UNIVERSITY OF GOTHENBURG

Protecting User Privacy in an Untrustworthy Environment

Galin Ivanov Zhelyazkov

Bachelor of Software Engineering and Management Thesis
ISSN: 1651-4769 Report No. 2009-063

Abstract

With more and more widespread use of web services in building distributed applications grows and the need to develop and implement adequate mechanisms for protection of such applications. The purpose of this paper is to investigate ways for achieving high security by using encryption during development of applications providing web services. The main task of the paper is to show a universal security model design guidelines for protecting user privacy in an untrustworthy Environment.

1. Introduction

1.1 Background

During the last decade the development of web applications increased rapidly and nowadays they are trying to replace the regular desktop applications. More and more people support the idea that one should be able to do everything what can be done with a desktop application on the web without taking care of operation system (OS), different platforms, software programs and file formats. The big software companies already start introducing this concept, e.g. Google docs, Photoshop express, etc. This concept implies the need of extremely high security, because any security leak found can result in bad reputation for these companies.

Security is an important part in the development and operation of any distributed software application. This applies with full force on applications representing or providing web services since they are built based on common standards, ensuring cross platform compatibility and transparent location of individual components. With more and more widespread use of web services in the development of distributed applications, the need of developing and implementing adequate mechanisms to protect these applications has increased.

Even more extreme is the situation, where the application can't trust its operating environment. In short, operating environment is the environment in which users run programs. Web-based applications, in particular, use the web browser and a web server as an operating environment. In this paper operating environment will refer only to the web server. Providing a secure, reliable, and efficient operating environment to support the web applications activities is a challenging task. Therefore the web applications should take this problem into consideration and provide mechanisms to prevent possible vulnerabilities. Such mechanisms can be encryption of every piece of data, that is vital to secure user's privacy.

1.2 Problem statement

In this paper I will concentrate my attention on data integrity in the web environment and issues of trust. In other words, I will show a constructive approach to design secure web based applications, which in the rest of the paper I will refer to as *security design model*. At the moment such set of guidelines on how to develop an application that needs to operate in an untrustworthy environment without compromising privacy are missing.

1.3 Purpose

The purpose of this paper is to investigate ways for achieving high security during development of applications providing web services. The motivation for doing this study comes from the fact, that the more popular web based applications are becoming, the more security vulnerabilities they are showing.
Another purpose of this paper is to explain the seriousness of security quality attribute in applications providing web services. And on the other hand to propose universal security design model, that can be applied to any application and still guarantee the same level of confidentiality.

1.4 Demarcation

For the purposes of this paper I ignore such issues as communication line security and operating systems security (for example reading data from system buffers). I confine my topic to that part of the system, which deals with encrypting and decrypting user's privacy data, stored on an untrusted web server.

This paper is organized into five major sections. Following this introduction, the next section reports on research methods. Section three summarizes the theoretical background. In section four I introduce "Alice", the target system of my case study and section five reports on the results from applying the security design model in it. Finally, in section six the conclusion is presented.

2. Methods

Like many predominantly technical projects, this project will follow the principles of Design Science (DS). DS puts strong focus gaining insight by means of proof-of-concept implementations, in order to combine research interests with practical needs. DS was chosen because it ensures the relevance and effectiveness of IS research, it seeks to create "what is effective" [13].

Another reason to choose DS was its characteristic trait, the strong orientation towards solving problems. "It [DS] is solution-oriented, using the results of description-oriented research from supporting (explanatory) disciplines as well as from its own efforts, but the ultimate objective of academic research in these disciplines is to produce knowledge that can be used in designing solutions to field problems." [20]. Other distinguishing characteristic is the normative or prescriptive nature of the outcome of a research program. Whereas the typical outcome of descriptive research are algorithmic prescriptions ("if you want to achieve Y in situation Z, then perform action X"). In DS prescriptions are of a heuristic nature. Given the background of this project, DS therefore promises to be a highly appropriate method to follow.

In the design sciences academic research objectives are justified by pragmatic validity [20]. Pragmatism arises out of actions, situations and consequences rather than antecedent conditions. It has "a concern with applications - what works - and solution to the problem" (Patton 1990). According to John W. Creswell [9], "instead of focusing on methods, [pragmatic] researchers emphasize the research problem and use all approaches available to understand the problem" (see Rossman & Wilson 1985). For a pure technical research, pragmatism seems to be the the most appropriate of all paradigms.

