on the client side, for all the other keys, Barbican (the OpenStack Hidden Storage) is the shop. The Encryption Layer uses a DEK to encrypt an object stored in the cloud store. Initially each object that belongs to one container will be encrypted with the same DEK. A policy update requires generating new DEK. EncSwift offers three ways for Over-Encryption on Swift objects. These methods need changes in Swift storage module. Each of the methods affects the data protection and downloads or uploads efficiency.

The method over-encryption is applied only on the data during the transfer between the client and server and is referred to as On-the-fly mode. The on-resource mode enables over-encryption on data when the objects are stored onto the cloud server. The third method of end-to-end mode over-encryption is applied with the combination of the protection of the prior two methods by over-encrypting of data both during the transmission and at rest. The proposed ePUB content protection framework use the third method (i.e end-to-end mode) to apply over-encryption on the swift objects saved on the server disks and maintain the SEL encryption during swift object transfer to the client.

3.2.3 Access Control List (ACL) in Swift

In order to control the access of stored objects, two levels of privileged access control are granted by Swift. The first level which is the account-level is the access control that enables the user to perform administrative level operations over an entire account (tenant) (e.g., assign rights like Read-Only and Read-Write access, to other users). The second level is the account-level that allow users to grant authorization to access objects throughout the account. They are also allowed to access account metadata and delete or create new containers. Container ACLs are set to allow following actions on the container: read or write or listing. Users with read (list and write) ACL can download objects from the container (list the container contents and upload objects to the container, resp.)

3.2.4 Handling Policy Update

In order to authorize an operation, the system is required to create a new (asymmetric) KEK for the authorized user and save it in the OpenStack Secret Storage (Barbican). The container (data) owner needs to create this KEK. In order to determine the procedure of revoking user access to a container, consider the mechanism to access swift object in the framework. When there is a request to retrieve a particular content (object), it is required to get the attribute of the DEK that is used to encrypt that particular object. The Swift Decryptor Module (EncSwift) read this DEK identifier in the beginning and grants access to the respective KEK and further determines corresponding DEK. For a revoke operation, there must a new DEK, with which the objects must be encrypted at the SEL level that should be unknown to the invoking users. Hence in order to suppress the use of locally stored KEK, the containers are assigned with two keys. The first key is associated at the client-side from the BEL layer. This key will be derived by entire users (authorized users and to be evoked users) originally authorized to access container. The non-invoked users only can derive from the second key at the server side (SEL). According to the data hierarchy of OpenStack swift, when a new object is added to the container, it derives the List of access control of an associated container. The swift object is also encrypted with the container-corresponding BEL DEK key. If the swift object is encrypted with the SEL DEK key, the swift object will be encrypted. object is considered to be revoke.

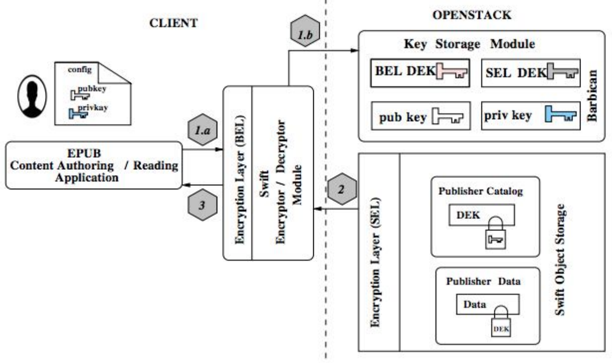


Figure 2: Architecture of encryption of content using EncSwift.

3.2.5 Data Content Protection Process

Figure 2 illustrates the proposed system architecture for protecting the ePUB content with policy- based encryption. Figure 2 also depicts the events when the client accessed a remote data of object stored in OpenStack swift.

