Volume 5, Issue 3, March 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Secure Multimedia Retrieval System Using Selective Ordered Bucketization

B. Brinda Devi, S. Gomathi, C. Nalini
Dept. of CSE, Bharath University
Tamil Nadu, India

Abstract— Bucketization is the splitting of large amount of data into smaller chunks and storing the parts in different sub servers called buckets. This paper deals with ordered bucketization (OB), which makes use of a random key generator algorithm and a deterministic bucketing algorithm, to propose a secure storage and retrieval system for multimedia files. In OB, plaintext-space is split into a number of disjoint buckets, from one to n, based on the ranges covered by them. A bucket number is allotted to each split range in ascending order. Based on the survey on various literature papers, it can be concluded that OB is considered more efficient than order preserving encryption (OPE) in terms of security. Further, asymmetric encryption is used to overcome key related security issues while retrieving the data. While enclosing the proposed method, it is concluded that multimedia retrieval system works effectively for quick retrieval of large multimedia files in a secure manner.

Keywords— multimedia retrieval, encryption, bucketization, cryptography, range query

I. INTRODUCTION

Ordered Bucketization is dealt as a cryptographic object. In OB, plaintext-space is split into a pre-defined range of buckets. With bucketization, along with OB, varied styles of SQL queries over encrypted data can be performed, if the bucket number corresponding to the initial plaintext is given with encrypted information [10]. For instance, if a client needs to get the data within a certain specified range, it first calculates the numbers of buckets whose union is that the smallest set covering the queried range. The client then sends the bucket numbers to the main server that acts as the database. The database server searches all the encrypted information whose bucket range matches one of the received numbers. The server then, sends the information back to the client. After decryption, correct results can be acquired by filtering out the data that are outside the specified range. During this case, a huge quantity of information is sent and received across the client and server than in the case where the database stores unencrypted information as a result of the false positives that occur in cases in which a bucket has the data required by the client and the data that it doesn't need.

This approach is very efficient compared to the case where the client receives all the encrypted information from the server and decrypts all information to get the proper query result. Therefore, this technique is very helpful when users cannot store unencrypted information like in cloud computing. Bucketization is very much useful for vary queries if the bucket range is assigned in an ordered manner. For instance, in the same case reported above, the client has to send only two bucket numbers: one that has start of the range value and the one that has the end of the range value. In this way, order preserving encryption (OPE) can be replaced by OB [6], [8].

OPE supports more efficient range queries compared to the case of OB. Besides, the results of range queries on ciphertexts encrypted by OPE doesn't produce false positives as a result of its comparison ability on ciphertexts which differentiates whether a ciphertext has the specified plaintext within the given range when the server has encryption of two borders of queried range in plaintext-space. But OPE doesn't support a weak version of IND-OCPA (INDistinguishability under Ordered Chosen Plaintext Attack) security [3]. This lack of security led to the consideration of using OB rather than OPE.

II. RELATED WORK

Paper [1] introduces encryption with ordered bucketization (EOB), a combination of OB and a symmetric encryption scheme, that provides reasonable range querying efficiency by showing that the bucket size distribution is not skewed and suggests a new security model for EOB called IND-OCPA-P which ensures high level security.

Paper [2] presents the first order-preserving scheme that achieves ideal security using the idea of mutable ciphertexts called mOPE, proposes a stronger notion of same-time OPE security, stOPE, that reveals only the order of the elements in encrypted database at the same time and presents versions of mOPE and stOPE that protect against a malicious server by using Merkle hashing.

Paper [3] deals with construction of real OPE scheme for specific plaintext domain and proves that it is secure under IND-OCPA and that ideal OPE is not secure under the same conditions and extends the concept of OPE to generalized OPE (GOPE) which is secure under IND-OCPA in polynomial-sized domains and then the security notion is weakened to IND-OLCPA to prevent the big jump attack in superpolynomial-sized domains.

Paper [4] deals with securely executing multidimensional range queries over outsourced data and provides a tunable data bucketization algorithm that allows the data owner to control the trade-off between disclosure risk and cost, identifies two distributional properties- variance and entropy, which when higher reduces the worst-case risk, proposes a trade-off algorithm called controlled diffusion (CDf) that increases variance and entropy values and carries out extensive experimentation to test the quality and performance of optimal partitioning and diffusion algorithms.

