2017

Security Analysis of ECC Based Protocols

Chanchal Khatwani
University of North Florida

SECURITY ANALYSIS OF ECC BASED PROTOCOLS

by

Chanchal Khatwani

A thesis submitted to the
School of Computing
in partial fulfilment of the requirements for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
SCHOOL OF COMPUTING

July, 2016

The thesis "Security Analysis of ECC Based Protocols" submitted by Chanchal Khatwani in partial fulfillment of the requirements for the degree of Master of Science in Computing and Information Sciences has been

Approved by the thesis committee: Date

Dr. Swapnoneel Roy
Thesis Advisor and Committee Chairperson

Dr. Daniel Dreibelbis

Dr. Karthikeyan Umapathy

Accepted for the School of Computing

Dr. Sherif Elfayoumy
Director of the School

Accepted for the College of Computing, Engineering, and Construction:

Dr. Mark Tumeo
Dean of the College

Accepted for the University:

Dr. John Kantner
Dean of the Graduate School

ACKNOWLEGEMENT

First, I would like to express my sincere gratitude to my advisor Dr. Swapnoneel Roy for his continuous support of my thesis work. His guidance helped me throughout the research and writing of this thesis. Besides my advisor, I would like to thank the rest of my thesis committee, Dr. Daniel Dreibelbis and Dr. Karthikeyan Umapathy, for their insightful comments.I would also like to thank Dr. Sherif Elfayoumy for his constructive feedback on my research during my thesis defense. Lastly, I would like to thank my parents for their constant encouragement.

CONTENTS

FIGURES

TABLES

ABSTRACT

Elliptic curve cryptography (ECC) is extensively applied in various multifactor authentication protocols. In this work, various recent ECC based authentication and key exchange protocols are subjected to threat modeling and static analysis to detect vulnerabilities, and to enhance them to be more secure against threats. This work demonstrates how currently used ECC based protocols are vulnerable to attacks. If protocols are vulnerable, damages could include critical data loss and elevated privacy concerns. The protocols considered in this work differ in their usage of security factors (e.g. passwords, pins, and biometrics), encryption and timestamps. The threat model considers different types of attacks including man in the middle, weak authentication, denial of service and SQL injection. Countermeasures to reduce or prevent such attacks are suggested. Beyond cryptanalysis of recent schemes and suggestion of new schemes, the proposed adversary model and criteria put forth a guideline for the methodic assessment of forthcoming multi-factor authentication proposals.

Chapter 1

INTRODUCTION

1.1 Background

In a cybersecurity domain, user authentication implements a perimeter device (proxy server, firewall, remote access server, VPN server, etc.) to judge if to allow an individual user's request to attain access to the network wall. The authentication is usually two-way, implying both entities (provider and user) validate themselves to each other (Khatwani and Roy, 2015). Client authentication requires security for remote login in the time the client's channel is making efforts to connect to the server's channel over unsafe networks. The identity and a secret password of a client are used for mutual authentication and access control. However, a password may be exposed while transmission if a suitable scheme is missing.

Elliptic curve cryptography (ECC) is an extensively applied technique in multi-factor authentication (Khatwani and Roy, 2015). ECC is a public key encryption approach set on elliptic curve theory that are used to create faster, smaller, and productive cryptographic keys. The keys are generated using the properties of the elliptic curve equation. This technology may be utilized in parallelism with almost all public key encryption techniques, like Diffie-Hellman and RSA. (Burr, 2016). ECC was introduced to minimize computational costs while providing equal layer of security as other familiar operations (such as modular exponentiation). The technique of ECC has applications in authentication protocols concerning RFIDs, digital signatures, wireless networks, smart cards, and other authentication techniques (Abidi et al., 2014; Choi et al., 2014; Chuang et al., 2015; Jin et al., 2015; Yeh et al., 2014; Zhang et al., 2015). However, the computational cost of one bilinear pairing (a crucial operation of ECC) is almost twice as high as a single modular exponentiation operation at the equal level of security (Farash and Ahmadian-Attari, 2014). Hence, the computationally intensive character of ECC yields a security flaw in the protocols that use it. An adversary may compel the client or server to repeatedly execute ECC operations in order to clog them, resulting in resources being wasted by performing unwanted calculations. In this thesis, we are looking at two different types of schemes that use ECC, one being multi factor authentication and the other being key-exchange.

Multi-factor authentication is a method wherein the user is required to render more than one form of validation to confirm user's identity and grant access to the structure. This takes leverage of a mix of many forms of authentication. The major forms contain verification by: (1) something a user knows (such as a password), (2) something the user has (such as a smart card or a security token), and (3) something the user is (such as a biometric characteristic). Due to their advance difficulty, authentication systems that use multi-factor verification are tougher to breach than those that use a single factor (Bhargav-Spantzel et al., 2007; Owen and Shoemaker, 2008; Khatwani and Roy, 2015; Sabzevar and Stavrou, 2008).

Key exchange is a cryptographic method by which secret keys are interchanged between two parties, with the use of a cryptographic algorithm. Public key cryptography, or asymmetric cryptography, is a cryptographic system that uses pairs of public and private keys. Each party has their own public key and private key. The message is encrypted using the public key and decrypted using one's private key. The public key is distributed and the private key is known only to the owner. If sender and receiver wish to communicate with each other, then a secret key is shared between them in order for communication to take place. Symmetric key algorithms use the same cryptographic keys for both encryption and decryption of text. The key exchange problem is how to exchange information so that no third party can obtain a copy. Typically, this has required trusted couriers or a secure channel.

Radio frequency identification (RFID) is a method of electronic identification which makes use of radio waves to detect, track, identify, and therefore manage a collection of objects. Despite the fact that this technology exists for over half a century, only recently have RFID privacy and security concerns started to invite awareness from corporate and academic research. RFID (Radio Frequency Identification) technology has become ubiquitous because of its low cost. Since it is used in many fields, any vulnerability detected in RFID technology, raises a threat in data privacy. Similarly, the potential for Smart Cards is enormous. However, by far the most serious problem for smart cards is the attacks that exploit security vulnerabilities caused by poor design or implementation. These vulnerabilities tend to be easy to exploit and replicate, and are, therefore, shared among the hackers community.

1.2 Problem Statement, Goal and Contribution

ECC is a multi-factor encryption technique currently used by the US government, Tor, Bitcoin, iMessage, and SSL/TLS. Such multi-factor authentication is required to produce higher level of security, however the establishment of other elements introduces more vulnerability in the protocols. Our goal is to detect these vulnerabilities before they are abused. The root causes of many common vulnerabilities like CPU resource-exhaustion, stack overflow, etc., are usually design flaws instead of programming (implementation) errors (Chang et al., 2009). Therefore developers perform static analysis on protocols to identify design flaws in order to ensure security before a program (software product) is launched.

Several multi-factor authentication protocols that involve RFIDs, smart cards, wireless networks, or digital signatures entrust ECC operations for their security. However, the computational cost of one bilinear pairing (a crucial operation of ECC) is almost twice as high as a single modular exponentiation at the equal level of security. Hence, the computationally intensive behavior of ECC yield a security loophole in the protocols. Therefore, a level of protection need to be added to ECC-based protocols to affirm overall security toward different types of attacks such as denial of service, man in the middle and database attacks.

The problem addressed by this thesis is that the vulnerabilities of ECC based protocols are not recognized. The first goal of this work is to perform static analysis on the underlying vulnerabilities and security threats that exist in ECC based protocols that are implemented in RFIDs and Smart Cards. The second goal is to design possible countermeasures to defeat the identified vulnerabilities in these protocols. The contribution of this research is to provide basis for future work in developing the security of ECC based protocols using dynamic analysis. The results of this work will contribute to the cybersecurity community in a considerable way.

1.3 Organization of this Thesis

The rest of the thesis is organized as follows. Chapter 2 contains the literature review and gives the necessary technical background. Chapter 3 introduces static vulnerability analysis and the threat model and Chapter 4 describes the cryptanalysis algorithms and discusses the type of protocols that were investigated in this work. Chapter 5 describes in detail the work done in this thesis to cryptanalyze various protocols. It describes the attacks that were carried out on the protocols, and proposes countermeasures for each attack. Finally, Chapter 6 discusses the results, conclusions and suggests future directions.