• Step 1.a - The EPUB Content Authoring or Reading System send the request to access the object to Encryption layer of Swift Decryptor Module at the Client side. There will be a configuration file that describes the path of the user's keys.The application sends this info after reading from this configuration file.
• Step 1.b - The Swift Decryptor Module access the private key of the user from the Barbican key storagemodule (Step 1.b). Alternatively the client can send private key with the request.
• Step 2 - The Swift Decryptor Module retrieves the KEK (from the Publisher's catalogue stored in Swift)matching to the DEK that is used to protect the object under consideration. The Swift Decryptor Module decrypts it to get the DEK. It will also obtain the encrypted object and decrypts it with the help of the retrieved DEK.
• Step 3 - The decrypted object from the Swift Decryptor Module is sent to the requested client application (EPUB Content Authoring or Reading System). The procedure to upload an object works is analogues to accessing the object, but with the APIs requiring the Publisher's keys to encrypt/decrypt the data.

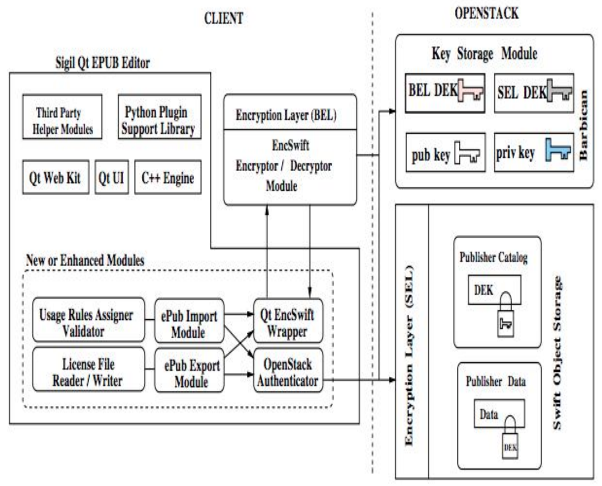


Figure 3: Block diagram of the secured EPUB eBook content using OpenStack cloud platform.

4. RESULTS AND DISCUSSION

The proposed system (as shown in Figure 3) It enables the producer of content to build and export protected content for safe sharing. The import module makes it possible to render the content and activate the contentThe device can also protect text files, media files, and related zip properties archive with encryption individually. The proposed implementation follows the embedded control architecture with external repository (Park et al., 2000). In this architecture, the control set is enclosed in the digital information. The digital container wraps the information and it is transferred to the external repository server (OpenStack). Also, in this work, an Based on Qt, a cross-platform technology framework to support the following aspects of the proposed system, the open source version of the Sigil EPUB editor is enhanced:
1. Content encryption during Export/Save.
2. Client side Swift Encryptor and Cloud Data upload
3. Content decryption during Import/Read.

The export or The EPUB editor save module is enhanced to introduce an EPUB multiple key encryption process using EncSwift Wrapper APIs (Bacis et al.,2016) that encrypts individual files as explained in section 3.2.5. The epub Export module encrypts each of the allowed files individually. It also formalizes and repackages the encrypted unit files based on the EPUB standard and invokes the encryption calls from the EncSwift library. The Import or Reading module decrypts individual files as explained in section 3.2.5 by calling the decryption APIs of EncSwift library through the wrapper module.

The prototype of the system is implemented using *Qt* C++ engine instead of python plug-ins. In the proposed system, *Qt* C++ engine is used to utilize the encryption and decryption modules during the authoring stage. *Qt*EncSwift Wrapper APIs is utilized in the encryption module to securely save the content in the cloud. While the decryption module assist in the importing of encrypted EPUB content with *Qt*EncSwift Wrapper APIs for decryption.