Paper [5] proposes a secure data storage scheme that can effectively defend against frequency-based attacks in wireless networks, designed a novel 1-to-n encryption scheme that utilizes the proposed dividing and emulating techniques that provides robust security guarantee against frequency-based attacks with low overhead and supports efficient query evaluation over encrypted data.

Paper [6] proposes a security notion in the spirit of pseudorandom functions (PRFs) with design of an efficient OPE scheme and prove its security under the notion POPF-CCA for pseudorandom order-preserving function against chosen-ciphertext attack and uncovers a connection between a random order-preserving function and the hypergeometric (HG) probability distribution using Lazy Sample algorithms.

Paper [7] involves construction of public-key systems using Hidden Vector Encryption to support comparison queries, subset queries, arbitrary conjunctive queries without leaking information on individual conjuncts and presents a general framework for analyzing security of searching on encrypted data systems.

Paper [8] proposes a new order preserving encryption scheme, OPES, that allows comparison queries on encrypted numeric data producing sound (no false hits) and complete (no false drops) query results, allows addition of new values added without requiring changes in the encryption of other values and designed to operate in environments in which the intruder can get access to the encrypted database, but does not have prior information such as the distribution of values thus providing robust encryption.

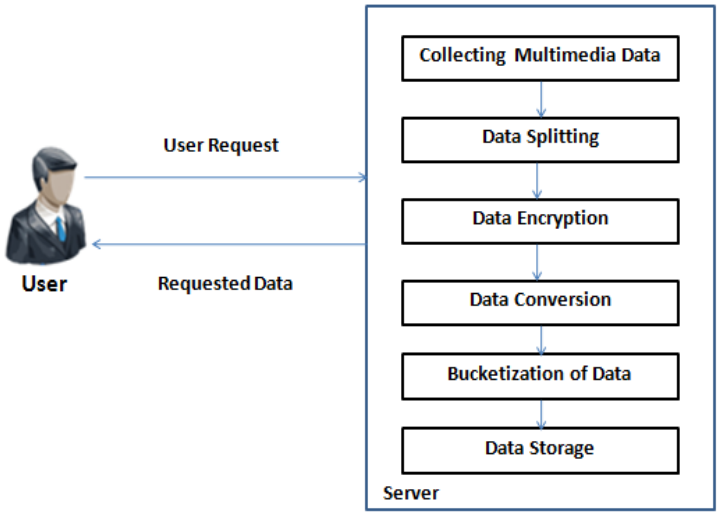
Paper [9] investigates data bucketization as a privacy-enhancing technique and highlighted the fundamental tradeoff between privacy and performance, derives an optimal algorithm for data partitioning that provably minimizes performance overhead in query processing, presents the controlled diffusion algorithm that lets the data owner fine-tune bucketization to achieve the desired level of data privacy by sacrificing the accuracy by a small measured amount.

Paper [10] addresses a data-privacy challenge where owner of database does not trust the service provider and so only encrypted data is stored at service provider, and enables service provider to execute SQL queries using a technique which deploys a coarse index that allows partial execution of SQL query on provider side, and proposes a technique to operate the SQL query and split it into server query and client query.

III. MULTIMEDIA RETRIEVAL SYSTEM

Multimedia bucketization is made possible, that is, text as well as video files are encrypted and split and can be stored in the same bucket. An asymmetric encryption is used so as to avoid key leakage between client and server. The plaintexts in text files are converted into ciphertexts on encryption whereas the video files are split into chunks based on their time-frames and stored as non-playable files in the cloud server. The data are thus evenly split into the predefined number of buckets and coefficient values for each bucket are stored separately as index file which helps in bucket identification for data retrieval.

A. Architecture Diagram



B. User Registration and Server Deployment

A user application is created to allow the user to access the data from the cloud server. The user needs to create an account to access the network. All the user details will be stored in the database of the server. A user interface frame is designed to communicate with the server through network coding using the programming language java. The server then establishes the connection to communicate with the users. Each user's activities will be updated in the database by the server. Server also checks for user authentication to prevent unauthorized users from accessing the application.

C. Data Splitting & Storage in Separate Buckets

By splitting the jobs the user requests are easily executed and work load of server is reduced. The server will split the data into multiple segments based on their sizes and store them in multiple buckets. These data are also stored in the replica servers as a back-up.

