SECURE PIN AUTHENTICATION IN JAVA SMART CARD USING HONEY ENCRYPTION

Sadeq mohammed
Department of Information Technology
Altinbas University
Aghalib57@gmail.com

Sefer KURNAZ
Department of Electrical And Computer Engineering Altinbas University
sefer.kurnaz@altinbas.edu.tr

Alaa Hamid Mohammed
Department of Electrical And Computer Engineering Altinbas University
Aallaaha12@gmail.com

Abstract— **Java card is a system that can allow the programming and improvement of smart card technology by incorporating the java language into a virtual machine, it also introduces the concept of applets as a mean of implementing various management and protection measures on the smart card data, but the java card system also introduces new security challenges to the stage, in this paper, we implement a security scheme based on the concept of honey encryption to overcome the threat of brute force attacks and denial of service attacks on the smart card password authentication.**

Keywords— Java Card, Smart Card, Denial of Service, Brute Force Attacks, Honey Encryption.

I. INTRODUCTION

The architecture of the Java Card system [1-3] improved three major aspects of the smart card technology, these aspects can be summarized as follows: first, portable java applications that can be transported to all java card platforms [4]; second, the java card can execute multiple applications simultaneously, which uses share object to communicate with each other [5]; third, the development of new applications referred to as applets [6] the figure below Figure 1 further explain the java card architecture.

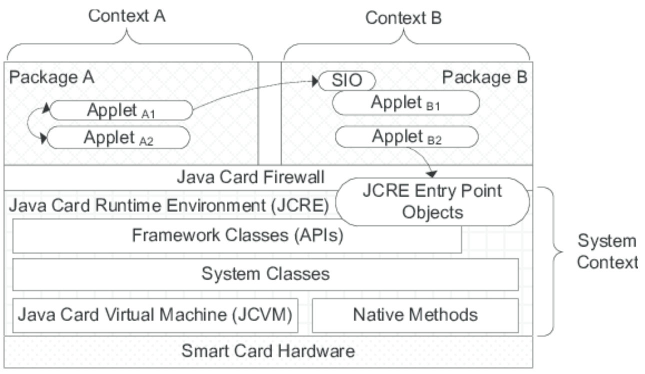


Figure 1: Architecture of Java card [6].

Smart cards store sensitive data that are targeted by attacks that aim to retrieve or manipulate that data [7], these attacks may be software attacks such as shareable interfaces mechanism abuse, CAP file manipulation, and transaction mechanism abuse [8-10]. Or hardware attacks such as fault attacks that uses physical manipulation to cause suspicious activity of the card which may lead to the leakage of sensitive information [11]. Software attacks are more common due to the fact that they are inexpensive attacks, software attacks aim to modify the software of the card without requiring any hardware implementation such as password brute force attacks and denial of service attacks [12-14]. [15] discussed the two of the most common smart card management architectures (GlobalPlatform and Multos). and explained why these two different architectures do not fully integrate to the GP-CCM and UCOM. [16] implemented a minutia-based matching algorithm for fingerprint matching in the smart card and reduced the computing time and required memory space.

[17] introduce a secure shell protocol (SSH) brute force attacks detection scheme by reviewing the server's unsuccessful attempts logs the drop in the firewall, [18] implemented a scheme to evaluate the effort and reliability in the brute-force attacks on a cipher using a statistical criterion, [19], reviewed the brute force attacks and how most of them exploit the software's vulnerability, [20] proposed a probabilistic encryption scheme that randomly generates cipher text to overcome the offline brute-force and dictionary attacks. most smart cards reduce the threat of brute force attacks by limiting the number of unsuccessful tries allowed [21], but this method makes it more vulnerable to the denial of service attacks. We propose a honey encryption based method to overcome both the brute force attacks and the DOS attacks, we implement our scheme by generating a random number of tries after every successful attempt to log in into the java smart card, and rather that resetting the smart card or shutting down the process, we generate new fake yet believable information to deceive the hacker into stopping the attack.

