IOSR Journal of Computer Engineering (IOSRJCE)
ISSN : 2278-0661 Volume 1, Issue 2 (May-June 2012), PP 18-23
www.iosrjournals.org

Document Management System with Enhanced Security

Prof M.K Kodmelwar[1], Mayur Agarkar[2], Ajinkya Borle[3], Ashwini Deshmukh[4], Munmun Bhagat[5]
[1,2,3,4,5]*(Comp Dept Sinhgad College of Engineering, University of Pune)*

ABSTRACT : *This paper provides the design of a Document Management System designed for a Small to Medium scale enterprises with a special emphasis on security, it also describes a new symmetric encryption algorithm Secured Quick Crypt and its unique block chaining mechanism. Along with this, various other small enhancements in terms of Compression, Abstraction and File Versioning have also been described in the paper.*

Keywords*— compression; document management system; file versioning; level of abstraction; security*

I. INTRODUCTION

The Enhanced Document Management system is a document management system for small to medium organizations for efficient management of electronic documents in their organization.

Up till now, there has not been a proper solution for managing documents that is affordable for small and medium sized enterprises. Documents are still transferred from one department to another by physical means (such as pen-drives and CDs), or are mailed via some web services (Email, file sharing solutions etc). Even if the organization has a file-server and a LAN setup, it still is not the most efficient way to manage the documents.

Thus, there arises a need for a "Document Management System" [1] for effective document management for various enterprises and government agencies. Various software solutions like documentum (from EMC Corporation) and sharepoint (from Microsoft) came into existence to solve this need.

However these existing systems are all web based and don't offer many benefits in terms of security, abstraction (level), compression [2, 3, 4], and affordability for small and medium scale organization.

Thus, in our paper, we are designing and giving a layout of an efficient Document Management System as solution that addresses the needs of the organizations in today's competitive world in a cost effective manner.

II. ARCHITECHTURE

In the following Architecture diagram we show how client machines are connected to the central server in order to perform different functionality for "EDMS" (Enhanced Document Management System) .The clients machine may be connected through wireless connection or wired connection to the server .The central server contain different module such as web server, database, third party server and secondary storage.

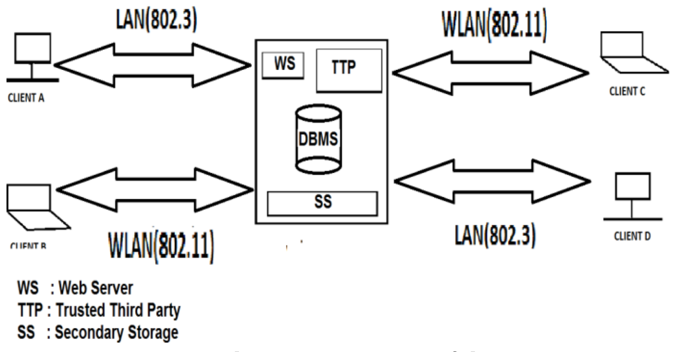

Fig A.1 Architecture Diagram of the EDMS

Although a simple Client/Server Model has been followed for the architecture, some of the features that are provided to enhance its capability will make it a unique and affordable system for the SME enterprises

III. SPECIAL SYSTEM FEATURES

The core task of any Document Management System is the storage and retrieval of documents from one or various sources in a centralized manner. The EDMS improves on this basic model by providing certain other features that are very important in today's world.

A. Encrypted Data Transfer

This feature allows the client machines to send their documents over to the main server for storage purposes in the organization. However, before the file can be transferred it must be encrypted for security purposes [5]. This is one of the most crucial functions of the system.

The technique of Digital enveloping is used for this feature. The Document(s) chosen for the transfer are themselves encrypted using a symmetric encryption algorithm which we have developed ourselves for Quick and secure encryption. The symmetric key itself shall then be encrypted by RSA encryption technique [6, 7] which is an asymmetric key algorithm.
We shall elaborate on this below.

Our Secured Quick Crypt Technique is based on the same basic principle of the AES technique. We are performing Block Ciphering; Each Block of Plaintext as well as that of the Cipher Key is 128 bits.

16 characters from the file (which is to be encrypted) are taken in a block and are converted into their respective ASCII representation. It is then transformed into the Cipher text by:

1. XORing each cell of the Block with the respective cell of the cipher key.
2. Perform the Sub-bytes operations on each cell as done in AES.
3. Shift the rows; Left Rotate 2nd row once, Left Rotate 3rd row twice and Left Rotate 4th row thrice.
4. Perform a simple matrix multiplication between Rijndael mix column matrix and the current block.
5. Generate the round key (directions given below) and XOR with each cell of the block.

