SECURITY ANALYSIS OF KLEPTOGRAPHIC ALGORITHMS

By

Anum Sajjad

A thesis submitted to the faculty of Information Security Department,
Military College of Signals, National University of Sciences and Technology,
Islamabad, Pakistan, in partial fulfillment of the requirements for the degree of MS in
Information Security

jan 2018

ABSTRACT

Kleptography is the study of stealing the secure data secretly and subliminally. It is the subdivision of crypto-virology. The concept of inserting backdoors was introduced two decades ago by Young and Yung but still it is a serious threat for modern cryptography. The attacker uses asymmetric cryptographic techniques to build the backdoor and later uses his own private key to reconstruct the secret key of the user. Different researches proved that exploiting implementation weakness of cryptographic algorithm needs less effort as compared to attacking its mathematical structure.

The SETUP (Secretly Embedded Trapdoor with Universal Protection) attack modifies the standard methods of generating public and private key pairs in such a way that the public information is meaningful for the attacker. Also, calculation of private key of the user in polynomial time is no more a hard problem for the attacker. The information leakage using backdoor does not require separate communication channel for the transmission of secret data. In present days, cryptography is using against the security of cryptosystems instead of protecting it. The user is not able to distinguish the output of an honest or a malicious cryptosystem.

This research presented the brief description of the history of backdoor attacks in practical cryptographic systems. Elliptic curve cryptographic concepts are discussed in this research. Then the term $Kleptography$ was introduced, followed by the definition of weak, regular and strong SETUP attacks. It also presents the proposed kleptographic attack strategy on a cryptographic algorithm based on elliptic curves algorithms i.e. Edwards-curve Digital Signature Algorithm, Elliptic curve Diffie-Hellman key exchange scheme, Elliptic curve Digital Signature Algorithm, Elliptic curve Integrated Encryption Scheme, Elliptic curve Menezes-Qu-Vanstone and Elliptic curve Qu-Vanstone implicit certificate scheme. In order to increase the security, the complexity of cryptographic algorithm's implementation is also enhanced. This makes extremely hard for the user to detect such malicious codes especially when they are introduced, in the code, very innocently. Finally, the strategy of running time analysis is presented in order to detect the presence of such kinds of backdoor attacks. The experimental results shows the successful detection of malicious code in an elliptic curve based protocols. The future work concludes the research work.

DEDICATION

This thesis is dedicated to

MY FAMILY, TEACHERS AND BELOVED COUNTRY PAKISTAN

for their love and endless support

ACKNOWLEDGMENTS

"In the name of Allah the Most Beneficial and the Most Merciful "

I am very grateful to God Almighty who gave me the strength to accomplish this thesis without His consent I could not have indulged myself in this task.

Secondly, i am very thankful to my supervisor, $Dr.MehreenAfzal$, for her support, useful discussion, suggestions and professional advice during my research. Her encouraging comments helped me in successfully completing my research work. She stayed a great source of inspiration for me all along and helped me grow into a better professional. I am really grateful for her worthless contributions in this research. I am also very thankful to my guidance committee members; Dr. Faisal Amjad and Mian Muhammad Waseem Iqbal for their support and corporation. I am grateful to my teachers; Dr.Abdul Ghafoor, Dr. Haider Abbas, Dr.Imran Rashid, Narmeen Shafqat, Dr.Rabia Latif and Waleed Bin Shahid for their support in my course work. Specially, I am very grateful to my best cryptography teachers Dr.Mehreen Afzal and Dr.Naveed Riaz.

I am very thankful to my parents and siblings for supporting me during my education. They have always stood by my dreams and ambition. Finally, i am very thankful to Military College of Signals.

Anum Sajjad

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

Advance Encryption Standard	AES
American National Standards Institute	ANSI
Dual Elliptic curve Deterministic Random Bit Generator	Dual-ECDRBG
Digital Signature Algorithm	DSA
Elliptic curve Diffie Hellman Key Exchange	ECDHKE
Elliptic curve Diffie Hellman problem	ECDHP
Elliptic curve Discrete Logarithmic Problem	ECDLP
Elliptic curve Digital Signature Algorithm	ECDSA
Elliptic curve ElGamal Encryption Scheme	ECEES
Elliptic curve Integrated Encryption Scheme	ECIES
Elliptic curve Qu-Vanstone Implicit Certificate	ECQVIC
Edward curve Digital Signature Algorithm	EdDSA
Federal Information Processing Standard	FIPS
International Electrical and Electronics Engineering	IEEE
International Organization for Standard	ISO
Key Derivation Function	KDF
Menezes Okamota Vanstone	MOV
Message Authentication Code	MAC
National Institute of Standards and Technology	NIST
National Security Agency	NSA
Pseudo-Random Number Generator	PRNG
Rivest Shamir Aldeman	RSA
Secretly Embedded Trapdoor with Universal Protection	SETUP
Secure Shell	SSH
Secure Sockets Layer	SSL
Transport Layer Security	TLS
Virtual Private Network	VPN

Chapter 1

INTRODUCTION

In September, 2013 [1], an article has been published by ProPublica, The New York Times and The Guardian, that reveal the secret documents belong to National Security Agency about an encryption project i.e. **SIGINT Enabling project** [2], with the code name of Bullrun. The aim of project is to weaken the encryption standards adopted by the world in oder to get the secret information about the web searches, emails, calls and messages.

NSA is spending billions of dollars to insert backdoors in cryptosystems to ensure surveillance. They inserted malicious codes inside the devices by requesting politely, through secret orders, by force or making compulsory for the government and private sector organizations [3]. Edward Snowden reveals that NSA paid RSA $10m to use malicious Dual Elliptic curve Deterministic Random Bit Generator in their cryptographic tools to bypass the security of secure devices [4]. They are playing the role of big brother to know the secrets of every state.

A system is always just as secure as the weakest link in the chain. So how much one can relay on properly implemented strong cryptosystems against Kleptographic attacks. Defending a cryptographic system is more difficult than attacking. It is a fact that the attackers does not publish each and every thing related to their research about the new types of possible cryptanalysis attacks against cryptographic devices. The defender of a cryptographic device only implements the security measures against known academic attacks. But resistance against the known cryptanalytical attacks are not enough to ensure the security of the cryptographic devices. NSA has more than 300 minds of best mathematicians and they are working to solve the puzzle of encryption,just imagine up to what extend they can go [4]. According to Filiol [4]:

"Why would USA present a secure encryption algorithm (Advance Encryption Standard) without any form of control"

1.1 Problem Statement and Objectives

Revelations over the past couple of years highlight the importance of understanding malicious and surreptitious weakening of cryptographic systems. Few research is done on the techniques of detection of cryptographic backdoors that are a serious threat for the security. In this thesis, an efficient way of detecting the presence of backdoors in elliptic curve based cryptographic algorithms is proposed.

Objectives of this thesis are:

- To give a firm, comprehensible detail of the Kleptographic attacks, threats and associated vulnerabilities, with an emphasis on the Secretly Embedded Trapdoor with Universal Protection attacks on very well-known elliptic curve based cryptosystems.
- To analyze the security of EdDSA signature algorithm against kleptographic attacks.
- To develop a detection strategy using Timing analysis technique against such types of attacks.

1.2 Contributions

The contributions of this thesis are summarized as,

- The Kleptographic attack technique is presented against elliptic curve based cryptographic protocols.
- Detection mechanism of the presence of cryptographic backdoor, on the bases of running time analysis of an honest and malicious implementation, is presented.

The high-level overview of proposed techniques is given here, their details are given in chapters to follow.

Trap door inserting mechanism is very attractive research topic in present days. With the increase in the size of key bits, the effort of encryption and decryption of data increases while the work of cryptanalyst is increased too. So instead of performing cryptanalysis techniques, it is easy for the attacker to insert backdoors in order to minimize the effort of finding secret key. US National Security Agency is putting effort to sabotage cryptographic standards. Products have showed that this type of attacks constitutes a real threat especially for military and defense agencies, and effective countermeasures are needed to protect security classified

information. In this untrusted world the user should always be suspicious if the internal system is secure or not. The trapdoor, which is used for government surveillance purpose, can be exploited by the criminals having bad intension because no one knows how to build them reliably.

The Chapter 2, presents the history of backdoor, how the backdoors are exploited by the attackers and the definition of term Kleptography along with literature review. The basic concepts of the mathematics involved in cryptography are presented in the Chapter 3, along with the introduction to elliptic curve cryptography and the complex mathematical concepts of point addition and multiplication. In olden days the cryptography is used to secure the communication channel against threats and world would trust blindly on the implementation of cryptographic algorithms. But now trusting the manufacturer is the most difficult task. The Secretly Embedded Trapdoor with Universal Protection attack mechanism is presented in Chapter 4. The manufacturer uses the asymmetric cryptography to insert the backdoor. The public key of the attacker is inserted inside the victim's device. The private key of the user is calculated by using the public key of the attacker. Later, the backdoor information enables the attacker to rebuild the secret key of the user. In a tamper proof cryptosystem if the malicious code is inserted in an efficient way then it is nearly impossible for the user to detect the presence of backdoor by only observing the output of the cryptosystem.

The idea of running time analysis, of both an honest and malicious implementation, is used to detect the presence of backdoor in elliptic curve based cryptographic algorithms. The test methodology is presented in Chapter 5. The running time of malicious and an honest implementation of elliptic curve based algorithms is observed and finally the coefficient of variance is calculated to analyzed the deviation of time. The Chapter 6 concludes the thesis along with the future research directions and countermeasures against backdoor techniques.

1.3 Motivation

In olden days cryptography is used to secure the communication against threats and world trust blindly on the implementation of cryptographic algorithms and different protocols in the cryptosystem. But now it is a big challenge to make sure whether the cryptosystem ,one is using to secure the communication, is trust able or not. In under developing country, IT industry is not very well developed so it is not possible for them to design their own

blackbox cryptosystems for secure communication. It is important for us to study different malicious ways of leaking key bits so that we can build appropriate trust level on the foreign cryptographic devices after analyzing the device against such threats. In present, we have to trust on the providers and vendors. But the least we can do is to make sure that the blackbox we are using is actually secure after testing that system. Cyber security pointed out the serious threat related to the insertion of backdoors that bypass the encryption security and are exploited by the criminals or state hackers. A system is always just as secure as the weakest link in the chain. So it is important to get awareness of such attacks to prevent the most secure systems from this type of threats. Foreign security agencies are very active in research and practical implementations of such types of attacks. Such backdoors helps them to covertly get secret key to exploit the confidentiality and for other offensive aims. Kleptography and its defenses will become more significant in future because now more organizations are searching hidden ways to exploit the security of their enemy instead of wasting their time, money, resources and effort on traditional cryptanalysis techniques.

1.4 Thesis Outline

This thesis is divided into six chapters:

- Chapter 1: This chapter contains the introduction, objectives and motivation of thesis. It also explains the overview of the contributions we have made in this thesis report that includes the proposed kleptographic attack on elliptic curve based protocols and timing analysis techniques for detection of such backdoors.

- Chapter 2: This chapter contains the brief history of backdoor, the way they are exploited by the attackers in practical scenarios and the literature review of the thesis. It also includes introduction of the term Kleptography and Secretly Embedded Trapdoor with Universal Protection (SETUP) attack.

- Chapter 3: In this chapter, review of literature related to an elliptic curve cryptography and background mathematics is given along with the introduction of elliptic curves protocols and its mathematical explanation.

- Chapter 4: This chapter deals with the brief description of proposed backdoor inserting technique in an elliptic curve based algorithms.

- Chapter 5: In this chapter the test methodology used in our research was proposed along with the experimental results of the detection of backdoors in an elliptic curve based cryptographic algorithms.
- Chapter 6: This chapter concludes the report and future work is proposed along with the countermeasures against kleptographic attacks.

Chapter 2

HISTORY OF BACKDOORS

Introduction

This chapter gives the history of backdoors present in hardware or software. In section 2.1, the introduction of backdoors along with its types are discussed. In section 2.2, the brief description of historical backdoors present in different devices are presented. In section 2.3, the introduction of the term Kleptography is discussed along with the definition of regular, weak and strong SETUP attack. In section 2.4, the literature review is presented.

2.1 What are backdoors?

According to Edward Snowden, our secure hardware and software can be used against us in order to leak our private information. In 2013, the Edward Snowden reveals about the US campaign of inserting backdoors in cryptographic devices. The security agencies are spending years to identify the methods of detecting the presence of backdoor in a cryptographic devices. It is important for the security analyst to analyze the mathematical model of the cryptographic algorithm against inserting trapdoors attack [5]. It is hard to detect the presence of backdoors in practical systems if the backdoor is inserted carefully [4].

Backdoor is a malicious way of bypassing the encryption or authentication mechanism of a system. Mostly the backdoors are inserted, by the attackers, as a hidden code to get unauthorized access and to compromise a system. Backdoors are created in the secure system by inserting some special credentials that are hard coded in the device or by changing the security parameters. Major types of backdoors are:

- System Backdoor
- Application Backdoor
- Cryptographic Backdoor

System backdoors mostly allow the unauthorized access of the data at system level. In application backdoor, the attacker modifies the application or a software to bypass the security controls. The cryptographic backdoor aims the creation of cryptographic weakness in an algorithm that can be easily exploited by the attacker to get the public and private keys of the user [6].

2.2 The History of Backdoors

The word "*backdoor*" has a long history. In old age, the backdoors are created in the castles to ensure the security and safe exit during wars. In 1330 AD, the designers of Nottingham castle build a secret tunnel inside the castle as a secure and safe exit in case of attack [7]. That backdoor was later used by the attackers to destroy the castle. Also in Maginot Line Fortifications, the secret passage was created that helps them in war times. Similarly people insert the backdoor inside the cryptographic codes that leaks the secret data for the attacker without being noticed [7].

In 1980's, the secret accounts were created that allow unauthorized remote access to the system. In 1990's the government proposed a fair cryptosystem i.e. the key escrow proposal in which it is decided that the government and all other private companies save the copy of their private keys that are used to decrypt the data. Later on any authorized third party can access those keys for the justified reason. The information related to the private key is only given to the authorized third party when the court orders are presented. Some of the examples from the history are discussed here in which presence of backdoor in different hardware and software products is detected [8] .

2.2.1 Clipper Chip

Clipper chip, officially known as MYK-78, is the voice and data encryptor device of 1990's [9]. It was proposed by NSA, implemented for cryptographic phone and other secure equipments. The users save the copy of the private keys in the database of the government according to key escrow project. Such implementation permits the security agencies to get the copy of the secret key and decrypt any traffic of their choice. The Clipper Chip is shown in Fig.2.1.

The cryptographic algorithm used for encryption of data in the device is Skipjack (proposed

Figure 2.1: MYK-78 ”Clipper Chip”

by NSA) while key establishment takes place by using Diffie-Hellman key exchange protocol. The Skipjack algorithm was made public in 1998, it uses 80 bit length of key and a symmetric algorithm for encryption. They manufacture it as a blackbox device in which secret keys are inserted inside the device and a copy of that key is also saved in escrow. In order to avoid misuse of the private keys by the security agencies the key is stored after splitting it. Each security agency has one part of the key and to build the whole key the other part is also required. So it is necessary to access the database of other agencies to recover the encrypted data. The 80 bit secret key is build by XORing the other part of that key.

The Session key is established by using Diffie-Hellman key exchange protocol. The security agencies need 128 bit key in order to rebuild the private key used for decryption of data. To decrypt the message using clipper chip it is necessary to have a valid hash code of session key. This 128 bit key is stored in the following manner. First the 128 bit key is divided into parts and then each security agency stores one part in the database, this will avoid the misuse of the key. When ever the security agency wants to retrieve the key they made the request to access the databases of other agencies to get the other part. All the parts of the key are then XORed to establish the original 128 bit key. Fig 2.2 shows the process of establishment of 128 bit key.

According to the Law enforcement agencies this helps to increase the security. This project fails because of the development of other alternative cryptographic packages like PGP, Nautilus etc. According to the cryptographers, if the best cryptographic algorithms are freely

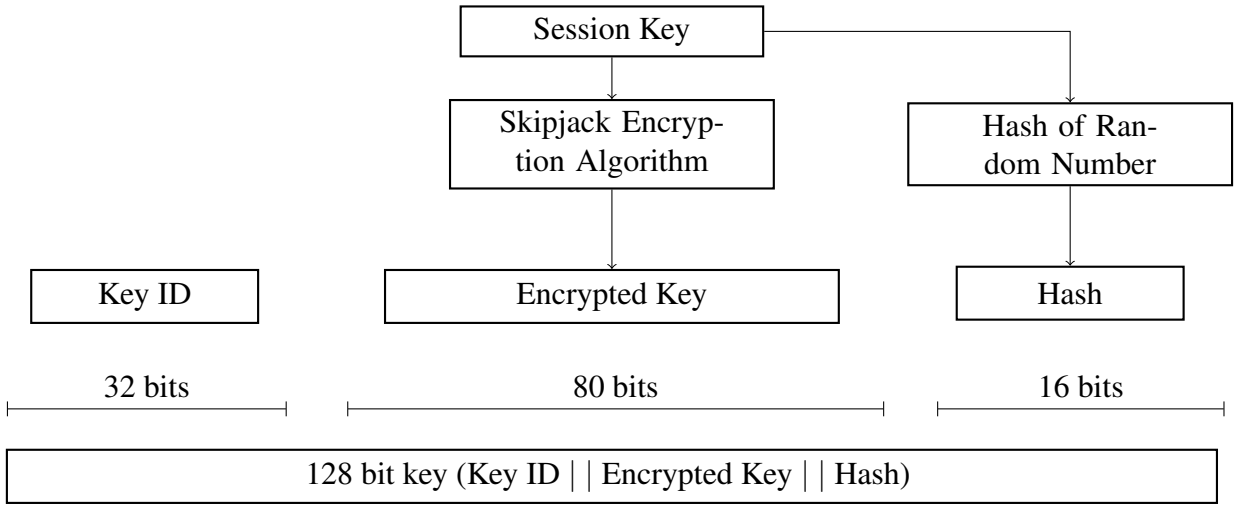

Figure 2.2: Key Establishment in Clipper Chip

available then the use of clipper chip can be avoided [10]. In April, 1993 the project of Clipper chip was announced. Then in July, 1993 NIST publish the review report to support the project. In 1994, FIPS 185 approved the use of Escrowed Encryption Standard [11].