D. Index File Maintenance

The co-efficient values of each batch of data are taken. The batch files and the co-efficient values are stored in a single child. The index information of all the data are maintained in the child, which handles the query from the user. The server uses the index information to easily identify which bucket a file is stored in and retrieve the required data quickly.

E. Data Retrieval

The server retrieves the data the user needs. The retrieval of data is quick because the server knows the data location through the child. This largely reduces the time consumption in the transmission of data. The user first selects the file type, after which the server displays a list of files of that type for the user to select the required file. The decrypted data is received only if the user is authorized, i.e. if the user provides the correct key for decryption. In case of data loss, the server gets the back-up data from the replica servers.

F. RSA Algorithm

The RSA algorithm is used for both public key encryption and digital signatures. It is the most widely used public key encryption algorithm. The basis of the security of the RSA algorithm is that it is mathematically infeasible to factor sufficiently large integers. The RSA algorithm is believed to be secure if its keys have a length of at least 1024-bits.

1) Key Generation Algorithm: The steps for key generation in RSA are given below.

1. Choose two very large random prime integers:
p and q
2. Compute n and φ (n):
$n = pq$ and $\varphi(n) = (p-1)(q-1)$
3. Choose an integer e, $1 < e < \varphi(n)$ such that:
gcd (e, φ (n)) = 1(where gcd means greatest common denominator)
4. Compute d, $1 < d < \varphi(n)$ such that:
$ed \equiv 1 \pmod{\varphi(n)}$

- the public key is (n, e) and the private key is (n, d)
- the values of p, q and φ(n) are private
- e is the public or encryption exponent
- d is the private or decryption exponent

2) Encryption: The cipher text C is found by the equation '$C = M^e \bmod n$' where M is the original message.

3) Decryption: The message M can be found form the cipher text C by the equation '$M = C^d \bmod n$'.

4) Example: This is an extremely simple example and would not be secure using primes so small, normally the primes p and q would be much larger.

1. Select the prime integers p=11, q=3.
2. $n=pq=33$; $\varphi(n)=(p-1)(q-1)=20$
3. Choose e=3
    - Check gcd(3,20)=1
4. Compute d=7
    - $(3)d \equiv 1 \pmod{20}$

Therefore the public key is (n, e) = (33, 3) and the private key is (n, d) = (33, 7).

Now say we wanted to encrypt the message M=7

- $C = M^e \bmod n$
- $C = 7^3 \bmod 33$
- $C = 343 \bmod 33$
- $C = 13$

So now the cipher-text C has been found. The decryption of C is performed as follows.

- $M' = C^d \bmod n$
- $M' = 13^7 \bmod 33$
- $M' = 62{,}748{,}517 \bmod 33$
- $M' = 7$

The message has been encrypted and decrypted the final message M' is the same as the original message M. A more practical way to use the algorithm is to convert the message to hexadecimal and perform the encryption and decryption steps on each octet individually.

IV. EXPERIMENTAL SETUP AND RESULT

A Hard Disk Drive of 80 GB and a RAM memory of 512 MB are used for the implementation. Java JDK 1.7 is used as the front-end and MySQL 5.0 is used as the back-end with Windows 7 operating system.

A. Screenshots

The following screenshots show the sample output for encrypting and bucketizing multiple files and gives an idea on splitting of the multimedia files.

Fig. 1 Public Key on User Registration

Fig. 1 shows a public key provided on user registration. It is used only for encryption and uploading of the file.

Fig. 2 File Upload by Admin

Fig. 2 shows the upload page that enables adding multiple file at once on providing the user's public key. Various text and video files are added to see how they are split into the buckets.

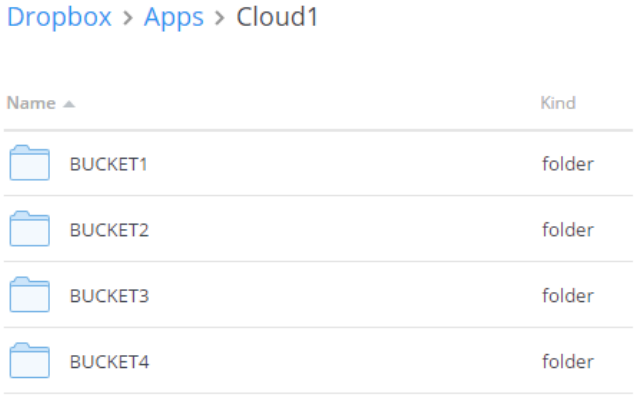


Fig. 3 Buckets in Cloud Storage

Fig. 3 shows four predefined buckets in the cloud that are used in splitting the work of the server.