After performing these 5 Steps, we receive a block of the cipher text (in Unicode). Only one round is applied for each block.

We are also providing a chaining mechanism by sending the current round key to be the cipher key of the next round.

The method to generate the round key is as follows:
1. Swap the first and last columns of the cipher key.
2. Right rotate 2nd row of the cipher key once.
3. Left rotate 3rd row of the cipher key twice.
4. XOR First and last rows of the cipher key.

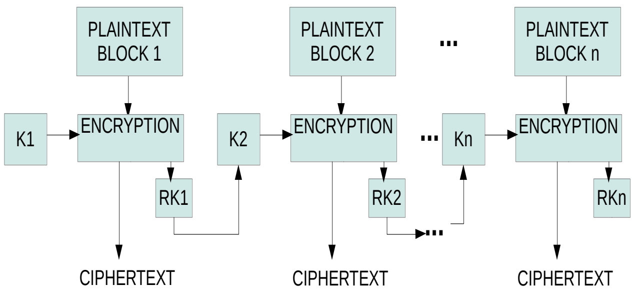

Fig A.3 Block Diagram of the Secured Quick Crypt Encryption

As seen in Figure 3, the mode of operation is slightly different from the usual modes. Since we have to compensate for the less number of rounds that each block of plaintext undergoes, we are sending the round key of the last round to be the cipher key of the next round. Thus a level of pseudo-randomness is generated amongst the entire text. A Crypt analyst would find different Cipher text for the same plaintext repeated in the course of document [8].

Another major advantage of have this unique mode of cipher is because unlike Code Block Chaining (CBC), we can parallelize the operation of encrypting or decrypting the file while keeping the pseudo randomness intact.

Let us now see how it would benefit us in the chaining operation. We shall consider the overview of the entire encryption process:

Speedup(n) = TS/TP

Where TS : Time required by doing encryption serially
TP : Time required by doing encryption in parallel

Now,
TP = TRk(n) + TB(n) + TC(n-1)

Where TRk(n) : Time for Round key generation for n blocks
TB(n) : Time for Encryption for n blocks
TC(n-1) : Time for Concatenation of n blocks, i.e. n-1 concatenation operations.

The generation of the round key cannot be parallelized since the next round key is entirely dependent on the previous round key. But it is independent of the actual encryption operation, so it can be done separately. After we have prepared all the round keys required for the all the blocks of the plaintext. We can send those blocks for their actual encryption in parallel.

Thus, if we theoretically have p processors encrypting n blocks simultaneously we'll have the time reduced to:
TP = TB(n/p) + TC(n-1) + TRk(n) for n mod p = 0
Or,
TP = TB(n/p + 1) + TC(n-1) + TRk(n) for n mod p $\neq$ 0
Also note that n/p is an integer division where the result is the quotient only.
Thus total Speedup S(n) is :
S(n) = TS(n) / TRk(n) + TB(n/p) + TC(n-1) for n mod p = 0
Or,
S(n) = TS(n) / TRk(n) +TB(n/p+1) + TC(n-1) for n mod p = 0

Thus, we are limited in our efforts to parallelize the encryption process due to Amdahl's Law of Speedup, i.e. it's the sequential part of our mode of operation which is preventing it from being completely parallel.

The same calculations apply for decryption as well, instead of encryption there shall be decryption of the blocks.

Thus our Enhanced Block Chaining is aimed at getting the advantages of Code Block Chaining and Electronic Code Book as well.

Given below is a table that compares our Secured Quick Crypt Technique with the standard AES encryption technique.

TABLE I
COMPARISION BETWEEN SQC AND AES

Advanced Encryption Standard	**Secured Quick Crypt**
Requires 10 Rounds per Block : Slower	Requires 1 Round per Block : Faster
Converts hexadecimal values (0-FF) to different hexadecimal values(0-FF) : no diffusion	Converts hexadecimal values (0-FF) to Unicode values in range (0-FFFF) : diffusion
Loop unrolling not present : no CPU optimization	Loop unrolling is present : CPU cycle optimization
Complicated mix column step	Simpler mix column step
Normal mode of operation :	Normal mode of operation :

ECB (no block chaining)	unique block chaining
Round key generation using universal RCON	Original method to generate round key

According to the comparison, Secured Quick Crypt works more effectively than the Advanced Encryption Standard in terms of the diversity that the output gets, thus we see the avalanche effect.