2.2.2 Lotus Notes

In 1990's, the US government introduces the backdoor feature in $Lotus\ Notes$, called differential cryptography. Because of the presence of backdoor, the effective key length of an algorithm reduced to 40 bits. The device contains the public key of NSA in it. Commonly the device is named as "MiniTruth" or also called Big Brother.

The basic idea of the backdoor feature is that, when ever the device encrypts the plaintext, 24 bits out of 64 bits of the symmetric key used to encrypt the plaintext is encrypted with the public key of the NSA. Now the final ciphertext comprises of the ciphertext of the plaintext along with the ciphertext of the encryption key. The decryption on the other side will fails if the ciphertext contain wrong ciphertext of the key part calculated using NSA public key. The fact about the Lotus notes is public and mostly the users are aware of this weakness. Now the NSA or the owner of the public key have to brute force for 40 bits only rather than full 64 bits of key in order to recover the plaintext [12]. It is difficult for the adversary to exploit the backdoor if they don't have the knowledge of the private key required to decrypt the 24 bits of key part. NSA requested Lotus to intentionally weaken the encryption system so that the security agencies can easily decrypt the documents, emails, messages, secret chats etc. Fig 2.3 describes the process of encryption carried out in Lotus Notes.

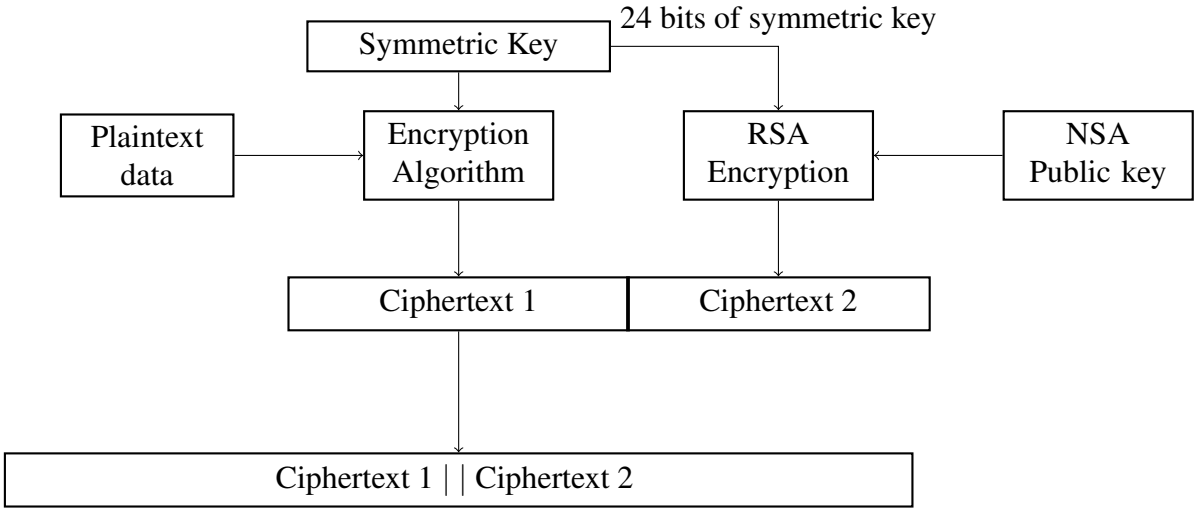

Figure 2.3: Example of Lotus Notes

2.2.3 Dual EC DRBG

$Dual\ EC\text{-}DRBG$ was a Federal Institute of Processing Standards (FIPS) standard up till 2014. Dual elliptic curve Deterministic Random Bit Generator contain a cryptographic backdoor that is only accessible to the creator of the algorithm i.e.NSA. This include a specific selection of elliptic curve points that later leaks the information of random bits generated using that device [13]. Moreover this random number generator is very slow and the output is biased. After 2007, the mathematicians indicates the presence of backdoor that leaks the random bit information and the flaw was made public. It was a surprise that the algorithm who faces a lot of criticism was included in the list of approved random number generator. According to [14], NSA paid 10 million dollars to use this backdoored random number generator as a default in BSafe project. The process of random number generation in Dual EC-DRBG is shown in Fig. 2.4.

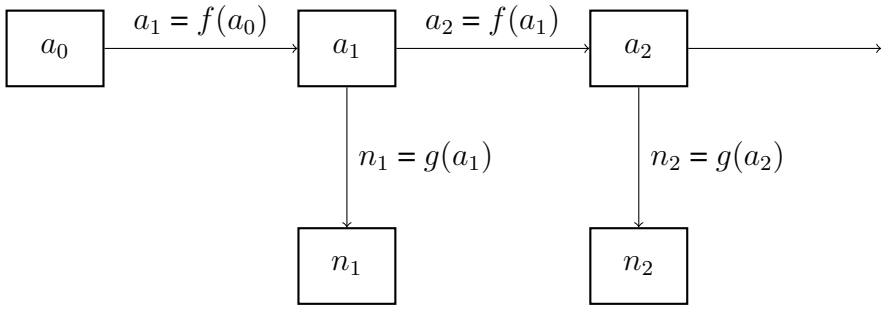

Figure 2.4: Dual Elliptic Curve Deterministic Random Bit Generator schematic diagram

In the Fig. 2.4, the f and g functions are elliptic curve point multiplications. When ever a random number is generated the internal state of the EC-DRBG is updated i.e. form a_0 to a_1. The updated value is calculated by using the elliptic curve point multiplication function f and g. If the attacker learn the internal state at any instant then he can easily predict the stream of random numbers generated by the EC-DRBG. NSA inserted a backdoored g function that enables the attacker to learn the internal states and then calculate the bits of generated random number.

According to NIST's standard elliptic curve, the malicious elliptic curve points P and Q are generated in such a way that $P = dQ$. These two points are preselected points and knowledge of d opens a backdoor for the attacker. The f and g functions return only the x coordinate of the elliptic curve point. Fig. 2.5 describes the malicious generation of random bits using Dual EC-DRBG.

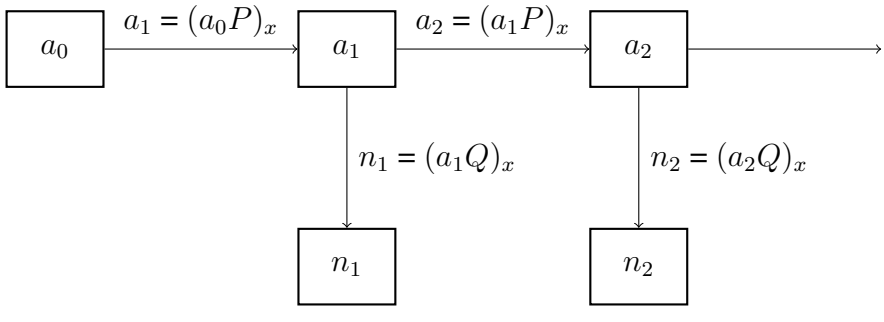

Figure 2.5: Malicious Dual Elliptic Curve Deterministic Random Bit Generator

Now from n_1 16 most significant bits are removed and remaining 30 bits are generated as random stream. So the attacker can easily know the 30 bits of the generated random data from the channel [15]. 16 bits can be easily brute force to recalculate the corresponding y coordinate of n_1 i.e. $N = (n_{1x}, n_{1y})$. Attacker knowns the secret value of d, so he can calculate the next internal state by using the following Equations 2.1, 2.2 .

$$N = (a_1 Q) \tag{2.1}$$

$$dN = d(a_1 Q) = a_1 dQ = a_1 P = a_2 \tag{2.2}$$

2.2.4 Debian OpenSSL PRNG

In 2006 [16], the non standard memory manipulation vulnerability was identified in $Debian\ OpenSSL\ PRNG$. The keys are generated in a malicious way that enables the attacker to recover TLS or SSH communication keys.

A pseudo random number generator takes the random seed as an input to introduce the randomness in the output stream. If the seed is same then the output bits of the pseudo random number generator is same. To introduce the randomness in OpenSSL, the processor ID i.e. OpenSSL function $RAND\_Bytes$ and some part of memory i.e. OpenSSL function $RAND\_add$ are used as the seed. In 2006, two lines of the source code were commented to fix the warnings generated by a security tool. This results in the decreasing the space of seed to 32,767 possible values [16]. Now the keys generated using OpenSSL is not random and can be easily predictable because the randomness is created by the processor ID only. The following code is commented form the $md\_rand.c$ file [17].

```
MD_ Update(& m,buf,j);
[....]
MD_ Update(&m,buf,j); /* purify complains */
```

2.2.5 Heartbleed

In 2014, the vulnerability was introduced in $Heartbleed$. It enable the adversary to read the protected memory of the system if vulnerable version of OpenSSL software was used [18]. The problem is in the OpenSSL cryptographic library used in the implementation of TLS or SSL. This malicious implementation lacks the capability of handling the Heartbeat Extension packets that cause buffer over read and the attacker can read the protected memory of the system.

2.2.6 Juniper Firewall

In 2015, the Juniper disclosed that in the result of an attack someone added a malicious code in the operating system of NetScreen VPN routers. As the result of this malicious code two vulnerabilities are created i.e. First, the attacker can bypass the authentication mechanism to get remote access and secondly, the attacker can decrypt the VPN data [19]. The Fig.2.6 shows the administrative password i.e. $<<< \%s(un =' \%s') = \%u$, to gain remote access.

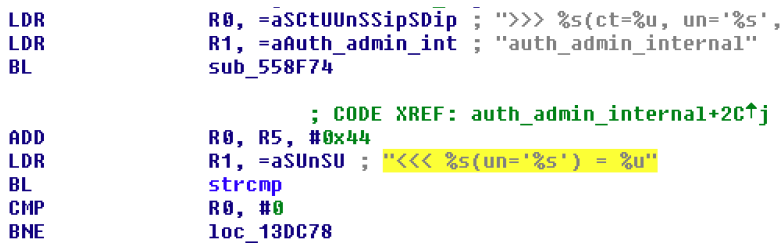

Figure 2.6: Juniper Hard-coded Password

The second vulnerability is because of the backdoor present in Dual EC-DRBG. The attacker inserts the Q i.e. the elliptic curve point, of his own choice that enables him to guess the generated random bit stream and then to decrypt the VPN traffic. In 2013, the updated version of Dual EC-DRBG was introduced that closes the backdoor for the attacker, but the Juniper was still using the older version of the Dual EC-DRBG at the time of attack. According to Langley [20]:

"In short, they used a backdoored RNG but changed the locks. Then this attack might be explained by saying that someone broke in and changed the locks again"

2.2.7 WhatsApp Backdoor

In 2016, WhatsApp introduces an end to end encryption service. According to the security experts it also introduces a backdoor that can be exploited by an attacker or the security agencies [21]. This backdoor enables the attacker to intercepts and read messages. When Alice sends the message to bob she has the public key of Bob. The message is encrypted by the private key of Alice and then the public key of Bob. Now on the other side first Bob uses his private key then the public key of Alice to read the message as shown below:

Alice	**Bob**
$S = E_{pubB}(E_{priA}(message))$	
$\xrightarrow{Sends(S)}$	
	$message = E_{pubA}(E_{priB}(S))$

If Bob is off line and Alice sends the message to Bob and before he receives the message, let he reconfigure his WhatsApp account. As a result Bob regenerate new pair of public and

private keys. When he becomes on line he will receive the message send by Alice. The message is re encrypted automatically by using the new keys generated by Bob. Now if the attacker sends his own key to Alice then the message for Bob is now encrypted by the attacker's key. The attacker can easily read the message that is originally meant for Bob.

2.2.8 Bitcoin Wallet Backdoor

The digital currency like Bitcoins are stored securely in wallets. But if the backdoor is inserted in the wallet the attacker can easily get the information related to private key of the user. According to Berlin [22], the attack is successful if the attacker only watches the blockchain until two malicious signatures of victim comes. The signature algorithm used in the Bitcoin protocol is Elliptic curve digital signature algorithm. According to [22], if the malicious version of signature algorithm is used then the attacker can easily construct the private key and nobody other than the attacker can exploit the backdoor to rebuild the secret key.

From past three years the $Underhanded\ Crypto\ Contest$ is being organized and its theme is to design the crypto backdoors in practical implementations that are difficult to detect [23]. In 2014, the winner of contest designs a backdoor for AES implemented in Internet of Thing devices. In 2015, the contest includes the discussion on GunPG and Password hashing. In 2016, the winner of the contest designs Backdoor for cryptocurrency that produces a valid signature for an invalid transaction.

Some other software and hardware backdoor includes the vulnerabilities in Back Orifice [24], DSL gateway made of Sercomm hardware [25], PGP full disk encryption [26], WordPress plug in [27], Joomla plug in [28], borland interbase [29], tcpdump [30], CowerSnail [31], facebook server [32], NetSarang's server management software [33], Android smart phone [34], Factoring as a service project [35], Backdoor in SAT [36], DES [37, 38], Monkey cipher [39] and many more. After Snowden revelations, the cryptographic research community has begun to admit that we are facing more strong adversary and they have more resources and computing power.

2.3 The Term 'Kleptography'

In today's modern world, Cryptography plays an important role in achieving security. It is very hard to design a cryptosystem that ensures proper implementation of cryptographic

protocol. In past, the cryptographic softwares and devices enable the users to gain advantage of cryptographic security without facing the trust issues but now the correctness of the cryptographic algorithm's implementation is itself a question mark. In blackbox cryptosystem the code is inaccessible to the user and inputs/outputs are the only things that are controlled by the user of that device. It is impossible to differentiate the output of an honest device from the output of a malicious device. So, blackbox cryptography can easily exploit the cryptographic device without being noticed.

A SETUP is a secure mechanism of inserting a trapdoor in a cryptosystem. The attacker uses public key cryptography to change the code in such a way that attacker's public key is inserted inside the cryptographic algorithm that is used to compute the private key of an honest user, while the private key of the attacker is used to derive the private key of the user. The SETUP attacks are of three types i.e. regular, weak and strong SETUP attack. Following are the features of regular, weak and strong SETUP attack in 2.3.1, 2.3.2, 2.3.3.

Definition 2.3.1. Let's suppose C is a tamper proof cryptosystem whose public parameters are known to adversary. In a regular SETUP attack the algorithm C s is changed to make it C" so that the following properties are satisfied.

- The input parameters of an altered algorithm C" and an honest algorithm C, are same.
- Attacker's public key, which resides with in the cryptosystem, is used in the calculation of parameters that are changed in C".
- Attacker's private key is known only to the attacker and it does not exist with in the cryptosystem C".
- The cipher text produced by C" fulfills the public specifications of the output of an algorithm C. The cipher text must contain the meaning full data about the secret key of the user which can be easily derived by the attacker.
- The output of C and C" is computationally indistinguishable in polynomial time for the adversary except for the attacker.
- If the presence of backdoor is exposed by the reverse engineer even then the adversary is unable to use that backdoor information to find the past and future keys, except the attacker.

Definition 2.3.2. A weak SETUP attack fulfills all the conditions of regular SETUP attack except that the output of C and C"is computationally indistinguishable in polynomial time not only for the attacker but also for the owner of the device.

The weak SETUP attack is useful when the controller of the device is interested to leak his own private key and after that he can securely communicate using that channel. He is aware of the presence of the backdoor but except attacker and owner of device, no one else can compromise the device or use that information to derive the private key of the cryptosystem. In strong SETUP attack, we assume that in the cryptosystem the SETUP code is applied with 50% probability and with 50% probability the cryptosystem runs an honest code for generating the output. It is strong type of attack compared to regular SETUP. As in this attack if the output is given to the user still he is unable to guess whether or not the output is meaningful.

Definition 2.3.3. A strong SETUP attack fulfills all the conditions of regular SETUP attack along with an additional notion. We assume that the cryptosystem is in access of the user and he can reverse engineer that device any time but still he is unable to get the knowledge of past and future keys also he cannot guess by only seeing the output that the SETUP code is applied or not.

In SETUP attack, the information leakage using backdoor does not require separate communication channel for the transmission of secret key. In the presence of backdoor, the private key of the user can be calculated by the attacker in a polynomial time. The reverse engineer can only detect the presence of the backdoor but he is unable to use that trapdoor information to reconstruct the future or past private key of the user.

2.4 Literature Review

Kleptography is the study of stealing secure data secretly and subliminally. It is the subfield of cryptovirology [73]. The concept of getting information without actually breaking the cryptosystem was introduced by Adam Young and Moti Yung two decades ago [74]. In present, still the stealing of information in an unnoticeable way is a serious threat for the modern cryptography. Strength of the cryptosystem was defined by analyzing it's security against cryptanalysis attacks that takes an average of 2^{128} operations [75] for the key size

of 128 bits. Different researches proved that exploiting the implementation errors of cryptographic algorithm needs less efforts as compared to attacking its mathematical structure.

