International Journal of Software Engineering and Computer Systems (IJSECS)
ISSN: 2289-8522, Volume 3, pp. 71-87, February 2017

DOI: http://dx.doi.org/10.15282/ijsecs.3.2017.5.0027

REVIEWING AND APPLYING SECURITY SERVICES WITH NON-ENGLISH LETTER CODING TO SECURE SOFTWARE APPLICATIONS IN LIGHT OF SOFTWARE TRADE-OFFS

Kamel Hussein Rahouma
Department of Electrical Engineering
Faculty of Engineering, Minia University
Minia, Egypt
e-mail: kamel_rahouma@yahoo.com

ABSTRACT

Important software applications need to be secured by choosing the suitable security services. In this paper, a shopper program is designed and implemented using VB.NET to follow up the movement of goods in the store and the shopping place. The program uses five files to store: the passwords, the information of goods in the store and the shopping place. A literature review is carried out to collect the information system and software trade-offs. The applied security services are then analyzed in light of these trade-offs. For security purposes, files and folders are hidden and files are set to Read-Only. The files' contents are encrypted by applying XOR operation with a random key generator. The file's contents are hashed and signed by the user to protect the integrity and authenticity of the files' contents. The applied security services do not result in much complexity and thus it does not affect the system resources. The program usability is easy to handle. The software is not freely available and in case of hunting a copy, it would be hard to run it without disclosing the needed keys. Applying the Arabic letter coding adds more credit to the program usability and availability.

Keywords: Software Protection; Security services; Encryption/decryption; Hashing; Digital signature

INTRODUCTION

Software piracy and tampering is a well-known threat the world is facing. Researchers gave many suggestions, techniques and algorithms to protect software from reverse engineering and tampering. Software developers and crackers are seemingly in a war to get the upper hand over each other. Software industry suffer always from software piracy, especially after the advent of the Internet and the availability of many software analysis and hacking tools (Wang et al., 2001).

Computer software companies' investment of time, money and intellectual capital is demanding to protect their assets of software production. Risk, theft and misuse are main threats to the software once it is produced. Tens of billion dollars are estimated as the loss by the software industry due to software piracy alone. Not only piracy, but also software tampering is a worrisome possibility (Jan M et al., 2007).

Software protection can be done to the code and/or to the executable programs and/or the output results of running software programs. Attacking software programs may be done in the form of modifying or omitting critical checks, such as license

checks, or key checks of functionality. Software developers suggested and employed many approaches to protect software such as: block and multi-block hashing schemes, the widely used hardware based approaches, obfuscation, Guards, cryptographic techniques, watermarking techniques. Information security concerns with protecting information availability, privacy and integrity. Computer database include highly business and individuals information which need to be kept confident, secret and not given for public (Wang, 2005; Koko and Amin, 2015).

In this paper, we are interested in protecting the executable software programs and their output information. This is done by applying cryptographic and hashing techniques. The cryptography techniques are classified on the basis of their key selection into:

Symmetric (Private) Cryptography:
In symmetric or private-key or secret-key encryption the same key is used for encryption and decryption. Data are encrypted using the private key to become unintelligent. Exclusive OR is the slightest algorithm to be applied for that purpose and it can make the system nearly tamper proof. Users must update the main and sub keys. Exclusive OR algorithm is effective and fast as compared to asymmetrical key cryptography. Keys in symmetric key cryptography are at the encryption side and sent to the receiver side to carry out the decryption algorithm (Singh and Mandeep, 2014).

Asymmetric (Public) Cryptography
In asymmetric cryptography, a pair of keys is used. The first key is used to encrypt a message and the second one is used to decrypt it. One of the keys is made public and the other is kept private. The two keys are generated simultaneously and they are prime numbers. Asymmetric cryptography is more secure that private key cryptography but it consumes more computational power and takes more processing time such that extra hardware is required (Singh L et al., 2014).

Modern Cryptography
In modern cryptography, a combination of both public key and private key is used. Once key is used to encrypt the plain message and the other one is used to decrypt the encrypted message. When the private key is used to encrypt the plain message, the process is entitled under the digital signature. Using the private key helps to have a fast speed process and using the public key helps to secure it. Using the pair of keys helps to avoid the key transportation and provides the power to the users to generate their own keys of variable length. This also gives the flexibility to upgrade the key at any interval of time. In modern cryptographic techniques; certification authority is used to keep the track of the entire system and keys (Singh and Mandeep, 2014).

The generation, modification and transportation of keys have been done by the cryptographic algorithm. There are many cryptographic algorithms available in the market and their strengths depend upon certain criteria. Some important cryptographic algorithms include: Data Encryption Standard (DES), International Data Encryption Algorithm (IDEA), Blowfish, Triple DES (TDES), Advanced Encryption Standard (AES), Twofish, RSA, Diffie-Hellman, Elliptic Curve Cryptography (ECC), Pretty Good Privacy (PGP), Public key infrastructure (PKI) (Singh and Mandeep, 2014).

