Computer Technology and Application 2 (2011) 730-735

Multi-Level Cryptographic Functions for the Functionalities of Open Database System

Akinwale Taofiki Adio, Adekoya Felix Adebayo and Ooju Emmanuel Oluwafemi
Department of Computer Science, University of Agriculture, Abeokuta, Nigeria

Received: July 26, 2011 / Accepted: August 08, 2011 / Published: September 25, 2011.

Abstract: The purpose of this paper is to design and implement a secure open database system for organizations that are increasingly opened up their information for easy access by different users. The work proposed some functionalities such as open password entry with active boxes, combined encryption methods and agent that can be incorporated into an open database system. It designed and implemented an algorithm that would not allow users to have free access into open database system. A user entering his password only needs to carefully study the sequence of codes and active boxes that describe his password and then enter these codes in place of his active boxes. The approach does not require the input code to be hidden from anyone or converted to place holder characters for security reasons. Integrating this scheme into an open database system is viable in practice in term of easy use and will improve security level of information.

Key words: Database system, cryptography, agent, random number, peer-to-peer, algorithm.

1. Introduction

An open system does not comprise of central administration components like intranet and internet. It is a system that regularly exchanges feedback with its external environment (i.e., the users communicating with the system). Any system termed "open" has its boundaries through which feedback can readily be exchanged and understood porous and security insensitive. In this kind of system, authenticated users must be able to access the system at any time and be able to retrieve information from the database based on a pre-defined authorization, without restriction. Recent research shows that databases in any organization are increasingly opened up to multiplicity of suppliers, customers, partners and employees—a feat that seemed unachievable years back. Many applications and their associated data are now possibly accessed by different users requiring different level of access via different devices and channels.

Corresponding author: Akinwale Taofiki Adio, Ph.D., research field: database system. E-mail: aatakinwale@yahoo.com.

Open network database system entails leaving our database open on the internet, intranet, etc. for users to access, but one question that comes to mind remains. Is the information in this open database fully secured and thoroughly protected from it being compromised and manipulated for other benefits outside the proposed purpose? Is the system user(s) solely in control as regards the issuance of information? If the answer to these questions is no, then what security measure should be put in place and what is its strength? Since the database is left open and proper security is not placed on it and the operations to be performed on it, how secure and protected is the system?

Having known that open system allows for free access to database, bearing in mind its integrity preservation, the functionality of this database system must incorporate the capability of management security and prevent it from unauthorized modification or extraction. All these form the basis for the research aimed at actualizing a conceptualized security measure that can be integrated into such database systems. The need to secure and protect relational database system

for open system, say, Peer-to-Peer, is growing beyond imagination. While the importance of security and privacy is generally taken strict, it is still largely ignored as a result of the problem being considered impractical to solve, given the limited computational resources available at user level. The crucial aim of this paper is to design an application to simulate how open database network system can be secured and prevented from been compromised, using a multi-level hash function. The remaining part of the paper is structured as follows: Section 2 presents literature review on open database system. In section 3, the methodology of the system is presented while implementation environment is also described in section 4. Evaluation of the system is in section 5 and conclusion is presented in section 6.

2. Literature Review

Security and functionality of applications and computer system are critical aspect of any information technology strategy and must be taken very serious with all measures. Beyond ensuring the security of the host operating system, a database administrator needs to ensure that the database management software itself is secure, and that it provides the full complement of features to enable the security [1]. Database usually contains an organization's (e.g., banks, schools, companies, etc.) most valuable information assets, and if compromised, could cause disaster [2]. Hence securing a database allows organization to protect the corporate data from threats and external sources. Database security is a serious issue, and if not implemented correctly, the consequence can be costly to organizations if their vital data is hacked into, or their clients' data leaks out, which can even lead to cases of identity theft [3].

Data accessibility is a major concern in database security. Many organizations cannot work properly if their databases are down: They are what we know as critical-mission systems. To make the data available implies to provide the security mechanisms to ensure authentication, authorization and auditing procedures [4]. According to Ref. [5], securing database entails:

(1) Preventing unauthorized access to classified data by just anybody;

(2) Preventing unauthorized users from committing mischief through malicious deletion, update or tampering of data;

(3) Monitoring user access of data through auditing techniques.

One of the most basic concepts in database security is authentication, which is quite a simple process by which a system verifies a user's identity. According to Ref. [6], the first step in encryption project is to determine what data to protect and from whom to protect it. The sensitivity of data will logically determine the need for the use of encryption. There are things to consider when thinking of implementing encryption:

- Will the data stored in the database need be encrypted or just the user password?
- Will you need to encrypt the data only in the local instance of the database or do you need to also encrypt in transit? [7]

All users and systems create traces of their activity in the form of log files. Logs are generated at an astounding rate by information technology components such as firewalls, routers, server and client operating systems, databases, and even business applications as usefulness for detecting and troubleshooting security and system operations issues. Dynamically monitoring log data will help protect businesses not only from external security threats but also from potential threats lingering inside the organization [8].