In 1996 [74], Young and Yung introduces the mechanism of inserting trapdoor inside the blackbox cryptosystem. The backdoor is only useful for the manufacturer or the attacker and no one else from the adversary can exploit that backdoor to derive the private key information. They proposed the Secretly Embedded Trapdoor with Universal Protection (SETUP) for RSA, ElGamal, DSA, and Kerberos. They also proved that the blackbox implementation of cryptosystem in no more secure.

In 1997 [76], the definition of weak, strong, regular SETUP attack along with the m out of n leakage bandwidth was introduced. They also presented the SETUP attack against the algorithms based on discrete logarithm problem. They implemented the proposed SETUP attack strategy against diffie hellman key exchange protocol. They also improved the strong SETUP attack against RSA public key algorithm. In 1997 [77], strong SETUP attack against ElGamal, DSA, Schnorr signature algorithm. Menezes Vanstone, MTI two pass protocol and Girault key agreement protocol was presented. They also introduced the term $Kleptogram$.

In 2001 [78], a new attack methodology was presented in which the attacker knows one part of the information from which the private key of the user is easily derived. They implemented the proposed attack against ElGamal and DSA. In 2003 [79], they presented four schemes of inserting backdoor in RSA key generation method. In three schemes the p and q are generated using standard manner but d and e are generated in such a way that they only appear to be random but in real they can easily be factorized by the attacker or the manufacturer. While in the last scheme half bits of p are represented in n that makes the factorization possible using Coppersmith's method.

In 2003 [80], the backdoor inserting mechanism in symmetric key ciphers is discussed. The Monkey cipher leaks the information related to the symmetric key of the user. The attacker only needs sufficient amount of ciphertext of unknown plaintext encrypted using same symmetric key to get the knowledge of secret key. In 2005 [81], strength of inserting backdoors is explained against RSA key generation algorithm. They also highlight the importance of analyzing the security of code as well as it's implementation for securing the overall cryptosystem instead of trusting the manufacturer of the cryptosystem. In 2005 [82], the implementation of trapdoor in RSA key generation is proposed that reduces the subliminal

channel. The proposed method is a lightweight solution so it can easily implemented in hardware or software.

In 2006 [83], they present the backdoor inserting mechanism against elliptic curve cryptosystem that is based on factorization problem i.e. elliptic curve version of RSA encryption and signature algorithm. In 2010 [84], an improved version of SETUP attack against RSA using elliptic curve was proposed using a new primitive i.e. covert key exchange. They used small bandwidth while the error probability is also very small. In 2015 [85], the stateless algorithm substitution is introduced. The technique is used to introduce symmetric backdoors in encryption algorithms that helps in mass surveillance. In 2016 [86], the subversion model was presented along with different kleptographic attack models.

Conclusion

In this chapter, the history of backdoor attacks is discussed along with the method to exploit such types of attacks. Introduction to the term Kleptography is also presented along with the definition of regular, weak and strong SETUP attack. This chapter also includes the review of the literature related to the concept of kleptography.

Chapter 3

ELLIPTIC CURVE CRYPTOGRAPHY

Introduction

This chapter gives the basic introduction of Elliptic curve cryptography. The section 3.1, describes the basic mathematics of an elliptic curves. In section 3.2, the cryptographic protocols based on an an elliptic curves are discussed. Finally, in section 3.3, some applications of an elliptic curve cryptography are presented.

3.1 Elliptic curves

Elliptic curve is an algebraic curve defined over a non singular Equation 3.1. This type of equation is called Weierstrass Equation [40]. It is an important field of number theory. Many applications of elliptic curves are present in the field of cryptography [41].

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{3.1}$$

Where a_1, a_2, a_3, a_4, a_6 are real numbers and E is the elliptic curve defined over the field F. The equation is non singular if its discriminant is non zero.

Definition 3.1.1. If α is the root of the polynomial than the discriminant of a polynomial defined as:

$$\triangle = \prod_{i \neq j}(\alpha_i - \alpha_j)$$

Definition 3.1.2. If F is the field of the polynomial than the characteristics of field is the smallest positive integer s such that $s \times 1 = 0$ where 1 is multiplicative identity of the field.

The non singularity of an elliptic curve ensures that it does not have self intersecting points. The equation 3.1 can be simplified on the bases of characteristics of the underlying field [40]. If the characteristic of the field is $\neq$ 2 or 3, then the equation 3.1 can be transformed in Equation3.2.

$$y^2 = x^3 + ax + b \tag{3.2}$$

Where a,b are real numbers. So the $\triangle = -16(4a^3+27b^2)$. If characteristics of the underlying field is 2 then there are two possible cases:

- The curve is non supersingular and it has the equation of the form:

$$y^2 + xy = x^3 + ax^2 + b \tag{3.3}$$

  where a,b are real numbers and its discriminant is b

- The curve is supersingular and it has the equation of the form:

$$y^2 + cy = x^3 + ax^2 + b \tag{3.4}$$

  where a,b,c are real numbers and its discriminant is c^4.

3.1.1 Group Law

An elliptic curve supports the addition and multiplication of two points in order to find the third point residing on the curve. Let $W = (x_1, y_1)$ and $Z = (x_1, y_1)$ are two points on an elliptic curve having characteristic $\neq 2$ or 3, then the sum of these two points i.e. $W + Z = (x_3, y_3)$ and $w \neq \pm Z$, can calculated by using the Equation 3.5 3.6 [40].

$$x_3 = (\frac{y_2 - y_1}{x_2 - x_1})^2 - x_1 - x_2 \tag{3.5}$$

$$y_3 = (\frac{y_2 - y_1}{x_2 - x_1})(x_1 - x_3) - y_1 \tag{3.6}$$

If an Elliptic curve is supersingular then Equation 3.7 3.8 is used for the addition of two points i.e. $W + Z = (x_3, y_3)$ [40].

$$x_3 = (\frac{y_1 + y_2}{x_1 + x_2})^2 + x_1 + x_2 \tag{3.7}$$

$$y_3 = (\frac{y_1 + y_2}{x_1 + x_2})(x_1 + x_3) + y_1 + c \tag{3.8}$$

If an elliptic curve is non supersingular then Equation 3.9 3.10 is used for the addition of two points i.e. $W + Z = (x_3, y_3)$ [40].

$$x_3 = (\frac{y_1 + y_2}{x_1 + x_2})^2 + (\frac{y_1 + y_2}{x_1 + x_2}) + x_1 + x_2 + a \tag{3.9}$$

$$y_3 = (\frac{y_1 + y_2}{x_1 + x_2})(x_1 + x_3) + x_3 + y_1 \tag{3.10}$$

To add a point to itself i.e. If $W = (x_1, y_1)$ then $W + W = 2W = (x_3, y_3)$ can be calculated using the Equation 3.11 if the characteristics is $\neq 2$ or 3. 3.12 [40].

$$x_3 = (\frac{3x_1^2 + a}{2y_1})^2 - 2x_1 \tag{3.11}$$

$$y_3 = (\frac{3x_1^2 + a}{2y_1})(x_1 - x_3) - y_1 \tag{3.12}$$

If an Elliptic curve is supersingular then Equation 3.13 3.16 is used for the point doubling i.e. $W + W = (x_3, y_3)$ [40].

$$x_3 = (\frac{x_1^2 + a}{c})^2 \tag{3.13}$$

$$y_3 = (\frac{x_1^2 + a}{c})(x_1 + x_3) + y_1 + c \tag{3.14}$$

If an Elliptic curve is non supersingular then Equation 3.15 3.14 is used for the point doubling i.e. $W + W = (x_3, y_3)$ [40].

$$x_3 = (\frac{x_1 + y_1}{x_1})^2 + (\frac{x_1 + y_1}{x_1}) + a \tag{3.15}$$

$$y_3 = x_1^2 + (\frac{x_1 + y_1}{x_1})(x_3) + x_3 \tag{3.16}$$

If the point $W = (x_1, y_1)$ then negative of the point is : $-W = (x_1, -y_1)$. When the point is added to its negative then the result is the point of infinity i.e. $W + (-W) = 0$, where 0 is the additive identity of Elliptic curve [41].

The points on an elliptic curve must have a primitive element that is used to generate the entire group elements. The group having a generator or the primitive element of the group is called the cyclic group. Some other properties of the group are:

Definition 3.1.3. A Group is a set of elements G along with the operation $*$ on G such that the following properties should hold:

- The group operation $*$ is closed, such that if the operation is performed on an elements that belong to group G then the result is also in G i.e. for all $x, y \epsilon$ G,x * y = zϵ G.
- The group operation $*$ is associative for all elements that belong to G i.e. for all x, y, z ϵ $G, x * (y * z) = (x * y) * z$.
- There exists an identity element 1ϵ G such that for all $x\epsilon$ $G, x * 1 = 1 * x$.
- There exists an inverse of an element that belong to G such that for all $x \epsilon$ $G, x * x^{-1} = x^{-1} * x = 1$.
- The group is commutative under operation $*$ i.e. for all $x, y \epsilon$ $G, x * y = y * x$.

If the group satisfy all the properties, then the group is abelian under the operation $*$. In cryptography both ,the additive in which the group operation $*$ denotes addition and multiplicative groups in which the group operation $*$ denotes multiplication, are used [42].
Total number of points on an elliptic curve defines the order of the group. Following example shows the procedure to find the total number of points on an elliptic curve.

For example: $E : y^2 = x^3 + x + 3$ mod 7 ,and the primitive point is $W = (x_1, y_1) = (4, 1)$

- 2W= W+W = (4,1)+(4,1) = (x_3, y_3) = (6,6)

$x_3 = (\frac{3(4)^2+1}{2(1)})^2 - 2(4)$ mod 7	$y_3 = (\frac{3(4)^2+1}{2(1)})(4-6) - (1)$ mod 7
$x_3 = ((3(4)^2+1)(4))^2 - 2(4)$ mod 7	$y_3 = ((3(4)^2+1)(4))(4-6) - (1)$ mod 7
$x_3 = ((49)(4))^2 - (8)$ mod 7	$y_3 = (49)(4)(-2) - (1)$ mod 7
$x_3 = ((0) - (8)$ mod 7	$y_3 = (0) - (1)$ mod 7
$x_3 = 6$	$y_3 = 6$

- 3W= 2W+W = (6,6)+(4,1) = (x_3, y_3) = (5,0)

$x_3 = (\frac{1-6}{4-6})^2 - (6) - (4)$ mod 7	$y_3 = (\frac{1-6}{4-6})(6-5) - (6)$ mod 7
$x_3 = ((5)(4))^2 - (6) - (4)$ mod 7	$y_3 = (5)(4)(6-5) - (6)$ mod 7
$x_3 = (20)^2 - (10)$ mod 7	$y_3 = (20)(1) - (6)$ mod 7
$x_3 = ((36) - (3)$ mod 7	$y_3 = ((6) - (6)$ mod 7
$x_3 = 5$	$y_3 = 0$

- 4W= 2W+2W = (6,6)+(6,6) = (x_3, y_3) = (6,1)

$x_3 = (\frac{3(6)^2+1}{2(6)})^2 - 2(6)$ mod 7	$y_3 = (\frac{3(6)^2+1}{2(6)})(6-6) - (6)$ mod 7
$x_3 = ((3(6)^2+1)(3))^2 - 2(6)$ mod 7	$y_3 = ((3(6)^2+1)(3))(6-6) - (6)$ mod 7
$x_3 = ((109)(3))^2 - (12)$ mod 7	$y_3 = (109)(3)(0) - (6)$ mod 7
$x_3 = ((25) - (12)$ mod 7	$y_3 = ((0) - (6)$ mod 7
$x_3 = 6$	$y_3 = 1$

- 5W= 4W+W = (6,1)+(4,1) = (x_3, y_3) = (4,6)

$x_3 = (\frac{1-1}{4-6})^2 - (6) - (4)$ mod 7	$y_3 = (\frac{1-1}{4-6})(4-6) - (1)$ mod 7
$x_3 = ((0)(-2))^2 - (6) - (4)$ mod 7	$y_3 = (0)(-2)(4-6) - (1)$ mod 7
$x_3 = ((0)(3))^2 - (6) - (4)$ mod 7	$y_3 = (0)(-2)(-2) - (1)$ mod 7
$x_3 = ((0) - (10)$ mod 7	$y_3 = ((0) - (1)$ mod 7
$x_3 = 4$	$y_3 = 6$

Now finally, for finding 6W ,we add 3W and 3W in the following way:

- 6W= 3W+3W = (5,0)+(5,0) = (x_3, y_3) = (O)

x_3 = $(\frac{3(5)^2+1}{2(0)})^2 - 2(5)$ mod 7	y_3 = $(\frac{3(5)^2+1}{2(0)})(\propto -6) - (0)$ mod 7
x_3 = $((3(5)^2+1)(\propto))^2 - 2(5)$ mod 7	y_3 = $((3(5)^2+1)(\propto))(0-6) - (0)$ mod 7
x_3 = O	y_3 = O

So the order of the group is 6 as six points lie on the curve. The Hasse's theorem 3.1.1 can also be used to find the approximate number of points on an elliptic curve.

Theorem 3.1.1. If E mod n, is an elliptic curve then the number of points on an elliptic curve is defined in the following range:

$$n + 1 - 2\sqrt{n} \leq \text{no of points} \leq n + 1 + 2\sqrt{n}$$

3.2 Elliptic Curve Discrete Logarithm Problem

In 1985, Neal Koblitz and Victor S.Miller introduced the use of elliptic curves in cryptography [43]. In an elliptic curve the public key cryptography was based on Elliptic Curve Discrete Logarithm Problem (ECDLP) 3.2.1.

Definition 3.2.1. If E is an elliptic curve defined over finite field. So, given the points of an elliptic curve i.e. P and G, discrete logarithm problem is to find x such that $xP = G$.

The x is the private key information in elliptic curve cryptosystem [44], where as P and G are the points of an elliptic curve. So the adversary needs to find how many times P is added to itself to get G. If the attacker is able to solve the Elliptic curve discrete logarithm problem then he can successfully break the elliptic curve based public key cryptosystem.
Elliptic curve discrete logarithm problem is stronger than the discrete logarithm problem of finite fields. Under certain conditions, exponential time algorithm was the only algorithm that reduces the complexity of solving elliptic curve discrete logarithm problem until 1990 [45]. While, many algorithms exists to solve the discrete logarithm problem i.e. baby step giant step algorithm [46], Pollard's rho method [47], index calculating method [48]etc. Later in 1993, Menezes, Okamoto and Vanstone (MOV) method was introduced by Albert Menezes, Tatsuaki Okamoto and Scott Vanstone to reduce the complexity of solving elliptic curve discrete logarithm problem but it only works for supersingular elliptic curves [49].

The point G can be found by performing point multiplication of x and p. In elliptic curve the point multiplication is similar to exponentiation function. An efficient method to compute xP is $Double\ and\ Add\ algorithm$. The working of double and add algorithm is explained in Algorithm1.

Algorithm 1 Double and Add Algorithm

procedure DOUBLE AND ADD
  Input: An elliptic curve E, an elliptic curve point P and an integer x.
  Initialization: P=G and D= O
  While x > 0
  If x = 1 mod 2 , D= D+ G
  $G = 2G$ and x = $\lfloor \frac{x}{2} \rfloor$
  If x > 0, go to step 4
  Return
  R
  Output: $R = G = xP$.

For example : Let an elliptic curve is E: $y^2 = x^3 + 14x + 19$ mod 3623. If P= (6,730). Find 13P?

Table 3.1: Double and Add algorithm to compute : 13P

Step i	x	G= 2^i P	D
0	13	(6,730)	O
1	6	(2521,3601)	(6,730)
2	3	(2277,502)	(6,730)
3	1	(3375,535)	(2398,3047)
4	0	(1610,1851)	(1330,144)

So 13 P = (1330,144). There are many elliptic curves based protocols that are used to encrypt messages, to sign message, to establish shared key among two or parties and to generate random numbers. Some Elliptic curve based protocols are described in the following section.

3.2.1 Edwards Curve Digital Signature Algorithm

Edwards Curve digital signature algorithm was developed in 2012 based on Twisted curves [50]. It is a variant of Digital Schnorr's signature. It runs faster than other existing signature algorithm while avoiding security issues. Following are some of the features of Edward curve Digital signature algorithm (EdDSA).

- The algorithm for verification of single signature works very fast.

- The algorithm used for the verification of a batch of signatures takes less time as compare to existing signature verification algorithms.
- The signature algorithm also runs very fast.
- The public and private key generation algorithm is also very fast.
- The security level is equivalent to the security level provided by th existing algorithms .i.e.2^{128}.
- The key generation require random numbers but new signatures are deterministic. It has fool proof generation of session keys.
- Collision of hash function does not compromise the security of the signature algorithm.
- It produces small size signatures and also uses small key for signature .i.e 64 byte signature and 32 bytes key.

In EdDSA the twisted Edward curves are used instead of weierstrass curves [51]. If f is non zero constant and e=1, then twisted Edward curve is described in Definition 3.2.2.

