International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-03, Issue-12, Mar 2018

Mobile Self Encryption Techniques

[1]Mitesh Parmar, [2]Summedh Kharat, [3]Nilesh Palve, [4]Chetan deokate, [5]Prof. Yogesh shahare

[1,2,3,4,5]Department of Information Technology, Mahatma Gandhi Mission College of Engineering and Technology, Kamothe, Navi Mumbai, Maharashtra, India.

***Abstract*—This paper allows users to store sensitive data on their mobile phones without having to worry about is confidentiality even if the mobile is lost. This system is developed so that employees and other mobile users can store and operate on sensitive data on their mobile phones without having to worry of it being leaked. This software project concentrates on securing data on mobile phones by storing it in an encrypted form. This data is encrypted with a stream cipher whose key is stored on a server. When the mobile device is lost it sends a report to the server and the server then destroys the respective key so that the data on that mobile can never be encrypted and remains confidential.**

Keywords – encryption technique, Mobile, security, confidential.

I. INTRODUCTION

The pervasive use of wireless networks and mobile devices has been changing our living style significantly, Along with great convenience and efficiency, the progress of technology also brings new challenges in protecting sensitive and/or private information carried in these devices. New vulnerability results from unique characteristics of mobile devices. For instance, due to constraints imposed by limited computing power, storage space, and battery lifetime, a light-weight rather than computing intensive and complex encryption algorithm is desired in the mobile devices.

In addition, portability makes mobile devices prone to being stolen or lost. It is very challenging to protect the weakly encrypted information on a mobile device, which might end up in the hands of an adversary, who could then use powerful cryptanalysis tools to break the encryption. Therefore, security solutions developed for general distributed data storage systems cannot be adopted directly for this new frontier. Statistics show that 22% of PDA owners have lost their devices, and 81% of those lost devices had no protection. Even worse, 37% of PDAs have sensitive information on them, such as bank account information, corporate data, passwords, and more. For this reason, some companies do not allow employees to use PDAs or similar mobile devices to store company data . However, effective protection that would enable the full and convenient use of these devices without the fear of losing or compromising data would be a much better scenario.

The most challenging part of mobile device data protection lies in the conflicting requirements for the data encryption scheme. While it should be computationally infeasible for adversaries to decrypt the data in captured mobile devices, the encryption/decryption operation should be reasonably efficient for legitimate users. Furthermore, the required computations should not consume too much energy so as to minimize battery drain. This research proposes a novel stream cipher scheme called self-encryption (SE) to address this dilemma. Treating the data set as a binary bit stream, we generate the keystream by extracting n bits in a pseudorandom manner based on a user's unique PIN and a nonce. The length of the keystream n is flexible and depends on the security requirements. Then we encrypt the remaining bit stream using this keystream. The encrypted remainder is stored in the mobile device, whereas the keystream is stored separately. It is very difficult to recover the original data stream from the ciphertext even if an adversary has the knowledge of the encryption algorithm. The variable length keystream makes brute force attacks infeasible, and the decrypted data stream is still unrecognizable unless the keystream bits are inserted to the original position.

The rest of the paper is organized as follows: Section 2 provides a brief review of related work. Section 3 presents the framework of our self-encryption (SE) scheme and the detailed SE design. Section 4 proposes a protocol that specifies the behavior of both mobile devices and the secure server to support the SE operations. Section 5 summarizes this paper with an introduction to our on-going efforts.

II. BACKGROUND

Cloud is the most disruptive change in computing in the past 10 years. Its emergence is fast with an all-encompassing reach that will affect all of us in some way. It is clear that Cloud Computing has achieved escape velocity from the massive hype that initially had it tethered in low orbit to the point that it has now joined the ranks of popular culture.

Everyone, from technicians to technophobes, speak of 'The Cloud,' each with their own interpretation of why and how it matters personally. Whether you're an end-user, a small business, a large enterprise or government, Cloud Computing has something to offer everyone. Of course, arguments still arise over the vagaries of vocabulary and the definition of 'Cloud,' and downright religious crusades have erupted with dogmatic contention offering notions of 'true' versus 'false' Cloud. Luckily, over the last few years, we

 DOI : 10.18231/2454-9150.2018.0073

have seen these discussions become more empirical and move from 'what is Cloud' to 'how can I use it?'

A key concept in a DRP is the physical separation of the primary and backup sites. We do analysis the current IT based System Architecture and try to emphasize its existing problems. In spite of maximum use and high level of maturity, there are still many problems in existing Architecture. Here, we try to address only main problems like network traffic, disaster recovery, and interoperability. Among these three main problems we consider more on disaster recovery because very less work has been done in this area. We commence on discussing on proper solution for disaster recovery with Disaster Recovery Plan (DRP).

It represents the past and present status of disaster recovery. In most of the traditional DRP's system, multiple replication of a same system is made and placed into different geographic location as a copy of primary system. If disaster happens in primary location, then the secondary location will provide the information. Mostly this is done in two ways. One is online backup i.e. if the system is down then another same type of system will be up and provide the information which is also known as Disk Mirroring. Another approach is offline backup. Here, the system of another location does not start immediately like online but the data are brought back to the victimized system and wait till the system gets ready to take data.