Chapter 2

REVIEW OF THE LITERATURE

ECC is one of the most accomplished and widely used, however least understood, cryptography tools (Sullivan, 2013). It is the future generation of public key cryptography. It provides significantly more security than first-generation public key cryptography systems like RSA (Sullivan, 2013).

2.1 ECC Background

ECC is a technique in public key cryptography set on the algebraic arrangement of elliptic curves over finite fields. Compared to non-ECC cryptography, ECC provides equivalent security with smaller keys (Hankerson et al., 2006) (Graham et al., 2016). The elliptic curve cryptosystem (Hankerson et al., 2006) was initially proposed as a basis for public key cryptosystems and it has proved out to be an important unit of current cryptography (Koblitz, 1987). ECC utilizes the mathematics of elliptic curves. The security of ECC lies in the complexity of working the elliptic curve discrete logarithm problem. An analysis of ECC theory and its computational problems are stated below.

2.2 Theory of Elliptic Curves

As shown in Figure 2.1, Elliptic curves ($E_q(a,b)$) are set of points defined by the solutions to the equation $y^2 \equiv x^3 + ax + b(\mod q)$, where a and b are elements of the field k together with a point at infinity **O** (Koc, 2013). There is also a condition such that

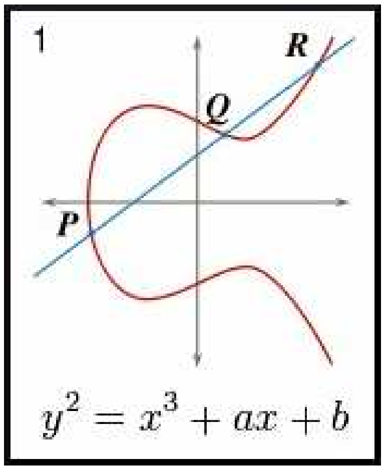


Figure 2.1: Elliptic Curve Addition (Guide, 2014)

$4a^3 + 27b^3 \neq 0 \pmod{q}$ where q is a prime number (Koc, 2013). This equation must be satisfied for the elliptic curve to have a well-defined group structure. This forms an additive cyclic group $E = \{(x, y) \in E_q(a, b)\} \cup \{\mathbf{O}\}$, where **O** serves as an additive identity element of the group (Koc, 2013). If P is a point in $Eand$k is a positive integer, then the point multiplication is computed by repeated addition, such as, $k{\cdot}P = P{+}P{\cdots}{+}P$, where k is a large integer and P is added to itself k times.

2.2.1 Computational Nature of ECC

ECC is a computationally intensive operation. Its scalar multiplication is one-way, making it computationally infeasible to trace the original number. For Example:Let P be a point in E, and let Q be a point such that $Q = kP$. The elliptic curve discrete log problem is the following: knowing the values of P and Q, determine the value of k. If the modulus q is large, the ECDLP is computationally infeasible. ECC is based on this problem. Even if P and Q are known, determining k such that $Q = kP$[1] is computationally infeasible. Hence the elliptic curve discrete log problem makes k difficult to compute.

[1] kP and $k \cdot P$ has the same meaning in ECC multiplication

2.2.2 Strengths of ECC

The strengths of Elliptic Curve Cryptography are as below:

- Elliptic curve discrete logarithm: Its security lies in its one way multiplication, which gives it a strong trapdoor function (Blake et al., 1999; Vanstone, 1997). A trapdoor function is a function that is simple to calculate in one direction, but difficult to calculate in the opposite direction (i.e finding its inverse) without sufficient information, called the "trapdoor". Trapdoor functions are extensively used in cryptography (Wikipedia, 2016c).
- Similar strength: It offers similar strength as compared to RSA with a small key size.
- More secure: It is considered to be the most secure cryptosystem (Blake et al., 1999; Vanstone, 1997).
- More efficient: It is proven to be more efficient than the first generation cryptography systems (Blake et al., 1999; Vanstone, 1997).
- Uses less CPU resources: It causes less overhead as compared to RSA, as it uses shorter encryption keys, and uses less memory than other schemes such as Diffie-Hellman, and therefore is faster than RSA (Gura et al., 2004; GlobalSign, 2015; Lauter, 2004).

2.2.3 Comparison between ECC and RSA

The RSA cryptosystem is an alternate technique used in encryption and authentication protocols that was discovered by (Rivest et al., 1978). RSA uses a technique called modular exponentiation to guarantee security. Table 2.1 compares the sizes of ECC and RSA keys. The key sizes for ECC are considerably smaller than RSA for the same level of security.

ECC Key Size (bits)	*RSA Key Size (bits)*	*Key Size Ratio*
163	1,024	1:6
256	3,072	1:12
384	7,680	1:20
512	15,380	1:30

Table 2.1: Comparison of Key Sizes.

2.2.4 Applications of ECC Based Protocols

After 25 years of their existence in cryptography, the factual benefits of using elliptic curves have finally been realized. ECC is progressively used in public key cryptography protocols, like for instance for carrying out digital signatures and key agreements. It is also used in digital signatures, pseudo-random generators, encryption, and for integer factorization algorithms such as Lenstra elliptic curve factorization (Bos et al., 2014).

Chapter 3

STATIC VULNERABILITY ANALYSIS

3.1 Introduction to Static Vulnerability Analysis

This section introduces static analysis and explains how it can be practiced to expose security vulnerabilities. Static analysis assesses application software without running it. This holds in contrast to dynamic analysis which executes the code and examine its runtime nature. Conventional approach used in static analysis consists of data flow analysis and model checking. Data flow analysis was introduced by Gary A. Kildall (Kildall, 1973), analyzes each sentence in a code to generate a set of information related to that sentence. The information may be values, variable names, or mathematical expressions, depending on the target of definitive analysis (Kildall, 1973). In this work, static vulnerability analysis of various ECC based authentication and key exchange protocols was performed, and possible vulnerabilities were highlighted using data flow analysis (a static vulnerability analysis techique).

The static vulnerability testing in this work followed a three step approach:

1. A threat model was first formulated in which the assumptions were articulated and the threats under investigation were identified.
2. The actual threat (vulnerability) analysis was performed on the protocols to find whether they were indeed vulnerable to the threats.
3. Solutions to make the protocols secure against the threats were identified, and were then applied to the protocols to test whether they made the protocols secure

against the identified threats.

3.2 Analysis of ECC Based Protocols

The first step in the analysis of ECC based protocols which we consider in this thesis is to design a threat model. Threat modeling is a method for evaluating the security of a software application. It looks at a system from a possible attacker's mindset. As shown in Fig. 3.1, to construct a threat model, it is necessary to specify the assumptions under which the protocols will be analyzed to identify threats (vulnerabilities) in them. The threat model also contains the various threats that pose security risks to the protocol.

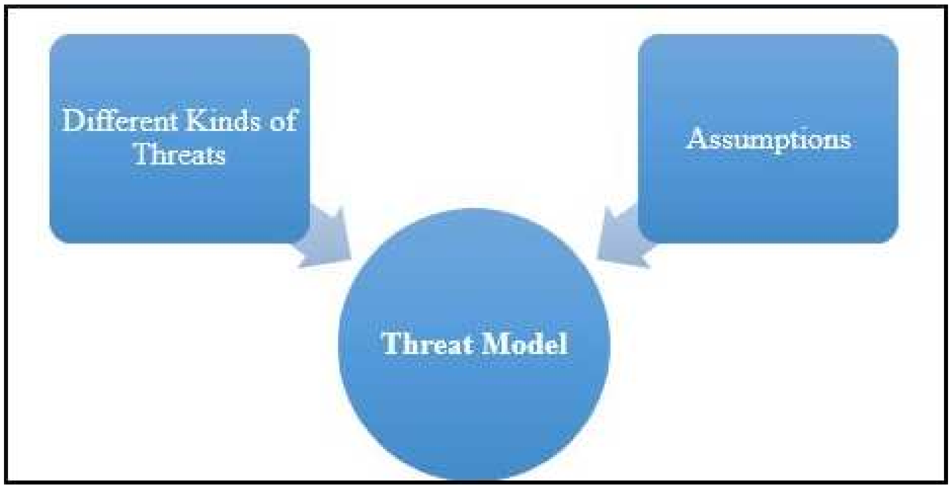


Figure 3.1: Threat Model

The next step (Fig. 3.2) is to choose authentication protocols to perform static crypt-analysis. This work focused solely on protocols that use ECC. Some of the other criteria used for selecting protocols include:

1. Protocol recency.
2. Variation in usage of authentication factors (e.g. smart cards, RFIDs, memory drives, etc.).
3. Variation in techniques to implement security (e.g. timestamps, nonce, encryption, hashes, etc.).