The evaluation environment for evaluating the proposed content protection platform A selection of virtual machines and clients for mobile devices includes. OpenStack Swift and Barbican deployments with 4 storage nodes and a proxy node and 4 storage nodes were used to set up a virtual machine (VM). The proxy node with Ubuntu 14.04 was equipped with Intel Xeon CPU and 4 GB RAM and storage nodes (Ubuntu 14.04) with Intel Xeon E5-2403 processors and 1 GB RAM. Another VM with the enhanced solution of Sigil with EncSwift is setup on another Ubuntu 14.04,that was equipped with Intel Xeon CPU and 2 GB RAM. The empirical evaluation and comparison of the system is measured based on the following criteria:

1. Computation Cost with impact of cryptographic operations
2. Communication Cost
3. Computation complexity of a group update

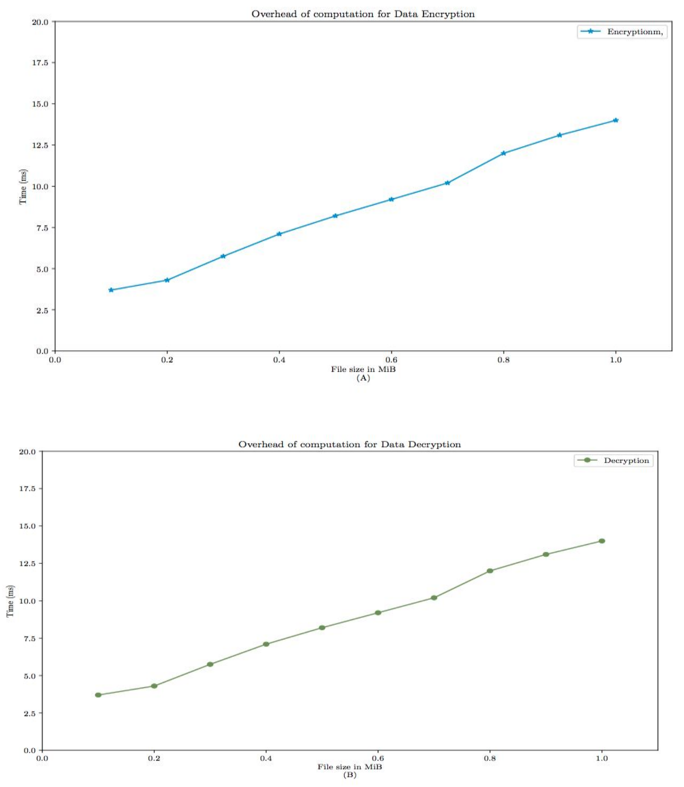
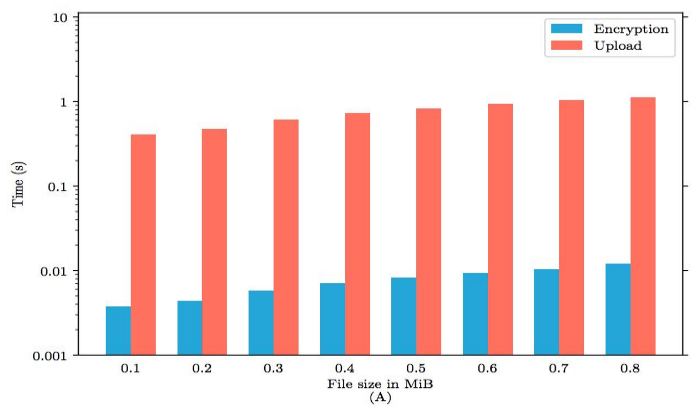


Figure 4: Overhead of computation for Data (A) Encryption and (B) Decryption at the client side

4.1 Evaluation of Computational Cost

In orderto test the performance of the system at the client side, we conducted local data processing tests for encryption and decryption. Figure 4 depicts the client side perceived computation overhead for data encryption and decryption for varying sizes of data content. We can observer that the time taken to encrypt 1MB file is approximately 12 ms, which make the less computation cost with less resources and better security for the outsourced cloud data. We can notice that the time usually increases with the size of file data and end-to-end mode takes more time, since In the end-to - end mode, SEL encryption and decryption function on the client side.