2.1 Data collection

This section is used to describe the process of preparing and collecting data. Different data was collected to obtain information, to keep on record, to make decisions on important issues and to pass the information on to others. Primarily, the data was collected to provide information regarding specific web application security topics. The process of data collection took place early in the project, and was formalized through a data collection plan, which contained the following activities: literature study, interviews and project documentation. In the coming subsections I will concentrate more on the details.

2.1.1 Literature study

The literature study was the first action I took in order to obtain the theoretical background needed for this research. 13 articles on different web security topics were read. This study gave a broad overview of the problem, starting from servers OS security, going through network communication security and concluding with web based services security. The relevant findings were collected and summarized. Next action was to narrow down the research problem and to select the most interesting topic for the reader. In addition to serving as a basic for the case study "Alice" (see chapter 4), the literature study was done to position my research. Since nobody provides fully complete set of guidelines on how to protect user privacy especially in an untrustworthy environment, I've dedicated this article to that topic.

A more detailed literature study was carried on using search engines, strongly oriented towards academic sources, e.g. Google Scholar, IEEE Xplore, ACM digital library (Assosiation for

computing machinery), oba digital library (Amsterdam library), etc.

A small number of the keywords used follows: web application security, web security, ISO 9126, master-slave architecture, security guidelines, security models, web environment, web services, web based service security, web application operating environment, distributed software applications, distributed applications, distributed web applications, web application vulnerabilities, security leaks in dynamic web page generation, security quality attributes, symmetric and asymmetric encryption, public key encryption, rsa, w3 security, etc.

2.1.2 Interviews

The study comprised of three initial interviews with the previous developer of the Alice 1.0 project used as an introduction to the problem. These were open question interviews, which were recorded and transcripted later on. Another important source of information was the meeting with the teams whose projects will involve the intended web interface to the Alice 2.0 system. There were also presentations for the stakeholders after each big milestone. During the whole process there were regular meetings with a security expert allocated to follow, approve or disapprove and test the security mechanism I've proposed. These meetings were held on regular basis, twice a week.

The three introduction meetings explain what "Alice 1.0" was used for, how did it perform, what strengths and weaknesses it had. The different trade-offs and design decisions were clarified in detail also. The purpose of these meetings was to serve as an introduction to the project as well as to elicit requirements. While in the first meeting there were more open questions and broad explanation, in the second and the third the question become more and more precise and there were even some brainstorming sessions. Each of the interviews were concluded with reading the summary and deciding the topics for the next meeting.

After the second milestone, there was one structured meeting with one member from each of the teams whose projects will involve the Alice 2.0 system. These were held, in order to facilitate the integration process. Graphic design decisions were discussed as well as dependencies between different software libraries. This meeting provided constructive knowledge to all of the teams for the integration phase.

At the end of each milestone, a presentation with the stakeholders was held. On those presentations the results from each milestone was shown as well as a direct feedback was given from the stakeholders. From business point of view these were very beneficial for the project, because important sales decisions were discussed.

Finally I will talk about the meetings with the security expert. These were short (1h) meetings, held twice a week. Their purpose was to verify the security mechanisms proposed and to check for any possible improvements. They primarily included code review sessions, but also small brainstorming sessions. These were mostly beneficial for me, as the developer of Alice 2.0, since I had the chance to get direct feedback about my research and the work I've done.

2.1.3 Alice documentation

Access to the Alice system source code and detailed project documentation were provided at the beginning of the project. The documentation provided information on requirement specifications, architectural design (incl. different views covered by the architecture), design documents, and exemplary database feed for testing.

3. Related research

This chapter presents the theoretical technical background needed to make the following chapters easier to grasp as well as to position the knowledge this paper brings towards other existing sources.

3.1 Distributed software applications

"Web applications [services] are important, ubiquitous distributed" [21] software applications, using standardized technologies and formats/protocols that simplify the exchange and integration of large amounts of data over the Internet. They make it easier to conduct work across organizations regardless of the types of operating systems, hardware/software, programming languages, and databases that are being used. Once defined and installed in the Web, these applications, also called services, can be discovered and used by other web services and computer programs.
As modern standards to achieve an overall 'program-to-program' communication model for business applications in Web environment are HTTP, XML, SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) and UDDI (Universal Description, Discovery and Integration) [7] [8] [14] . SOAP is an XML-based protocol for exchanging data in distributed systems using a transport protocol to HTTP. WSDL can be considered as an XML-based grammar for the specification of Web services.
Presentation of this meta-information and the type of service is in the records maintained by the standard UDDI (Universal Description Discovery and Integration). UDDI developed standardized specifications for the detection and description of Web services. Interactions between client, server, description of the service WSDL and registers UDDI, which contain meta descriptions of services are realized through specific communication protocols on a higher level. SOAP - UDDI is actively developed for software companies and large technology users in the implementation of major Web portals in different areas of 'e-business' (e-commerce, e-learning, e-health, etc).