Definition 3.2.2. Suppose $E_{E_{e,f}}$ is a twisted elliptic curve over a finite field having characteristic $\neq 2$, then it is defined by the following equation:

$$E_{E_{e,f}}\colon ex^2 + y^2 = 1 + fx^2y^2$$

Let (x_1, y_1) and (x_2, y_2) be any two points on an elliptic curve then the addition of the points i.e. (x_1, y_1)+ (x_2, y_2)= (x_3, y_3), on $E_{E_{e,f}}$ is calculated by using the Equation 3.17

$$(x_3, y_3) = (\frac{x_1y_2 + y_1x_2}{1 + fx_1x_2y_1y_2}, \frac{y_1y_2 - ex_1x_2}{1 - fx_1x_2y_1y_2}) \tag{3.17}$$

In twisted Edward curves, if $P = (x_1, y_1)$ is a point on the curve then the negative of the point P is $(-x_1, y_1)$. Point doubling or multiplication function of the point i.e. $2(x_1, y_1) = (x_3, y_3)$, is performed by using the Equation3.18.

$$(x_3, y_3) = (\frac{2x_1y_1}{ex_1^2 + y_1^2}, \frac{y_1^2 - ex_1^2}{2 - ex_1^2 - y_1^2}) \tag{3.18}$$

The security of the EdDSA strongly depends on the selection of an elliptic curve parameters that were selected to sign the message. Edward Digital Signature algorithm has the following parameters:

- An integer b such that $2^{b-1} > q$ and it defines the number of bits of public key.
- A one way hash function $Hash$ that gives the output value of 2b bits.
- An odd prime power q that defines the finite field for the elliptic curve while (0,1) is the neutral element of the group.
- The (b-1) bit encoding Enc of an elements of finite field.
- A non-square element d from the finite field.
- A large prime l such that $l[G] = 0$ and the generator of a group is G [52].

The 2b bit string of $(Enc(R), Enc(Sig))$ is send as the signature of the message msg using the secret key k of b bits. Let $Hash$ of secret key k is:

$$Hash(k) = (h_0, h_1, ...h_{2b-1})$$

If a is the private key of the user than a is calculated as :

$$a = 2^{b-2} + \sum_{3 \leq i \leq b-2} 2^i h_i$$

$A = a[G]$ is the public key of the user. The symbol $a[G]$ represents the elliptic curve point multiplication. r is the one time key, unique for every message is calculated as:

$$r = Hash((h_b, ...h_{2b-1}), msg)$$

In EdDSA r must be kept secret and random for each message. In the calculation of r, msg is different so it introduce randomness while $(h_b, ...h_{2b-1})$ is same because it is one of the half of secret key k. It is noted that if same r is used for two different messages then attacker can recover the private key information [50] that is further misused to create a valid signature of a fake message. The Enc is a b bit little endian encoding function. Little and Big endian encoding is just the way of representing the numbers in bytes. In this convention the least significant bytes comes first. Last three bits of the signature are 0 because mod l fits

the signature in b-3 bits. In order to forge the signature the attacker must wanted to gain the knowledge of a and for this he have to solve the elliptic curve discrete logarithm problem. The signature of message msg and verification process of that signature is described as follows:

Sender		**Verifier**
$A = a[G]$		
$R = r[G]$		
$x = Hash(Enc(R), Enc(A), msg)$		
$Sig = (r + (x)a) modl$		
	$\xrightarrow{(Enc(R), Enc(Sig))}$	
		$8Sig[G] = (x_1, y_1)$
		$x = Hash(Enc(R), Enc(A), msg)$
		$8R + 8xA modl = (x_2, y_2)$
		$Verified$
		$If : (x_1, y_1) = (x_2, y_2)$

3.2.2 Elliptic Curve Diffie Hellman Key Exchange Scheme

Elliptic curve diffie hellman key exchange protocol is an anonymous key exchange protocol that enables two honest parties to build the shared secret using their public and private key information in order to communicate over an insecure channel.

In Elliptic curve diffie hellman (ECDH) key exchange protocol, both sender and receiver first agreed on the domain parameters. The domain parameters of ECDH are: E is an elliptic curve define over a finite field, G is an elliptic curve point also called the generator of the group. An integer K_{sen} is the private key while $A = K_{sen}[G]\ \epsilon\ \{1, 2, ..., n-1\}$ is the public key of the sender that is calculated by multiplying K_{sen} by a elliptic curve point G similarly an integer $K_{rec}\ \epsilon\ \{1, 2, ..., n-1\}$ is the private key while $B = K_{rec}[G]$ is the public key of the receiver. n is the order of the group. Key is the final shared secret key between two parties that is used for communication. The key exchange protocol is explained below:

Sender		**Receiver**
$Choose : K_{sen} \epsilon \{1, ...n-1\}$		$Choose : K_{rec} \epsilon \{1, ...n-1\}$
$A = K_{sen}[G]$		$B = K_{rec}[G]$
	$A \longrightarrow$	
	$\longleftarrow B$	
$Key = K_{sen}[B] = K_{sen}K_{rec}[G]$		$Key = K_{rec}[A] = K_{rec}K_{sen}[G]$

For example: If E: $y^2 = x^3 + 2x + 2$ mod 17, total no of points on an elliptic curve are 19 and G=(5,1). Let the sender's private key is $K_{sen} = 5$ and receiver's private key is $K_{rec} = 9$. Then the protocol works in the following way:

Sender		**Receiver**
$Choose : K_{sen} = 5$		$Choose : K_{sen} = 9$
$A = 5(5,1) = (9,16)$		$B = 9(5,1) = (7,6)$
	$A \longrightarrow$	
	$\longleftarrow B$	
$Key = 5(7,6) = (0,6)$		$Key = 9(9,16) = (0,6)$

The adversary only has the knowledge of public keys of sender and receiver. So, one can not calculate the final shared key from public key information only. The adversary can find the secret key if he has the algorithm to solve elliptic curve diffie hellman problem in polynomial time [53]. The diffie hellman problem (ECDHP) is explained in Definition 3.2.3.

Definition 3.2.3. Suppose the adversary has the knowledge of an elliptic curve E define over the finite field, the elliptic curve point G, the public keys of sender and receiver $A = k_{sen}[G]$,

$B = K_{rec}[G]$ for some unknown values of k_{sen}, K_{rec}. Then elliptic curve diffie hellman problem is to find an elliptic curve point Key = $k_{sen}K_{rec}[G]$ in a polynomial time.

If one can find any algorithm that can solve elliptic curve discrete logarithm problem then that algorithm can definitely solve elliptic curve diffie hellman problem. So ECDLP is as hard as the solving ECDHP. The diffie hellman key exchange protocol is secure as long as ECDHP is a hard problem in polynomial time.

3.2.3 Elliptic Curve Digital Signature Algorithm

Elliptic curve digital signature algorithm (ECDSA) is an elliptic curve version of digital signature algorithm [54]. ECDSA used small key size as compare to DSA but provides same security level. In 1998, American National Standards Institute (ANSI) uses ECDSA as their standard algorithm for signature [55]. Conceptually the ECDSA resembles with DSA but computationally both are very different.

If msg is the message and $e = Hash(msg)$, then Elliptic curve digital signature is calculated in the following way:

Signer		**Verifier**
$(x_1, y_1) = k[G]$		
$r = x_1 mod n (r \neq 0)$		
$Sig = k^{-1}(z_e + rd_a) mod n (Sig \neq 0)$		
	$\xrightarrow{msg, (r, Sig)}$	
		$w = Sig^{-1} mod n$
		$u_1 = z_e w mod n$
		$u_2 = rw mod n$
		$(x_1, y_1) = u_1[G] + u_2[Q_a]$
		$If : r = x_1 mod n$
		$Return : True$

Initially both the sender and receiver agreed on the specific elliptic curve parameters: an

elliptic curve E define over finite a field, an elliptic curve point G also called the base point or the generator of the group. The order of the group is n such that $n[G] = 0$. Signer choses his private key $d_a \; \epsilon \; \{1, 2, ..., n-1\}$, while $Q_a = d_a[G]$ is his public key. If l_n is the bit length of n then $z_e = l_n$ leftmost bits of e. $Hash$ is one way hash function. k is cryptographically secure random number unique for every signature otherwise one can easily calculate the private key of the sender by using the Equation 3.20. So while signing the message using an elliptic curve digital signature algorithm, the sender should be ensure that the k is randomly selected. The receiver receives the Signature, first he will verify that if the $r, Sig \epsilon \{1, 2, ..., n-1\}$. Then perform the verification steps.

For example: Let E is an elliptic curve defined over a finite field having equation: $y^2 = x^3 + 2x + 2$ mod 17, the order of the group is $n = 19$. The base point is G =(5,1), the private key of the sender is $d_a = 7$ while his public key $Q_a = 7(5,1)$=(0,6). Let hash of $msg = e =$ 26 while $k = 10$ then signature of the message is calculated below:

Signer		**Verifier**
$(x_1, y_1) = 10(5, 1) = (7, 11)$		
$r = x_1 mod n (r \neq 0)$		
$Sig = 10^{-1}(26 + 7 \times 7) mod 19 = 17$		
	$msg, (r, Sig)$ $\longrightarrow$	
		$r, Sig \epsilon \{1, 2, ..., n-1\}$
		$w = 17^{-1} mod 19 = 9$
		$u_1 = 9 \times 26 mod 19 = 6$
		$u_2 = 9 \times 7 mod 19 = 6$
		$(x_1, y_1) = 6(5, 1) + 6(0, 6)$
		$(x_1, y_1) = (7, 11)$
		$If : 7 = 7 mod 19$
		$Return : True$

Let (r, Sig) and (r, Sig') are two signatures of messages msg and msg', signed by same k. So, attacker calculates $k =$ by using Equation 3.19. The value of z_e and z'_e can be easily calculated by the attacker as the message is known to him while Sig and Sig' can be captured through channel. Now putting the values in Equation 3.20, the attacker can get the private key information of the sender.

$$k = \frac{z_e - z'_e}{Sig - Sig'} \tag{3.19}$$

$$d_a = \frac{Sig \times k - z_e}{r} \tag{3.20}$$

3.2.4 Elliptic Curve Integrated Encryption Scheme

Elliptic curve Integrated Encryption scheme is one of the best schemes used in different cryptographic standards [56]. ECIES was developed in 2001 by Victor Shoup and was adopted as a standard in American National Standards Institute (ANSI), Institute of Electrical and Electronics Engineers (IEEE), International Organization for Standardization (ISO) etc [57, 56]. ECIES hides data as well as the symmetric key, used to encrypt or decrypt the data, using an asymmetric key [58]. If msg is the desired message then the encryption and decryption is performed in the following way:

Sender		**Receiver**
$R = r_s[G]$		
$P = (p_x, p_y) = r_s[Q_a]$		
$k_{enc}\|\|k_m = KDF(p_x\|\|S_1)$		
$ct = Encrypt(k_{enc}, msg)$		
$tag = MAC(k_{mac}, ct\|\|S_2)$		
	$\xrightarrow{R\|\|ct\|\|tag}$	
		$P = (p_x, p_y) = k_a[R]$
		$k_{enc}\|\|k_{mac} = KDF(p_x\|\|S_1)$
		$tag = MAC(k_{mac}, ct\|\|S_2)$
		$msg = Encrypt_{-1}(k_{enc}, ct)$

In order to send the message to the receiver, first both parties agreed on the following domain parameters: any standard Key Derivation Function (KDF), Message Authentication Code (MAC) and a symmetric encryption scheme. KDF is a function that can build one or more secret keys using a master key, passphrase or a password [59]. MAC is a piece of data used to ensure authentication of message while symmetric encryption scheme is a mechanism in which same key is used for encryption and decryption.

E is an elliptic curve defined over a finite field, G is an elliptic curve point also called base point, n is the order of the group, $Q_a = k_a[G]$ is the public key of the sender while $k_a \epsilon \{1, ...n-1\}$ is his private key. $Q_a = k_a[G]$ represents an elliptic curve point multiplication. $r_s \epsilon \{1, ...n-1\}$ is a random number. k_{enc} is the key used for symmetric encryption function, k_{mac} is the key for MAC function. S_1 and S_2 is the pre-shared information and its optional [60]. $||$ notation is used as a symbol of concatenation of two values. $Encrypt$ is a symmetric key encryption function. p_x is the x coordinate of an elliptic curve point.

The security of elliptic curve integrated encryption scheme depends on the hardness of elliptic curve diffie hellman problem. It provides security against chosen plaintext and ciphertext attack [60].

A lot of attacks against elliptic curve integrated encryption schemes are presented but they are only based on theoretical assumptions [61]. The most important practical attack against elliptic curve integrated encryption scheme is Benign malleability. This vulnerability arises when the public key of sender R is not included in the KDF function instead only its x coordinate is included. This can cause a successful chosen ciphertext attack. So far no practical attack has been launched that exploit this vulnerability but it is important to avoid such mistakes [57].

Solved example of elliptic curve based integrated encryption scheme is presented below.

For example: Let $E$is an elliptic curve defined over a finite field having equation: $y^2 = x^3 + 2x + 2$ mod 17, the order of the group is $n = 19$. The base point is G =(5,1), the private key of the sender is $r_s = 3$ while his public key $R = 3(5,1)$=(10,6).The private key of the receiver is k_a=5 while his public key is $Q_a = k_a[G]$ =5(5,1)=(9,16). Let S_1 =1 and S_2 =2.Let msg =10 then signature of the message is calculated below:

Sender		Receiver
$P = 3(9, 16) = (3, 16), S = 3$		
$k_{enc}\|\|k_m = KDF(3\|\|1) = 4\|\|5$		
$ct = Encrypt(4, 10) = 9$		
$tag = MAC(5, 9\|\|2) = 8$		
	$((10, 6)\|\|9\|\|8)$ $\longrightarrow$	
		$P = 5(10, 6) = (3, 16), S = 3$
		$k_{enc}\|\|k_{mac} = KDF(3\|\|1) = 4\|\|5$
		$tag = MAC(5, 9\|\|2) = 8$
		$msg = Encrypt_{-1}(4, 9) = 10$

3.2.5 Elliptic Curve Qu-Vanstone Implicit Certificate

In cryptography, the public key certificates are used to verify the ownership of the public key information present in the certificate. The Implicit certificates are used as the replacement of traditional public key certificates. This technique is suitable for highly constrained environments. Implicit certificates enables the person to derive the public key and the private key is known to the party whose identity is associated with the certificate, there is no rule that nobody knows the private key [62]. Elliptic curve Qu-Vanstone (ECQV) is a kind of implicit certificate. The certificate is generated in the following way [63]:

In order to request the CA to generate the certificate first it is necessary to agree upon the domain parameters. E is an elliptic curve defined over a finite field where as n is the order of the group. G, the generator, is an elliptic curve point. $C_{ca} = c[G]$ is the public key of certificate authority (CA) while c is his private key. $c[G]$ represents the elliptic curve point multiplication. ID_a is the identity of the user. $ENCode$ is an encoding function while $Hash$ is one way hash function. The user generate a random number $a \, \epsilon \{1, ...n-1\}$ while $b \, \epsilon \{1, ...n-1\}$ is the random number generated by the certificate authority(CA). The generation of public and private key pair is shown below: [63]

Alice		**Certificate Authority**
$A = a[G]$		
	$A, ID_a \longrightarrow$	
		$D = A + b[G]$
		$certificate = ENCode(D, ID_a)$
		$E = Hash(certificate)$
		$R = Eb + cmodn$
	$\longleftarrow R, certificate$	
$P_{pub} = E[D] + C_{ca}$		
$P_{pri} = Ea + Rmodn$		

Implicit certificates contain the information like ID, public key and the digital signatures but the certificate is very small in size [62]. Certificate Authority's (CA) signature are not required if the implicit certificates are in use. The user extract the public key and perform the verification operation, if the certificate is expired the operation will fail.

In traditional public key certificate the certificate authority signs the certificate and the user knows the public and private key but in implicit certificate it is very easy to generate a certificate with out knowing the private key. So this will create the possibility of denial of service attack. As according to the protocol, the attacker does not know the value of the private key associated with the fake public key. The knowledge of the private key is only possible if the attacker knows the private key of the certificate authority.

For example: Let E is an elliptic curve defined over a finite field having equation: $y^2 = x^3 + 2x + 2$ mod 17, the order of the group is $n = 19$. The base point is G = (5,1), Alice generates a random number $a = 4$ and the random number selected by certificate authority is b =2. The private key of the certificate authority is $cs = 3$ while his public key is $C_{ca} = c[G] = 3(5,1) = (10,6)$. The public and private keys for Alice using elliptic curve Qu-Vanstone implicit certificate is calculated below:

Alice		**Certificate Authority**
$A = 4(5,1) = (3,1)$		
	$(3,1), 2$ $\longrightarrow$	
		$D(3,1) + 2(5,1) = (16,13)$
		$certificate = ENCode((16,13), 2)$
		$certificate = 13$
		$E = H(13) = 12$
		$R = 12 \times 2 + 3 mod 19 = 8$
	$8, 13$ $\longleftarrow$	
$P_{pub} = 12(16,13) + (10,6) = (5,16)$		
$P_{pri} = 12 \times 4 + 8 mod 19 = 18$		

3.2.6 Elliptic Curve ElGamal Encryption Scheme

ElGamal encryption scheme was proposed by Taher ElGamal in 1985 [64]. Elliptic curve based ElGamal encryption scheme was based on the hardness of elliptic curve discrete logarithm problem. This is an asymmetric encryption scheme that provides an extra layer of security by encrypting the key. The encryption of the message msg is performed in the following way:

Sender		**Receiver**
$c_1 = a[G]$		
$c_2 = msg + a[R]$		
	c_1, c_2 $\longrightarrow$	
		$msg = c_2 - r[c_1]$

Let E is an elliptic curve define over a finite field. G is an elliptic curve base point where as n

is the order of the group. The public key of the receiver is $R = r[G]$ where $r \epsilon \{1, ...n-1\}$, is his private key. $r[G]$ represents the elliptic curve point multiplication. $a\epsilon \{1, ...n-1\}$ is the random number chosen by the sender.

