Image Data Integration for Integrity Detection in Database Systems for M-Learning Applications

MIHAI DOINEA, PAUL POCATILU, CRISTIAN CIUREA
Department of Economic Informatics
Bucharest Academy of Economic Studies
Piata Romana 6, Bucharest
ROMANIA
mihai.doinea@ie.ase.ro, ppaul@ase.ro, cristian.ciurea@ie.ase.ro

Abstract: - The paper presents the framework for an integrity detection model, describing the database elements needed for the implementation. The BMP image format is illustrated and the process of how data is encoded in the image is presented. An integrity detection model is proposed for an m-learning application along with the advantages and disadvantages brought to the system. Conclusions are drawn and future work is presented.

Key-Words: - Security, Digital Image, Data Integrity, Optimization, Steganography, M-learning application.

1 Introduction

The objective of this paper is to present the ongoing research about one of the data characteristics which helps in conserving their initial quality, meaning and scope. An algorithm for detecting integrity changes is presented based on a data set stored in a database table and on a watermarking procedure for encrypting a unique identifier into a digital content. The scenario for which this algorithm is developed and tested is based on a web based distributed system that has implemented in the background a database storing a table with binary large objects fields.

M-learning applications usually, based on their nature of activity, use extensively multimedia content. For this reason the use of this combination between watermarking and digital content, aka images, videos, sounds and other forms of it, is justified and not very expensive because of the use of redundant information found at this level.

The paper describes the security issues that appear in this kind of systems, the database structure used to implement the algorithm, the structure of a BMP image format, the algorithm used to hide the information and the actual algorithm for detecting changes that affect the integrity of data stored in the database and also the integrity of the multimedia content by means of using a fragile steganography system. Steganography is an applied form of watermarking used to embed messages into digital content without making the user aware of its presence. A form of fragile steganography is used to prevent that any modifications of the digital content in which the message is embedded not pass undetected.

2 Security issues of M-learning applications

The security inside mobile learning applications is an important issue that must be analyzed in order to discover possible vulnerabilities or threats and to avoid loss of data.

M-learning applications use sensitive information related to user personal data, payment information, tests answers, users' results and other.

Security of m-learning applications requires the existence of the followings characteristics [1]:

- *confidentiality*, which means protecting data leaking to unauthorized parties, like personal identification data, credit card information;
- *integrity*, that suppose avoiding data corruption;
- *availability*, which means to ensure that data and applications are always available to authorized entities with no interferences.

Along with the increase of network technologies, the security problems of mobile learning applications are becoming more complex. In order to guarantee the availability of resources offered by an application, the concept of *intrusion detection* is proposed. An intrusion detection system deals with intrusive data collection process, data analysis and response. Such kind of system uses the mobile agent's characteristic of small code size. The real-time and scalability of the intrusion detection

are advantages that allow to use it in solving security issues of m-learning applications. At the same time, the intrusion detection system has some learning capabilities and can dynamically auto-adapt to new environment characteristics [2].

As more and more sensitive data is processed by m-learning services, in [3] is proposed a solution that will secure the Short Messages Service (SMS) content having minimal impact on the device performance. The solution implements a symmetric encryption scheme being used by a distributed m-learning architecture.

3 Database structure used for the integrity algorithm

In [4] is defined the relational model as a model based on a mathematical set theory. A ***relational database*** is a database that conforms to the relational model. A ***relation*** is described as a set of tuples, called table. A ***tuple*** is an unordered set of ***attribute*** values. Having this, a table is described as a two-dimensional representation of a relation in the form of *rows* or *tuples* and *columns* or *attributes*. Each row in a table has the same set of columns. A relational database is a database that stores data in relations, figure 1.

Fig. 1 – Relational database components

Let *CC* be a collectivity, with $CC = \{cc_1, cc_2, ..., cc_n\}$, *n* - number of elements of a collectivity *CC*, stored in a database and set of characteristics described by $CA = \{ca_1, ca_2, ..., ca_m\}$, *m* - number of attributes for the relation *CC*, like IDMEAL, DESCRIPTION, NOKCAL, NOPROTEINS, NOLIPIDS, NOCARBO, NOFIBERS, TYPE, IMAGE, ALLERGY.