3. Methodology

Multi-level cryptographic hash functions such as Rivest, Shamir and Adleman (RSA) algorithm and Secure Hash Algorithm (SHA-512) were employed to encrypt not only the password and user identity but also all the information about every peer in the database including the message that can be shared within the members if necessary. The essence of this is to prevent

any fraudulent member who might have access to the database server from retrieving others' security accounts, thereby hacking into others' personal information, editing details, backdating events and the likes.

3.1 SHA-512 Algorithms

SHA-512 can be used to hash a message, M, having a length of N bits, where $0 \leq N < 2^{128}$. The algorithm uses a message schedule of eight 64-bit words and eight working variables of 64 bits each together with a hash value of eight 64-bit words. The final result of SHA-512 is a 512-bit message digest. The words of the message schedule are labeled $W_0, W_1, \ldots, W_{79}$. The eight working variables are labeled a, b, c, d, e, f, g and h. The words of the hash value are labeled $H^{(i)}, H_1^{(i)}, \ldots, H_7^{(i)}$, which will hold the initial hash value, $H^{(0)}$, replaced by each successive intermediate hash value (after each message block is processed), $H^{(i)}$, and ending with the final hash value, $H^{(N)}$. SHA-512 also uses two temporary words, T_1 and T_2 [9]. The logic in each of the 80 rounds of the processing of one 512-bit block is defined by the following set of equations:

$T_1 = h + \text{Ch}(e, f, g) + (\sum_1^{512} e) + W_t + K_t$

$T_2 = (\sum_0^{512} a) + \text{Maj}(a, b, c)$

$a = T_1 + T_2; b = a; c = b; d = c; e = d + T_1; f = e; g = f; h = g.$

3.2 RSA Algorithm

The RSA Scheme is a block cipher in which makes use of an expression with exponentials. The plaintext is encrypted in blocks, with each block having a binary value less than some number n, most preferably 1,024. That is, the block size must be less than or equal to $\log_2(n)$. In practice, the block size is i bits, where $2^i < n <= 2^{i+1}$. Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C:

$C = M^e \bmod n$

$M = C^d \bmod n = (M^e)^d \bmod n$

Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d. Thus, this is a public key (PU algorithm with PU = $\{e,n\}$ and a primary key (PR) = $\{d,n\}$ [10].

Further research into the security strength of open database management system shows that leaving information shared among users open may later pose security threats despite the secured open password applied, and therefore, we thought it well a good research to fortify the security level of the most recent of the models of the open database in peer-to-peer by employing the algorithms of RSA and SHA-512. Consequent upon this, all the information about all users or peers is being double encrypted, first by RSA, then by SHA-512 before they are sent to the database. This ensures that even if the system is hacked, it will be very difficult for the hacker to record a successful cryptanalysis and understand what is kept in the database. Each level of encryption/decryption requires different keys. These keys are given to the users at the point of registration made available whenever it is required or needed.

3.3 Open Login Details for Open Database System

Having known that access to the database system in question is thrown open, a means of ensuring that the system is not misused is to apply:

An open system where all alphanumeric keys of the keyboard can be seen on the screen by everyone who tries to register for access to the system. This is to ensure that users do not have access to the system in a simple way. It would therefore be assumed that anybody that can go through the rigour of registering possesses level of trust to use the system. Detail input segmentation where the users will have to calculate which of the input field he will have to input the details string given him in the above process.

3.4 Algorithm for Details Input Segmentation

Step 1: Let the any of the Details (i.e., User Name, Peer Id and Password) = A; where A is a string with a minimum of eight characters.

Step 2: Let the Randomized generated Details = B.

Step 3: Let Details integer = C; where C ∈ segments 1 to 25.

Step 4: The segment C is randomly generated that is at each key stroke, it is sure of having eight (8) even and four (4) old segment and vice versa.

Step 5: Repeat if Details generated is odd, input it into C odd segment and if details generated is even, input it into C even segment.

Step 6: Repeat two (2) times minimum, else put the details in an available segment.

3.5 Creation of Register Database

The register database which is the peer-to-peer information table of the database contains the information of any peer that has interest in using the system. The first time user is required to fill the necessary information in the registration form. After registration, when the peers demand for access to be granted to the system they will be asked to provide some vital information in a given form provided. The system then compares the information in the register database with the newly provided one, if it matches, the access is granted else access is denied.