This paper includes five sections. Section one is an introduction and section (2) introduces the security services to be applied in this paper. In section (3), a literature review is carried out to collect the information system and security trade-offs and then an analysis and a discussion of the applied security services is done in light of these trade-offs. Results of the analysis are given in section (4) and some conclusions are highlighted in section (5) and a list of the used references is given at the end of the paper.

APPLYING THE DECURITY SERVICES TO PROTECT SOFTWARE

Mostly, owners of software applications need some security that helps them control and follow up the events running by their software. Owners of software need to guarantee the protection of their applications against insider and outsider attacks. For instance, a shopper's owner may like to guarantee that the shopper employees are not cheating on. In this paper, some security services are applied to fulfill these needs such as:

i. Hiding the files and folders of transactions (by using the attribute "hide"),
ii. Denying the access of these files and folders (by setting the attributes of some files and folders to Read-Only),
iii. Using encryption,
iv. Using hash functions (to guarantee the integrity of the files' contents) and
v. Using digital signature (to record the signature of the users).

Each of these services helps the software owner to control and follow up the use of the software. However, the security services can be applied to any similar software. Visual Basic DOT NET programming language is used for the implementation. Analysis of the security services, applied to the software, will be discussed in light of the security criteria. Next section discusses the application of the security services and section (4) discusses the security criteria, while section (5) analyzes the applied security services in light of the security criteria. The shopper program is designed for the purposes of storing goods, selling goods, following up the movement of goods in a daily basis. VB NET is used to implement the program. The program has the following operations:

i. Preparing the program for the first time
   a) The passwords b) Data and database files
ii. Movement of goods in and out of the stores.
   a) Adding new goods. b) Changing information of goods
   c) Making a report
iii. Movement of the goods in and out of the shopping center.
   a) A new customer service. b) Adding new goods.
   c) Changing information of goods d) Making reports

Some of the last operations and processes need to be protected and some of them do not. When the program starts running for the first time, a test is done to check if the folder "c:\shpdat\" is found or not, if not, then it is created. Then, the password of running the program is set by entering it twice to verify its correctness. A file is opened by the name "file1.txt" and used to save the date and password of running the program, in their plain text and encrypted forms and their hash block. Then, the files (file2.txt – files3.txt – file4.txt) are created. The files are used as follows:

i. File1.txt is used to records the Input/Output information including encrypted passwords, hash blocks, date of accessing files.
ii. File2.txt is used to record the movement of good in/out of the store. Contents of the file include records for every good, such as: the number, the name, the price, the quantity.
iii. File3.txt is used to record the movement of goods in/out of shopping center. Contents of the file include records for every good, such as: the number, the name, the price, the quantity.
iv. File4.txt is used to record the daily movement of good in the shopping center. Contents of the file include records for every good, such as: the number, the name, the price, the quantity.

In the following subsections, we discuss a group of the protection tools and techniques that can be used with any software. These tools and techniques include:

i. Passwords of running software and accessing the different files.
ii. Controlling the files and folder attributes such as: 1) hiding, 2) Read only.
iii. Encryption/Decryption of data and information.
iv. Hashing the files' contents.
v. Signing certain parts in the used files.
vi. RSA Cryptography Algorithm.
vii. Coding the Arabic letters

Passwords of Running Software and Accessing Files

Passwords are a tool that protects the access of applications and/or software. Choosing passwords is critical and crucial. Experts advise users of computerized access to select their passwords in clever ways. The following tips are important for password uses (Stallings, 2013):

i. Capital letters and small ones, as well as numbers and symbols should be included in the passwords.
ii. Length of the password can as long as it can be and it is advisable to change it from time to time.
iii. Passwords must be changed from time to time with short periods of gabs between them.
iv. Passwords are not saved in plain text but they are encrypted. It should be noted that passwords may include symbols and non English letters. This means that a suitable coding is needed. The coding with utf8 is found to be suitable (Stephens, 2005).

Controlling the Files and Folders Access

Attributes of files and folders control their access. Hiding and Read-Only are two important attributes. These attributes can be implemented according to the used programming language. In the following, we explain how to control these attributes (Stephens, 2005):

1) Hide Attribute:

When the file attribute is Hidden, the user can not see the file or the folder. This means that no one can access the folders or files contents. In VB NET, the following code is used to hide the file or folder.

```
Dim attribute As IO.FileAttributes = IO.FileAttributes.Hidden
System.IO.File.SetAttributes("c:\shopdat\", attribute)
```

Figure 1: Code of hiding a folder

2) Read Only Attribute:

When the file attribute is Read Only, the user can open the file and read it. No changes are allowed to be done to the contents of the file. This means that no deletion or addition of any information can happen to the contents. In VB NET, the following code is used to change the attribute of the file and folder to Read Only.

```
Dim attribute As IO.FileAttributes = IO.FileAttributes.ReadOnly
System.IO.File.SetAttributes("c:\shopdat\file1.txt", attribute)
```

Figure 2: Code for file Read_Only attribute

Encryption/Decryption of data and information

Encryption changes information to an un-understood form and thus they can be protected from being disclosed. The most secure cryptographic systems are very much affected by specifications or programming errors. No amount of unit testing will uncover a security' vulnerability in a cryptosystem. Keys can be discovered and consequently the cipher texts can be deciphered. Thus, the difference between a security system and another lies in how long it is needed to break the system and/or obtain the keys. The success of attacking a security system depends on the used keys, as well as the used algorithm. Length of the used key is an important standard that affects the speed of running the software. The speed of software may be also affected by the encryption algorithm. Thus, we compromise between the used keys and algorithms and speed (Boneh and Shoup, 2015). There are two main types of algorithms for encryption. These are the symmetric key encryption, and the public key encryption. In the following, we describe how to generate the keys in both of the types (Stalling, 2013).