The table below is an example of the encryption outputs of Secured Quick Crypt and AES for the same key : 880079511D768FA81C3D93901BDFB3C8.

TABLE II
ENCRYPTION EXAMPLE OF SQC

Advanced Encryption Standard	**Secured Quick Crypt**
Input Word : IEEE	Input Word : IEEE
U2FsdGVkX19S7Zr8aDuYQlhWx+ohNPPvCaqjS2vjlsc=	ƻʸɦńƑŷńLjﬁŋyϰ̋ȕũɰ

B. Data (file) Compression

This feature shall compress the size of the encrypted file which resides over on the server. The encrypted file has a size that is larger than the plaintext file. Thus compression shall allow us to optimize the size of the encrypted document.

The system uses the gzip compression technique as it gives us a fair ratio of the size of the data (file) and the time required to compress it when compared with other popular compression techniques.

C. Level of Abstraction

This feature ensures that the document access is restricted to the authorized department (even within the organization) and for the authorized employee only.

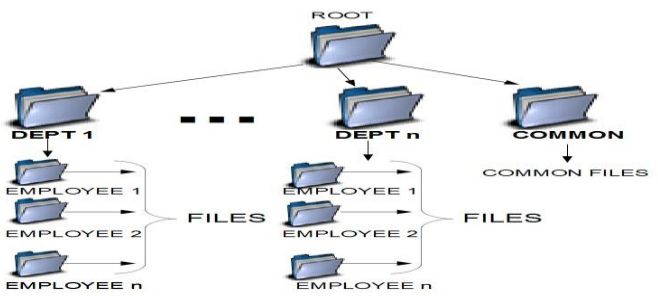

Fig A.4 Hierarchy of the Directories used for storing files on server

Fig.4 shows the fundamental structure of directories for the central server used for the purpose of file storage.

An Employee working on a client machine will store and retrieve documents within his/her own directory stored inside the department. The system allows an employee access to only his/her files and not of any other employee. Also files from one particular department will not be authorized for employees of the other department. Documents that are common to all departments should be put in a common directory to which every employee has access.

For E.g.: A Marketing employee cannot access files from the Engineering Department of an organisations. However files containing public information must be put in the common directory folder, so that all the employees from all the departments can have access of it.

D. File Versioning

This feature of the system keeps the track of the versions of a particular document over time[9][10]. Employees working on a particular document over a period of days may want to revert to a previous version of the same file. For this the server will keeps multiple copies of the same file on it.

Fig.5 shows the directory structure which provides the file versioning capability. Every Document has a hidden folder (in the same parent directory) of the same name. Whenever the user posts a fresh version of a particular file, the system not only replaces the old file but also places a copy of the file in the hidden folder associated with that particular file. The files inside the Hidden Folder maintain the versions of the file with the date of posting concatenated with the actual name of the file.

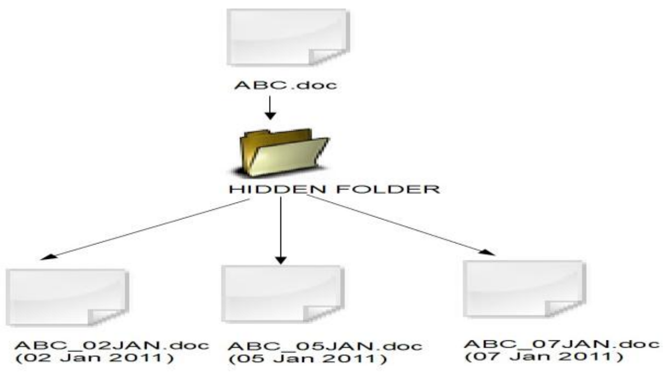
ABC.doc
HIDDEN FOLDER
ABC_02JAN.doc (02 Jan 2011)
ABC_05JAN.doc (05 Jan 2011)
ABC_07JAN.doc (07 Jan 2011)

Fig A.5 Directory structure for File Versioning

E. Searching Technique

Every good Document Management System must provide a search feature that enables its users to quickly find a particular file kept in the system. So a general approach would be to keep all the file information (file name, location, user,etc) in a table inside the backend DBMS, whenever a user will search for a file, its entry would be checked in that table. And the file would be retrieved for the user, depending on relevance to the user that asked for it. This approach has a drawback; as the number of files in the system keeps on increasing, the time taken to fetch a file increases as well. We have a simple and cost effective solution for the above shortcoming.

In EDMS, we are using a searching technique whose idea is taken from hash partitioning. The goal is to increase the speed of searching a file kept centrally.