3.6 Creation of Log File

In this system, the log in time, log out time, and time used by different users are documented in correspondence to their usernames. This helps in tracing, tracking and auditing any suspicious activity or unauthorized action by any user, who logs in within that range of time in question. As soon as a user starts using the system, his/her username, the system present time, the system time and the computer system media access control address number when he/she logged out is recorded in the log file. During the tracking, any user found responsible, will be barred from using the system by granting a denial of service to such user.

3.7 An Agent

An agent in this system is an object program that decides who should be registered as a user. The agent is an intermediary between the user and the database system and between users to users. There are few functions assign to the agent, among which are

(1) It decides if a new user qualified for registration or not;

(2) It checks if any posted data/file is relevant, if not it discard it;

(3) It is the one who makes sure the information are been multi-leveled encrypted; and

(4) It alerts the administration if there is an update request. Figs. 1-2 show other functions to be performed by the agent.

4. Implementation Environment

Java programming language was used for the implementation of RSA, SHA-512 and input segmentation algorithms, creation of agent, log files, register files and open login detail interface. From the algorithm, a form interface was generated which shows

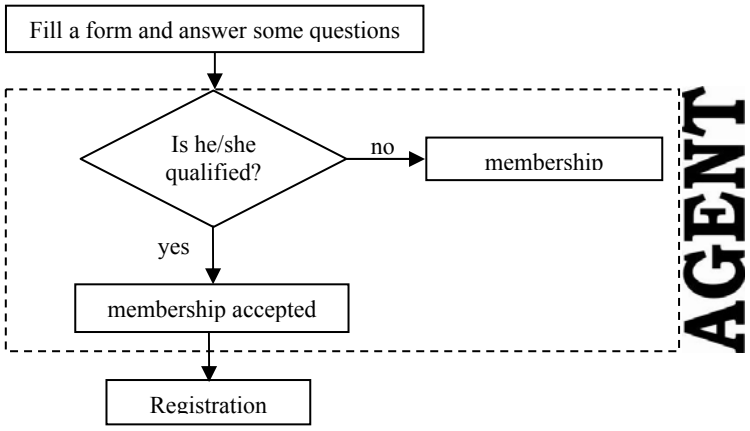

Fig. 1 Registration architecture.

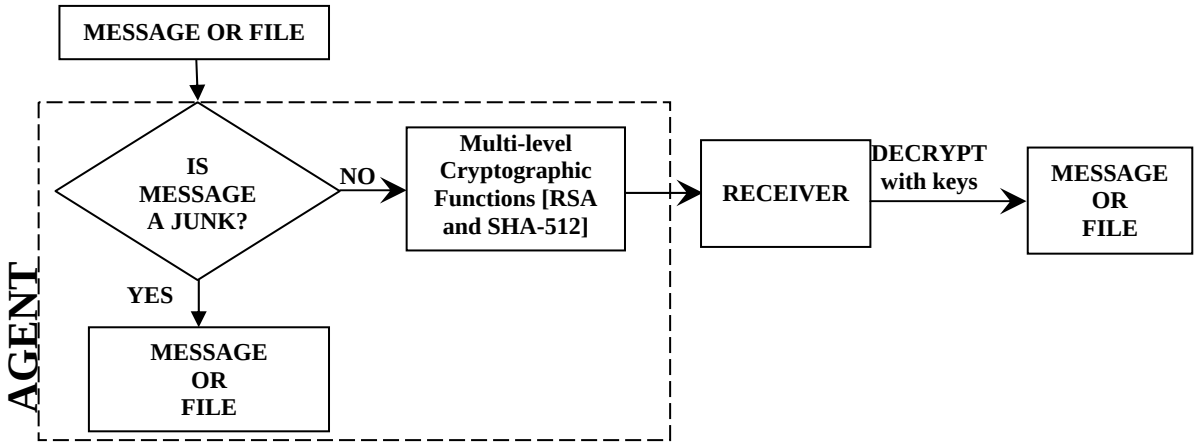

Fig. 2 File/Data transfer architecture.

the password. The interface contains alphanumeric A-Z, a-z, 0-9 and special characters (&, %, $ @, …) each of which is labeled with a single numeric digit code between 0 and 9. The single digit labels are generated randomly and are equally distributed. Any user must study the interface form carefully because of its openness and then enter the numeric digits corresponding to his actual details. Upon completion of the input the user hits the enter key. This triggers the algorithm to regenerate a new input form followed by authentication of the last input details.