For example: Let $E$is an elliptic curve defined over a finite field having equation: $y^2 = x^3 + 2x + 2$ mod 17, the order of the group is $n = 19$. The base point is G = (5,1),Sender generates a random number $a = 3$. The private key of the receiver is r=5 while his public key is $R = r[G]$ =5(5,1) = (9,16). The message is msg = (11,10) and the encryption is performed in the following way:

Sender		**Receiver**
$c_1 = 3(5,1) = (10,6)$		
$c_2 = (11,2) + 3(9,16)$		
$c_2 = (11,10) + (3,16) = (11,7)$		
	$\xrightarrow{c_1, c_2}$	
		$msg = (11,7) - 5(10,6)$
		$msg = (11,7) - (3,16) = (11,10)$

Any one can calculate c_2 if he knows the value of msg so it is compulsory to select different and unique random number a for every message in order to improve the security. The ElGamal encryption scheme is vulnerable to chosen ciphertext attack so specified measures should be taken to reduce the possibility of chosen ciphertext attack [65].

3.3 Application of Elliptic curve Cryptography

The elliptic curve cryptography is widely used in the world to perform encryption, signature, secure key agreements and generating random numbers [66,64]. Elliptic curve cryptography is famous for its small key size, fast implementation of encryption or signature functions and better security as compared to other public key cryptosystems. In 1999, NIST recommended 15 safe elliptic curves [67]. In 2007, some backdoors are introduced in an elliptic curves [68], so the safecurve project was started to introduce the elliptic curves that are secure and

easy to implement and have less chances of the presence of backdoors [69].

No known algorithm can solve the elliptic curve discrete logarithm problem or elliptic curve diffie hellman problem in polynomial time, so it is secure to use elliptic curve cryptography. They consumes less storage space while encryption, so they perform efficiently in constrained environments. Hyperelliptic curves used more smaller key size as compared to elliptic curves. Mathematics involved in elliptic curves is little bit difficult so its hard to implement and explain the mathematical concepts to public. Elliptic curve cryptography is vulnerable to quantum attacks but supersingular isogeny curves are secure for post quantum era [43].

Some challenges are also associated with an elliptic curve cryptography. Selection of elliptic curve parameters, that provide the required level of security, is a big challenge. Interoperability is also a big issue that can be solved by using standard elliptic curve parameters [70]. Most of the time, cryptosystem using elliptic curve cryptography is vulnerable to side channel attacks [71]. Here is a list of algorithm or platforms where elliptic curves are used [72].

Conclusion

This chapter describes the concept of an elliptic curve cryptography along with the basic mathematics of an elliptic curve. The brief description of an elliptic curve based cryptographic protocols is also given. This chapter also discussed the importance of an elliptic curve cryptography along with its practical application.

Chapter 4

PROPOSED KLEPTOGRAPHIC ATTACKS

Introduction

This chapter gives the basic knowledge of the procedure of inserting backdoor mechanism. In section 4.1, proposed implementation strategy of inserting backdoors in protocols based on an elliptic curves is discussed along with the key recovery mechanism. Following are the protocols along with their relevant sections: Edward curve digital signature algorithm in 4.1.1, Elliptic curve diffie hellman key exchange algorithm in 4.1.2, Elliptic curve digital signature algorithm in 4.1.3, Elliptic curve integrated encryption scheme in 4.1.4, Elliptic curve qu-vanstone implicit certificate in 4.1.5 and Elliptic curve elgamal encryption scheme in 4.1.6 section.

4.1 Proposed Kleptographic Attack on Elliptic Curve Cryptography

As it is discussed that the by inserting malicious code in an honest device, the attacker or the controller of the device can easily get the secret key information. It is important for the crypto security analyst to be aware of various types of malicious code inserting mechanism that are discussed in the different literature and academics. To detect the presence of backdoor one should be expert of analyzing the code, at least, against known kleptographic attacks.

In this section, kleptographic attacks on an elliptic curve based signatures and encryption algorithms have been presented. To explain the SETUP attack on an elliptic cure based cryptographic algorithms, both an honest and malicious versions of an algorithm are presented along with the attacker's key recovery mechanism.

4.1.1 Edwards Curve Digital Signature Algorithm

The details of Edward curve digital signature algorithm is explained in Section 3.2.1. Here the Algorithm 2 is presented, that describes an honest method of signing message $'msg'$ using Edward curve digital signature algorithm.

Algorithm 2 Edward-curve Digital Signature Algorithm

procedure SIGNATURE OF MESSAGE : (R, Sig)
  $Hash(k) = (h_0, h_1, \cdots\cdots, h_{2b-1})$
  $a = 2^{b-2} + \sum_{3 \leq i \leq b-2} 2^i h_i$
  $A = a[G]$
  $r = Hash(h_b, \cdots\cdots, h_{2b-1}, msg)$
  $R = rG$
  $Sig = (r + Hash(Enc(R), Enc(A), msg)a) mod\ l$
  return (R, Sig)

Suppose the private key of the attacker is v, and the corresponding public key of the attacker is V such that $V = v[G]$ where $v[G]$ represents an elliptic curve point multiplication. G is the same base point that is used in the signature algorithm of an honest device. k is the original secret key of the user. A is known to the attacker as it is a public parameter. The Algorithm 3, describes the procedure of calculating malicious signature of the message msg. $H(Z_x)$ represents the hash of x coordinate of an elliptic curve point Z. The (R', Sig') is the required malicious signature.

Algorithm 3 Malicious Edward-curve Digital Signature Algorithm

procedure MALICIOUS SIGNATURE OF MESSAGE: (R', Sig')
  $Hash(k) = (h_0, h_1, \cdots\cdots, h_{2b-1})$
  $a = 2^{b-2} + \sum_{3 \leq i \leq b-2} 2^i h_i$
  $A = a[G]$
  $Z' = H(Z_x)$ where $Z = a[V] = av[G]$
  $r' = Hash(Z', msg)$
  $R' = r'[G]$
  $Sig' = (r' + Hash(Enc(R'), Enc(A), msg)a) mod\ l$
  return (R', Sig')

The attacker can recover the private key $'a'$ of an honest user by using the backdoor information. The attacker can capture (R', Sig') from the channel and can easily find $(Enc(R'), Enc(A)$. A is a publicly known parameter and $'msg'$ is also known. The procedure of key recovery is shown in Algorithm 4.

Algorithm 4 Recover Key information (EdDSA)

procedure PRIVATE KEY RECOVERY:(a)
  $Z' = H(Z_x)$ where $Z = v[A] = va[G]$
  $r' = Hash(Z', msg)$
  $a = (Sig' - r')\ [(Hash(Enc(R'), Enc(A), msg)) mod\ l]^{-1}$
  return a

4.1.2 Elliptic Curve Diffie Hellman Key Exchange Scheme

The details of Elliptic curve diffie hellman key exchange scheme is explained in Section 3.2.2.Here the Algorithm 5 is presented that describes an honest method of establishing a symmetric key using an elliptic curve diffie hellman key exchange scheme.

Algorithm 5 Elliptic-curve Diffie Hellman Key Exchange Algorithm

procedure KEY GENERATION : (Key)
  $K_{sen}\ \epsilon\{1, \cdots, n-1\}$
  $A = K_{sen}[G]$, send to receiver
  $K_{rec}\ \epsilon\{1, \cdots, n-1\}$
  $B = K_{rec}[G]$, send to sender
  $key = K_{rec}[A] = K_{rec}K_{sen}[G]$, is receiver's Key
  $key = K_{sen}[B] = K_{sen}K_{rec}[G]$, is sender's Key.
  return (Key)

Let $Y = x[G]$ is the public key of the attacker while x is his private key. $Hash$ is a cryptographic secure one way function. w is a fixed odd integer, $t\ \epsilon\ \{0,1\}$ and a, b are fixed constants. The fixed constants known to attacker w, a, b are just to introduce some randomness in z that makes hard for owner of the device to distinguish between the outputs of an honest device and the outputs of a malicious device [87]. $Hash(z_x)$ represents the hash of the x coordinate of an elliptic curve point z. The malicious key generation is performed by the Algorithm 6. Key_{mal} is the required malicious shared secret key.

Algorithm 6 Malicious Elliptic-curve Diffie Hellman Key Exchange Algorithm

procedure MALICIOUS KEY GENERATION : (Key_{mal})
  $K_1\ \epsilon\{1, \cdots, n-1\}$
  $M_1 = K_1[G]$, store K_1 in memory
  Now for second usage
  $z = (K_1 - wt)[G] + (-aK_1 - b)[Y]$
  $K_2 = Hash(z_x)$
  $M_2 = K_2[G]$, send to receiver
  $K_{rec}\ \epsilon\{1, \cdots, n-1\}$
  $B = K_{rec}[G]$, send to sender
  $Key_{mal} = K_{rec}[M_2] = K_{rec}K_2[G]$, is receiver'Key
  $Key_{mal} = K_2[B] = K_2K_{rec}[G]$, is sender'Key
  return (Key_{mal})

The attacker captures M_1 and M_2 from the channel. They are used to find key K_2. So the attacker implements Algorithm 7 to recover the key information and to reconstruct the shared key between two users. $Hash(z_{1x})$ represents x coordinate of z_1.

Algorithm 7 Recover Key information (ECDHKEA)

procedure PRIVATE KEY RECOVERY:(k_2)
    $r_{mal} = a[M_1] + b[G]$
    $z_1 = M_1 - x[r_{mal}]$
    If $M_2 = (Hash(z_{1x})[G]$
    return $Hash(z_{1x})$
    $z_2 = z_1 - w[G]$
    If $M_2 = (Hash(z_{2x}))[G]$
    return $Hash(z_{2x})$

4.1.3 Elliptic Curve Digital Signature Algorithm

The details of Elliptic curve digital signature algorithm is explained in Section 3.2.3. Here the Algorithm 8 is presented that describes an honest method of signing the message msg using an elliptic curve digital signature algorithm.

Algorithm 8 Elliptic-curve Digital Signature Algorithm

procedure SIGNATURE OF MESSAGE : (r, Sig)
    $e = Hash(msg)$
    Let l_n is the bit length of n then $z_e = l_n$ leftmost bits of e
    $k \, \epsilon \{1, \cdots, n-1\}$
    $(x_1, y_1) = k[G]$
    If $r = x_1 mod\, n = 0$, go to step 4
    $Sig = k^{-1}(z_e + rd_a) mod\, n$, where $Sig \neq 0$
    return (r, Sig)

The Algorithm 9 is used to verify the signature of the message msg.

Algorithm 9 Signature Verification of ECDSA

procedure SIGNATURE VERIFICATION OF MESSAGE (T/F)
    Q_a is a valid curve point and $nQ_a = 0$ where 0 is identity element
    $r, Sig \, \epsilon \{1, \cdots, n-1\}$
    $e = Hash(msg)$
    $z_e = l_n$ leftmost bits of e
    $w = Sig^{-1} mod\, n$
    $u_1 = z_e w mod\, n$ and $u_2 = rw mod\, n$
    $(x_1, y_1) = u_1[G] + u_2[Q_a]$
    If $(x_1, y_1) = 0$
    return $False$
    If $r = x_1 mod\, n$
    return $True$

The signature verification algorithm is valid for both an honest as well as the malicious signature scheme. Let x is the attacker's private key while Y is his public key such that $Y = x[G]$. The private key of the signer is d_a. $Hash(z_x)$ represents the x coordinate

of an elliptic curve point. w, a, b, t are fixed odd constants known to the attacker, where $t\epsilon\{0, 1\}$. These constants introduced randomness in malicious key selection. The malicious implementation of an elliptic curve digital signature algorithm is given in Algorithm 10. The required signature is (r_2, Sig_2).

Algorithm 10 Malicious Elliptic-curve Digital Signature Algorithm

procedure MALICIOUS SIGNATURE OF MESSAGE: (r_2, Sig_2)
  $K_1 \epsilon\{1, \cdots, n-1\}$, store K_1 in memory.
  $e_1 = Hash(msg_1)$
  $z_{e1} = l_n$ leftmost of e_1.
  $(x_1, y_1) = K_1[G]$
  If $r_1 = x_1 mod\ n = 0$, go to step 2
  $Sig_1 = K_1^{-1}(z_{e1} + r_1 d_a) mod\ n$, where $Sig_1 \neq 0$
  Signature is : (r_1, Sig_1)
  now for second usage
  $e_2 = Hash(msg_2)$
  $z_{e2} = l_n$ leftmost of e_2.
  $z = (K_1 - wt)[G] + (-aK_1 - b)[Y]$
  $K_2 = Hash(z_x)$.
  $(x_2, y_2) = K_2[G]$
  If $r_2 = x_2 mod\ n = 0$, go to step 13
  $Sig_2 = K_2^{-1}(z_{e2} + r_2 d_a) mod\ n$, where $Sig_2 \neq 0$
  return (r_2, Sig_2)

The attacker captures r_1 and r_2 from the channel to recover the value of K_2.The attacker use Algorithm 11 to recover the key information.

Algorithm 11 Recover Key information (ECDSA)

procedure PRIVATE KEY RECOVERY:(k_2)
  Using r_1 and r_2 find their ordinate, (x_1, y_1) and (x_2, y_2)
  $z_1 = (x_1, y_1) - x[(a(x_1, y_1) + b[G])]$
  If $(x_2, y_2) = Hash(x_2)G$
  return $Hash(x_2)$
  $z_2 = z_1 - w[G]$
  If $(x_2, y_2) = Hash(x_2)[G]$
  return $Hash(x_2)$

Now the attacker knows the value of K_2 so the calculation of d_a is carried out by using Equation 4.1. Sig_2 should not be 0.

$$d_a = \frac{Sig_2 k_2 - z_{e2}}{r_2} \tag{4.1}$$

4.1.4 Elliptic Curve Integrated Encryption Scheme

The elliptic curve integrated encryption scheme is a hybrid encryption scheme. The details of an Elliptic curve integrated encryption scheme is explained in Section 3.2.4. Here the Algorithm 12 is presented that describes an honest method of encrypting the message msg using an elliptic curve integrated encryption scheme. $r_s[G]$ represents an elliptic curve point multiplication.

Algorithm 12 Elliptic curve Integrated Encryption Scheme

1: **procedure** ENCRYPTION OF MESSAGE:$(R||ct||tag)$
2: $R = r_s[G]$
3: $P = (p_x, p_y) = r_s[Q_a]$
4: $S = p_x$, the abscissa of P
5: $K_{enc}||K_{mac} = KDF(S||S_1)$.
6: $ct = Encrypt(K_{enc}, msg)$
7: $tag = MAC(K_{mac}, ct||S_2)$
8: **return** $R||ct||tag$

The decryption Algorithm 13 of an elliptic curve integrated encryption scheme is used to recover the message msg on the other side of the communication network. The receiver receives $(R||ct||tag)$ form the sender.

Algorithm 13 Decryption of message in ECIES

1: **procedure** DECRYPTION OF MESSAGE:(msg)
2: Calculate $P = (p_x, p_y) = K_a[R]$
3: $S = p_x$, the abscissa of P
4: $K_{enc}||K_{mac} = KDF(S||S_1)$.
5: $tag = MAC(K_{mac}, ct||S_2)$
6: $msg = Encrpt^{-1}(K_{enc}, ct)$
7: **return** msg

The attacker wants to knows the symmetric encryption key as well as the key used in MAC. If x is the private key of the attacker while Y is his public key such that $Y = x[G]$ and $x[G]$ represents an elliptic curve point multiplication. Let w, a, b, t are some fixed odd constants known to the attacker, where $t\epsilon\{0, 1\}$. These constants are used to introduce randomness. S_1 is a preshared value and it is optional [60], so we can ignore this value. K_1 is selected in an honest way as r_s was chosen in Algorithm 12. The malicious encryption of the message msg is performed using the Algorithm 14.

The Algorithm 15 is used to recover the key information K_2. For this the attacker needs R_1 and R_2, that he can capture from the communication channel, to recover the values of K_{enc}, K_{mac}. The attacker first reconstruct S that is further user to recalculate the values of K_{enc}, K_{mac}. For t=0 K_2 is the $Hash(z_{1x})$, where $Hash(z_{1x})$ means x coordinate of an elliptic curve point z. For t=1 K_2 is the $Hash(z_{2x})$.

Algorithm 14 Malicious Elliptic curve Integrated Encryption Scheme

procedure MALICIOUS ENCRYPTION OF MESSAGE:$(R_2||ct_2||tag_2)$
    $R_1 = K_1[G]$, store K_1 in memory
    $P = (p_x, p_y) = K_1[Q_a]$, $S = p_x$ abscissa of P
    $K_{enc}||K_{mac} = KDF(S)$
    $ct_1 = Encrypt(K_{enc}, msg)$
    $tag_1 = MAC(K_{mac}, ct_1||S_2)$.
    return $R_1||ct_1||tag_1$
    For second time R_2 is calculated in a malicious way
    $z = (K_1 - wt)[G] + (-aK_1 - b)[Y]$
    $K_2 = Hash(z_x)$
    $R_2 = K_2[G]$
    $S = p_x$ from $P = (p_x, p_y) = K_2[Q_a]$
    $K_{enc}||K_{mac} = KDF(S)$
    $ct_2 = Encrypt(K_{enc}, msg)$
    $tag_2 = MAC(K_{mac}, ct_2||S_2)$
    return $R_2||ct_2||tag_2$

Algorithm 15 Recover Key information (ECIES)

procedure PRIVATE KEY RECOVERY:(K_2)
    $r_{mal} = a[R_1] + b[G]$
    $z_1 = R_1 - x[r_{mal}]$
    If $R_2 = Hash(z_{1x})[G]$
    return $Hash(z_{1x})$
    $z_2 = z_1 - w[G]$
    If $R_2 = Hash(z_{2x})[G]$
    return $Hash(z_{2x})$

4.1.5 Elliptic Curve Qu-Vanstone Implicit Certificate

The details of Elliptic curve Qu-Vanstone Implicit Certificate is explained in Section 3.2.5. The certificates are used to generate the secure public and private keys through an honest third party. Here the Algorithm 16 is presented that describes the honest method of generating the public and private keys of the user.