Each element of the relation and the set of attributes is represented in Table 1, where cc_i is represented by *m* values associated to each characteristic who describe it, $cc_i = \{c_{i1}, c_{i2}, ..., c_{im}\}$.

Table 1 – Relational model of a table

	ca_1	ca_2	…	ca_j	…	ca_m
cc_1	c_{11}	c_{12}	…	c_{1j}	…	c_{1m}
cc_2	c_{21}	c_{22}	…	c_{2j}	…	c_{2m}
…	…	…	…	…	…	…
cc_i	c_{i1}	c_{i2}	…	c_{ij}	…	c_{im}
…	…	…	…	…	…	…
cc_n	c_{n1}	c_{n2}	…	c_{nj}	…	c_{nm}

A set of database vulnerabilities together with a list of solutions to be implemented are depicted in Table 2:

Table 2 – Database security analysis

Database vulnerabilities	Vulnerabilities measures
secrecy reveal described as any database object who has read access and are retrieved by unwanted users who have no right to do so;	*cryptographic controls* meant to hide sensitive information from unwanted retrievals;
integrity violation presented as any database object which is modified without having the permission, losing the consistency of it and possible its relations with others;	*access controls* to assure access by users who are authorized to do so; SQL injection preventions;
availability minimization represents the denial of database services due to different types of hardware or software problems.	*backup procedures* and redundancy control features, like cycle redundancy check meant to restore information in case of damage.

In [5] an automatic process of creating SQL Injections and Cross-Site Scripting Attacks, XSS is described, representing the way through which database instances could be altered without any obvious knowledge.

The SQL injection is a technique of code injection that exploits a security vulnerability occurring in the database layer of an m-learning application. The vulnerability appears when user credentials are either incorrectly filtered for string literal escape characters, which are embedded in the SQL statements, or when user credentials are not strongly typed and thereby unexpectedly executed.

In order to increase the security inside an m-learning application, the protection against SQL injection is realized by minimizing and controlling the input data given by the user at the application interface and by replacing the special characters associated with an SQL statement executed at the request of a user [6]. The following code represents an example of how SQL injections can be avoided when login credentials are given for accessing the applications resources.

```
string userName =
TextBox1.Text.ToLower().Replace("'",
"''");
string passWord =
TextBox2.Text.ToLower().Replace("'",
"'");
```

As no user has writing privileges on the database, the database is considered to be static, the problem of checking the consistency regarding any undesired modification on the database is considered just for threats that are intentionally triggered to alter inside database information by different means.

4 Image representation

Based on the reasons presented in previous sections, an algorithm is presented in order to cope with these kinds of troubles. The algorithm is meant to check the database consistency at regular bases, must be implemented as an external process that is scheduled to run independently without interrupting the functionality of the database or the data availability.

For understanding the way that application encrypts a message we first must understand the structure of a bitmap file which consists in a 3 or 4 parts depending of the way that color information is presented:

- **BITMAPHEADER** contains various information about the header of the file such as: the signature of the file which is BM; file size, a reserved 4 bytes zone and the offset to which image representation begins stored on 4 bytes; bitmap header size is 14 bytes;
- **BITMAPINFOHEADER** with a total of 40 bytes used presents the size of the info header zone, value 28 in hexadecimal; image dimensions represented by height and width; number of planes; the color depth given by the number of bits per pixel; compression, if there is any; total image size if compression was specified; horizontal and vertical resolution stored on 8 bytes both; number of colors used; the number of important colors;
- **OPTIONAL PALLETE** represented by the color table, is present only if the number of bits per pixel is less or equal to 8;
- **IMAGE DATA** the actual zone where useful information is stored.

Fig. 2 – Hexadecimal view of a bitmap

In figure 2 are presented the first 54 bytes which are selected, meaning the information of both bitmap header and bitmap info header from a bitmap file with the following attributes:

- signature with a length of *2* bytes, value *BM – ASCII* and *4D42h*;
- file size with a length of *4* bytes, value *0010B476h* meaning *1.094.774* bytes;
- *4* bytes reserved;
- data offset, *4* bytes in length, value *36h* meaning that the actual image data will begin after *54* bytes at the end of the selected zone;
- info header size with a length of *4* bytes, value *28h* equivalent of *40* bytes;
- width, *4* bytes length field, value *26Eh* meaning *622* pixels;
- height, *4* bytes length field, value *1B8h* meaning *440* pixels;
- other useful information.