Symmetric (private key) cryptography:

For this type of algorithms, one secret key is used for encryption and decryption. The secret key can be generated by using any good random number generator. The output of the random number generator is used as a key to encrypt the text. When we need to decrypt the cipher, we simply run the random number generator and use its output to decrypt the message and obtain the plain text. Thus, if the random number generator is really good, it will give good results. Good results here mean that no way, an attacker can use output random numbers to obtain the original information the generator uses. In this case, we can simply use the XOR operation to encrypt the plain text at the sender and decrypt the cipher at the receiver to obtain the plain text.

However, with computers, we can not pretend that the generated keys are really random, but to be fair, we can say they are pseudo-random keys. Thus, depending on the cycle of the pseudo-random generator, we can assure that the used keys are strong or not. A good pseudo-random generator can be designed using the following chaotic equation (Boneh and Shoup, 2015; Rahourma, 2000):

$$X_{n+1} = r * x_n * (1 - x_n) \quad (1)$$

With $3.57 \leq r \leq 4$ and $0 \leq x_0 \leq 1$, the obtained sequence of numbers is really pseudo-random. The time of logging into the program is used to generate these values. Seconds are divided by 60 to give x0. Minutes and hours are concatenated (as strings) and then taken as a number and divided by 6024. If the result is greater than 0.4 then it is multiplied by 0.4 and added to 3.57 and if it is less than 0.4 it is directly added to 3.57. This gives the value of r. X_{n+1} is set to a double format and it is treated to obtain the encryption key as follows:

i. The information are divided into fixed length blocks (e.g., L=16 bytes).
ii. The right L bytes from xn+1 are taken as the key which we XOR with the plain text to obtain the encrypted cipher.
iii. The encrypted block is XORed with the same key to obtain the original plain text.

The equation (1) uses an initial seed x_0 and a bifurcation factor r to compute recursively values x_{n+1}. Knowing the numbers of the sequence does not help any attacker gets the values of r and x_0. Thus, the obtained numbers are used as encryption/decryption keys. With such good keys, we use a block cipher algorithm of the simple XOR operation for encryption/decryption. The files' contents that are used by the program are divided into fixed length blocks. Each block is encrypted using a different key that is generated from the random number generator. Figure 3 gives the block diagram of the encryption process.

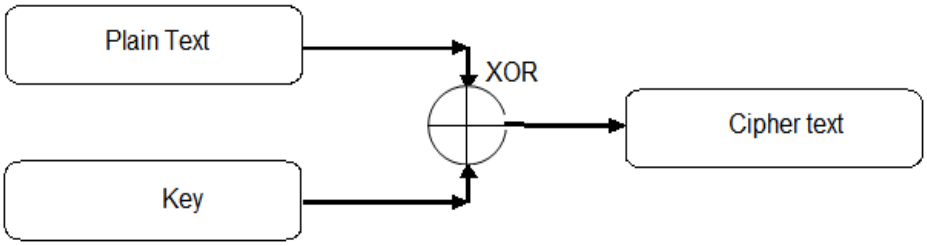

Figure 3: Block diagram of the. XOR encryption process

Asymmetric (public key) cryptography

For this second type, two keys are used: one of them is kept secret, and the other is published. One key is used for encryption and the other key is used for decryption. The two keys can be gene. Using the pair of public and private keys plays an important role in asymmetric cryptography. The public key is published and used by people while the private key is kept secret and used by the owner. Public key cryptography depends always on the cryptographic algorithm. Thus, a hard mathematical problem lies behind the used algorithm. Examples of such hard problems: the discrete logarithm problem, the elliptic curve problem, etc. However, the two keys have a very important characteristic. Encrypting a message with the public key is reversed (i.e., decrypted to the same message) by using the private key. Thus, the effect of the public key in the encryption algorithm is cancelled by the effect of the private key in the decryption algorithm. Because of the computational complexity of asymmetric encryption, it is usually used only for small blocks of data, typically exchanging the symmetric cryptography keys, digital signature, message authentication codes [Stinson, 2003; Smart, 2013; Ruohonen, 2014).

Hashing of Plain Texts and Message Authentication Code (MAC)

When the symmetric key cryptographic technique is used to provide message authentication, it is called a message authentication code (MAC). To establish a MAC process, a private key K is shared by the sender and the receiver. A MAC is simply an encrypted checksum generated from the processed message and sent along with it to ensure its authentication. Figure (4) presents the generation of the MAC and figure (5) explains how to use that MAC to authenticate the plain text message.