Algorithm 16 Elliptic curve Qu-Vanstone Implicit Certificate

procedure PRIVATE KEY GENERATION:(p_{pri}, P_{pub})
    User finds $A = a[G]$, and send A, ID_a to CA
    CA performs the next calculations
    $B = b[G]$
    $D = A + B$
    $certificate = ENCode(D, ID_a)$
    $E = Hash(certificate)$
    $R = Eb + c$ mod n
    Certificate authority sends $(R, certificate)$ to the user
    $P_{pub} = E[D] + C_{ca}$, public key of the user
    $p_{pri} = Ea + R$ mod n, private key of the user
    return p_{pri}, P_{pub}

The Algorithm 17 is used to generate malicious key pairs. Let the private key of the attacker is x and the corresponding public key is Y, such that $Y = x[G]$. In this method the third party i.e. the Certificate Authority, is an honest party. The malicious code is inserted on the user's side who requests for the key pairs.

Algorithm 17 Malicious Elliptic curve Qu-Vanstone Implicit Certificate

procedure MALICIOUS KEY GENERATION:(p_{mal_pri}, P_{mal_pub})
    The user generates a random number a_1 and stored in the memory
    $A_1 = a_1[G]$, sends A_1, ID_a to CA.
    CA is honest and sends $(R, certificate)$
    For the second time the random number is generated using malicious way.
    $z = (a_1 - wt)[G] + (-aa_1 - b)[Y]$
    $a_2 = Hash(z_x)$
    $A_2 = a_2[G]$
    CA performs the next calculations
    CA selects a random number b and finds $B = b[G]$
    $D = A_2 + B$
    $certificate = ENCode(D, ID_a)$
    $E = Hash(certificate)$.
    $R = Eb + cmod\, n$
    Certificate authority sends $(R, certificate)$ to the user
    $P_{mal\_pub} = ED + C_{ca}$, public key of the user
    $p_{mal\_pri} = Ea_2 + Rmod\, n$, private key of the user
    return p_{mal_pri}, P_{mal_pub}

The attacker uses Algorithm 18 to recover the secret key information.The attacker gets the value of A_1 and A_2 from the communication channel and his aim is to recover the value of the key i.e. a_2.

Algorithm 18 Recover Key information (ECQIC)

procedure KEY RECOVERY:(K_2)
    $r_{mal} = a[A_1] + b[G]$.
    $z_1 = A_1 - r_{mal}[x]$
    If $A_2 = Hash(z_{1x})[G]$
    return $Hash(z_{1x})$
    $z_2 = z_1 - w[G]$
    If $A_2 = Hash(z_{2x})[G]$
    return $Hash(z_{2x})$

4.1.6 Elliptic Curve ElGamal Encryption Scheme

The details of Elliptic curve ElGamal encryption scheme is explained in Section 3.2.6. Here the Algorithm 19 is presented that describes the honest method of encrypting the message msg using elliptic curve ElGamal encryption scheme.

Algorithm 19 Elliptic curve ElGamal Encryption Scheme

procedure ENCRYPTION OF MESSAGE:(c_2, c_1)
    $c_1 = a[G]$
    $c_2 = Msg + a[R]$, where R = r[G]
    c_2, c_1 is send to the receiver

Let the private key of the attacker is x, while his corresponding public key is Y such that $Y = x[G]$. $x[G]$ represents an elliptic curve point multiplication operation. The malicious code, inserted by the attacker, is described in an Algorithm 20. To decrypt the message the attacker needs the private key of the receiver i.e. r. The sender is an honest party while the receiver is malicious.

Algorithm 20 Malicious Elliptic curve ElGamal Encryption Scheme

procedure MALICIOUS ENCRYPTION IN ECEES:(c_{mal1}, c_{mal2})
    Store r_1 in the memory
    Sender calculates and sends (c_1, c_2) to receiver
    For the second time R_2 is calculated using malicious code
    $z = (r_1 - wt)[G] + (-ar_1 - b)[Y]$
    $r_2 = Hash(z_x)$
    $R_2 = r_2 G$
    Sender perform honest calculations
    $c_1 = a_2[G]$, a_2 is random number chosen by sender
    $c_2 = Msg + a_2[R_2]$
    The sender sends c_{mal2}, c_{mal1} to the receiver
    return c_{mal2}, c_{mal1}

In Algorithm 20, r_1 is the private key of the receiver that is generated in an honest way

while r_2 is generated in a malicious way. w, a, b are fixed odd constants known to the attacker, while $t\epsilon\{0, 1\}$. All these constants are used to introduce randomness in malicious key generation. $Hash(z_x)$ represents the x coordinate of z.

The Algorithm 21 is used to recover the key information using R_1, R_2.

Algorithm 21 Recover Key information (ECEES)

1: **procedure** KEY RECOVERY:(r_2)
2: $r = a[R_1] + b[G]$.
3: $z_1 = R_1 - r[x]$
4: If $R_2 = Hash(z_{1x})[G]$
5: **return** $Hash(z_{1x})$
6: $z_2 = z_1 - w[G]$
7: If $R_2 = Hash(z_{2x})[G]$
8: **return** $Hash(z_{2x})$

Conclusion

This chapter presents the proposed mechanism of inserting backdoor malicious code in an honest elliptic curve based cryptographic protocols. The public key of the attacker is inserted in the device of the user. The private key of the user is calculated on the basis of public key of the attacker. The private key of an attacker is used to generate the private key of the user. The reverse engineer only detects the presence of the backdoor but can not calculate the public key information of the attacker to get the private key of the user. Only the person in possession of the private key of an attacker can get the advantage of a backdoor information. Both an honest and malicious versions of an algorithms are presented. The key recovery mechanism, that is used by the attacker to recover private key information, is also given.

Chapter 5

TEST METHODOLOGY WITH RESULTS

Introduction

This chapter gives the brief description of test methodology along with the results. The section 5.1, describes the detail explanation of the methodology of testing technique implemented in this thesis research in order to detect the backdoor presence. In section 5.2, the experimental results of detection of backdoor in an elliptic curve based algorithms are presented.

5.1 Test Methodology

Detection of the presence of backdoors is very difficult process. As discussed previously, the reverse engineer can only detect the presence of backdoor but reverse engineering itself is very hard procedure. So detection of kleptographic implementation in a device is a complex task and very few researches can be found in this direction.

Analysis of the deviation of running time of an honest algorithm and malicious algorithm is identified as a possible way of detecting kleptographic implementations. In [88], the author discuss the run time analysis technique and provides the experimental results for ElGamal, DSS, Diffie Hellman and RSA protocol based on discrete logarithm problem.

The deviation of running time analysis technique along with coefficient of variance is used to detect the presence of kleptographic backdoor in an elliptic curve based protocols.

In probability theory and statistics, the term coefficient of variance is defined as the measure of variation from the mean value. It is also called relative standard deviation [89]. It is commonly used in the field of probability i.e. renewal theory, reliability theory etc. It is expressed as a ratio of standard deviation and the mean value. If SD is the standard deviation then the coefficient of variance i.e. CoV is calculated by using the Equations 5.1.

$$CoV = \frac{SD}{Mean} \tag{5.1}$$

Let f is the frequency of occurrence then the standard deviation SD is calculated by Equation 5.2.

$$SD = \sqrt{\frac{\sum(fx^2)}{\sum f} - \frac{(\sum fx)^2}{(\sum f)^2}} \tag{5.2}$$

The mean of a sample $x_1 + x_2 + x_3 + \cdots x_n$ is calculated as:

$$Mean = \frac{x_1 + x_2 + x_3 + \cdots x_n}{n} \tag{5.3}$$

For example: The data set is shown in Table 5.1 then the coefficient of variance is calculated as:

Marks	13	14	15	16	17
No of students	4	10	28	7	3

Table 5.1: Data set

Using the above mentioned formulas the coefficient of variance is calculated as:

Marks (X)	No of students (F)	X^2	FX^2	FX
13	4	169	676	52
14	10	196	1960	140
15	28	225	6300	420
16	7	256	1792	112
17	3	289	867	51
	$\sum F$= 52	$\sum X^2$=1135	$\sum FX^2$=11595	$\sum FX$=775

Now Standard deviation is :

$$SD = \sqrt{\frac{\sum FX^2}{\sum F} - \frac{(\sum FX)^2}{(\sum F)^2}}$$

$$SD = \sqrt{\frac{11591}{52} - \frac{(775)^2}{(52)^2}}$$

$$SD = \sqrt{222.98 - 222.12}$$

$$SD = 0.925$$

Mean value is:

$$Mean = \overline{X} = \frac{\sum FX}{\sum F}$$

$$\overline{X} = \frac{775}{52}$$

$$\overline{X} = 14.903$$

Now finally the Coefficient of variance is :

$$CoV = \frac{SD}{Mean}$$

$$CoV = \frac{0.925}{14.903} = 0.066\%$$

It is useful to calculate the coefficient of variance because it is the ratio of two terms having same unit so it is independent of the unit, or in other words, the ratio does not depend on the efficiency of the measurement setup [89]. So it is a suitable parameter that compare two data sets independent of the measuring unit. The test methodology is explained in Figure 5.1. According to the flow chart there are two implementations of a cryptographic algorithm i.e. first an honest implementation while second one is malicious. Same plaintext dataset and keys are given to an honest as well as a malicious implementation. Both algorithms returns a ciphertext and the running time of an algorithm. It is observed that ciphertext of an honest implementation is different from the output of the malicious implementation because of the presence of kleptographic code. Running time of both algorithms is also different for every input plaintext. Finally, the amount of deviation is calculated between the running time of an honest and malicious implementation of the cryptographic code by measuring the coefficient of variance.

5.1.1 Experimental Setup

To differentiate the malicious algorithm from an honest one, we implemented both i.e. malicious and an honest, versions of Edwards curve digital signature algorithm, Elliptic curve Diffie Hellman key exchange, Elliptic curve digital signature, Elliptic curve integrated encryption, Elliptic curve qu Vanstone and Elliptic curve Elgamal encryption schemes in Java using Eclipse Integrated Development Environment. MATLAB-9.2 is used to analyze the results of running time of an algorithm by representing the data in a graph form. The tests were conducted on Linux operating system on personal computer core i7, 16GB memory. For the analysis 1,00,000 random plaintext are generated using secure random number generator library in Java. These random numbers are given as an input to an honest and a malicious algorithms, while a random key is selected and that fixed key is used every time. Running time of an algorithm is measured in milliseconds.

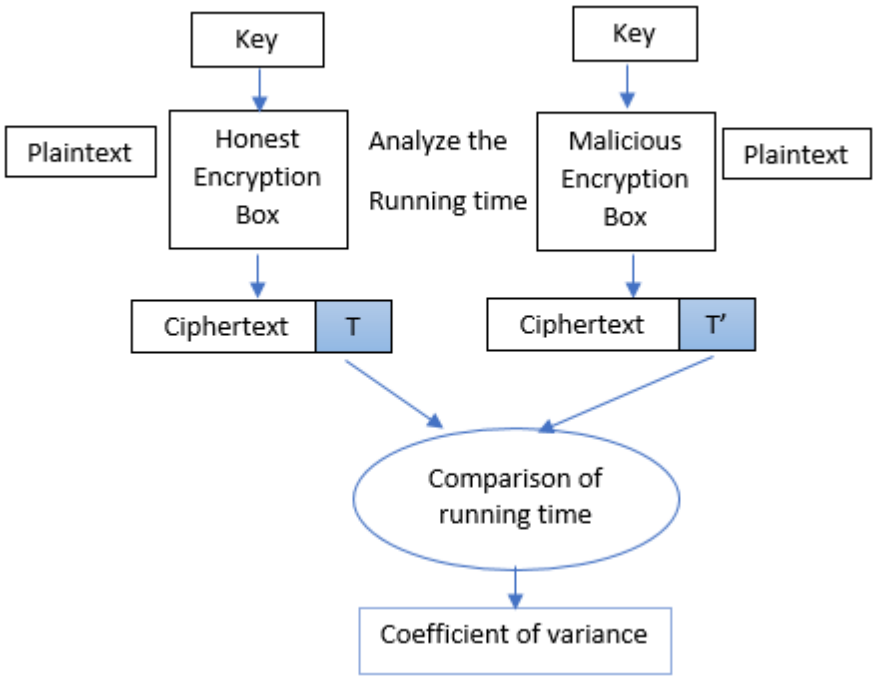

Figure 5.1: Flow chart of Test Methodology

5.2 Experimental Results

This section shows the experimental results for each analysis. The Figure5.2 shows the curve of an honest and malicious implementation of Edwards curve digital signature algorithm. As it is shown that the peaks of both graph is almost similar. So, in Edward curve digital signature algorithm the backdoor code does not require extra computation time to run. The malicious code is inserted in such a way that it take very less extra computation for generating a valid signature of the given message. Inserting a backdoor perform an extra elliptic curve multiplication operation.

Theoretically, by inserting one extra elliptic curve point multiplication the difference between the coefficient of variances of an honest algorithm and malicious algorithm must be approximately 0.58. In Table 5.3 the value of coefficient of variance is shown that is calculated by using the data shown in Table 5.3, 5.4. The difference of coefficient variance is approximately equal to the theoretical value of coefficient of variance presented above i.e. 5.8. Table 5.3 shows the data of running time of an honest implementation while Table5.3 shows the data of malicious implementation of Edward digital signature algorithm.

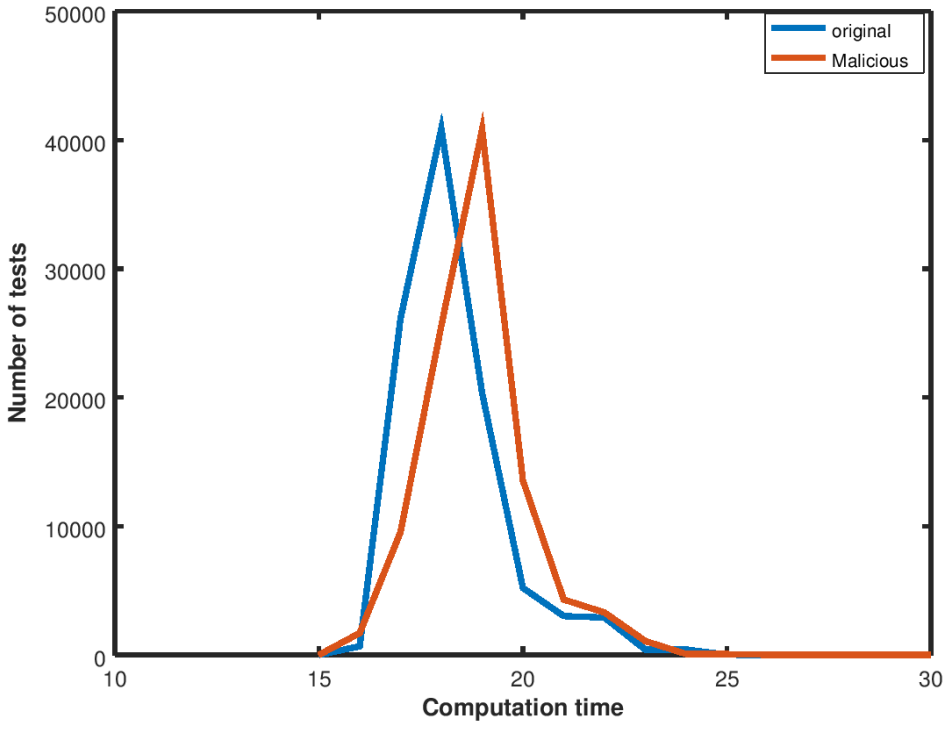

Figure 5.2: Experimental Results of Edward Digital Signature Algorithm

Table 5.2: Coefficient of variance of EdDSA

CoV of honest implementation	0.70
CoV of malicious implementation	0.060

Table 5.3: Results of Original implementation of EdDSA

Computation Time	15	16	17	18	19	20	21
Number of Tests	0	674	26130	40874	20346	5205	3017

Computation Time	22	23	24	25	26	27	28	29	30
Number of Tests	2905	407	422	17	1	1	1	0	0

The Figure 5.3 shows the results of the running time of an honest and a malicious implementation of an Elliptic curve Diffie Hellman Key Exchange protocol. As shown, the curve plotted against the malicious implementation shows small peak at its peak time, as compared to the curve that represent the running time of an honest implementation. The reason of this small peak at its peak time is very simple. The malicious implementation of the algorithm has two parts, first part of the code of malicious implementation is similar to the implemen-

Table 5.4: Results of Malicious implementation of EdDSA

Computation Time	15	16	17	18	19	20	21
Number of Tests	0	1701	9537	25574	40815	13553	4302

Computation Time	22	23	24	25	26	27	28	29	30
Number of Tests	3296	1075	75	68	2	1	1	0	0

tation of an honest algorithm, but the other part of the code is repeated every second time in malicious implementation only. So, the first part is shown in the small curve as it takes less time to run, while the disproportion between curves of malicious implementation shows that the device is running second part of the code repeatedly after every second call to generate the backdoor information for the attacker.