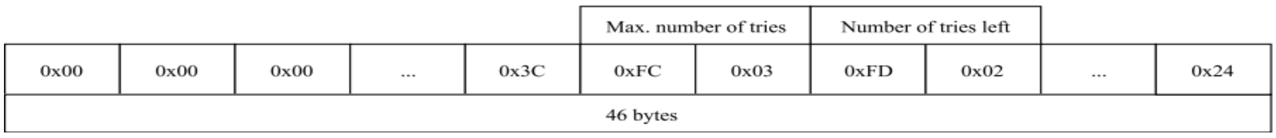
The rest of the paper is organized as follows. Section 2 describes the types of software and logical attacks on java card, Section 3 explains the design of proposed method, Section 4 deals with the simulation results our method, and finally we conclude our work in Section 5.

II. SECURITY CHALLENGES

The java card security is derived from the security of the java programming language, it features some security measures such as type mismatch checking, limiting the access of an array indices to the memory space of its allocated boundaries, doesn't allow pointer arithmetic unlike C and C++, classes, methods, and fields access is controlled no matter what the defined level of access [22] . The java card also has some security measures such as, bytecode verifier BCV, and global platform GP, and the java card firewall [23]. Because these security measures protect sensitive data, it is most often the target of software and hardware attacks, in this section, we review two of the most popular attacks that try to bypass the security measures of both the java card and the java programming language:

A. *Brute-force attacks*

Brute force is not an attack itself but rather a concept that malicious applets can perform in order to gain unauthorized access to the sensitive data in the java card, brute force attacks aim at guessing the right PIN or crypto-key or resetting the try counter by gaining access to the memory space allocated for the try counter container [24]. Of course, if the owner does not implement aby runtime counter measure for securing the try counter of the PIN these types of attacks will succeed ever time. It is not easy to specify which place in the memory the try counter is stored but it is easy to notice the four bytes in the data structure that indicate the maximum number of tries allowed as shown in the figure 2 below, the other downside of limiting the maximum number of failed tries is the vulnerability to denial of service attacks DoS [25].



					Max. number of tries		Number of tries left			
0x00	0x00	0x00	...	0x3C	0xFC	0x03	0xFD	0x02	...	0x24
46 bytes										

Figure 2: The 4-digit try counter stored in the memory space

B. *DoS attacks*

Denial of service attacks are the types of attacks that aim at shutting down a system by overwhelming it with information in order to make it inaccessible to the intended user, these attacks exploit the security measures of limiting the number of information or in this case the "try counter" to specific number, there are two types of Dos attacks [26]:

1. Flooding DoS attacks: which aim at flooding the services with unnecessary and redundant data in order to slow down a system or stop it from providing the service.
2. Crash DoS attacks: which aim at crashing a system by overloading its processors with by an overwhelming number of data.

III. PROPOSED COUNTERMEASURE

In this paper, we propose a honey encryption base scheme in order to countermeasure the brute force attacks aimed at guessing the user's password without having to face the risk of DoS attacks which results from limiting the maximum number of tries. Honey encryption is the concept of presenting the attacker with fake yet believable data that resemble the sensitive hidden data in the smart card but does not indicate the type of information that is hidden [27-30], the aim of this method is to prevent the attacker from proceeding in the brut force attack thinking that the brute force attack has successfully guessed the real PIN or password, we implement our method by presenting the attacker with feasible garbage information stored in the smart card after a random number of tries in not guessing the correct PIN or password, then we after each presented information or "honey" we generate a new random number for the maximum number of tries allowed, this way the random number does not indicate to the attacker that information presented is fake, the proposed method is further explained in the figure below Figure 2.