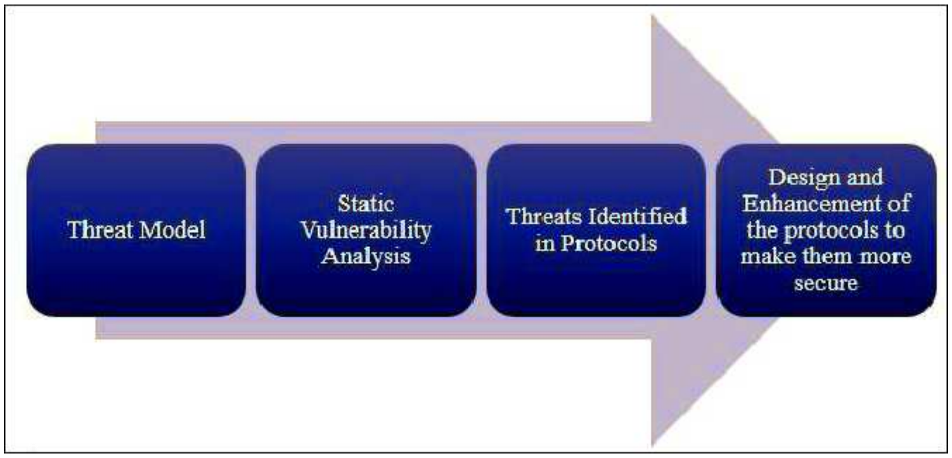

Threat Model
Static Vulnerability Analysis
Threats Identified in Protocols
Design and Enhancement of the protocols to make them more secure

Figure 3.2: The Threat Model Analysis Steps

Threat vulnerability analysis was performed on the protocols identified (discussed in the next chapter) to find whether they are indeed vulnerable to the threats. The last step is to suggest solutions to prevent the identified attack in order to make the protocols more secure.

3.3 Application of the Threat Model to Analyze Protocols

A threat model helps assess the probability, potential harm, and priority of attacks on a given ECC based protocol, and thus helps minimize threats in the protocols. It is often useful to define many different threat models for a system (of protocols), with each model representing a different set of analysis, where each set contains different types of vulnerabilities. Threat identification is intended to identify potential threats in system components, that might lead to a breach in the overall security of a system. The absence of security against a threat could denote a vulnerability whose risk could be reduced with the application of a countermeasure. Threats in protocols are identified by performing vulnerability analysis and finding flaws in protocol design of the protocol. Analysis results are used to suggest improvements to the protocol to prevent possible attacks and make it more secure.

3.4 Adversary Model and Evaluation Criteria

Many papers reviewing smart-card-based password authentication schemes have been published lately. But, in many of these findings, the publishers demonstrate attacks over conventional design's and introduce new protocols along with affirmation of their exceptional approach of their design's, while avoiding gain that their design fails to provide, hence ignoring extensions on which it fares low (Wang et al., 2012). Inspite of the lack of widespread assessment criteria, one more familiar attribute of these studies is that there isn't an appropriate security confirmation, that explains why protocols earlier believed to be secure show up to be vulnerable (Wang et al., 2012). The research history of this domain is summarized in Fig 3.3. The history of *break* and *fix* has been such that the new protocol comes into the picture after the existing protocol *breaks*.

In 1981, Lamport began the research on protocols using single factor authentication (password). In 1991, Chang Wu continued the research using two-factor authentication, which is password combined with smart card. This contributed to the growth of the research tree. The dotted line shows a clear demarcation between the tamper resistance and non-tamper resistance based protocols which use smart card. The protocols above the dotted line are established under the presumption which states that the smart card is tamper resistance and the protocols below the dotted line are established under the presumption which states that the smart card is non-tamper resistance. Hence, the protocols which have come into existence after making a practical presumption which states that the smart card is non-tamper resistance are more secure. The left side of the research tree displays the ECC. This research is expected to extend this research tree.

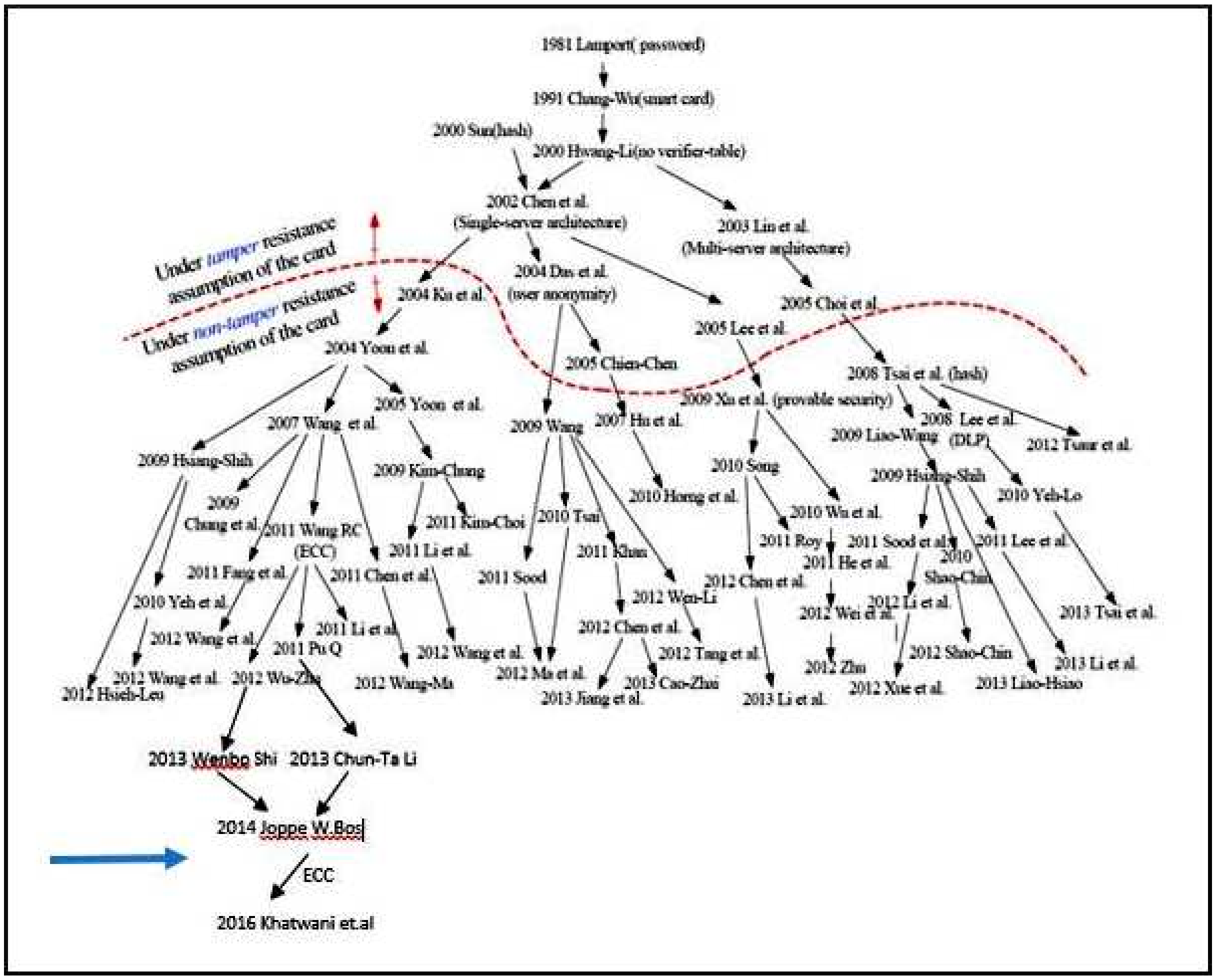


Figure 3.3: Research on Security Protocols

In common password authenticated key exchange protocols, the potential adversary is assumed to hold total authority of the communication channel such as by eavesdropping, intercepting and changing any broadcasted messages over the public network. Although this assumption can be considered correct for password-based authentication protocols, it might be less likely for password-based remote authentication systems that use smart cards. As investigated by Wang et al., harmful card readers can also deepen the security breakdown of these schemes (Wang, 2012).