In general circumstances, there will be 36 tables from a-z/A-Z and 0-9 which will store the file(s) according to their initial alphabet or number into appropriate table. E.g. a file named 'sales_q4' will be saved in table 'S'. Whenever user will enter the file name for searching, the system will check the first character of file name with table and the list of files starting from that character will be displayed. System will go on matching the file name with existing file character by character within the table.

To simulate the efficiency of this technique, we have taken a list of 2468 entries in a single table, and also created three different tables containing entries from the original table starting with 'A', 'M' and 'Y' respectively. We shall now compare the time taken to execute the query for search on the big single table and in the separate tables.

TABLE III
COMPARISON OF SEARCHING TECHNIQUES

Entry searched	Single Table	Separate Tables
Abhimani	0.0022 seconds	0.0013 seconds
Martand	0.0018 seconds	0.0009 seconds
Yoganand	0.0015 seconds	0.0007 seconds

From the table, you can see that even the worst case (entry beginning with A) is 1.69 times faster than search done on a single table. The entry beginning with M is searched twice as fast as in the single table (average case). And finally, the entry beginning with Y is searched 2.14 times faster (best case).

IV. GENERAL SYSTEM REQUIREMENTS

The requirements for the system were based on the findings of the literature review done as well as from the interview sessions done. After a careful analysis of the data collected, the findings of the analysis is used to derive the following application requirements:

1. The Client Application must be a desktop application that must have web browser integration.
2. The Server Application would be a combination of a DBMS, a webserver and a third party server (to generate RSA keys).
3. The Client and the Server Machines must be connected via the 802.3 or 802.11 standards.

V. CONCLUSIONS

In this paper, the authors wish to design and implement a document management system for a small to medium scale organization. This paper acknowledges the fact that merely having a simple storage/retrieval system is not enough, and hence we have stressed upon some features to enhance the basic DMS that are very useful in terms of security, optimal disk usage, level of abstraction and productivity of the employees within an organization.

In summary we provide a collection of features that enhance the traditional document management systems in such a way that is relevant to today's organizational needs.

ACKNOWLEDGMENT

It is with great pleasure that we are presenting this paper. This paper aims to explain such a system, along with its goal, in a simple language. Firstly, we would like to thank all people who have contributed to this effort. We are deeply indebted to our Principal Dr. S.D.Lokhande for providing us all the required facilities. We are thankful to our Head of Department Prof.P.R.Futane for his support. We owe our sincere gratitude to our project guide Prof.M.K.Kodmelwar, for his valuable Guidance and co-operation. He has been of great help with his valuable and timely inputs to the Project .Last but not the least we would like to thank all the staff members from department and my colleagues for their indispensable help all time.

REFERENCES

[1] HEDIYEH BABAN,SALIMAH MOKHTAR, "ONLINE DOCUMENT MANAGEMENT SYSTEM FOR ACADEMIC INSTITUTES", 28 NOV. 2010, VOLUME : 4,315-319

[2] Konecki, M.; Kudelic, R.; Lovrencic, A., "Efficiency of lossless data compression",23-27 May 2011, 810-815

[3] Zia, Z.K.; Rahman, D.F.; Rahman, C.M., "Two-Level Dictionary-Based Text Compression Scheme", 24-27 Dec. 2008, 13 – 18

[4] Jonghyun Lee, Marianne Winslett, Xiaosong Ma, Shengke Yu, "Enhancing Data Migration Performance via Parallel Data Compression", 2002, 444 – 451

[5] Xin Zhou ; Xiaofei Tang, "Research and implementation of RSA algorithm for encryption and decryption", 22-24 Aug. 2011, Volume : 2, 1118 – 1121

[6] Bhargav Balakrishnan, "Three Tier Encryption Algorithm for Secured File",19-21 Mar 2010, Volume : 2, 259 – 263

[7] Gang Hu, "Study of File Encryption and Decryption System using Security Key",16-18 April 2010, Volume : 7, V7-121 - V7-124

[8] Hiroki Endo, Yoshihiro Kawahara, and Tohru Asami, "A Self-Encryption Based Private Storage System Over P2P Distribution File Sharing Infrastrure", 12-13 May 2008, 69 – 76

[9] Xiang Xiao-Jia; Shu Ji-Wu; Xue Wei; Zheng Wei-Min, "Design and Implementation of an Efficient Multi-version File System", 29-31 July 2007, 277 – 278

[10] Maohua Lu; Chiueh, T., "File Versioning for Block-Level Continuous Data Protection", 22-26 June 2009, 327 - 334