3.2 ISO 9126 model

The International Organization for Standardisation (ISO) is organization that facilitates international trade, international coordination and unification of industrial standards by providing a single set of standards that would be recognized and respected [2]. ISO 9126 was developed in 1991 to provide a framework for evaluating software quality and then refined over a further ten year period [4]. Many studies criticize ISO 9126 for not prescribing specific quality requirements, but instead defining a general framework for the evaluation of software quality [19]. Chua and Dyson believe that this is in fact one of its strengths as it is more adaptable and can be used across many systems [6]. The ISO 9126 model defined six product characteristics: Functionality, Reliability, Usability, Efficiency, Maintainability and Portability. These six characteristics are further subdivided into a number of sub-characteristics. "ISO 9126 defines security, which is a sub-characteristic [of Functionality], as a set of software attributes which relates to its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data." [3]

3.3 "trust" in web services

For the Internet it is necessary to establish a foundation of trust among the participants, in order to be accepted as a viable platform, where the big companies can migrate their current

desktop software applications. Trust is often mentioned in the context of e-commerce and has been developed over time through the formation of appropriate policies, procedures and practices to safeguard transactions and company assets [16]. The objective of security should be to manage risks by anticipating what is probably going to happen and limiting the risk exposures that could injure an organization's reputation. Perfect security is hard to achieve, infinitely expensive and often not a rational goal.[18]

3.4 Cryptography

The most common way to secure information is encryption. Encryption is the process of scrambling data using mathematical procedures that make it extremely difficult and time consuming to decrypt, except for the authorized recipient - those with the correct decryption key. Proper encryption guarantees that the information will be safe even if it stored on a insecure machine [5].

The encryption is strong enough only when proper encryption and decryption keys are chosen. Proper keys are dependent on their size. "The size of keys are measured in bits and the difficulty of trying all possible keys grows exponentially with the number of bits used. Adding one bit to the key doubles the number of possible keys; adding ten increases it by a factor more then a thousand" [5].

There are two encryption classes - symmetric and asymmetric. Each of these has broad classes of algorithms for cryptography.
Symmetric key algorithms use trivially related cryptographic keys, also called password, for both encryption and decryption. In other words, a password is used to encrypt some information, and that same password is used later on to retrieve (decrypt) the information. Under the hood is an algorithm or cipher, which simply put, is a mathematical function that transforms the data into something obscure and vice versa. This key algorithm is appropriate only for small systems with little number of users. [11]

Public-key cryptography, using asymmetric key algorithms, follows the public/private key pair technique. One key, the public key, encrypts data and the other, the private key, decrypts it. Because the public key can not be used to decrypt data, it can be safely given out or installed on a web site, allowing anyone to encrypt data to be sent to the owner of the private key. The private key is kept safe and is typically symmetric encrypted with an additional password. [10]

Asymmetric key algorithm does not require a secure initial exchange of one or more secret keys as is required when using symmetric key algorithms. Asymmetric key algorithm is more expensive than the symmetric key algorithm when it comes to performance. The following table presents the numbers that famous security expert Bruce Schneier has quoted in his classic "Applied Cryptography" [17]:

Algorithm type	Operation	Thumb rules
Symmetric key encryption	DES	45,000 64-bit blocks can be encrypted per second
Asymmetric key encryption	RSA	Encryption in 0.03 seconds, decryption in 0.16 seconds, Digital signature in 0.16 seconds, verification in 0.02 seconds

Another way to secure information is by using *one way function*. One way function is a function "easy to compute, but hard to invert on a noticeable fraction of instances" [12]. It has very high performance and is very practical to use in users' authentication process.