When the causes of the primary failure have been addressed and the switch is made back to the primary, the switch is termed a failback. A number of options arise depending on the nature of the backup site and how it links to the process at the primary site. The backup situation is often described as follows.

- Cold standby: Recovery in such a case requires hardware, operating system and application installation. This recovery can take multiple days
- Hot standby: This requires a second data center that can provide availability within seconds or minutes. A hot site can take over processing while the main site is down. A complete copy of the primary process may sometimes exist in the backup, with no need to install either the OS or the application.
- Warm standby: A tradeoff between a hot and a cold site. It should be noted that the terms "hot" and "warm" are sometimes defined differently.

In order to satisfied the continuity of application and the security of data, the structure of disaster recovery system is "distributed computing, centralized storage". According to different requirements, disaster recovery has three levels. They are data-level disaster recovery, system level disaster recovery, and application-level disaster recovery. Data-level disaster recovery is the most basic and it can ensure the security of the application data. System level disaster recovery disaster has further requests for operating system of application server, making disaster recovery time can be as short as possible. System-level disaster recovery requests real-time by which users could not feel that any disaster occurred.

III. EXISTING METHODS

4. SE Protocol Design

To secure the sensitive data in mobile devices, a protocol set is mandatory to support the functionalities of the SE stream cipher, the AD agent, and the server. In addition, the protocol specifies the behavior of the whole system. At the mobile device side, the major functions include:

1) Setting up connection with the remote server;
2) Retrieving the keystream and nonce for local decryption;
3) Generating a new keystream with a new nonce and encrypting the document; and
4) Transferring the updated keystream and new nonce back to server.

At the server side, the SE protocol supports two working model: normal model and emergent model. As implied by its name, the normal model (NM) consists of the working flow when the mobile device is used normally by the legitimate user. The emergent model (EM) is a status that is triggered when a mobile device is reported lost. In fact, EM specifies the countermeasures to be executed when the device is in the hand of an adversary.

IV. CONCLUSIONS AND DISCUSSIONS

Lack of effective protection of sensitive data in mobile devices is a major concern that prevents the mobile devices from being used widely as part of enterprise networks or personal area networks. The proposed SE system will remove the barrier and enable employees to enjoy the high efficiency and convenience brought by mobile devices. It will lead to another wave of prosperity of wireless networks and pervasive computing. Physical attacks have been proved effective in breaking some well designed ciphers in practice [13]. Unfortunately, it is challenging to designers to theoretically investigate the robustness of a cipher scheme against various physical attacks. To address this problem, a prototype is going to be implemented on top of reconfigurable hardware devices (i.e. FPGAs). Particularly we will study the behavior of our SE prototype under local non-invasive attacks including timing analysis and differential power analysis (DPA) [20].

IV. FIXING THE SELF-ENCRYPTION SCHEME

We think that the core idea of the Self-Encryption scheme, i.e. exploiting the availability of a secure server not accessible to the adversary in order to strengthen the encryption, is brilliant and can solve some of the problems which currently affect mobile device security. Mobile devices are feature-crippled if compared to their desktop

 DOI : 10.18231/2454-9150.2018.0073

counterpart. Processing power and battery capacity limitations do not allow the execution of complex tasks, and the limits imposed by their input interfaces are well known. However, the advances in embedded processors and the availability of fast stream ciphers, such as RC4 [15], Rabbit [16] or Salsa20 [17], make the use of such ciphers viable on resource constrained platforms. We believe that such ciphers can be used as an efficient building block for a strongly secure encryption scheme, and we detail one of the possible constructions in the following sections.

A. The RC4 Stream Cipher

Stream ciphers are well known for their high speed and small memory footprint compared to block ciphers. Our interest goes towards RC4, a small pseudo-random generator, due to its fame and widespread availability. RC4 was introduced in 1987 by Ron Rivest [15]. Today, among other uses, it is one of the available ciphers for SSL/TLS [22] (secure socket layer / transport layer security). It is composed of a key scheduling phase, where the secret key – typically of length between 40 and 256 bits – is used to initialize an internal state. The stream sequence is then produced by outputting specific values from the internal state, which is updated after each byte of output is produced. Unlike many other stream ciphers, such as those in eStream [23], RC4 does not take an initialization vector together with the key.

Therefore each implementation must specify how to combine the initialization vector and the key in order to generate each time a different RC4 keystream. A survey of some of the approaches to combine the key and the initialization vector is given in [24]. RC4 is widely considered secure when used appropriately. Specific care must be invested in order to make sure that the pseudorandom generator is used properly, since "plain" RC4 is now known to have some weaknesses. For one, the first few bytes of its output are known to be biased [25]. Therefore, when RC4 is used, the first 128 or so bytes of its output should be discarded. Together with this weakness, there are resynchronization problems (see for example [26] and [27]). We assume that the implementation of a systems which uses RC4 takes the appropriate countermeasures against known weaknesses.