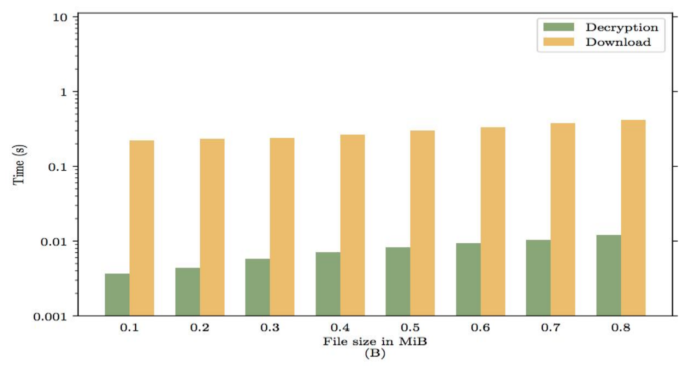
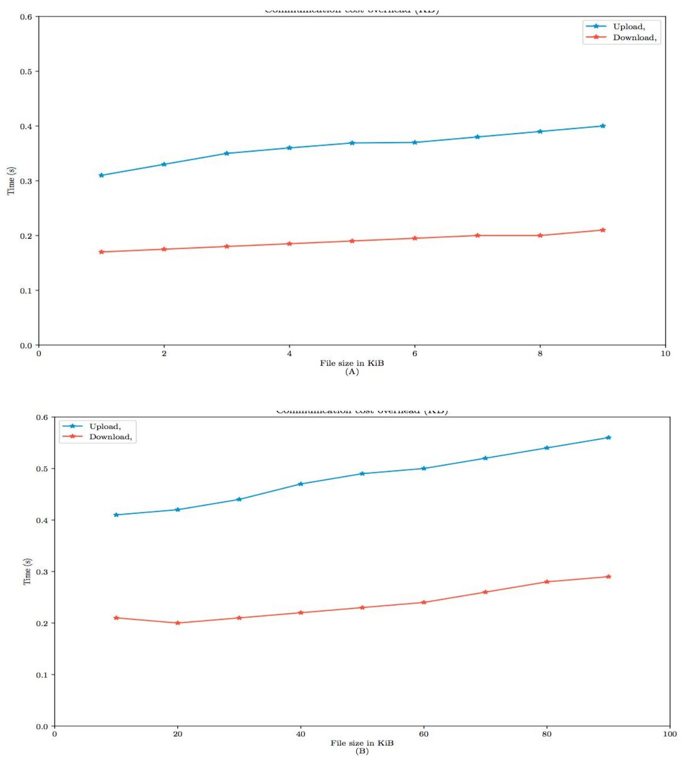
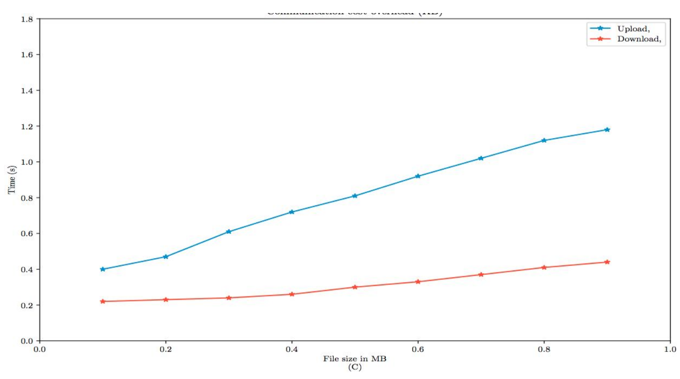


Figure 5: Effect of cryptographic operations (A) Encryption and (B) Decryption at the client side.