4. Case study "Alice"

This case study illustrates how the security design model, the one this paper is proposing, has been used successfully to achieve web application security during Alice implementation, developed for Bob company. Bob is a worldwide leading company on the GPS car navigation systems market. The company is developing a test environment, called Alice, which allows test users (Tom) to give feedback on their journeys. Alice is holding user privacy data, e.g. user's driven routes, user personal data, etc., which is supposed to be highly secured. The case study Alice included three big milestones and the overall project lasted three months. The precise specification of the Alice project, used to investigate the research problem, reads as follows:

"As part of the evaluation process of the IQ Routes development end user experience was acquired. In short, a number of end users (Tom) received SD cards with a special build for their Bob personal navigation devices (PNDs), used them, and then send back their feedback to Bob. In contrast to most previous beta tests, the IQ Routes test supplied an online mode in which users could give feedback per trip basis. The corresponding web application Alice 1.0 was used for this.

As the concept proved itself to be very useful for in-depth analysis of the project at hand, there was a strong wish to extend Alice in such a way, that applying the approach to other projects boils down to an easy configuration task. This required a much more flexible setup of all components included in the system (feedback DB, and test build for PNDs).

Furthermore, as the Alice DB contained personalized trip data the DB must be highly secured and must not be accessible from the extranet. These assumptions led to a master-slave architecture: The master application handled the Alice DB and provided an internal View on the per trip data. This application resided in the intranet. The slave application targets the end user feedback request only. It was providing access to a limited set of data. Furthermore, the slave application needed to be hosted on an insecure host (Sally) without compromising privacy." [1]

In that chapter I'll talk only about the slave application, since that is the part of the system which I fully developed myself from scratch.

4.1 Three scenarios

Let us consider the following three scenarios, which will present the main points that the security design model covers. Those scenarios use simple examples how the security design model facilitates high security. They all together present a complete security design guidelines pack for an application to operate in an untrustworthy environment.

4.1.1 Tom sign up at Alice

1. Alice generates a random asymmetric key pair, MK, and encrypts it's private key in an asymmetric cryptosystem with Tom's password, P, to produce P(MK). Alice saves P(MK) in the Database. The random asymmetric key, MK, is also called "Master Key".

2. Alice generates an unique password challenge, challengeP, using one-way function, F, and encrypts it with Tom's password, P, to produce P(challengeP). Alice saves P(challengeP) in the database.

3. Alice generate random cipher, also called "Value Key", VK. The cipher is encrypted in a symmetric cryptosystem with the *master key*, MK, to produce MK(VK). Alice saves MK(VK) in the database.

4. Alice encrypts all Tom's registration data in a symmetric cryptosystem with the *value key*, VK, to produce VK(data). Alice saves VK(data) in the database.

4.1.2 Tom sign in in Alice using his login and password, P

1. Tom send his login and password to Alice.

2. Alice confirm the login exist and then computes the one-way function, F, with Tom's password value, P, to produce F(P).

3. Tom is logged in, only if the value of F(P) match the P(ChallengeP) record in the database.

4.1.3 Tom changes his registration data

1. Tom sign up in Alice using his login and password, P.

2. Alice store the password, P, in an encrypted session.

3. Alice decrypt the *master key*, MK, using Tom's password, P, to produce P(MK).

4. Alice decrypt the cipher, C, using the *master key*, MK, to produce MK(C).

5. Alice decrypt the *value key*, VK, using the cipher C, to produce C(VK).

6. Alice decrypt the registration data, data, using the *value key*, VK, to produce VK(data).

7. Tom change his registration data.

4.2 Computation performance

The three scenarios shown above illustrate the security design model this papers presents. In this paragraph I will concentrate the attention on the computation performance behind this method. First, "Since the function F must be calculated once per login, its computation time must be small. A million instructions seems to be reasonable limit on this computation. If we could ensure, however, that calculation of F $^{-1}required10^{30}$

or more instructions, someone who had subverted the system to obtain the password, P, could not in practice obtain P from F(P), and could thus not perform an unauthorized login" [21]. If F(P) is provided as an input for , F(F(P)) will be calculated and the result will still not match the F(P) sored in the database. Second, storing the registration data asymmetrically encrypted will cost a lot of performance time to decrypt (see chapter 3.4), therefore the security design model suggest the following structure: The master key (MK), asymmetrically encrypted with the user's password, unlocks the value keys (VK) for each table, symmetrically encrypted, and the VK unlocks the data, symmetrically encrypted. That structure implies certain database schema, the master key is stored in the users table and value key/s are stored in the table/s we want to secure. Like this the asymmetric decryption is performed only once, when the user logs in, and not for every database query. And third, since we can't trust the server where our application and database are deployed, the password, unlocking the master key, is stored in an encrypted session on the client side, e.g. browser's cookies. This stops the server from catching users' passwords, the most important keys overall.
This method provides security per database record per user and allows the application to be freely hosted on a untrustworthy machine. In mean time, the logic is easy to apply in any web application, e.g. e-commerce, e-government, online banking, etc.