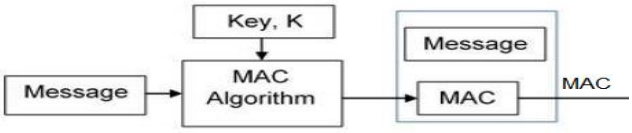

Figure 4: Generating the message authentication code (MAC) at the sender.

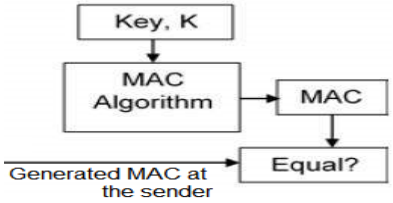

Figure 5: Regenerating MAC at the receiver and authenticating the received message.

The MAC algorithm

The steps of MAC algorithm are (Stallings, 2013):

i. At the sender side, some publicly known MAC algorithm is used to process the message and with a secret key K to produce a MAC value.
ii. A MAC function is used to compress any arbitrary long input message into a fixed length output block.
iii. The plain text message and the MAC block are sent to the receiver for authentication. If confidentiality is needed, the message and its MAC are encrypted.
iv. At the receiver side, the message is processed in the same way like it was processed at the sender side using the same private key, to produce a new MAC block.
v. The receiver compares the received MAC block to the new computed MAC block. If they match, then the received message is accepted as an authentic and if they do not match then, the original message (which was sent by the sender) is known to be changed.

The contents of a file can be hashed as follows:

i. The file contents are divided into fixed length blocks (e.g., L=16 bytes).
ii. An initial hash block is generated, from the pseudo-random number generator, and used to hash the blocks with the XOR function.
iii. A final block is obtained which is saved as the hash of the file.

To check the file contents for any change, we simply obtain the contents hash and compare it to the one saved in step (3). In case they are equal then the file contents are not changed and if not, then the contents are surely changed.

Limitations of MAC

There are two major limitations of MAC, both are due to its symmetric nature of operation (Stallings, 2013):

i) Establishment of Shared Secret.

a) Authentication is done between only legitimate users.

b) Shared secret channels are needed prior to use of MAC to exchange the secret keys.

ii) Inability to Provide Non-Repudiation

a) A MAC technique does not prove the origin of a received message.

b) Thus, the sender can deny the responsibility about sending the message and claim that the receiver forged it.

Both of the above limitations can be overcome by using the public key based digital signatures discussed in following section.

Signing the MAC with RSA Cryptosystem

Digital signature is a security service allows people to sign documents electronically. Two keys are needed to complete this process. These are the private key and the private key. The sender uses the private key to encrypt the plain text message and the receiver uses the public key to decrypt the received encrypted message and obtain the original one. Because only the sender knows the private key, he cannot deny sending the message if the receiver succeeds to obtain the original message using the sender's public key. This is because only the sender's public key can be used to decrypt the received encrypted message. This confirms the signature of the sender. Generation of the secret and public keys has its rules and algorithms [10-12]. Figure (6) explains the signature-verification process.

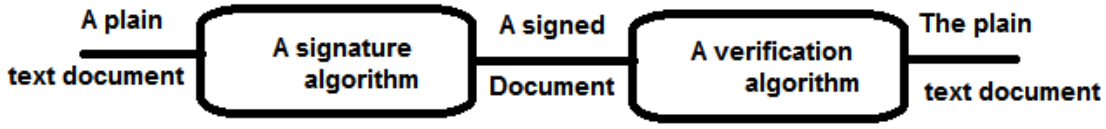

Figure 6: The signature-verification process

To generate the key pair (e, d), choose two random prime numbers p and q of equal lengths (for maximum security). Then compute:

$$n=p * q \quad \text{and} \quad \psi=(p-1) * (q-1) \tag{2}$$

A value e is randomly chosen as the encryption public key. This is done such that: 1) the greatest common divisor gcd(e, ψ)=1 and

2) e * d=1 mod ψ where d is the decryption private key.

This means that:

$$d=e^{-1} \bmod \psi \tag{3}$$

Thus (e,n) are made public and d is made private. We notice that the two numbers p and q are no longer needed. However, the keys e and d can be interchanged.

To encrypt a MAC (M) where: M is smaller than n, with (M, n) are of the same size, the block M is processed according to the following equation:

$$C=M^{e} \bmod n. \tag{4}$$

The idea here is to transform e into a binary form and M = the symbol (') which has the ascii representation of 96

Assume p=17 q=19 N = p*q = 17*19 = 323

ψ=(p-1) * (q-1) = 16*18 = 288 e = 13 $d=(13)^{-1} \bmod 288 = 133$

The RSA Cryptography Algorithm

Rivest et. Al., in (1978) invented the RSA. The RSA gets its security from the difficulty of factoring large numbers. The public key pair (*e, d)* are functions of a pair of large prime numbers *p* and *q* (100 to 200 digits or even larger). Obtaining the plain-text message from its cipher-text form using the public key is equivalent to factoring n to its factors *p* and *q* (Stinson, 2003; Smart, 2013; Ruohonen, 2014). Encrypting M is done as follows:

$C = (96)^{13} \bmod 323 = (96)1101 \bmod 323 = 96 * 96\text{^}4 * 96\text{^}8 \bmod 323$