The engraving algorithm uses this image representation for a simplified analysis upon the data that is being changed in the process. The BMP image file format is capable of representing non-compressed images with the pixels written in reverse order on a height * weight matrix.

5 Encoding process

An application of invisible watermarking is steganography, a method through which messages are hidden in multimedia content by modifying redundant information without the possibility of discovering it unless special analyze techniques are used.

The algorithm is used to engrave a control code into a binary large object to check if the content has been altered in previous transactions. Invisible

watermarking algorithms have peculiar characteristics that give the strength of the algorithm:

- the embedded message is difficult to perceive by a human observer;
- the embedded message is difficult to remove; the image may be subject to processing like geometric distortion, resizing, data loss compression, enhancing, adjustments; in such cases, the message should be still recognizable;
- the algorithm is applicable to all multimedia types;
- by decoding the embedded image the owner is identified accurately.

Fig. 3 – Encoding algorithm

The current implementation is an example of fragile watermarking which refers to a message embedded that is easily altered when the image suffers changes, more if the record is modified in an online transaction than the hashed value encrypted inside will not match the one for the modified values. If the decoded message is altered by altering the BLOB field, the recipient becomes aware that the image has been modified from its initial state and it is not the version issued from the authentic sender.

For hiding the hashed value in the binary large object field, a password is needed representing the actual step at which parts of the message will be stored by modifying the redundant pixels found in the image.

For achieving the array of bytes which further will be hidden into the BLOB field, as a unique identifier for all the data which are part of a record, the following code is used.

```
MD5CryptoServiceProvider md5 = new
MD5CryptoServiceProvider();
byte[] hash =
md5.ComputeHash(messageStream);
```

The process of encrypting the message is presented by the following algorithm:

A. calculating the message length;

B. storing the length of the message, only 2^{24} bytes meaning that the message is limited by the maximum value stored in 3 bytes, representing the RGB colors;

C. total number of pixels count, width*height – 1;

D. calculating the X and Y coordinates based on the value of the ASCI code of every byte read from the password code and the dispersion component calculated based on the length of both message and password;

E. writing every byte of the message stream along with a byte of the password code read in reverse order at calculated positions.

The process of hiding the message *HIDDEN MESSAGE* using the password *PASS* in a BLOB field that contains a BMP image of 100x100 pixels will take every byte of the message and write it to the image as follows:

- the dispersion value is 8, calculated for a stream length of 28, 2 bytes for every character due to the encoding restrictions, and a length of the key password of 8 bytes, meaning that every step at which a byte of the message will be written will be multiplied with the value 8; the dispersion value is determined based on the total number of pixels of the image and the number of pixels required by the message in order to write it;
- the first character of the message, H, ASCII code 72, will be written after the first 54 bytes along with S, the first character in reverse order of the key password, at an offset of 80*8, where 80 is the ASCII value of the first character in the key password, P and 8 is the dispersion value with the results in figure 4.

Fig. 4 – First character encryption

The steps of engraving the hash stream into the image field is described above as a function of writing sensitive information at color bit level on the whole surface of the image.

```
private void btnHide_Click(object
sender, System.EventArgs e)
{
  …….…
try
{
SteganoGraphy.HideMessageInBitmap(hash
, bitmap, keyStream,
chkGrayscale.Checked);
picImage.Image = bitmap;
SaveFileTo24RGB(fileName);
 }
```

```
catch (Exception ex)
 {
MessageBox.Show("Exception:\r\n" +
ex.Message);
}
  …. . …
}
```

The total amount of memory used for adding integrity detection codes to each record is strictly dependent upon the number of records and a fixed cost returned by the hash function equal to **16 bytes** per record.

If the encryption process is used upon a 100x100 pixels BMP image format the following results depicted in figure 5 are visible.

Fig. 5 – Encryption results

In the field of cryptography, the MD5 (Message-Digest 5) encryption algorithm is a widely used cryptographic hash function with a 128-bit hash value, but which is not suitable for applications like SSL certificates or digital signatures [6].