B. Ciphers Performances on Mobile Devices

In order to determine the performance impact of encrypting documents on the device using a standard cipher, we measured the speed of some widely used stream and block ciphers on a modern smartphone. Our testbed is the Apple iPhone 3G. Released in 2008, it is an interesting device, since it is a full Unix machine in a very small package. It is powered by a 416MHz Samsung 32-bit RISC CPU, which is a modern low powered in-order CPU based on the ARM instruction set, rated 0.45 mW/MHz (with cache), with 16KB of level 1 instruction cache and 16KB of level 1 data cache [18].

The iPhone 3G comes with 128MB of on-board RAM. ARM based CPUs are currently the most widely used CPUs in mobile devices, accounting for approximately 90% of all embedded 32-bit RISC processors. Similarly clocked or even faster ARM-based processors can be found on many Nokia E and N series smart phones, Microsoft Zune, Nokia N800 and N810 tablet, Motorola RIZR, most Windows Mobile smartphones, Palm Pre, Samsung Omnia, Blackberry devices, and Google's Android-based devices, just to cite some. We were able to install OpenSSL [19] on the iPhone in order to perform some tests. OpenSSL is a well known open source library which provides efficient implementations of many symmetric and asymmetric cryptographic algorithms, together with various utility functions. Since the only stream cipher available with OpenSSL is RC4, we were unable to compare it with other stream ciphers like Rabbit or Salsa20 at this time.

However, an implementation which uses OpenSSL as a cryptographic library would necessarily have to use RC4, therefore we think that our measurements are meaningful. We compared RC4 with the AES and DES block ciphers using a 128-bit and a 56-bit key respectively, in order to provide reference values.

Second for The Specified Plaintext Size. The Block Ciphers Are Used In Cbc Mode.

64 Bytes 256 Bytes 1 KB 8 KB
RC4 19,847 20,438 21,294 20,934
AES-128 5,303 5,504 5,740 5,611
DES 3,492 3,559 3,681 3,623
Triple DES 1,269 1,324 1,266

V. CONCLUSIONS AND DISCUSSIONS

Lack of effective protection of sensitive data in mobile devices is a major concern that prevents the mobile devices from being used widely as part of enterprise networks or personal area networks. The proposed SE system will remove the barrier and enable employees to enjoy the high efficiency and convenience brought by mobile devices. It will lead to another wave of prosperity of wireless networks and pervasive computing. Physical attacks have been proved effective in breaking some well designed ciphers in practice [13]. Unfortunately, it is challenging to designers to theoretically investigate the robustness of a cipher scheme against various physical attacks. To address this problem, a prototype is going to be implemented on top of reconfigurable hardware devices (i.e. FPGAs). Particularly we will study the behavior of our SE prototype under local non-invasive attacks including timing analysis and differential power analysis (DPA) [20].

REFERENCE

[1] J. Al-Muhtadi, D. Mickunas, and R. Campbell, "A Lightweight Reconfigurable Security Mechanism for 3G/4G Mobile Devices," *IEEE Wireless Communications*, April 2002.

 DOI : 10.18231/2454-9150.2018.0073

[2] D. J. Bernstein, "Which eSTREAM ciphers have been broken?" http://www.ecrypt.eu.org/ stream/, submitted 2008- 02-21.

[3] A. Biryukov, "Block Ciphers and Stream Ciphers: The State of the Art," *Lecture Notes in Computer Science, in Proceedings of the COSIC Summer course*, 2003.

[4] A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data tradeoffs for stream ciphers," in *Proceedings of Asiacrypt'00*, no. 1976 in Lecture Notes in Computer Science, pp. 1–13, Springer-Verlag, 2000.

[5] W. Daniel, T. Pintaric, F. Ledermann, S. Dieter, "Towards Massively Multi-User Augmented Reality on Handheld Devices", *International Conference on Pervasive Computing, Munich*, Germany, 2005.

[6] D. E. Denning and D. K. Branstad, "A Taxonomy for Key Escrow Systems," *Communications of the ACM*, Vol. 39, Issue 3, 1996.

[7] eSTREAM, ECRYPT Stream Cipher Project, http://www.ecrypt.eu.org/stream.

[8] N. Fournel, M. Minier, and S. Ubeda, "Survey and Benchmark of Stream Ciphers for Wireless Sensor Networks," the *Workshop in Information Security Theory and Practices* (WISTP'07), Crete, Greece, May 8-11, 2007.

[9] A. O. Freier, P. Karlton, and P. C. Kocher, "The SSL Protocol, Version 3.0," *Internet draft*, Networking Group, March 1996.

[10] C. Galdi, A. Del Sorbo, and G. Persiano, "Distributed Certified Information Access for Mobile Devices," *Workshop in Information Security Theory and Practices* (WISTP'07), Crete, Greece, May 8-11, 2007.

 DOI : 10.18231/2454-9150.2018.0073