We can notice that the process is simply squaring the first multipliers to get the second multiplier and squaring the second multiplier to get the third multiplier, and so on. Then, for k binary bits the total multiplication value x^y is computed as follows:

Assume that $y)_{10} = (b_n\ b_{n-1} \ldots\ldots\ b_2\ b_1)_2$

$x^y \bmod = MULT\ (a\text{^}b_i*(2\text{^}(i-1))), = 1, 2, 3, \ldots.., k$

Thus:

$96^{13} = 96 * (96\text{^}2 * 96\text{^}2) * (96\text{^}2 * 96\text{^}2)\text{^}2 \bmod 323 = 96 * (9216 * 9216) * (9216 * 9216 * 9216 * 9216) \bmod 323 = 39 * 9216 * 9216 * 9216 * 9216 * 9216 \bmod 323 = 248 * 9216 * 9216 * 9216 * 9216 \bmod 323 = 20 * 9216 * 9216 * 9216 \bmod 323 = 210 * 9216 * 9216 \bmod 323 = 267 * 9216 \bmod 323 = 58$

Thus, the cipher of (') = (:)

To decrypt a received cipher C compute:

$M' = C^d \bmod n = 58^{133} \bmod 323 = 96$

The obtained M' is the same as M because

$C^d = (M^e)^d = (M)^{e.d} = (M)^{k(p-1)(q-1)+1} = M . (M)^{k(p-1)(q-1)} = M .1 = M .$

The RSA is very easy to understand and implement. The crypt-analysis can't give any argument about RSA security and instead, it suggests a confidence level in the algorithm. The RSA technique can also be applied to encryption and digital signature. In this case the following points are considered to make the use of RSA algorithm more secure (Stallings, 2013):

i. Random messages received from strangers are not signed unless a one-way hash function is firstly used.
ii. Sharing a common a common modulus n among a group of users is not preferred.
iii. The processed block must be smaller than n and they must be of the same size. Padding with certain values may be needed if the last message block size is less than that of n.
iv. The decryption exponent (*d*) should be large.

Note:

1- Two parties (A, B) use their special keys (e_A, d_A, n_A) and (e_B, d_B, n_B) respectively.
2- The large modulo number from (n_A, n_B) is used first.

3- Some people have mixed RSA with conventional crypto-systems (e.g. block cipher ones) and digital signatures.

The following algorithm is used in RSA (Rivest et al., 1978; Rajdeep, 2015),

i. Choose p and q as two random big prime numbers.
ii. Calculate n = p * q
iii. Calculate φ(n) = (p - 1) * (q - 1)
iv. 4. Choose e such that: 1) $1 < e < \varphi(n)$ 2) (e, n) are co-prime.
v. Calculate a value (d) where (d * e) % φ (n) = 1.
vi. Make (e, n) as the public key.
vii. Make (d) , n) as the secret key.
viii. For encryption, calculate $C=m^e \bmod n$
ix. For decryption, calculate $m =C^d \bmod n$)

The above algorithm is applied to the plain text to obtain the encrypted form or cipher text. Then, the encrypted message is decrypted to plain text. The main disadvantage in RSA cryptographic is its encryption speed. The RSA algorithm consumes a lot of time to encrypt data. This is disadvantage is almost common between all the asymmetric key algorithms because of the need to compute the exponent of big blocks to big numbers. However, the RSA provides a good level of security but it is slow for encrypting files. Another threat in this algorithm is to change the decryption key and thus it must be kept secret. When a file contents are hashed, the final block is signed using the RSA system. Thus, every user of the program will have a private key and a public key. These keys are stored in the file "file1.txt" and the public keys of the users are known to everyone while the private keys are kept secret to their owners. The program checks the signature of the hash by applying the RSA algorithm.

Encoding Arabic Alphabetics

When the password contains arabic letters, it needs to use the utf-8 coding not the ascii coding. Then, every letter in the password will be transformed into two hexadecimal digits. The followuing code is implemented for coding and decoding Arabic alphabetical strings.

```
Dim utf8Encoding As Encoding = Encoding.UTF8
Dim stringvalue, stringspw, stringsd As String
Dim bytes(300) As Byte
stringvalue = strings
MsgBox("Strings to encode= " & stringvalue)
Dim written As Integer = utf8Encoding.GetBytes(stringvalue, 0, stringvalue.Length, bytes, ind)
ind = ind + written
npw = ind
stringspw = ShowByteValues(bytes, ind)
MsgBox("Encoded bytes= " & stringspw)
Dim newStringd As String = utf8Encoding.GetString(bytes, 0, ind)
MsgBox("Decoded= " & newStringd)
```

Figure 7: Encoding/decoding Arabic letters

Example of Using utf-8

Assume the password is: "تأمين برنامج المتجر" ===➔ meaning "securing the shopper program"

Number of Arabic letters = 17 letters + 2 spaces

Number of bytes = 36

Notice that the space is treated as one byte because it is already included in the ascii code. Thus we got

Encoded bytes = D8 AA D8 A3 D9 8A D9 86 20 D8 A8 D8 B1 D9 86 D8 A7 D9 85 D8 AC 20 D8 A7 D9 84 D9 85 D8 AA D8 AC D8 B1

Figure 8: Results of encoded Arabic letters

ANALYSIS AND DISCUSSION

Information Systems and Security

Over the past 20 years, our society has become increasingly dependent on software. Today, we rely on software for our financial transactions, our work, our communications, even our social contacts. A single software flaw is enough to cause irreparable damage, and as our reliance on software increases, so does our need for developing systematic techniques that check the software we use for critical vulnerabilities (Athanasios, 2014).