4.2Effect of Cryptographic Operation

Next we analysed the effect of cryptographic operations while uploading or downloading the content through the EPUB Sigil IDE (Creator/Reader) to or from the cloud storage. Figure 5 depicts the comparison of time taken for encryption process and the OpenStack swift upload operation. We observed that the content encryption operation at the client side is tolerable. As an example, an EPUB file with the size of 0.8 MiB required 1.12 seconds for upload and only 0.1 seconds for encryption. Hence less than 1% overhead is used for file content encryption over the OpenStack swift and Barbican upload overhead. Figure 5 also depict the comparison of time taken for decryption process and the OpenStack swift download. Similar to encryption, the content decryption operation at the client side is tolerable in contrast to the download operations and involved less than 1% overhead compared to file content decryption over the OpenStack swift download.





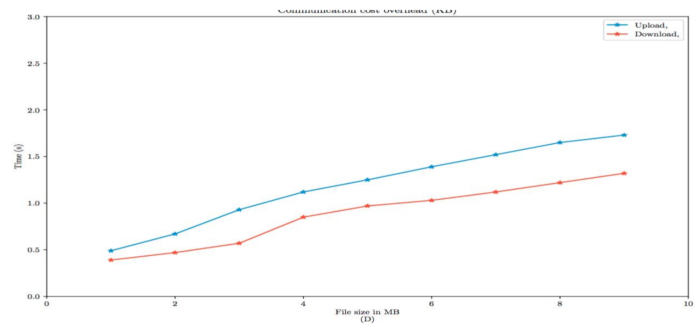


Figure 6: OpenStack upload and download overhead with different data size

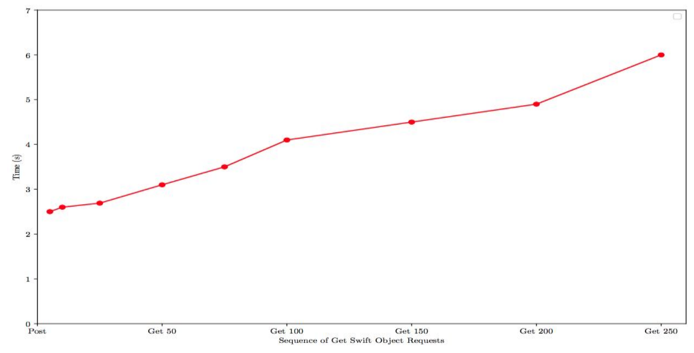


Figure 7: Performance for serving multiple requests after a policy update.

4.3 Communication Cost

We also examine the communication cost, which is the overhead cost when the content owner stores the EPUB file content to cloud store and then request to access secured data. Figure 6 shows a variation of the latency for upload and download operations with respect to different file sizes. We can observe that the average latency is stable for small file size and gradually increase for large data content.

4.4 Computing complexity of a group update

We investigated the communication overhead, when the content owner needs to update sharing policy of the content and carry out a Group update to share content to a new user. In the evaluation we analysed the time taken for updating the policy over a 100MB container. The test analyse the effect of applying a policy change for a 100MB container, consisting of object size of 1MB . The system then requests a sequence of get object request to the cloud store. As depicted in Figure 7, There is an immediate overhead on the initial post container due to re-encryption of the saved objects. End-to -end mode is used by the proposed ePUB content security system to apply over-encryption to the speedy objects stored on the server discs. It should be noted that the overhead will decrease as there are less policy updates than the object access requests.

5. CONCLUSION

It provides a proposed method for securing and managing the content of the EPUB. The prototype is also implemented using the Sigil Desktop Editor and enhanced to support the proposed EPUB security framework in terms of integration of modified version python-swiftclient library (EncSwift) for encrypting and decrypting the swift objects. The implementation is evaluated for Computation Cost with impact of cryptographic operations, Communication Cost and Effect of policy update. Our proposed system provides a suitable method for protecting and sharing the EPUB content with current cloud technology by utilizing the open and modular architecture of OpenStack Swift object storage.