5. Results

In this paper I've presented a security design model strongly oriented towards data integrity in a web environment and issues of trust. I've merged the knowledge found in different sources and created complete guidelines, that can be applied to any system providing web services, which ensure high data security in an untrustworthy environment. Particular techniques for user authentication and data encryption/decryption were addressed. The model was applied in a web application called "Alice", to prove its ability to cope with web service vulnerabilities. The following data flow diagrams will illustrate the scenarios' processes of encryption and decryption shown in chapter 4:

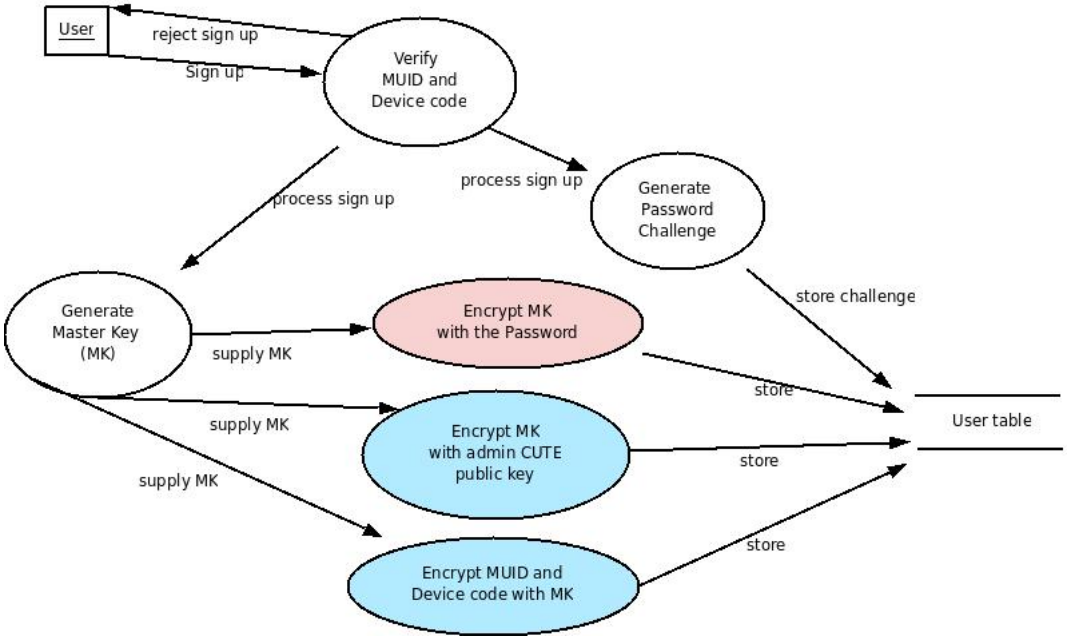

Figure 5.1: User signup

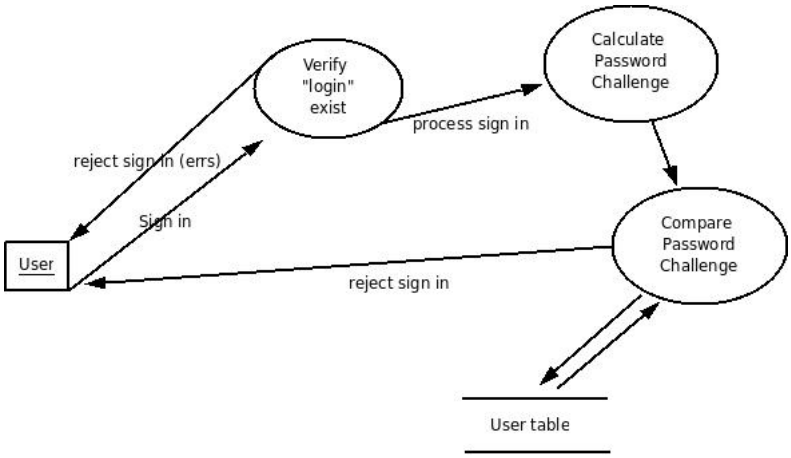

Figure 5.2: User signin

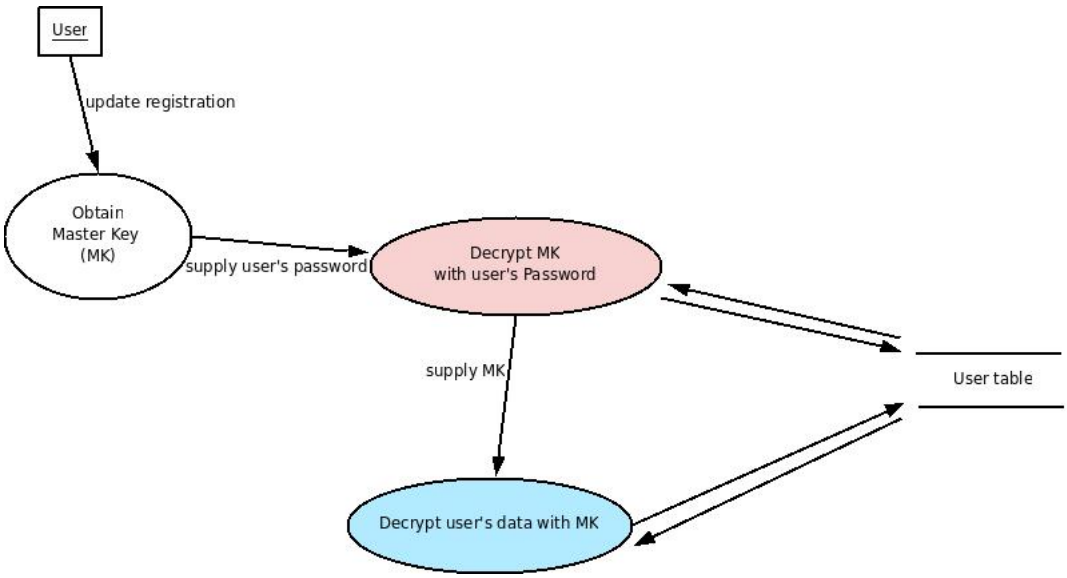

Figure 5.3: User update data

6. Conclusion

In this paper I have discussed security model guidelines and their approaches to disseminate and exchange information securely. However, those are design guidelines and considerations and not comprehensive framework to address the multifaceted security issues related to Web-based applications.

Achieving web application security in a untrustworthy web environment is a difficult task, because the environment can spy and follow all the data flows. The security design model don't stop the environment to see and to follow all the data, but ensures that the data is securely encrypted during the whole process. The security design model has several desirable features such as flexibility, adaptability, better support for security management and administration extendability and other aspects that make it attractive candidate for developing secure Web-based applications.
Let me explain the security design model in simple words. Every user has a password, that only he knows. Instead of storing the password in the database encrypted with the common encryption algorithms, e.g. SHA, MD5, etc., which are easily unlocked by dictionary attacks, combinatoric attacks, etc., the security design model suggest to generate an unique "challenge" for the user's password, using one-way function, and to store that in the database instead. That feature ensures the user's authentication process is highly secured.