The software architectural solution should meet the security requirements as well as the other quality attributes such as performance, availability, usability, modifiability, etc (Hassan, 2013). Selecting a design solution among multiple options involves making trade-offs among competing requirements. Security is one critical requirement among many, which can cause critical trade-offs and severe costs. Damages from security attacks can be overwhelming and the costs increase every year. The threat of vulnerabilities and their exploitation by potential adversaries call for careful analysis of security risks and trade-offs that security solutions impose, from the viewpoints of both defenders and attackers.

Since software developers and analysts are usually not security experts, detecting potential threats within software systems can be problematic. Even when threats are known, the risk factors, either the probability of a successful attack or the resulting damage of a successful attack, are not always known or numerically measurable. Selecting proper security solutions can be challenging, when mitigating impacts and side-effects of solutions are often not quantifiable (Tiwari and Karlapalem, 2005).

If the organization depends on information technology, information security becomes very important. Confidentiality, Integrity and Availability (CIA) aspects of the system must be protected security of the used computer based information system. Increasing the dependence of business processes on information technology increased the number of attacks against CIA aspects. Achieving a perfect security system is monetarily and practically infeasible. Thus, organizations use risk management concepts to forego perfection. Hence, they make tradeoffs in pursuit of security goals (Tiwari and karlapalem, 2005). However, trade-offs between systems engineering and security engineering can be summarized in (Reed, 2015) as follows:

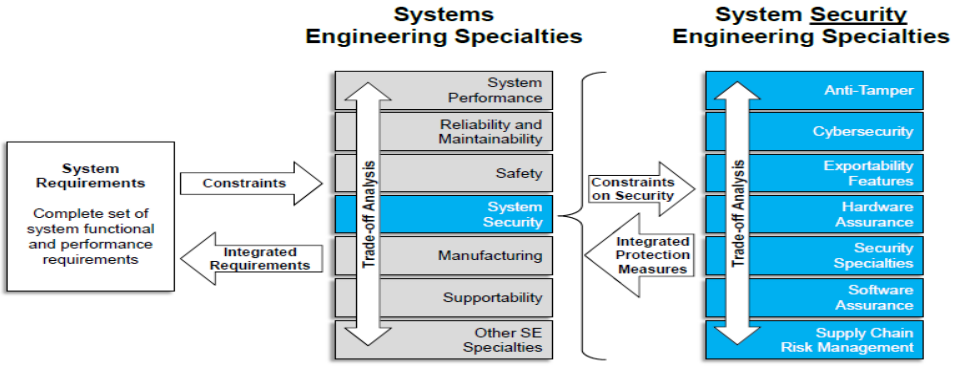

Figure 9: System engineering and system security engineering trade-offs

Analysis of security services in light of the software trade-offs

In the following, we discuss the software trade-offs and analyze the security services introduced in this paper.

Security and Complexity

Complexity of software is affecting the utilization of the information system resources, quality, performance and run time. In the following we give some hints about these effects:

i) Security Effects

Increasing the software complexity may result from many primitives such as using: counters and hashing as well as arbitrary code fragments. These primitives require significant bandwidth, memory and processing resources. The required resources can affect the accuracy of the eventual measurement (Moshref et al., 2013). The software complexity affects the software quality. A wide range of solution proposals have been considered to solve this problem. However, a greater empirical research is required to evaluate and compare these solution proposals (Sebastian et al., 2012).

Nowadays, hundreds to thousands of cores per processor are seen by many as a natural evolution of multicore processors. These multicore processors require a productive parallel programming model and an efficient runtime. A scalable synchronization mechanism may be considered as a critical prerequisite for an efficient runtime to support task coordination at different levels of granularity. However, several approaches of phase implementation are suggested using software, hardware and a combination of both to explore their portability and performance (Yonghong, 2011).

ii) Security and Complexity

In the following we discuss the used services according to the complexity and mathematical computations:

i. Generating the keys of operation is carried out by running a simple chaotic recursive equation.
ii. Applying the encryption/decryption is carried out mainly by applying symmetric key cryptography which uses the simple XOR operation. This means that no much computation is needed and consequently no complexity would result.

iii. Hashing is carried out by using the simple XOR operation between the hashed blocks. This also does not complicate the computations.
iv. The RSA cryptography is used only for performing the digital signature ad this makes its complexity effect low.

However, cryptography requires efforts to fulfill security conditions in: the generation of keys, encryption/decryption and transmission of the messages or saving them. These issues are always translated into overheads including financial overheads, less communication channel bandwidth, heat dissipation affecting processors, power consumption and time delay of processing. For the present work we can highlight the following points:

Security and Usability

Usability and security become core issues in the design of modern computer software. Many studies have been conducted in different combinatorial ways of these issues. However, still there is a room to improve the relationship between security and usability in the sense of appropriate deployment of these features in software applications (Sahar, 2013). Also, applying the Arabic letter coding adds more credit to the usability of the program.