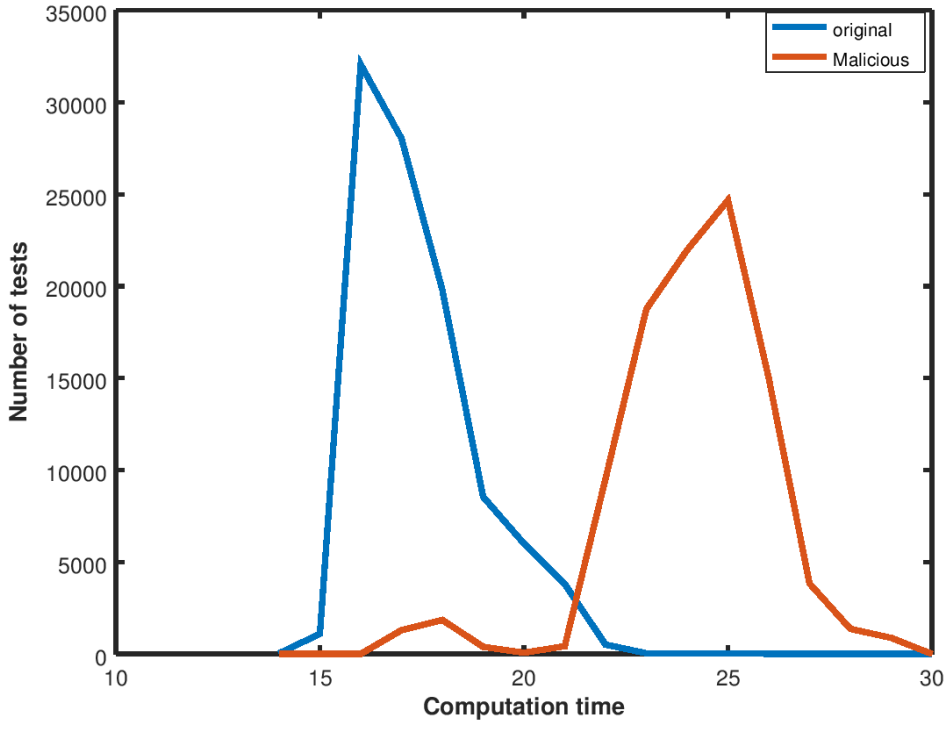

Figure 5.3: Experimental Results of Elliptic curve Diffie Hellman key exchange protocol

In an elliptic curve diffie hellman key exchange protocol, the malicious implementation involves three elliptic curve multiplication operation while in an honest protocol, only one elliptic curve multiplication operation is required for creating single key. So, the theoretical difference between the coefficient of variance of an honest and a malicious implementation should be approximately 1.73.

The Table 5.5 shows the coefficient of variance calculated by the data given in Table 5.6,5.7.

Table 5.5: Coefficient of variance of ECDHKE

CoV of honest implementation	0.04
CoV of malicious implementation	0.136

Table 5.6: Results of Original implementation of ECDHKE

Computation Time	14	15	16	17	18	19	20
Number of Tests	32	1095	32057	28030	19804	8546	6005

The difference between them is 1.45 that is approximately equal to the value calculated theoretically. The Table 5.6,5.7 represents the result of the computation time taken by an honest and malicious implementation of an elliptic curve diffie hellman key exchange protocol.

Computation Time	21	22	23	24	25	26	27	28	29	30
Number of Tests	3807	509	37	33	27	7	7	2	1	1

Table 5.7: Results of Malicious implementation of ECDHKE

Computation Time	14	15	16	17	18	19	20	21	22
Number of Tests	0	0	1	1294	1849	389	68	420	9530

Computation Time	23	24	25	26	27	28	29	30
Number of Tests	18753	21963	24674	14994	3827	1364	874	0

The Figure 5.4 shows the experimental results of an honest and a malicious implementation of an Elliptic curve Integrated Encryption Scheme. The graph shows that the behavior of an honest implementation of an Elliptic curve Integrated Encryption Scheme is similar to the behavior of an elliptic curve diffie hellman key exchange protocol. The curve of an honest implementation has smaller peak at its peak point. In malicious implementation two extra elliptic curve point multiplications are involved that results in the formation of second peak. That malicious part of the code is called at every second instance.

Table 5.8: Results of Original implementation of ECIKE

Computation Time	8	9	10	11	12
Number of Tests	28	97	106	4589	54358

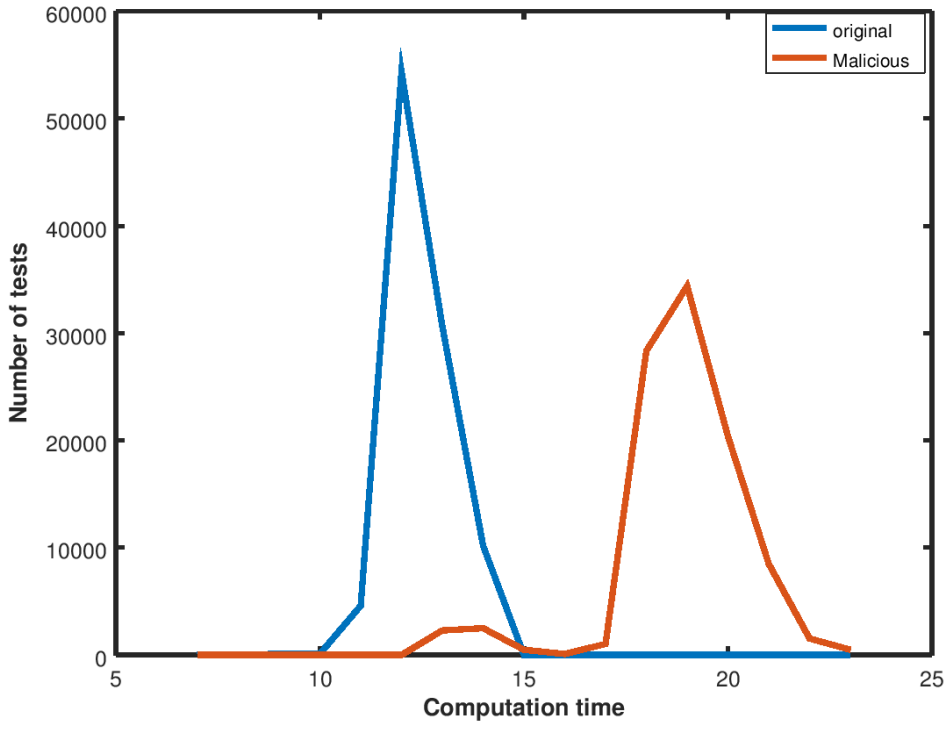

Figure 5.4: Experimental Results of Elliptic curve Integrated Encryption Scheme

Computation Time	13	14	15	16	17	18	19	20
Number of Tests	30636	10177	2	2	2	1	1	1

The theoretically observed value of the difference between the coefficients of variation is 0.58 for one time encryption. The difference of the coefficient of variance, calculated by the data recorded in Table 5.8,5.9, gives the value of 0.43 that is close to the desired value. Table 5.8 contain the data of the calculated running time of an honest algorithm while Table 5.9 contain the data of the calculated running time of a malicious implementation of an algorithm.

Table 5.9: Results of Malicious implementation of ECIKE

Computation Time	12	13	14	15	16	17
Number of Tests	1	2284	2494	485	89	1028

Computation Time	18	19	20	21	22	23
Number of Tests	28341	34384	20385	8493	1533	483

The Figure 5.5 shows the graphical representation of the experimental results of malicious and an honest implementation of Elliptic curve Digital Signature Algorithm. The honest implementation involve two elliptic curve point multiplication while the malicious implemen-

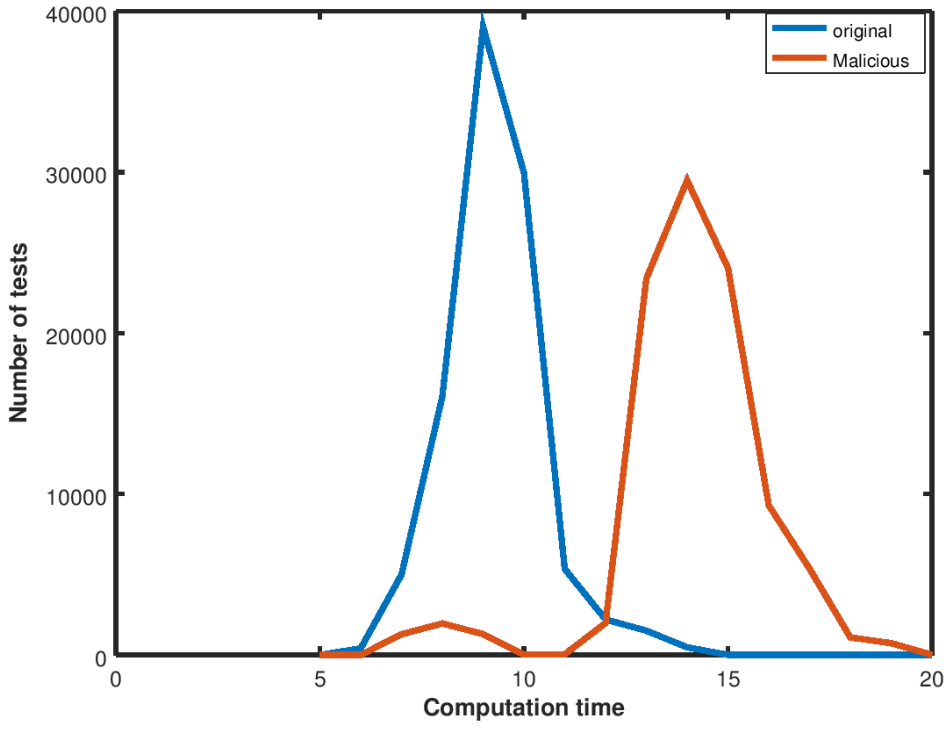

Figure 5.5: Experimental Results of Elliptic curve Digital Signature Algorithm

tation involve an additional point multiplications. The curve shows small peak just because the malicious implementation does not runs every time when the digital signatures are performed. Instead the malicious code runs at every second call of signature. This behavior is shown in the graph in the form of two peaks of different sizes.

Table 5.10: Coefficient of variance of ECDSA

CoV of honest implementation	0.4091
CoV of malicious implementation	0.0691

The theoretically observed value of the difference between the coefficients of variance is 0.58 that is approximately equal to the value calculated by using the data of Table 5.11,5.12. The Table5.11 gives the experimental results of the computation time of an honest implementation of a code while Table 5.12 gives the computation time of the malicious implementation of an elliptic curve digital signature algorithm.

Figure 5.6 presents the graphical representation of an Elliptic curve Qu-Vanstone Implicit Certificate. The Table 5.13 represents the measured data of an honest implementation while

Table 5.11: Results of Original implementation of ECDSA

Computation Time	6	7	8	9	10	11	12	13	14	15
Number of Tests	409	4995	16057	39030	29974	5345	2205	1507	474	4

Table 5.12: Results of Malicious implementation of ECDSA

Computation Time	7	8	9	10	11	12	13	14
Number of Tests	0	1958	1294	17	38	1986	23390	29495

Table5.14 represents the data of a malicious implementation of an elliptic curve qu-vanstone implicit certificate. The theoretically calculated value of difference between the coefficient of variance is 0.58 while the coefficient of variance that is calculated using the measured data in Tables 5.13 5.14 is 0.38.

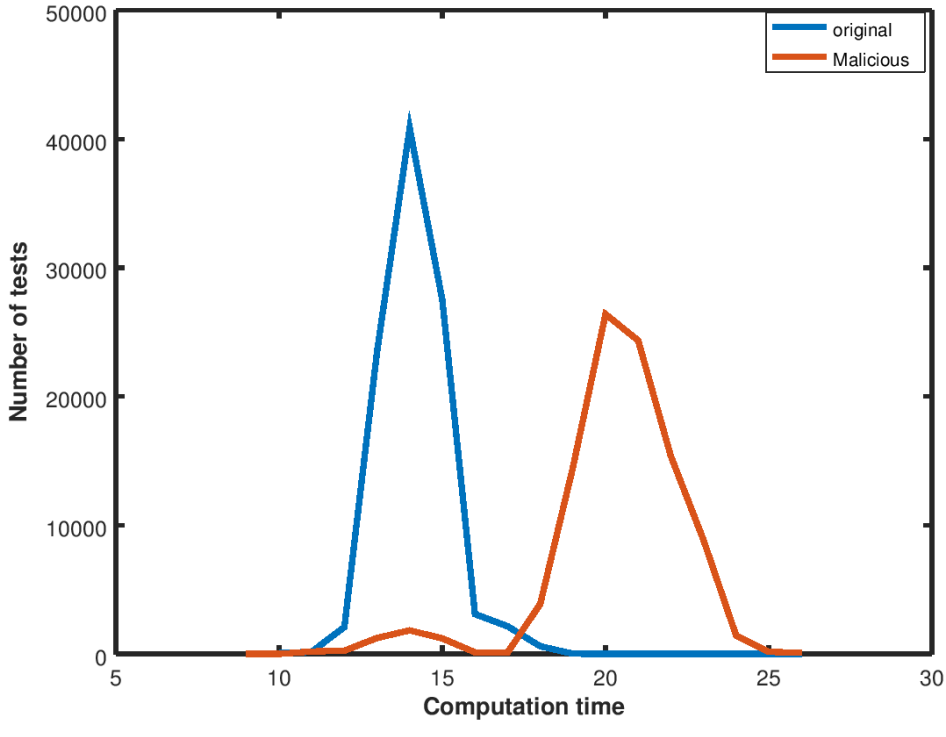

Figure 5.6: Experimental Results of Elliptic curve Qu-Vanstone Implicit Certificate

The Figure 5.7 shows the graphical representation of the results of timing analysis of an honest and a malicious code implementation of an Elliptic curve ElGamal Encryption Scheme. The Table 5.15 shows the computational results of an honest implementation of the code while Table 5.16 shows the computational results of the malicious implementation of an Elliptic curve Elgamal encryption scheme. The practically calculated value of the coefficient of variance is 0.401 while, the theoretically calculated value of the coefficient of variance is

Computation Time	15	16	17	18	19	20
Number of Tests	24052	9298	5361	1086	734	7

Table 5.13: Results of Original implementation of ECQVIC

Computation Time	9	10	11	12	13	14
Number of Tests	0	63	169	2096	23476	40847

Computation Time	15	16	17	18	19	20	21	22
Number of Tests	27486	3096	2148	597	17	3	1	1

Table 5.14: Results of Malicious implementation of ECQVIC

Computation Time	10	11	12	13	14	15	16	17	18
Number of Tests	0	194	251	1242	1832	1204	128	112	3883

Computation Time	19	20	21	22	23	24	25	26
Number of Tests	14484	26392	24352	15374	8864	1436	168	84

0.58.

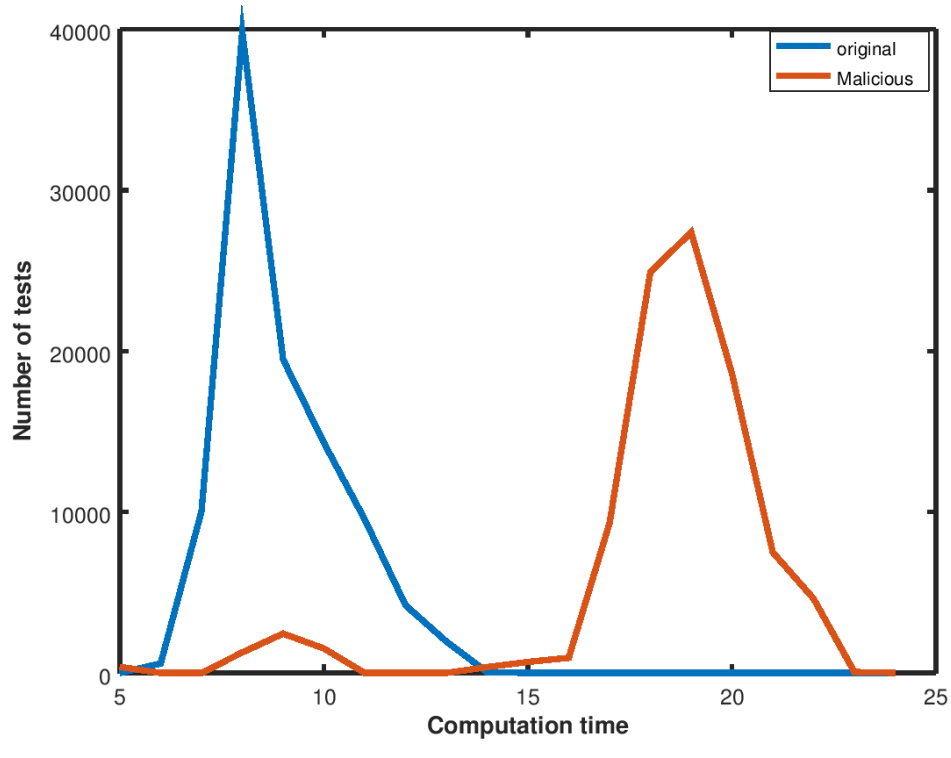

Figure 5.7: Experimental Results of Elliptic curve ElGamal Encryption Scheme

Table 5.15: Results of Original implementation of ECEES

Computation Time	5	6	7	8	9	10
Number of Tests	0	592	10038	39830	19485	14302

Computation Time	11	12	13	14	15	16	17	18
Number of Tests	9547	4205	1958	40	1	1	1	0

Table 5.16: Results of Malicious implementation of ECEES

Computation Time	5	8	9	10	14	15	16	17
Number of Tests	374	1274	2443	1523	367	685	940	9303

Computation Time	18	19	20	21	22	23	24
Number of Tests	24903	27392	18610	7503	4621	61	1

Conclusion

This chapter presents describes the proposed mechanism for the detection of malicious code in an elliptic curve based cryptographic protocols. Difference between the running time of an honest and malicious algorithms is calculated. It is observed that the malicious code takes more time to run as compared to the an honest algorithm. So, careful analysis of running time can help the security analyst to detect the presence of malicious code in the cryptographic device. The reverse engineering of the code needs more effort to detect the presence of malicious code as compare to analyzing the running time of an algorithm. This chapter also presents the practical results of running time of an algorithm.