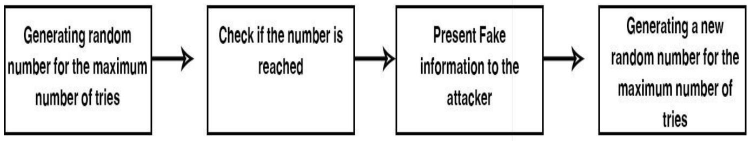

Generating random number for the maximum number of tries
Check if the number is reached
Present Fake information to the attacker
Generating a new random number for the maximum number of tries

Figure 3: The proposed honey encryption method

We use OwnerPIN() to set the maximum number of tries to enter the pin code, then we use the pinTriesRemaining() to return the remainder of the random number we generated of incorrect times to enter the PIN code, if the attacker maxed out the random number, we present the attacker with

fake data relevant to the type of information the attacker is trying to access yet irrelevant to any data stored in the smart grid as shown in the general code below:

```
if(pinTriesRemaining == -1){ System.out.println("Access
Granted") System.out.println("Fake Cardholder Name");
System.out.println("Fake Account Number");
System.out.println("Fake Expiry Date");
System.out.println("Fake Security Code");
}else{
System.out.println("Try again");
}
```

As shown in the code above, instead of blocking the attacker from re-entering the incorrect PIN after the random number of PIN tries is maxed out, we present the attacker with fake data that resemble that data that the attacker is trying to get, yet unrelated to any data stored in the smart card, we store multiple fake data in case of an offline attack to present to the attacker multiple times until the attack is stopped, this way we not only reduce the threat of brute force attack, we also eliminate the threat of DoS attacks related to blocking of the brute force attacker which is shown in the code below:

```
if(pinTriesRemaining == -1){
return true;
}else{
return false;
}
}
```

Due to the fact that one of the aspects of java card is the ability to install a modified applet, the attacker can install an applet which implements a brute-force attack with OwnerPIN object in order to reinput a 4 digit PIN that can be cracked within 15 minutes or less, some smart card restrict the access to the memory and performing a runtime try counter as an additional line of defense, although these methods are considered to be more effective in preventing side channel attacks rather than logical attacks [31,32].

IV. SIMULATION AND RESULTS

We simulated our attack on 5 various cards with different settings to simulate the different setting on various manufacturing techniques in smart cards, we implemented various attacks using the brute force approach in order to gain access to the data stored in the card, these attacks include: modifying metadata, using getstatic b, 'spoofing' references, using 'illegal' opcodes, and accessing arbitrary objects using arrayCopyNonAtomic,only Card_d was able to countermeasure all these attacks using our proposed method described in section 2 the table below Table 1 shows the result of these attacks on various simulated smart cards:

Table 1: The Modifications caused by the attacks implemented on various cards

Attack	Card_a	Card_b	Card_c	Card_d
Changing a PIN try counter	✓	✗	✓	✗
Retrieving a plaintext PIN	✓	✗	✓	✗
Retrieval a plaintext RSA key	✓	✗	✓	✗
APDU buffer array reference	✓	✓	✗	✗

The changing of the pin counter in Card_a and Card_c was successful because the brute force attack created a type confusion in the memory space where the counter lies, while failed in Card_b because the card restricted the access to the memory space and used a small size of 64KB memory to store the counter. The APDU buffer array referencing failed in Card_c because the card restricted the authorization of installing of CAP files with library package, Card_d is the "honey card" where we implemented our scheme resisted the attacks and was able to stop the brute-force attack without allowing any of the modifications inflicted to the other cards..

V. CONCLUSIONS

Java smart cards are a technology that inherited the security measures of both the smart card technique and the java programming language but due to the fact that it stores sensitive and valuable data it is in most cases the target of inexpensive logical attacks, namely the denial of service attacks and the brute force attacks, in this paper we propose implementing the concept of honey encryption on java smart card programming to prevent both of these attacks, we simulated our method and compared it to some security implications of the brute force attacks and found that our method not only reduces the threat of brute force attacks rather prevent it from gaining access to the sensitive data by stopping it permanently. In the future, instead of using pre-stored fake data, we propose a Deep learning approach such as the generative adversial network GAN to generate data from the sensitive data stored in the memory that resemble the nature of the stored data but does not indicate any information related to it.