Security and Availability

Software availability is reduced with increasing it security. Mostly, software programs are not freely publicly available. The shopper software is secured and consequently is not freely available. However, a free trial limited version can be published for demonstration. Also, applying the Arabic letter coding adds more credit to the availability of the program.

Security and Anti-Attacking

Using the security services, discussed in the last sections, gives a good protection to the software. In the following we discuss the different points in this regard (Kakkar et al., 2012; Kakkar et al., 2010):

i. Hiding the files which are used by the software, makes it hard for the insider/outsider attackers to reach them.
ii. Using the Read-Only attribute makes it hard for the attackers to change the files' contents unless the program is legally used.
iii. Using encryption/decryption makes it hard for the attackers to understand what the contents of the files mean.
iv. Using the digital signature makes it easy to trace people who used the software and made any changes.
v. Hashing the file contents makes it easy to discover if the contents were illegally changed or not.
vi. Using the chaotic equation for key generation, pseudo-randomly, makes it hard to guess the passwords and keys. Also, it is hard to collect the generated numbers and reversely find the factors of the chaotic equation.
vii. Encoding Arabic alphabets helps the administrator to use Arabic letters in passwords and keys.

Key management

In modern, users have their own keys which need to be kept secret. Also, information about the data, which are used to obtain the keys, must be kept secret. In practice, Individual users use different keys. A security systems is much affected by the number of keys, length of keys and their generation and transportation (Kakkar et al., 2010; Verma et al., 2012). An attacker may succeed to attack the system but fail to expose the information because they are protected. Attacker may then try to crash the information by damaging them. This means, some techniques are needed to protect the information from being crashed. This can be done by protecting the documents and files from being accessed and/or changed. Multiple keys for individual user maybe then a good solution to increase information protection but this increases the calculations and consequently slowdown the speed of information processing.

RESULTS

Securing software programs lies in a critical area of research. Research has been done to determine the trade-offs and criteria for software production and to include or embed security services in software programs. Increasing software security reduces its complexity and consequently exhausting its resources and running time. Practically, people compromise between increasing security and speeding up the program execution speed. However, in the following, we give some of the advantages of using the security services in the present work.

i. The attributes of "Hide" and "Read-Only" are used to protect the files and folders from being freely accessed by illegal users.
ii. Using the chaotic equation for key generation, pseudo-randomly, makes it hard to guess the passwords and keys. Also, it is hard to collect the generated numbers and reversely find the factors of the chaotic equation.
iii. Encoding Arabic alphabets helps the administrator to use Arabic letters in passwords and keys.
iv. Encrypting the files' contents makes it hard for to get the plain information of these contents without using the same keys XORed with the cipher. However, using the XOR keeps the encryption/decryption processes fast.
v. Hashing the file contents yields a possibility of checking up the integrity of these contents.
vi. Signing the hash block using the RSA algorithm allows us to follow up the users who used the program. It is well known that breaking the RSA system is hard because it depends on reversing the modulo-exponent.
vii. The applied security services do not result in much complexity and thus running the software does not affect the system resources. The program usability is easy to handle but its availability is not much because the software is not freely available and in case of hunting a copy from the software, it would be hard to run it without disclosing the used keys.
viii. Applying the Arabic letter coding adds more credit to the program usability and availability.

CONCLUSIONS

Software programs have spread in all sectors of our daily life. Software owners always like to secure their programs such that their data files are safe and cannot be easily attacked. The shopper program was designed by the author and used here to apply some security services for protection. The program uses 5 files to store its data to follow up the movement of goods in the store and in the shopping place. The first file is used to save the information of the users. The second and third files are used to record the movement of goods in the store and in the shopping place. The fourth and fifth files are used to record the daily movement of goods in the store and the shopping place. Visual Basic Dot NET is used to produce the application. From the security service applied in this paper: 1) Hiding the file and folders, 2) Read-Only attribution for the files' contents, 3) Fast Stream Cipher Encryption using a random key generator, 4) Hashing the contents of files, 5) Signing the hash of the file. A literature review was done to collect the software trade-offs. Then, the applied security services were analyzed in light of the software trade-offs. Results show that software complexity is considered a main trade-off because it affects the systems resources, usability and availability. Applying the Arabic letter coding adds more credit to the program usability and availability. Data files and folder can be protected by using the attributes of "Hide" and "Read-Only". Using the XOR encryption/decryption does not affect the software performance and run time. Using the chaotic equation to pseud-randomly generate the used keys made it hard to attack the software because collecting these numbers does not help to disclose the equation factors (bifurcation factor r, the constant a and the initial value of x_0). The contents of files are encrypted before saving them and decrypted when the programs reads them for use. Contents of files are hashed and then signed. This makes it easy to trace the users who used the program and makes it easy to check up if the contents were changed or not. The RSA is known sign the hash block of the file contents.