The security design model implies certain database schema. Every table storing user's private data should have a column called "value key", which is used to store the symmetrically encrypted value key. The value key is a symmetric key, which encrypt and decrypt user's private data in that table's records. In addition, the value key is encrypted with the asymmetric key, called master key. The master key is unique for every user and is stored in the user's table, the "master key" column. The master key is also encrypted with cipher only the user knows, and this is the password he choose during sign up.
Because all these master and value keys are randomly generated, there is no way for the environment to know them. Still, the user's password is the most important key, because it can be used to decrypt the master key, and the master key can decrypt the value keys and the value keys can decrypt user's data. But that key is not stored in the database and the environment doesn't know it. The security design model strictly followed guarantees that the data in the system is secured per user.
A weakness of the security design model is password retrieve process. If the user losts or forgets his/her password, there is no way for the system to retrieve it, but the user's data is not unrecoverable encrypted (lost). The security design model propose to keep a journal of the database activities. This journal entries are encrypted with an asymmetric public key, its corresponding asymmetric private key is own by the system administrator and he/she can unlock the all the users' data with it.

Some researches claim that if the same asymmetric key is used to encrypt many values, a pattern can be discovered and data leak is possible. The security design model is protected against this vulnerability. As i said earlier the asymmetric key, master key, encrypt the value keys for every table, but even if the system's database has 100 tables, these are still not enough values to discover a pattern, consequently not possible to hack the system in that way.