6. ACKNOWLEDGMENT

Authors would like to thank the Faculty of Informatics and Computer Science, Universiti Putra Malaysia for its assistance. This research is funded under the Fundamental Research Grant Scheme (FRGS) by Malaysia's Ministry of Education. (FRGS/1/2015/ICT04/UPM/02/07) and Universiti Putra Malaysia under the Putra Grant Scheme-Putra Graduate Initiative (GP-IPS/2016/9478900).The authors also would like to thank to King Fahd University of Petroleum & Minerals (KFUPM) and Center for Communications and IT Research, Dhahran, for all the support in producing this research paper.

REFERENCES

[1] Bill Rosenblatt. EPUB Lightweight Content Protection: Use Cases & Requirements, 2012.http://idpf.org/epub-content-protection.[Online; accessed 24-June-2019].
[2] Brian Huffman. Self-publishing digital books: options, considerations, and insights. 2013.
[3] Chao Chiang-Nan, Niall Hegarty, and Abraham Stefanidis. Global Impacts And Challenges Of Paperless Books: A Preliminary Study. International Journal of Business and Social Science, 3(11), 2012.
[4] Chen-Yuan Chuang, Yi-Bing Lin, Zhihao Julie Ren, and Yu-TienYeh. User-Generated E-Books From Facebook Contents. In 2011 7th International Wireless Communications and Mobile Computing Conference, pages 1918–1922.23 IEEE, 2011.

[5] David Bounie, Bora Eang, Marvin Sirbu, and Patrick Waelbroeck. Superstars And Outsiders In Online Markets: An Empirical Analysis Of Electronic Books. Electronic Commerce Research and Applications, 12(1):52–59, 2013.

[6] Enrico Bacis, Sabrina De Capitani di Vimercati, Sara Foresti, Daniele Guttadoro, Stefano Paraboschi, Marco Rosa, Pierangela Samarati, and Alessandro Saullo. Managing Data Sharing In Openstack Swift With Over-Encryption. In Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, pages 39–48. ACM, 2016.

[7] EriKataoka, Toshiyuki Amagasa, and Hiroyuki Kitagawa. A System For Social Reading Based On Epub3. In Proceedings of International Conference on Information Integration and Web-based Applications and Services, page 72. ACM, 2013.

[8] Eun-Bum Kim, Kyung-Il Kim, Tae-Hyun Kim, and Seong-Hwan Cho. A Study Of Partial Preview Control Method Of Epub-Based Ebook DRM. The Journal of The Institute of Internet, Broadcasting and Communication, 12(1):249–256, 29 2012.

[9] Eun-Bum Kim, Kyung-Il Kim, Tae-Hyun Kim, and Seong-Hwan Cho. A Study Of Partial Preview Control Method Of Epub-Based Ebook DRM. The Journal of The Institute of Internet, Broadcasting and Communication, 12(1):249–256, 29 2012.

[10] Greg Williams. Epub: Primer, Preview, And Prognostications. Collection Management, 36(3):182–191, 2011.

[11] Hugh McGuire and Brian O’Leary. Book: A Futurist’s Manifesto: A Collection of Essays from the Bleeding Edge of Publishing. ” O’Reilly Media, Inc.”, 2011.

[12] IskandarIshak, Sidi, F., Jabar, M. A., Sani, N. F. M., Mustapha, A., Supian, S. R., & Apau, M. N. 2012. A survey on security awareness among social networking users in malaysia. Australian Journal of Basic and Applied Sciences, 6(12), 23-29.

[13] Jaehong Park, Ravi Sandhu, and James Schifalacqua. Security Architectures For Controlled Digital Information Dissemination. In Computer Security Applications, 2000.ACSAC’00. 16th Annual Conference, pages 224–233. IEEE, 2000.

[14] James B Fenwick Jr, Barry L Kurtz, Philip Meznar, Reed Phillips, and Alex Weidner. Developing A Highly Interactive Ebook For CS Instruction. In Proceeding of the 44th ACM technical symposium on Computer science education, pages 135–140. ACM, 2013.