In [7] is considered that the security of encryption depends on the decided key length and the algorithm used. The key length is the essential element in choosing the security strength, to preserve the confidentiality of the message and the privacy of sender and recipient.

6 Integrity detection model

The algorithm of engraving a value into a binary large object uses a BLOB field as input and also the hashed value of a string composed by all the characteristics of a table row cc_i concatenated that has a unique value associated to each cc_i with a size of 128 bits presented in the diagram depicted in figure 6.

Fig. 6 – Integrity detection algorithm diagram

The algorithm takes each row of database table as depicted in figure 7 and calculates the hash value for that record.

	IDMEAL	DESCRIPTION	NOKCAL	NOPROTEINS	NOLIPIDS	NOCARBO	NOFIBERS	TYPE	IMAGE
1	123	1123	107,25	9,75	7,5	0,75	0	1	(BLOB)
2	67	167	347,5	123,2	19,265	188,75	26,45	1	(BLOB)
3	64	164	403,75	127,5	7,725	505,5	91,5	1	(BLOB)

Fig. 7 – Row table selection

Based on each row the final binary large object is the result of the process presented in figure 8.

Fig. 8 – The result of the encoding process

The algorithm for hiding row identification streams into a binary large object has no extra cost

because it uses the redundant data found at the digital content level. After each record was processed, any change in its structure is detected by recalculating the hash value of each row and comparing it with the value encoded in the binary large object.

7 Conclusions and Future Work

The algorithm uses the memory that is already assigned for the image field. No extra costs are involved in the process. The integrity checking procedure is made one in a while after a predefined number of processes. The algorithm is dependent on size of the image which must contain the length of the message digest function used. The engraving algorithm used can be refined so that the hidden information wouldn't be detected nor replaced.

The confidentiality of a message can be protected through the encryption process, but in order to protect the integrity and authenticity of the message, other techniques are needed. Through these techniques very important is the verification of a message authentication codes (MAC) or a digital signature.

One future work will include the security of Multimedia Message Service (MMS) in m-learning applications, taking in account the presented results.

Acknowledgement

This work was supported by CNCSIS –UEFISCSU, project number PNII – IDEI 2637/2008, project title: *Project management methodologies for the development of mobile applications in the educational system*.

References:

[1] P. Pocatilu, M. Doinea, C. Ciurea, Development of Distributed Mobile Learning Systems, *The 9th WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing (CSECS '10)*, Vouliagmeni, Athens, Greece, December 29-31, 2010, ISBN 978-960-474-262-2, ISSN 1792-7315, pp. 196-201.

[2] Jianxiao Liu, Lijuan Li, A Distributed Intrusion Detection System Based on Agents, *Pacific-Asia Workshop on Computational Intelligence and Industrial Application, PACIIA '08*, Vol.1, 19-20 Dec. 2008, pp. 553-557.

[3] C. Boja, P. Pocatilu, A. Zamfiroiu, Secure SMS Communications for M-Learning Services, *Proc. of the 5th European conference on European computing conference (ECC '11)*, Paris, France, 2011, pp. 480-484

[4] E.F. Codd, A Relational Model of Data for Large Shared Data Banks, *Communication of the ACM*, Vol. 13, No. 6, pg. 377 – 387, 1970

[5] Kiezun, A., Guo, J.P., Jayaraman, K., Ernst, D. M., Automatic Creation of SQL Injection and Cross-Site Scripting Attacks, *Proceeding of the ICSE'09*, Vancouver, Canada, May, 2009, ISBN: 978-1-4244-3452-7

[6] C. Ciurea, Implementing an Encryption Algorithm in Collaborative Multicash Servicedesk Application, *Open Source Science Journal*, Vol. 2, No. 3, 2010, ISSN 2066–740X, pp. 5-14.

[7] Nor Badrul Anuar, Lai Ngan Kuen, Omar Zakaria, Abdullah Gani, Ainuddin Wahid Abdul Wahab, Usability and Performance of Secure Mobile Messaging: M-PKI, *WSEAS Transactions on Information Science and Applications*, Vol. 6, No. 2, February 2009, ISSN 1790-0832, pp. 179-189.