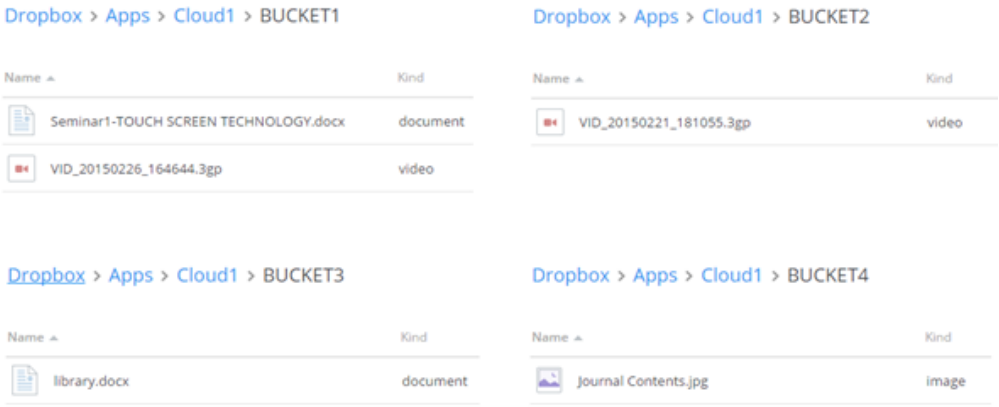


Fig. 4 Data Splitting in Buckets

Fig. 4 shows the multiple files uploaded that are evenly split into the sub-servers or buckets. A bucket can contain both text and video files together.

Fig. 5 Encrypted File

Fig. 5 shows an encrypted text file stored in a bucket. An encrypted video file is made non-viewable.

file_seq	download_url_txt	file_name_txt	file_owner_txt	pri_upload_location_txt	sec_upload_location_txt
1	https://db.tt/VcIZ9r0X	Journal.docx	brinda	BUCKET1	REPLICA1
2	https://db.tt/f4xGW1hH	VID_20150226_164644.3gp	brinda	BUCKET1	REPLICA1
3	https://db.tt/xoxvLcfP	VID_20150312_135810.3gp	brinda	BUCKET1	REPLICA1
4	https://db.tt/vfIZphCo	health.docx	brinda	BUCKET2	REPLICA2
5	https://db.tt/pdeWvWIOf	water pollution.docx	brinda	BUCKET3	REPLICA3
6	https://db.tt/iTyzFLMG	home.docx	brinda	BUCKET4	REPLICA4
7	https://db.tt/kQFr8SxV	Iris 402.docx	brinda	BUCKET1	REPLICA1
8	https://db.tt/4SpIHNts	Brinda Cert'12.jpg	brinda	BUCKET1	REPLICA1
9	https://db.tt/4xP84t9E	Bhobhalan Cert'13.pdf	brinda	BUCKET2	REPLICA2
10	https://db.tt/8iEAYn4d	VID_20150318_130312.3gp	brinda	BUCKET1	REPLICA1
11	https://db.tt/jlzXPyFz	ts documentation.docx	brinda	BUCKET1	REPLICA1
12	https://db.tt/ePwGvICG	evs_notes.docx	brinda	BUCKET2	REPLICA2
13	https://db.tt/Kk1ZXGcD	Arch.jpg	brinda	BUCKET3	REPLICA3
14	https://db.tt/3OLvB5O7	DBMS Lab Record.docx	brinda	BUCKET4	REPLICA4
15	https://db.tt/b4kUj4JT	library.docx	brinda	BUCKET1	REPLICA1

Fig. 6 Index File Maintenance

Fig. 6 shows the index file maintained in the database containing the location details of the bucketized files.