[15] Mao Chen, Hang Luo, Tengbaao He, and Sanya Liu. A Method to Improve the Annotation Mechanism based on Readium. In 2016 4th International Conference on Machinery, Materials and Information Technology Applications. Atlantis Press, 2017.

[16] Marc A Kaplan et al. IBM Cryptolopes, Superdistribution And Digital Rights Management. viewed at on, pages 1–10, 1996.

[17] Mohamed, K., Sidi, F., Jabar, M. A., &Ishak, I. (2013).A novel watermarking technique in data transmission between QR codes and database. Paper presented at the 2013 IEEE Conference on Open Systems, ICOS 2013, 95-99. doi:10.1109/ICOS.2013.6735055.

[18] Mohamed, K., Sidi, F., Jabar, M. A., Ishak, I., Salimin, N., Salleh, N. S. A., Hamzah, A.Q., Jarno, A.D., andPauzi, M. F. (2016b). Strengthening user authentication for better protection of mobile application systems. Journal of Theoretical and Applied Information Technology, 85(3), 416-424.

[19] Mohamed, K., Sidi, F., Jabar, M.A., &Ishak, I. (2016a). Protecting Wireless Data Transmission In Mobile Application Systems Using Digital Watermarking Technique. Journal of Theoretical and Applied Information Technology, 83(1): 52-63, 2016.

[20] Olin Sibert, David Bernstein, and David Van Wie. The digibox: A Self-protecting Container for Information Commerce. In USENIX Workshop on Electronic Commerce, 1995.

[21] Saad, N. H. M., Ibrahim, H., Alwan, A. A., Sidi, F., &Yaakob, R. 2014.A framework for evaluating skyline query over uncertain autonomous databases. Procedia Computer Science, 29 1546-1556. doi:10.1016/j.procs.2014.05.140.

[22] Saad, N. H. M., Ibrahim, H., Sidi, F., Yaakob, R., & Alwan, A. A. 2016. Computing range skyline query on uncertain dimension, doi:10.1007/978-3-319-44406-2_31.

[23] Siddharth Bhatt, Radu Sion, and Bogdan Carbunar.A Personal Mobile DRM Manager for smartphones. Computers and Security, 28(6):327–340, 2009.

[24] Sidi, F., Marzanah, A. J., Affendey, L. S., Ishak, I., Sharef, N. M., Zolkepli, M., Ming, T. M., AbdMokthi, M. F., Daud, M., Zainuddin, B. Z. and Hamid, R. A. 2017. A Comparative Analysis Study on Information Security Threat Models: A Propose for Threat Factor Profiling. Journal of Engineering and Applied Sciences, 12(3), 548-554. doi:10.3923/jeasci.2017.548.554.

[25] Tae-Hyun Kim, Hee-Don Yoon, Ho-Gab Kang, and Seong-Hwan Cho. A Study OnEbookDrm Interoperability Based On Idpf Readium SDK. The Journal of The Institute of Internet, Broadcasting and Communication, 15(2):15–21, 2015.

[26] Tae-Hyun Kim, Ho-Gap Kang, Yoon-Ho Kim, and Seong-Hwan Cho. A Study Of License Acquisition Method Supporting Mutual Compatibility Of Epub-Based Ebook DRM. The Journal of The Institute of Internet, Broadcasting and Communication, 13(1):205–214, 2013.

[27] Tae-Hyun Kim, Hui-Don Yun, Ho-Gab Kang, and Seong-Hwan Cho. A Study On Content Protection Framework For E-Book Drm-Agnostic Based On Readium SDK. The Journal of The Institute of Internet, Broadcasting and Communication, 16(1):7–14, 2016.

[28] Trude Eikebrokk, Tor Arne Dahl, and Siri Kessel. EpubAs Publication Format In Open Access Journals: Tools And Workflow. Code4Lib Journal, 24, 2014.