REFERENCES

Athanasios (Thanassis) Avgerinos (2014). *Exploiting Trade-offs in Symbolic Execution for Identifying Security Bugs*. A Doctoral Dissertation, Submitted to the Electrical & Computer Engineering, National Technical University of Athens, ii.

Boneh, D. and Shoup, V. (2015). *A Graduate Course in Applied Cryptography*. 43, 81-91, 125-142, 243-265, 301-370. Retrieve from https://crypto. stanford.edu/ ~dabo/cryptobook/draft_0_2.pdf.

Golnaz E. (2012). *Making Trade-Offs among Security and Other Requirements during System Design*. A Doctoral dissertation submitted to the Graduate Department of Computer Science University of Toronto, ii-iii.

Hassan R. (2013). Security Trade-off Analysis of Service -oriented Software Architecture. *World Journal of Computer Application and Technology 1(4)*, 110-120.

Jan M. Memon, Asma Khan, Amber Baig and Asadullah Shah (2007). A Study of Software Protection Techniques, *Innovations and Advanced Techniques in Computer and Information Sciences and Engineering*, 249–253, Springer.

Kakkar J., M. L. Singh, P.K. Bansal (2012). Comparison of Various Encryption Algorithms and Techniques for Secured Data Communication in Multinode Network. *International Journal of Engineering and Technology, 2(1)*, 87-92.

Kakkar A., Dr. M. L. Singh, Dr. P. K. Bansal (2010). Efficient Key Mechanisms in Multinode Network for Secured Data Transmission. *International Journal of Engineering Science and Technology, 2(5)*, 787-795.

Koko, S. O. and Amin B. Mustafa (2015). Comparison of Various Encryption Algorithms and Techniques for improving secured data Communication. *Journal of Computer Engineering (IOSR-JCE), 17(1)*, 62-69.

Moshref, M. M., Minlan Yu and Ramesh G. (2013). Resource/Accuracy Tradeoffs in Software-Defined Measurement. HotSDN'13, Hong Kong, China,73-78.

Rahouma, K. (2000). A chaos-based stream cipher algorithm for high speed networks and real time applications. *Presented and published in the Applied telecommunication symposium, as a part of the 2000 advanced simulation technologies conference (ASTC2000)*, Washington, D.C. USA, 16-20.

Rajdeep B. and Rahul Hans (2015). A Review and Comparative Analysis of Various Encryption Algorithms. *International Journal of Security and Its Applications. 9(4)*, 289-306.

Reed M. (2015). Systems Engineering and System Security Engineering Requirements Analysis and Trade-Off Roles and Responsibilities. *18th Annual NDIA Systems Engineering Conference Springfield*, VA, 1-4.

Rivest, R., Shamir, A. and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM, 21(2)*, 120-126.

Ruohonen, K. (2014). *Mathematical Cryptography: A Translation of Lecture Notes in Finnish Language*. 42-47. Retrieve from http://math.tut.fi/~ruohonen/MC.pdf.

Sahar, F. (2013). Tradeoffs between Usability and Security. *IACSIT, International Journal of Engineering and Technology, 5(4)*, 434-437.

Sebastian B., Kai Petersen, Mikael Svahnberg, Aybuke Aurum, Hamish Barney (2012), Software Quality Trade-offs: A Systematic Map. *Information and Software Technology, 54(7)*, 651-662.

Singh, L. and Er MandeepKaur (2014). Novel Technique of Cryptography algorithm for Improving Data Security. *Global Journal of Advanced Engineering Technologies, 3*(4), 394-398.

Smart, N. (2013) *Cryptography: An Introduction*. McGraw-Hill, 3rd ed., ebook, 109-119, 153-181.

Stallings, W. (2013). *Cryptography and Network Security, Principles and Practice*. Prentice Hall, London, Sixth Edition, 329-331.

Stephens, R. (2005). *Visual Basic® 2005 Programmer's Reference*. Wiley Publishing, Inc., 687-746.

Stinson, D.R. (2003). *Cryptography, Theory and Practice*. 3rd Edition, Chapman & Hall CRC, London, 233-273.

Tiwari, R.K., and Karlapalem, K. (2005). Cost Tradeoffs for Information Security Assurance. *4th Annual Workshop on the Economics of Information Security, WEIS. Harvard University*, Cambridge, 1-3.

Verma S., Rajnish Choubey, Roopali soni (2012). An Efficient Developed New Symmetric Key Cryptography Algorithm for Information Security. *International Journal of Emerging Technology and Advanced Engineering, 2(7)*, 18-21.

Wang, C., Davidson, J., Hill, J. and Knight, J. (2001). Protection of software-based survivability mechanisms. *The International Conference on Dependable Systems and Networks*, Goteborg, Sweden, *IEEE Press*, 193-205.

Wang, P. (2005). *Tamper resistance for software protection*. M.S. thesis, School of engineering Information and Communications University, Daejeon, 1-2. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.8556&rep=rep1&type=pdf

Yonghong Y., et. al. (2011). Hardware and Software Tradeoffs for Task Synchronization on Manycore Architectures. *17th international conference on Parallel processing, Euro-Par'11 - Volume Part II*, 112-123.