REFERENCES

[1] L. Audah, U. Tun, H. Onn, M. M. Hamdi, and S. Alani, "Prediction Based Efficient Multi-hop Clustering Approach with Adaptive Relay Node Selection for VANET," no. March, 2020.

[2] M. Hammouda, "Resource allocation for 5G technologies under statistical queueing constraints." Hannover: Institutionelles Repositorium der Leibniz Universität Hannover, 2019.

[3] M. M. Hamdi, L. Audah, and S. A. Rashid, "A Survey on Data Dissemination and Routing Protocol in VANET : Types , Challenges , opportunistic and Future Role A Survey on Data Dissemination and Routing Protocol in VANET : Types , Challenges , opportunistic and Future Role," no. May, 2020.

[4] A. T. Al-heety, M. Singh, J. Singh, M. T. Islam, and A. H. Ahmed, "MM - wave backhauling for 5g small cells," no. May, 2019.

[5] M. M. Hamdi, L. Audah, and S. A. Rashid, "Coarse WDM in Metropolitan Networks : Challenges , Standards , Applications , and Future Role Coarse WDM in Metropolitan Networks : Challenges , Standards , Applications , and Future Role," 2020.

[6] C. O. Thein, "A frequency domain approach to synchronization of filterbank multicarrier systems in practice." Hannover: Gottfried Wilhelm Leibniz Universität Hannover, 2015.

[7] Y. R. Mohammed, N. Basil, O. Bayat, and A. Hamid, "A New Novel Optimization Techniques Implemented on the AVR Control System using MATLAB-SIMULINK A New Novel Optimization Techniques Implemented on the AVR Control System using MATLAB-SIMULINK," no. May, 2020.

[8] S. Bharati, M. A. Rahman, P. Podder, M. A. Islam, and M. Hossain, "Bit Error Rate Analysis of M-ARY PSK and M-ARY QAM Over Rician Fading Channel," *arXiv Prepr. arXiv2002.07392*, 2020.

[9] N. Linthoingambi Chanu, A. Dinamani Singh, and N. Loyalakpa Meitei, "Performance Analysis of QAM over One Wave Diffused Power Fading Channels," *N., Perform. Anal. QAM over One Wave Diffus. Power Fading Channels (January 9, 2020)*, 2020.

[10] L. Wu, Z. Zhang, J. Dang, Y. Wu, H. Liu, and J. Wang, "Joint User Identification and Channel Estimation Over Rician Fading Channels," *IEEE Trans. Veh. Technol.*,

2020.

[11] C. Li, J. Yao, H. Wang, U. Ahmed, and S. Du, “Effect of Mobile Wireless on Outage and BER Performances Over Rician Fading Channel,” *IEEE Access*, vol. 8, pp. 91799–91806, 2020.

[12] S. A. K. Tanoli, A. Mustafa, F. Nawaz, I. Khan, M. Usman, and Z. A. Khan, “SER and throughput analysis of space–time analog network coded relaying system over shadowed Rician fading channels,” *Wirel. Networks*, vol. 25, no. 8, pp. 5045–5056, 2019.

[13] G. S. Prashanth, “Comparative Analysis of Various Digital Modulation Techniques for FHSS-WCDMA over AWGN and Fading Channels,” *Int. Res. J. Eng. Technol. (IRJET), e-ISSN*, pp. 56–2395, 2019.

[14] D. Sadhwani and R. N. Yadav, “On the average of the product of two Gaussian Q functions over η− μ and κ− μ fading channels using MRC diversity reception,” *IET Commun.*, vol. 13, no. 8, pp. 981–987, 2019.

[15] M. K. Soni and S. Thakral, “BER Computation of OFDM System Using Real Audio Signal for Rician Channel,” 2019.

[16] D. Das and R. Subadar, “Performance analysis of QAM for L-MRC receiver with estimation error over independent Hoyt fading channels,” *AEU-International J. Electron. Commun.*, vol. 107, pp. 15–20, 2019.