The input interface form regeneration is necessary to harden the details entry system and make it extremely difficult, if not impossible for attackers to spy. For example, there are twelve boxes for both peer_Id and password where five of the input fields are dormant while the remaining seven are active for peer_Id. Also, there are twenty-five boxes for the User Name where nine of the input fields are dormant while the remaining sixteen are active. The position of these fields is not fixed. Inputting into these fields requires little calculation, that is, if the first number given to you by the above process is even, then check the first available even active box and input it and vice versa if it is odd. Repeat this process until the given values for the details are exhausted. Sample of the input form is shown in Fig. 3. The agent performs multilevel encryption form for storage as shown in Fig. 4 and all other functions as illustrated in Figs. 1-2.

5. Evaluation of the System

This open system security proposed and implemented can be evaluated and graded by its security potentials against possible attacks or threats. These security parameters are hereby looked into. According to Ref. [11], if the size of a hash value is n, the brute force attack on that hash will require $2^{n/2}$ attempts to yield a possible collision. Therefore, since the size of our hash value is 512, a brute force attack on it will require 2^{256} attempts to yield a collision. From the birthday paradox in probability [12], which questions that "given a hash function H, with n possible outputs and a specific value

Fig. 3 Input interface for login.

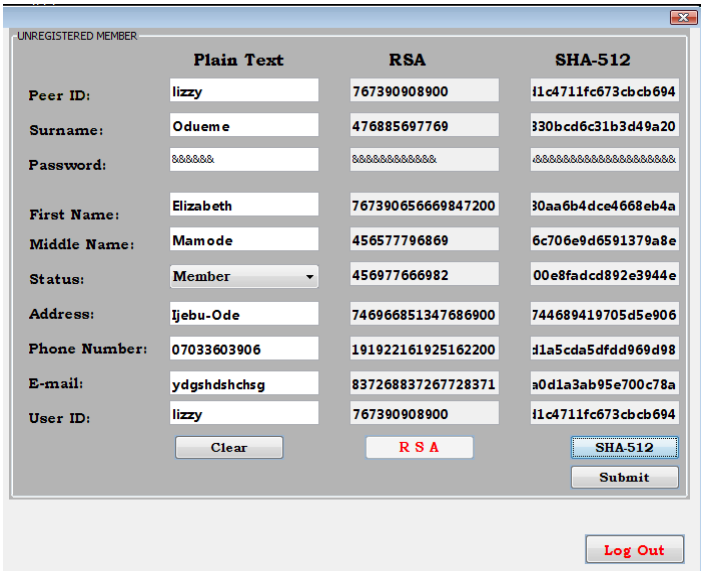

Fig. 4 Encryption of user details using RSA and SHA-512.

$H(x)$, if H is applied to k random inputs, what must be the value of k so that the probability that at least one input y satisfies $H(x) = H(y)$ is 0.5? The solution to this problem is solved to be $k = 2^{m-1}$ for an m-bit hash code with 2^m possible code. Hence, this attack requires 2^{511} possibilities to yield a collision.

For a large n with large prime factors, factoring will be hard. This keeps the RSA safe today until a better provable research can break this resistance.

6. Conclusions

For any information stored in a database to be security guaranteed, if the database will be left under an open architecture, functionalities like the creation of register database, application of multi-level cryptographic functions, application of activity manager and agents should be properly integrated. These features must be regularly checked to ensure efficiency, and new techniques need to be discovered and implemented where these features seem to be failing because as technology advances loopholes are discovered and fixed. Integrating these ideas into open database system in network environment will definitely guarantee a high degree of safety and functionalities of the system.

References

[1] Red Hat Database: Open Source Database and Security, Red Hat Inc., United States, 2002.
[2] W. Duane, Making Your Network Safe for Database, International Business Machine Corporation, 2001.
[3] B.S. Akshava, Database Security, Information Technology Toolbox, Toolbox Co. LLC, USA, 2008.
[4] A.T. Joaquin, Database Security in High Risk Environment, International Business Machine Corporation, USA, 2001.
[5] P. Zikopoulous, The Database Security Blanket, International Business Machine Corporation, USA, 2001.
[6] R. Mogul, The Ins and Outs of Database Encryption, Tech Target Security Media, 2008.
[7] K. Westphal, Secure Mysql Database Design Security Focus, available online at: http://www.securityfocus.com, 2003.
[8] A. Chuvakin, Network database and system log management: the what, why and how, Computer Technology Review, USA, 2008.
[9] FIPS, PUB 180-4, Secure Hash Standards, Federal Information Processing Standards Publication, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8900, 2008.
[10] R.S. Rivest, A. Shamir, L.M. Adleman, On digital signatures and public key cryptosystems, MIT Laboratory for Computer Science, Technical Report, IT/LCS/TR-212, 1979.
[11] W. Stallings, Cryptography and Network Security, Revised Edition, Englewood Cliffs N.J., Prentice Hall Inc., USA, 2005.
[12] K.H. Rosen, Discrete Mathematics and Its Applications, 4th ed., McGraw-Hill, 1999.