6.1 Future work

I have presented the security design model using user's password as a main key to encrypt/decrypt private data stored on untrusted machine. There are many security design models, that provide high security, but my main goal, however, was to be able to host the application and its database in an untrustworthy environment. I demonstrated through the case study "Alice" that it can be applicable to any system with focus on security, because of its flexibility and simplicity. The case study exhibited that it can be very beneficial and helpful for systems trying to reach their quality goals. Still, I expect some people might say that I haven't provide a solution without a problem. And yes, I can foresee few improvements which the security design model does not suitably address. First, user inadvertent disclosure of his password, e.g. weak password selection. This problem "cannot be prevented by any password protocol, since two individuals presenting the same password information cannot be distinguished by the system" [15]. Second, securing database record data per user can be improved by addressing new unique parameter to the model and like this securing the database record data per login session. Third, the application can fully avoid any data decryption on the untrusted server, by moving the decryption process to the user's web browser. This on the other hand will require the use of additional plug-ins or technologies like JavaScript to do the decryption. Finally the security design model needs to be validated by comparing it with the other already existing security models. All these are subject to a future research.

Acknowledgements

I would like to thank Ingo Schurr, Heiko Schilling and Andreas Priesnitz for their assistance and support. Ingo Schurr helped me by presenting the initial concept of the security design model, which allowed me to expand it to what it is now. Andreas Priesnitz made a number of helpful observations, especially about encryption per record per session. Heiko Schilling was my team leader and project manager for the Alice project. Special thanks to Per Zaring for helping to capture and present this knowledge to you in the most readable, understandable and easy to follow way.

References

[1] Srs "alice" project.

[2] T. P. Abel, D. E. Rout. The mapping of software quality engineering to iso 9126. *Australian Computer Journal*, pages 1–14, August 1993.

[3] Khelifi A. Suryn W. Seffah A. Abran, A. Consolidating the iso usability models. *Proceedings of 11th International Software Quality Management Conference (Springer)*, April 2003.

[4] Khelifi A. Suryn W. Seffah A. Abran, A. Usability meanings and interpretations in iso standards. *Software Quality Journal*, pages 325–338, 2003.

[5] Diffie W. Rivest R. Schneier B. Shimomura Ts. Thompson E. Wiener M. Blaze, M. Minimal key lenghts for symmetric ciphers to provide adequate commercial security: A report by an ad hoc group of cryptographers and computer scientists, 1996.

[6] L.E. Chua, B.B. Dyson. Applying the iso9126 model to the evaluation of an e-learning system. *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference*, pages 184–190, December 2004.

[7] D. Clark. Next-generation web services. *IEEE Internet Computing*, 6(2):12–14, March 2002.

[8] Januszewski K. Colgrave J. Using wsdl in a uddi registry, version 2.0.2. *OASIS*, 2004.

[9] John W. Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications, July 2002.

[10] A. C. Yao D. Dolev. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 1983.

[11] Don Davis. Kerberos plus rsa for world wide web security. *In Proceedings of the USENIX Workshop on Electronic Commerce*, July 1995.

[12] Levin L. A. Goldreich, O. A hard-core predicate for all one-way functions, 1989.

[13] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1), 2004.

[14] Heather Kreger. Web services conceptual architecture (wsca 1.0). *IBM Software Group*, May 2001. redebooks.ibm.com.

[15] Leslie Lamport. Password authentication with insecure communication, 1981.

[16] Furnell M. S. and Karweni T. ecurity implications of electronic commerce: a survey of consumers and businesses. *Internet Research: Electronic Networking and Policy*, 9(5):372–382, 1999.

[17] Bruce Schneier.

[18] A. Singhoi. Goriška eregion a new vision that can be accomplished by educating citizens and improving esecurity. *Preceeding of the fifteenths Blend Electronic Commerce Conference*, pages 629–635, 2000.

[19] Cucchiarelli A. Panti M. Valenti, S. Computer based assessment systems evaluation via the iso9126 quality model. *Journal of Information Technology Education*, pages 157–175, 2002.

[20] Joan Ernst van Aken. Management research as a design science. *British Journal of Management*, 16(1):19–36, March 2005.

[21] Martin E. Hellman Whitfield Diffe. New directions in cryptography.