In a real world, past session key(s) and components applied to compose the session key could be dropped on several grounds (Krawczyk, 2005), varying from a breach in the software/hardware system to a malicious plan of an insider or the arbitrary release of the particular session key at the time the session is damaged (Wang, 2012). Computing this capability to the vulnerability test allows the model to catch the risk of the known key attack. To examine the break of the server's long-term private key, the vulnerability test provides to the attacker the skill of mastering server's long-time private key. This method allows us to handle forward secrecy (Wang, 2012). In cryptography, forward secrecy is a attribute of secure protocols wherein compromise of long-time keys will not compromise previous session keys. It also saves previous sessions towards oncoming compromises of secret keys/passwords (Wikipedia, 2016b).

In remote user authentication schemes, a user is usually permitted to select their own identity accordingly in the registration phase. A user often usually selects an identity which can be easily remembered for convenience purposes. Therefore, these ID's are not so strong and hence can be speculated by an attacker A within polynomial time (Wang, 2012). Hence, it is fair to assume that A could guess all the (ID, PW) pairs in the Cartesian product $Did * Dpw$ within polynomial time where PW, Dpw and Did denote the password, password space and the identity space, respectively. Furthermore, it is favorable to presume that a decisive attacker may in some way may learn the victim's

ID. First, the user's ID is fixed and usually bound to follow a familiar pattern, also it can be easily guessed than the password (Bonneau et al., 2010). Second, in convention, unaware users tend to write down their ID on the card, and the adversary can learn the personal information of the user when they receive access to the card. Finally, the input ID is often presented in plain text on the screen's display and is prone to shoulder-surfing. As pointed out in (Yang et al., 2006), though the analysis of these schemes have a old history, no conventional series of attractive security properties were extensively identified for building such schemes (Wang, 2012).

Chapter 4

FRAMEWORK FOR CRYPTANALYSIS

In this chapter we will describe each of the protocols, the potential attacks on them, and the solutions against those attacks.

4.1 General Algorithms and Conditions for Various Attacks

In this section, general algorithms and conditions for the attacks that were successfully performed on the protocols chosen for this work are described.

4.1.1 Clogging Attack

The technique for almost all password authentication protocols is that the client (often a smart card reader, memory stick, or RFID) provdes its authorization to the server, which in turn computes particular arithmetic operations in order to validate the credentials. These protocols often function in multiple phases. The phases in which client and server authentication take place will be discussed in Chapter 5 after each protocol has been individually considered.

As shown in Fig. 4.1, the prime concept of the clogging attack is blockage of the message that contains login credentials between the client and the server (Garrett et al., 2015). This message is not encrypted in a few protocols, and encrypted in others. It may or may

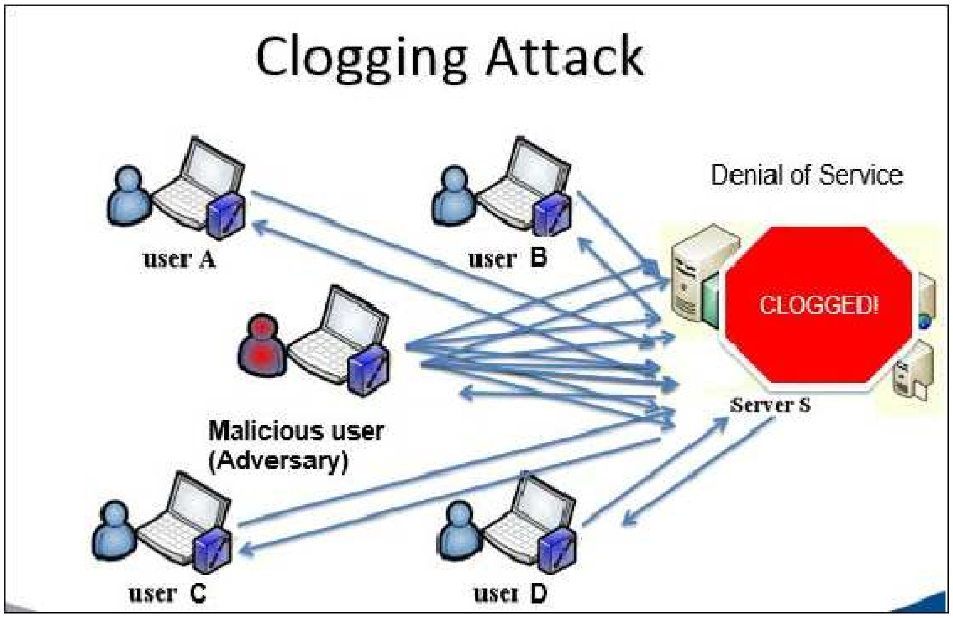


Figure 4.1: The Clogging Attack

not contain a timestamp. The attacker *replays* the intercepted message multiple times to enforce the server to carry out computationally intensive operations (in the case of (Khatwani and Roy, 2015), ECC operations), hence enforcing the server to lose its time and resources. Authorized users are blocked services in this way. Algorithm 1 depicts this type of attack.

Algorithm 1 The General Algorithm for Clogging Attack. (Garrett et al., 2015)

```
Intercept login message from client to server
if Timestamp is present then
    Change timestamp to suit requirements
else
    Keep message as is
end if
while The server is not completely clogged! do
    Replay the message to the server
end while
```

4.1.2 Application of Algorithm 1

The clogging elements we evaluate in this work rely on the computational and resource intensiveness of the operations in *elliptic curve cryptography* (ECC). The widely used ECC operations by most of the authentication schemes are:

1. Bilinear pairing
2. Scalar multiplication in group G
3. Map-to-point conversion

Let T_p, T_s, and T_{map} respectively be the time taken to perform a single bilinear pairing, scalar multiplication, and map-to-point conversation respectively. It has been shown in (Xu and Wu, 2015) that:

1. $T_p > T_s > T_{map}$
2. $T_p \approx 3 \times T_s$
3. $T_p \approx 4 \times T_{map}$

Further, let T_{modex} be the time taken by one modulo exponentiation operation. It has been proven (Farash and Ahmadian-Attari, 2014) that $T_p \approx 2 \times T_{modex}$ for the same level of security. The operation modular exponentiation has been shown to be very computationally intensive (Garrett et al., 2015). In fact, T_{modex} has been shown to be approximately a hundred times that of normal addition, multiplication, and bitwise XOR operations. We can, hence, conclude that all the ECC based operations bilinear pairing, scalar multiplication, and map-to-point conversation are computationally intensive.