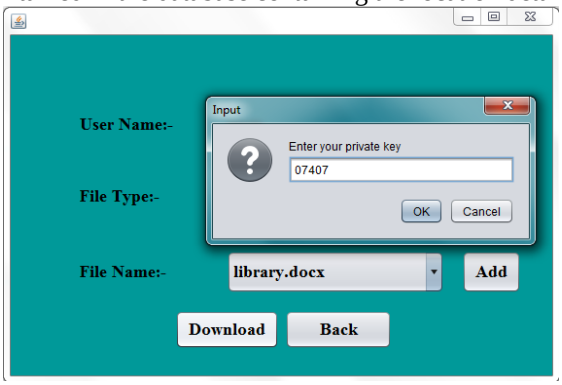


Fig. 7 File Download using Private Key

Fig. 7 shows the cloud server prompting the user for his private key. On receiving the correct key the required files are decrypted and downloaded.

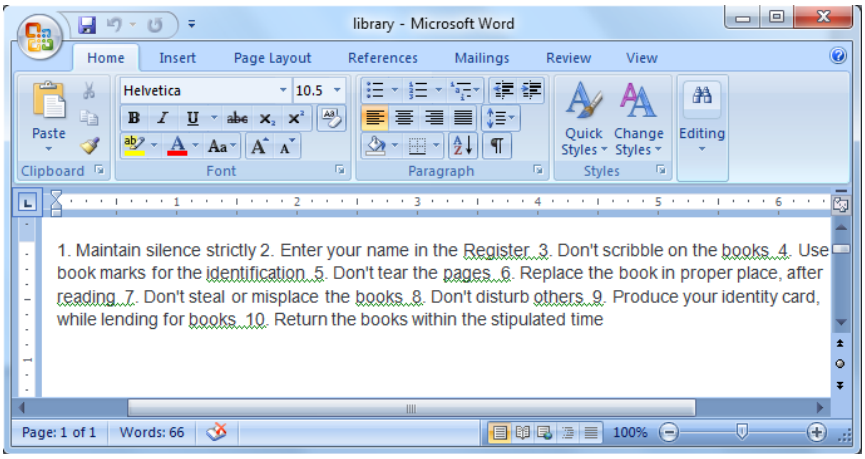


Fig. 8 Decrypted File

Fig. 8 shows the decrypted file with plaintext after downloading from the cloud server.

B. Comparison with Existing System

The bucketization concept that has been used only for plain-text files is modified so that video files are bucketized as well enabling video and text files to be stored in same bucket to prevent adversaries from knowing the file location. The existing system used symmetric cryptography for encryption and decryption, but proposed system uses asymmetric cryptography which does not involve exchange of key for decryption, thus preventing leakage of key.

V. CONCLUSION

This paper introduced the bucketization of multimedia files, especially video and text. The data were evenly split and securely stored in separate buckets with back-ups in the server after encrypting using an asymmetric key algorithm where key exchange is not essential and the keys generated were provided to the user. The data were easily identified and located by the server using their index details stored in the database and the user is authorized before encryption and decryption. Thus an effective and secure way for storage and retrieval of multimedia data is provided.

REFERENCES

[1] Younho Lee , "Secure Ordered Bucketization", *Dependable Secure Computing*, IEEE Transactions on, 2014, pp.292-303

[2] R. A. Popa, F. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proc. IEEE Symp. Security Privacy*, 2013, pp. 463–477.

[3] L. Xiao and I. Yen, "A note for the ideal order-preserving encryption object and generalized order-preserving encryption," *IACR ePrint Archive*, pp. 535–552, 2012.

[4] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu, "Secure multidimensional range queries over outsourced data," *The Very Large Data Bases J.*, vol. 21, pp. 333–358, 2012.

[5] H. Liu, H. Wang, and Y. Chen, "Ensuring data storage security against frequency-based attacks in wireless networks," in *Proc. 6th IEEE Int. Conf. Distrib. Comput. Sensor Syst.*, 2010, pp. 201–215.

[6] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-preserving symmetric encryption," in *Proc. 31st Annu. Int. Conf. Adv. Cryptology*, 2009, vol. 5479, pp. 224–241.

[7] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. 4th Conf. Theory Cryptography*, 2007, pp. 535–554.

[8] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xiu, "Order preserving encryption for numeric data," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2004, pp. 563–574.

[9] B. Hore, S. Mehrotra, and G. Tsudik, "A privacy-preserving index for range queries," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2004, pp. 720–731.

[10] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model" in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2002, pp. 216–227.