Chapter 6

CONCLUSION and FUTURE WORK DIRECTIONS

6.1 Conclusion

A simple and efficient technique of backdoor detection in an elliptic curve based cryptography is proposed. In this thesis, it is shown that by simple analysis of running time of an honest and a malicious implementation of code, the presence of kleptographic backdoor can be detected in an elliptic curve based cryptographic protocols. It is observed that, as the number of tests increased the result of timing analysis become more accurate. Total 1,00,000 plaintexts are taken for the analysis of running time and this number is very less as compared to the plaintext ciphertext pairs needed to break the mathematical structure of an algorithm using modern cryptanalysis attack. It can be seen that an attacker can easily implement the successful SETUP attack in the cryptographic devices. So, it is important to test the cryptographic code as well as the correct implementation of the an algorithm to ensure the security of the device. Blackbox cryptographic devices are no more secure because they are more vulnerable against kleptographic attacks. Reverse engineering the device detects the presence of the backdoor but the process itself is very difficult and mostly infeasible. The reveres engineer can use the backdoor information to get the knowledge of the secret key.

Some measures are proposed in different researches to prevent the kleptographic attacks like to avoid the use of black box devices, the critical security parameters should be chosen by the crypto officer/ user of the device instead of using default parameters. Code walk through activity should be conducted. Use several cryptosystems designed by different manufacturer and divide the task to avoid the possibility of attack. It is good to use indigenous cryptographic algorithm, along with any standard encryption algorithm, to add an extra step of security. The source code should be made public. Finally check the integrity and authentication of the cryptographic algorithm

For future directions, it is recommended to search the possible solution for the problems:

- In crypto-currency, ECDSA is used as a signature algorithm so one can implement the

backdoor attack in the protocol that helps to produce the valid signature and change the bitcoin amount in a malicious way.

- The other possible way to detect the presence of kleptographic backdoor can be proposed i.e. by analyzing the number of operation performed in an honest and malicious implementation of an algorithm.
- To build a integrity check mechanism that ensure the correct implementation of an algorithm in a cryptographic device.
- To proposed the detection mechanism for the practical cryptographic devices using side channel techniques.
- To propose some prevention measures to avoid the kleptographic attack.

To conclude, there are many other directions to work in the field of Kleptography. It is important to test the blackbox device against such contemporary attacks before trusting the system. Now a days, it is very easy to hide the mini backdoored processor inside the parent processor to exploit the security of a cryptosystems.

Bibliography

[1] "Revealed: The nsa's secret campaign to crack, undermine propublica," https://www.propublica.org/article/the-nsas-secret-campaign-to-crack-undermine-internet-encryption, (Accessed on 02/21/2018).

[2] "Sigint enabling project," https://www.documentcloud.org/documents/784285-sigint-enabling-project, (Accessed on 02/21/2018).

[3] "Prism — us-news — the guardian," https://www.theguardian.com/us-news/prism, (Accessed on 2/21/2018).

[4] "We need to talk about mathematical backdoors in encryption algorithms the register," https://www.theregister.co.uk/2017/12/15/crypto_mathematical_backdoors/, (Accessed on 02/20/2018).

[5] "A tour of cryptographic backdoors," https://cohney.info/backdoors/, (Accessed on 04/3/2018).

[6] C. Wysopal and C. Eng, "Static detection of application backdoors," *Black Hat*, 2007.

[7] "Backdoors.pdf," https://www.cs.bu.edu/~reyzin/teaching/f14cs538/Backdoors.pdf, (Accessed on 02/29/2018).

[8] B. Schneier, M. Fredrikson, T. Kohno, and T. Ristenpart, "Surreptitiously weakening cryptographic systems." *IACR Cryptology ePrint Archive*, vol. 2015, p. 97, 2015.

[9] "A history of backdoors a few thoughts on cryptographic engineering," https://blog.cryptographyengineering.com/2015/07/20/a-history-of-backdoors/, (Accessed on 02/20/2018).

[10] "Clipper chip," http://www.cryptomuseum.com/crypto/usa/clipper.htm, (Accessed on 04/02/2018).

[11] "The clipper chip," https://www.epic.org/crypto/clipper/, (Accessed on 04/02/2018).

[12] "Nsa backdoor key from lotus-notes," http://www.cypherspace.org/adam/hacks/lotus-nsa-key.html, (Accessed on 02/20/2018).

[13] "Dual_ec_drbg - wikipedia," https://en.wikipedia.org/wiki/Dual_EC_DRBG, (Accessed on 02/20/2018).

[14] "Exclusive: Secret contract tied nsa and security industry pioneer," https://www.reuters.com/article/us-usa-security-rsa/exclusive-secret-contract-tied-nsa-and-security-industry-pioneer-idUSBRE9BJ1C220131220, (Accessed on 04/02/2018).

[15] D. J. Bernstein, T. Lange, and R. Niederhagen, "Dual ec: A standardized back door," in *The New Codebreakers*. Springer, 2016, pp. 256–281.

[16] "The debian openssl bug: Backdoor or security accident?" https://freedom-to-tinker.com/2013/09/20/software-transparency-debian-openssl-bug/, (Accessed on 02/20/2018).

[17] "Github - g0tmi1k/debian-ssh: Debian openssl predictable prng (cve-2008-0166)," https://github.com/g0tmi1k/debian-ssh, (Accessed on 04/02/2018).

[18] "Heartbleed - schneier on security," https://www.schneier.com/blog/archives/2014/04/heartbleed.html, (Accessed on 02/20/2018).

[19] S. Checkoway, J. Maskiewicz, C. Garman, J. Fried, S. Cohney, M. Green, N. Heninger, R.-P. Weinmann, E. Rescorla, and H. Shacham, "A systematic analysis of the juniper dual ec incident," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 468–479.

[20] "Juniper firewall backdoor password found in 6 hours — securityweek.com," https://www.securityweek.com/juniper-firewall-backdoor-password-found-6-hours, (Accessed on 04/02/2018).

[21] "Whatsapp messages can be easily read by anyone, wikileaks' cia files show — the independent," https://www.independent.co.uk/life-style/gadgets-and-tech/news/whatsapp-messages-wikileaks-cia-files-not-encrypted-hacking-julian-assange-vault-7-a7616576.html, (Accessed on 04/02/2018).

[22] "Research: Hackers could install backdoor in bitcoin cold storage," https://www.coindesk.com/research-hackers-install-backdoor-bitcoin-cold-storage/, (Accessed on 04/02/2018).

[23] "The 2016 backdoored cryptocurrency contest winner underhanded crypto contest," https://underhandedcrypto.com/2016/08/17/the-2016-backdoored-cryptocurrency-contest-winner/, (Accessed on 04/02/2018).

[24] F. Apap, A. Honig, S. Hershkop, E. Eskin, and S. Stolfo, "Detecting malicious software by monitoring anomalous windows registry accesses," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2002, pp. 36–53.

[25] "The 12 biggest, baddest, boldest software backdoors of all time — infoworld," https://www.infoworld.com/article/2606776/hacking/155947-Biggest-baddest-boldest-software-backdoors-of-all-time.html#slide3, (Accessed on 02/20/2018).

[26] S. Lowman, "The effect of file and disk encryption on computer forensics," *http://lowmanio. co. uk/. Acesso em*, vol. 7, no. 05, p. 2011, 2010.

[27] "Five reasons you should not use pirated wordpress plugins - wpstarters," https://www.wpstarters.com/five-reasons-not-use-pirated-wordpress-plugins/, (Accessed on 02/20/2018).

[28] T. Canavan, *CMS Security Handbook: The Comprehensive Guide for WordPress, Joomla, Drupal, and Plone*. John Wiley and Sons, 2011.

[29] K. Poulsen, "Borland interbase backdoor exposed," *The Register, http://www. theregister. co. uk/2001/01/12/borland_interbase_backdoor_ exposed*, 2001.

[30] Y. Zhang and V. Paxson, "Detecting backdoors." in *USENIX Security Symposium*, 2000.

[31] "Cowersnail windows backdoor from the creators of sambacry linux malware," https://thehackernews.com/2017/07/cowersnail-windows-backdoor.html, (Accessed on 02/20/2018).

[32] "Hacker installed a secret backdoor on facebook server to steal passwords," https://thehackernews.com/2016/04/hack-facebook-account.html, (Accessed on 02/20/2018).

[33] "Creepy backdoor found in netsarang server management software the register," https://www.theregister.co.uk/2017/08/15/netsarang_software_backdoor/, (Accessed on 02/2/2018).

[34] "Pre-installed backdoor on 700 million android phones sending users' data to china," https://thehackernews.com/2016/11/hacking-android-smartphone.html, (Accessed on 02/2/2018).

[35] L. Valenta, S. Cohney, A. Liao, J. Fried, S. Bodduluri, and N. Heninger, "Factoring as a service," Cryptology ePrint Archive, Report 2015/1000, 2015, https://eprint.iacr.org/2015/1000.

[36] A. Semenov, O. Zaikin, I. Otpuschennikov, S. Kochemazov, and A. Ignatiev, "On cryptographic attacks using backdoors for sat," *arXiv preprint arXiv:1803.04646*, 2018.

[37] D. R. Stinson, *Cryptography: theory and practice*. CRC press, 2005.

[38] B. Schneider, *Applied cryptography: protocols, algorithms, and source code in C*. John Wiley & Sons, 1996.

[39] "bh-us-05-young-update.pdf," https://www.blackhat.com/presentations/bh-usa-05/bh-us-05-young-update.pdf, (Accessed on 04/3/2018).

[40] A. Kahate, *Cryptography and network security*. Tata McGraw-Hill Education, 2013.

[41] "Elliptic curve - wikipedia," https://en.wikipedia.org/wiki/Elliptic_curve, (Accessed on 02/10/2018).

[42] H. C. Martin and G. F. Carey, *Introduction to finite element analysis: Theory and application*. McGraw-Hill College, 1973.

[43] "Elliptic-curve cryptography - wikipedia," https://en.wikipedia.org/wiki/Elliptic-curve_cryptography, (Accessed on 02/17/2018).

[44] W. Stallings, *Cryptography and network security: principles and practices*. Pearson Education India, 2006.

[45] "Microsoft word - haraksingh.doc," https://ocw.mit.edu/courses/mathematics/18-704-seminar-in-algebra-and-number-theory-rational-points-on-elliptic-curves-fall-2004/projects/haraksingh.pdf, (Accessed on 02/18/2018).

[46] J. S. Coron, D. Lefranc, and G. Poupard, "A new baby-step giant-step algorithm and some applications to cryptanalysis," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2005, pp. 47–60.

[47] E. Bach, "Toward a theory of pollard's rho method," *Information and Computation*, vol. 90, no. 2, pp. 139–155, 1991.

[48] V. S. Miller, "Use of elliptic curves in cryptography," in *Conference on the theory and application of cryptographic techniques*. Springer, 1985, pp. 417–426.

[49] I. Blake, G. Seroussi, and N. Smart, *Elliptic curves in cryptography*. Cambridge university press, 1999, vol. 265.

[50] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, 2012.

[51] "Twisted edwards curve - wikipedia," https://en.wikipedia.org/wiki/Twisted_Edwards_curve, (Accessed on 02/17/2018).

[52] S. Josefsson and I. Liusvaara, "Edwards-curve digital signature algorithm (eddsa)," Tech. Rep., 2017.

[53] "Elliptic curve diffie-hellman — math programming," https://jeremykun.com/2014/03/31/elliptic-curve-diffie-hellman/, (Accessed on 02/18/2018).

[54] D. W. Kravitz, "Digital signature algorithm," Jul. 27 1993, uS Patent 5,231,668.

[55] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.

[56] V. G. Martínez, L. H. Encinas *et al.*, "A comparison of the standardized versions of ecies," in *Information Assurance and Security (IAS), 2010 Sixth International Conference on*. IEEE, 2010, pp. 1–4.

[57] V. Shoup, "A proposal for an iso standard for public key encryption (version 2.1)," *IACR E-Print Archive*, vol. 112, 2001.

[58] "Elliptic curve integrated encryption scheme - crypto++ wiki," https://www.cryptopp.com/wiki/Elliptic_Curve_Integrated_Encryption_Scheme, (Accessed on 02/19/2018).

[59] "Key derivation function - wikipedia," https://en.wikipedia.org/wiki/Key_derivation_function, (Accessed on 02/19/2018).

[60] "Integrated encryption scheme - wikipedia," https://en.wikipedia.org/wiki/Integrated_Encryption_Scheme, (Accessed on 02/19/2018).

[61] V. Gayoso Martínez, L. Hernández Encinas, and A. Queiruga Dios, "Security and practical considerations when implementing the elliptic curve integrated encryption scheme," *Cryptologia*, vol. 39, no. 3, pp. 244–269, 2015.

[62] "Implicit certificate - wikipedia," https://en.wikipedia.org/wiki/Implicit_certificate, (Accessed on 02/19/2018).

[63] D. Brown, "Standards for efficient cryptography, sec 1: elliptic curve cryptography," *Released Standard Version*, vol. 1, 2009.

[64] R. Sunuwar and S. K. Samal, "Elgamal encryption using elliptic curve cryptography," *Cryptography and Computer Security, University of Nebraska, Lincoln*, 2015.

[65] "Elgamal encryption - wikipedia," https://en.wikipedia.org/wiki/ElGamal_encryption, (Accessed on 02/20/2018).

[66] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[67] P. FIPS, "186-3," *Digital signature standard (DSS)*, 2009.

[68] B. Schneier, "Did nsa put a secret backdoor in new encryption standard?" *URL: http://www. wired. com/politics/security/commentary/securitymatters/2007/11/securitymatters*, vol. 1115, p. 2007, 2007.

[69] D. J. Bernstein, T. Lange *et al.*, "Safecurves: choosing safe curves for elliptic-curve cryptography," *URL: http://safecurves. cr. yp. to*, 2013.

[70] P. G. Shah, X. Huang, and D. Sharma, "Analytical study of implementation issues of elliptical curve cryptography for wireless sensor networks," in *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*. IEEE, 2010, pp. 589–592.

[71] "Why are elliptic curves used in cryptography," http://www.umsl.edu/~siegelj/information_theory/projects/EllipticCurveEncyiption.pdf, (Accessed on 05/06/2018).

[72] "Things that use ed25519," https://ianix.com/pub/ed25519-deployment.html, (Accessed on 02/20/2018).

[73] A. Young and M. Yung, *Malicious cryptography: Exposing cryptovirology*. John Wiley & Sons, 2004.

[74] ——, "The dark side of black-box cryptography or: Should we trust capstone?" in *Annual International Cryptology Conference*. Springer, 1996, pp. 89–103.

[75] K. Thissen and T. Lange, "Klepto for post-quantum signatures," 2016.

[76] A. Young and M. Yung, "Kleptography: Using cryptography against cryptography," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1997, pp. 62–74.

[77] ——, "The prevalence of kleptographic attacks on discrete-log based cryptosystems," in *Annual International Cryptology Conference*. Springer, 1997, pp. 264–276.

[78] ——, "Bandwidth-optimal kleptographic attacks," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2001, pp. 235–250.

[79] ——, "Backdoor attacks on black-box ciphers exploiting low-entropy plaintexts," in *Australasian Conference on Information Security and Privacy*. Springer, 2003, pp. 297–311.

[80] C. Crépeau and A. Slakmon, "Simple backdoors for rsa key generation," in *Cryptographers Track at the RSA Conference*. Springer, 2003, pp. 403–416.

[81] A. Young and M. Yung, "Malicious cryptography: Kleptographic aspects," in *Cryptographers Track at the RSA Conference*. Springer, 2005, pp. 7–18.

[82] A. L. Young and M. M. Yung, "Yygen: A backdoor-resistant rsa key generator," 2005.

[83] A. Young and M. Yung, "A space efficient backdoor in rsa and its applications," in *International Workshop on Selected Areas in Cryptography*. Springer, 2005, pp. 128–143.

[84] ——, "Kleptography from standard assumptions and applications," in *International Conference on Security and Cryptography for Networks*. Springer, 2010, pp. 271–290.

[85] M. Bellare, J. Jaeger, and D. Kane, "Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1431–1440.

[86] A. Russell, Q. Tang, M. Yung, and H.-S. Zhou, "Cliptography: Clipping the power of kleptographic attacks," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2016, pp. 34–64.

[87] M. Antheunisse, "Kleptography cryptography with backdoors," Ph.D. dissertation, Masters thesis, Eindhoven University of Technology, 2015. http://repository. tue. nl/801620.

[88] D. Kucner and M. Kutylowski, "Stochastic kleptography detection," in *Public-Key Cryptography and Computational Number Theory*, 2001, pp. 137–149.

[89] "Coefficient of variation - wikipedia," https://en.wikipedia.org/wiki/Coefficient_of_variation, (Accessed on 02/2/2018).