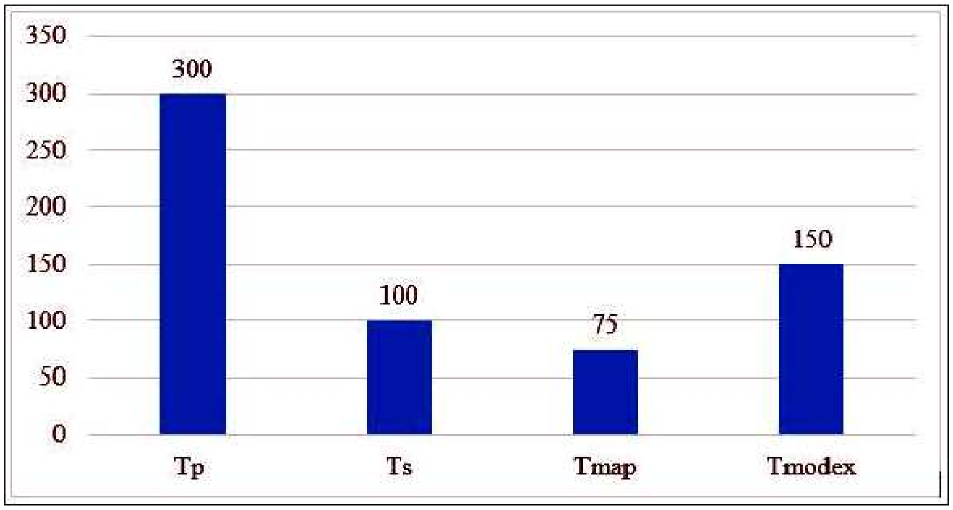


Figure 4.2: Comparison of ECC and Modex Operations

Hence, a protocol that uses ECC operation has a vulnerability to the *clogging attack*, a type of DoS in which the attacker abuse the computational intensiveness nature of ECC operations.

4.1.3 Database Attack

According to many experts, databases are still not secured properly in most organizations (Higgins, 2008). Database attacks go unnoticed as it takes less than a few seconds to hack in and out of a database. Therefore, it is not a wonder that a lot database attacks go undiscovered by large organizations until late after the information has been compromised. Attackers use clean methods to cause a breach in databases, such as exploiting weak authentication, using default passwords and exploiting familiar vulnerabilities (Higgins, 2008).

This analysis focuses on database connections which are weak and hence open to vulnerabilities. The front end client-server authentication stores passwords in the server's

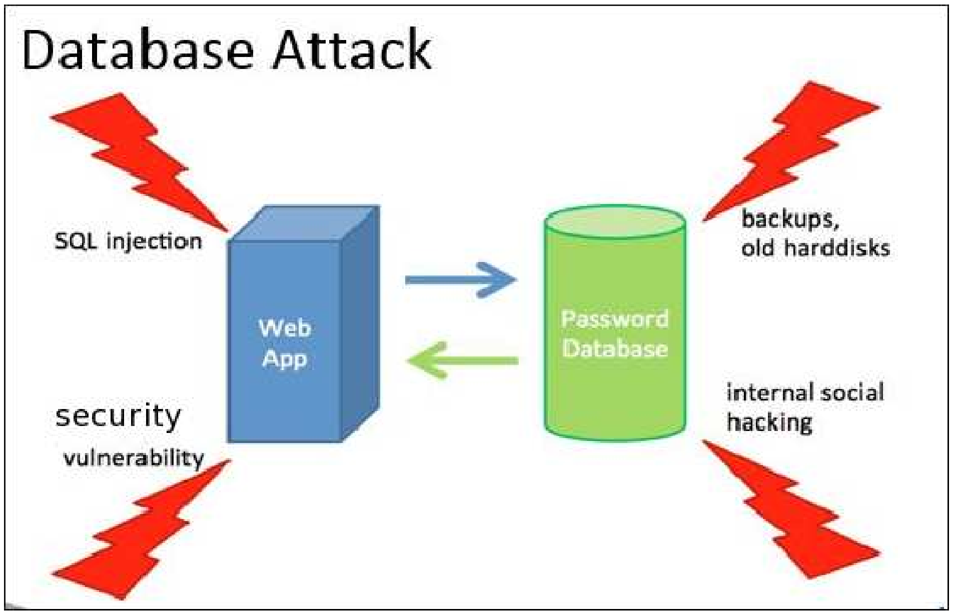


Figure 4.3: The Database Attack

back-end databases. If any password is compromised, then the database schema becomes vulnerable to attack which makes the protocol insecure (as explained in Algorithm 2). Passwords and their hashed forms are usually stored in relational databases. The most familiar approach to get unauthorized access to a database is to make a copy of the database by a technique called SQL injection. SQL injection attacks appears where the fields accessible for user input allow SQL statements through to query the database instantly. Web applications generally are the weakest link outside of the client's architecture (Higgins, 2008).

Internal attacks should also never be underestimated. There have been many cases of insider attacks which came as a result of a malicious user acquiring more system access than the user should have had (Higgins, 2008). Databases are usually attainable from inside organizations and passwords can easily be found in the source code or configuration files. This gives an opportunity to employees to access data and save it to a local disk or even transfer it to an external output device.

In the following chapters, database attacks to which recently used protocols are vulnerable are considered.

Algorithm 2 The General Algorithm for Database Attack

```
Intercept data access layer from application to back-end
if Encryption is present then
    Break the encryption to gain access to the database
else
    Access the database
end if
while The data are not corrupted and stolen do
    Inject malicious statements
end while
```

4.1.4 Man in the Middle Attack

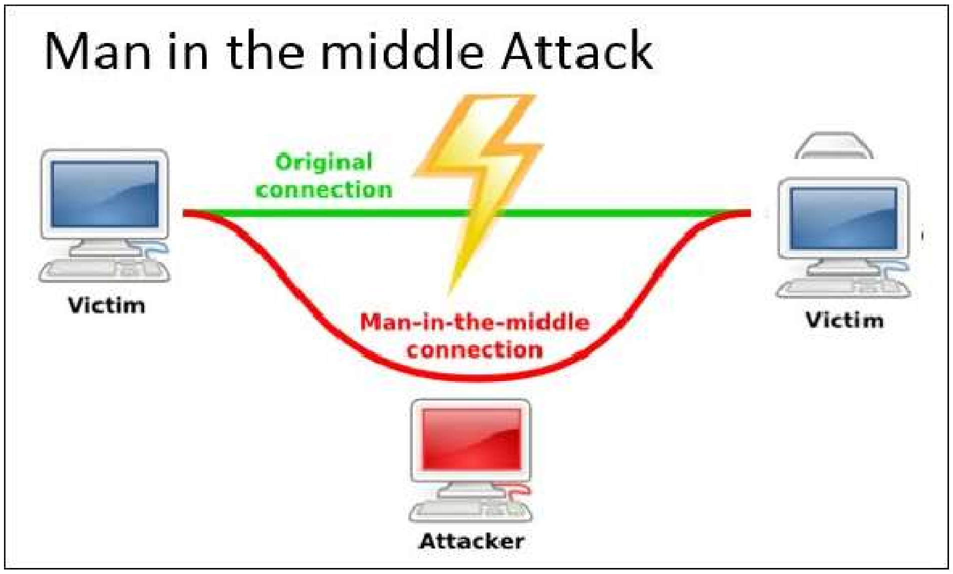


Figure 4.4: The Man in the Middle (MITM) Attack

As shown in Fig. 4.4, a man-in-the-middle attack can be used towards some of the cryptographic protocols. A man-in-the-middle attack needs an attacker to gain the capability to control and inject messages onto a communication medium. One example is eavesdropping, wherein the attacker atempts to make separate communications with the victims and relays messages between them to make them believe they are talking with each other over a private network, where as in reality the complete conversation is governed by the attacker. Attacker has the ability to intercept all the messages passing between the two victims and inject new ones. A few ECC based protocols claim to be secure against man-in-the-middle attacks. However, ECC protocols are still vulnerable to man-in-the-middle attacks, as shown in the next chapter.

Algorithm 3 The General Algorithm for Man in the Middle Attack

```
Intercept communication between two parties
if TTP is present then
    Acquires access and potentially alters the communication between two victims who
are bound to believe they are directly communicating with each other
else
    Acts as an invader who relays and modifies the message between two victims
end if
while The communication is not ended do
    Relay
end while
```

4.1.5 Application of Algorithm 3

An alternative of the Diffie-Hellman algorithm that uses elliptic curve cryptography, Elliptic curve Diffie-Hellman (ECDH) is an anonymous key agreement protocol that grants two entitites, each holding an elliptic curve public-private key pair, to originate a shared secret over a not so secure channel (Wikipedia, 2016a) (LaMacchia and Manferdelli, 2006), (Bos et al., 2009), (Sherwood et al., 2012). This shared secret can be used as a key, or it can be used to establish a different key, which may then be used to encrypt following interactions using a symmetric key cipher. The following illustration will show how a key is originated. For example, Alisa wants to establish a shared key with Rob, but the only medium available for both of them can be eavesdropped by a third party. Firstly, the domain parameters (that is, (p, a, b, G, n, h) need to be established. Let Alisa's key pair be (d_A, Q_A) and Rob's key pair be (d_B, Q_B). Each party must know the other party's public key before the initiation of the protocol.

Alisa computes $(x_k, y_k) = d_A Q_B$. Rob computes $(x_k, y_k) = d_B Q_A$. The shared secret which is same for both parties is x_k (the x coordinate of the computed point). The sole information about Alisa's private key that she gives out is her public key. Therefore, no other party except for Alisa can establish her private key, unless that interested party has the potential to solve the elliptic curve discrete logarithm problem. Rob's private key is equivalenty secure. Only Alisa and Rob can compute the shared secret (Wikipedia, 2016a).

Authentication is important to avoid man-in-the-middle attacks. If either of Alisa's or Rob's public keys is fixed, then man-in-the-middle attacks are defeated. Fixed public keys do not provide forward secrecy or key-compromise impersonation resilience, amid other improvised security measures. (Wikipedia, 2016a).

Chapter 5

CRYPTANALYSIS OF THE PROTOCOLS

This chapter describes each of the protocols selected for analysis in this work, the attacks on them, and ways of preventing the attacks. Two of the protocols mentioned in this chapter have also been illustrated in the already published work by the author (Khatwani and Roy, 2015). The following generic notations have been used to describe the protocols in this chapter. Specific notations for each individual protocol have been illustrated while describing each of them:

- $\mathbf{Z}_q^*$ denotes the finite field over q
- $\otimes$ denote (bitwise) exclusive OR
- $A \rightarrow B$: M denotes the propagation of the message M from user A to user B
- $\|$ denotes the concatenation operation
- In cryptography, a *nonce* is an arbitrary number that may only be used once

5.1 Choice of Protocols

The protocols chosen for analysis fall into the large domain of multi-factor authentication protocols. All of them employ user ID and password for authentication. Choice of these protocols is based on differences in the second authentication factor (smart cards, RFIDs, memory drives, etc .), and the tools used to serve confidentiality (timestamp,encryption, nonce, etc.).

Protocol	*Type*	*Factor Used*	*Confidentiality*
Moosavi	Authentication	RFID	Usage of Random Numbers
Xu	Authentication	Smart Card	Usage of Random Numbers
He	Authentication	RFID	Usage of Random Numbers
Hui	Authentication	Password	Usage of Encryption
Ammayappan	Key Exchange	–	Trusted Third Party

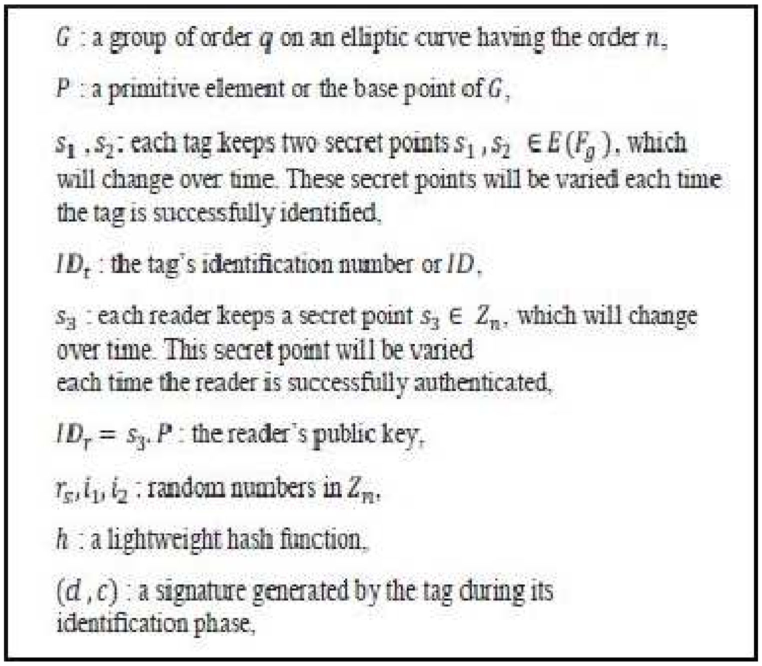
Table 5.1: Summary of the Differences in the Protocols

5.2 Moosavi et al.'s Protocol for RFID Implant Systems

The first protocol considered is that of (Moosavi et al., 2014). This is a mutual authentication scheme for an RFID implant system. (Moosavi et al., 2014) assert that their protocol is immune to various attacks including *denial of service* (DoS). But, their protocol is inherently vulnerable to clogging attacks (a form of DoS) that apply the algorithm of (Garrett et al., 2015). Most of the precursor protocols to that of (Moosavi et al., 2014) are vulnerable to clogging attacks. In this section, the mathematical groundwork that makes the protocols vulnerable to clogging attack is identified, and a desirable countermeasure is suggested.

5.2.1 Review of the Protocol

Moosavi et. al's protocol works in three phases: Reader Authentication and Verification, Tag Identification and Tag Verification. Figure 5.1 shows the notations used for the protocol.


G : a group of order q on an elliptic curve having the order n,

P : a primitive element or the base point of G,

s_1 , s_2: each tag keeps two secret points s_1 , s_2 $\in E(F_g)$, which will change over time. These secret points will be varied each time the tag is successfully identified,

ID_t : the tag's identification number or ID,

s_3 : each reader keeps a secret point $s_3 \in Z_n$, which will change over time. This secret point will be varied
each time the reader is successfully authenticated,

$ID_r = s_3.P$: the reader's public key,

r_s, i_1, i_2 : random numbers in Z_n,

h : a lightweight hash function,

(d , c) : a signature generated by the tag during its identification phase,

Figure 5.1: Notations for Moosavi et al.'s Protocol (Moosavi et al., 2014)

This protocol allows the two interacting parties, an *RFID implant tag* and a *reader*, to respectively validate and assure both identities. The assumption is that the communication between the reader and tag is weak (Khatwani and Roy, 2015). The protocol is shown in Algorithm 4.

5.2.2 Analysis of Moosavi et al.'s Protocol

This protocol is for an RFID implant system that has applications in microchip implant. The identities are the tag that is implanted, and the reader that verifies (and authenticates) the tag. Communication between the tag and the reader is through an insecure network. Additionally, the reader is connected to a database through a secured channel, so the reader and database is considered to be a single entity for analysis purpose (Khatwani and Roy, 2015). The protocol uses ECC techniques twice, once during tag identification (step I4 of Algorithm 4) and once during tag verification (step V6 of Algo-

Algorithm 4 Moosavi et al.'s Protocol for RFID Implant Systems

```
Reader Authentication and Verification
Reader R
1. Step A1. Select a random number $r_1 \in \mathbf{Z}_n$ and computes $R_1 = r_1 \cdot P$ as its public key.
2. Step A2. Initialize $i_1$ to 1.
3. Step A3. $R \rightarrow T$: $\{R_1, i_1\}$.
4. Step A4. Increment $i_1$ by $r_1$.
Tag T
1. Step A5. Verify if $i_1 \geq i_2$ ($i_2$ is initialized to 0).
2. Step A6. If the above is true, then set $i_2$ to $i_1$.
3. Step A7. Compute $r_3 = X(r_2 \cdot P) * Y(R_1)$, where $*$ is a non-algebraic operation over the abscissa of $(r_2 \cdot P)$ and the ordinate of $R_1$.
4. Step A8. $T \rightarrow R$: $\{r_3\}$.
Reader R
1. Step A9. Compute $R_2 = r_1 \cdot ID_t + r_3 \cdot s_3$.
2. Step A10. $R \rightarrow T$: $\{R_2\}$.
Tag T
1. Step A11. Verify if $(R_2 - r_1 \cdot ID_t) r_3^{-1} \cdot P = ID_r$.
2. Step A12. Reader $R$ gets verified if the above is true.
Tag Identification
Reader R
1. Step I1. Select $r_s \in \mathbf{Z}_n$, a random integer.
2. Step I2. $R \rightarrow T$: $\{r_s\}$.
Tag T
1. Step I3. Validate if $r_s \neq 0$. If success, then compute $s_2 = f(X(s_1)) \cdot P$.
2. Step I4. Select a random integer $k$ and calculate curve point $(x, y) = k \cdot G$
3. Step I5. Calculate $d = x \mod n$
4. Step I6. Validate if $d = 0$. If success, recalculate $d$ using a different $k$.
5. Step I7. Calculate value of ID as $ID_t = (Mb(X(s_1)) * Mb(X(s_2))) \cdot P$.
6. Step I8. Calculate $c = k \cdot (Hash(ID_t) + X(s_1) * d) \mod n$.
7. Step I9. Validate if $c = 0$. If yes, recalculate $c$ using a different $k$.
8. Step I10. $T \rightarrow R$: $\{ID_t, (d, c)\}$.
```

Moosavi et. al.'s scheme (contd.)

Tag Verification

Reader R

1. **Step V1.** Select a random integer $r_s \in \mathbf{Z}_n$.
2. **Step V2.** Compute public key $p_r = r_s \cdot P$.
3. **Step V3.** Verify if $d, c \in \mathbf{Z_n}$.
4. **Step V4.** If true, compute $h = Hash(ID_t)$.
5. **Step V5.** Compute $w = c^{-1} \mod n$, $u_1 = zw \mod n$, and $u_2 = dw \mod n$ respectively.
6. **Step V6.** Compute curve point $(x, y) = u_1 \cdot P + p_r$.
7. **Step V7.** Verify if $r = x \mod n$. If true, authenticate tag T.

rithm 4). However, in the course of this work it became clear that the tag verification phase has a dangling step in Step V6. The use of variable u_2 in the verification process is not mentioned in this protocol. Also, from (Moosavi et al., 2014) the description of verification variable r of Step V7 is not mentioned in this protocol.

5.2.3 Clogging Attack on Moosavi et al.'s Protocol

During this work, a security analysis of (Moosavi et al., 2014) was performed as in Section 5 of their paper (Moosavi et al., 2014). The adversary $\mathscr{A}$ has the same power as assumed by Moosavi et al. $\mathscr{A}$ needs to be able to read and modify the contents of messages over a not so secure medium during the Tag Identification and Tag Verification phases of the protocol (Khatwani and Roy, 2015). The line of attack in this work is a denial of service where the aim of $\mathscr{A}$ is to damage the whole RFID system. There are two steps (a map-to-point conversion in Step V6, and a scalar multiplication in Step V2) where ECC schemes are applied that $\mathscr{A}$ can try (Khatwani and Roy, 2015). However, it might be more profitable to render the Reader R useless, the line of attack selected here will try breaking the Reader R in a way that it will block services to authentic tags.

1. $\mathscr{A}$ intercepts an authentic message of $T \rightarrow R$: $\{ID_t, (d, c)\}$ from step **Step I10**.
2. As the message is not encrypted, $\mathscr{A}$ can always modify the (d, c) such that $d, c \in \mathbf{Z_n}$ holds (though $\mathscr{A}$ might not need to).
3. $\mathscr{A}$ simply relays $\{ID_t, (d, c)\}$ to R.

The following is performed by the Reader R:

1. **Step V1.** Select a random integer $r_s \in \mathbf{Z}_n$.
2. **Step V2.** Calculate public key $p_r = r_s \cdot P$.
3. **Step V3.** Validate if $d, c \in \mathbf{Z_n}$.
4. **Step V4.** If success, calculate $h = Hash(ID_t)$.
5. **Step V5.** Calculate $w = c^{-1} \mod n$, $u_1 = zw \mod n$, and $u_2 = dw \mod n$ respectively.
6. **Step V6.** Calculate curve point $(x, y) = u_1 \cdot P + p_r$.
7. **Step V7.** Validate if $r = x \mod n$. If success, authenticate tag T.

$\mathscr{A}$ would make the reader R re-run steps V1 through V7 to calculate the ECC operations many times. $\mathscr{A}$ has the potential to modify the incoming login requests from an authorized tag to R. As the ECC operations are computationally intensive (Farash and Ahmadian-Attari, 2014), the victimized Reader R spends considerable computing resources performing unwanted ECC computations (a map-to-point conversion in Step V6, and a scalar multiplication in Step V2) along with the other steps V1, V3 through V5, and V7 rather than any real work. Thus $\mathscr{A}$ clogs R with unwanted work and hence denies an authorized tag (user) any service. $\mathscr{A}$ needs only an ID of a single authentic tag to implement the clogging attack (Khatwani and Roy, 2015). It should be noted that

even DoS-resilient mechanisms are introduced on Reader R's side, it might be not a real obstruction for the attacker $\mathscr{A}$ as it can initialize new sessions with various identities in an interleaving pattern. Therefore, $\mathscr{A}$ may possibly implement the above attack procedure continuously. If distributed DoS attacks are implemented based on this strategy, the effect of it will be more serious (Khatwani and Roy, 2015).

5.2.4 Proposed Countermeasures from the Attack

In the early stages of the authentication phase, the reader may validate to see if the network address of the tag is authentic. It has to learn the network addresses of all the registered authentic tags. $\mathscr{A}$ could still deceit the network address of a authentic tag and replay the tag verification message. To prevent this deceit, a cookie exchange step may be inserted at the beginning of the tag verification phase of Moosavi et al.'s protocol (Khatwani and Roy, 2015). This step has been constructed as in the well known Oakley key exchange protocol (Orman, 1998).

1. The Tag T selects a pseudo-random number n_1 and sends it along with the message $\{ID_t, (d, c)\}$.
2. The Reader R upon receiving the message, acknowledges the message and sends its own cookie n_2 to T.
3. The next message from T must contain n_2, else T rejects the message and the Tag Verification request.

5.2.4.1 Security Analysis of the Fix

Had $\mathscr{A}$ spoofed T's IP address, $\mathscr{A}$ would not get n_2 back from R. Therefore $\mathscr{A}$ succeeds only in having R send back an acknowledgement, but not in launching the computationally intensive ECC based operations. Therefore the clogging attack is negated by these extra steps. It must be noted, though, that this process does not avoid the clogging attack but only resists it to some extent. This fix will completely work only if n_1, and n_2 are encrypted respectively by T's and R's private keys for a secure communication (Khatwani and Roy, 2015).

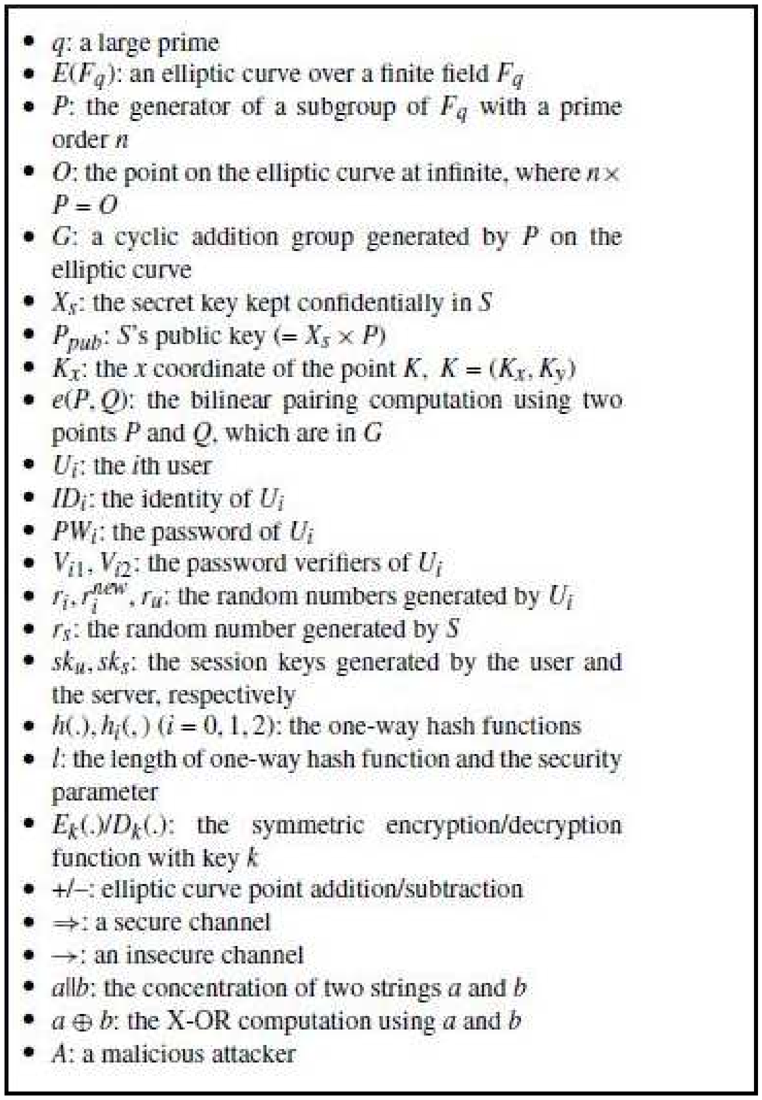
5.3 Xu et al.'s Smart Card Based Protocol

The next protocol we look at in this work is due to Xu et al. (Xu and Wu, 2015). It is an ECC based remote authentication protocol involving smart cards. Xu et al.'s protocol in (Xu and Wu, 2015) is an advancement over its predecessor, Li et al.'s protocol (Li, 2013), which (Xu and Wu, 2015) claimed to be vulnerable towards the off-line password guessing, user impersonation, and the denial of service (DoS) attacks. (Xu and Wu, 2015) introduced a new protocol, which they prove is immune against all three attacks.

We briefly present Xu et al.'s protocol. We then demonstrate a clogging attack on the protocol. Many protocols cited in their paper (Xu and Wu, 2015) are found to be vulnerable to clogging attack. We then suggest a suitable countermeasure towards clogging attack for this protocol (Khatwani and Roy, 2015).

5.3.1 Review of the Protocol

Xu et al.'s protocol works in five phases: Registration, Authentication, Password Change, Card Revocation, and User Eviction. Figure 5.2 shows the notations used for the protocol.



- q: a large prime
- $E(F_q)$: an elliptic curve over a finite field F_q
- P: the generator of a subgroup of F_q with a prime order n
- O: the point on the elliptic curve at infinite, where $n \times P = O$
- G: a cyclic addition group generated by P on the elliptic curve
- X_s: the secret key kept confidentially in S
- P_{pub}: S's public key $(= X_s \times P)$
- K_x: the x coordinate of the point K, $K = (K_x, K_y)$
- $e(P, Q)$: the bilinear pairing computation using two points P and Q, which are in G
- U_i: the ith user
- ID_i: the identity of U_i
- PW_i: the password of U_i
- V_{i1}, V_{i2}: the password verifiers of U_i
- r_i, r_i^{new}, r_u: the random numbers generated by U_i
- r_s: the random number generated by S
- sk_u, sk_s: the session keys generated by the user and the server, respectively
- $h(.), h_i(.)$ $(i = 0, 1, 2)$: the one-way hash functions
- l: the length of one-way hash function and the security parameter
- $E_k(.)/D_k(.)$: the symmetric encryption/decryption function with key k
- $+/-$: elliptic curve point addition/subtraction
- $\Rightarrow$: a secure channel
- $\rightarrow$: an insecure channel
- $a\|b$: the concentration of two strings a and b
- $a \oplus b$: the X-OR computation using a and b
- A: a malicious attacker

Figure 5.2: Notations for Xu et al.'s Protocol (Xu and Wu, 2015)

We present the *Registration, Authentication* phases of the protocol in Algorithm 5. The other phases are omitted as those are not needed to present the clogging attack (Khatwani and Roy, 2015).

Algorithm 5 Xu et. al.'s Scheme of Authentication

Registration Phase

User U_i

1. **Step R1.** Select identity ID_i, password PW_i.
2. **Step R2.** Select r_i
3. **Step R3.** Calculate $HPW_i = h_0(PW_i \parallel r_i)$
4. **Step R4.** $U_i \rightarrow S$: $\{ID_i, HPW_i\}$ through a *secure channel.*

Server S

1. **Step R5.** Validate ID_i, select N
2. **Step R6.** Calculate $k_i = h_0(X_s \parallel ID_i \parallel N)$ and $W_i = k_i \oplus HPW_i$.
3. **Step R7.** Store $\{W_i, P, P_{pub}\}$ into smart card.
4. **Step R8.** Store $\{ID_i, N\}$ in the server's database.
5. **Step R9.** $S \rightarrow U_i$: The smart card.

User U_i

1. **Step R10.** Upon receiving the smart card from S, U_i enters stores r_i in it.

Authentication

User U_i

1. **Step A1.** U_i inserts his smart card and inputs ID_i, and PW_i.
2. **Step A2.** Smart card chooses random $r_u \in \mathbf{Z}_n^*$.
3. **Step A3.** Compute $HPW_i = h_0(PW_i \parallel r_i)$
4. **Step A4.** Compute $R_1 = r_u \times P = (R_{1x}, R_{1y})$, $R_2 = r_u \times P_{pub} = (R_{2x}, R_{2y})$.
5. **Step A5.** Compute $B_1 = h_0((W_i \oplus HPW_i) \parallel h_0(R_{1x} \parallel R_{2x} \parallel R_{1y} \parallel R_{2y}))$.
6. **Step A6.** Compute $CID_i = ID_i \oplus h_0(R_{2x} \parallel R_{2y})$.
7. **Step A7.** $U_i \rightarrow S$: $\{CID_i, B_1, R_1\}$.

Server S

1. **Step A8.** Compute $R'_2 = X_s \times R_1$, $ID'_i = CID_i \oplus h_0(R'_{2x} \parallel R'_{2y})$, $k_i = h_0(X_s \parallel ID'_i \parallel N)$.
2. **Step A9.** Verify if $B_1 = h_0(k_i \parallel h_0(R_{1x} \parallel R'_{2x} \parallel R_{1y} \parallel R'_{2y}))$. Abort if not true, continue if true.
3. **Step A10.** Choose random $r_s \in \mathbf{Z}_n^*$.
4. **Step A11.** Compute $R_3 = r_s \times P = (R_{3x}, R_{3y})$, $K_s = r_s \times R_1 = (K_{sx}, K_{sy})$.
5. **Step A12.** Compute $B_2 = h_0(k_i \parallel h_0(R'_{2x} \parallel R'_{2y}))$.
6. **Step A13.** Compute $B_3 = h_1(R_{1x} \parallel R_{1y} \parallel R_{3x} \parallel R_{3y} \parallel ID_i \parallel S \parallel B_2 \parallel K_{sx} \parallel K_{sy})$.
7. **Step A14.** Compute session key $sk_s = h_2(R_{1x} \parallel R_{1y} \parallel R_{3x} \parallel R_{3y} \parallel S \parallel ID_i \parallel B_2 \parallel K_{sx} \parallel K_{sy})$.
8. **Step A15.** $S \rightarrow U_i$: $\{R_3, B_3\}$.

Xu et. al.'s scheme (contd.)

<u>`User` U_i</u>

1. **Step A16.** Compute $K_u = r_u \times R_3 = (K_{ux}, K_{uy})$ and $B_2' = h_0((W_i \oplus HPW_i) \parallel h_0(R_{2x} \parallel R_{2y}))$.
2. **Step A17.** Verify if $B_3' = h_1(R_{1x} \parallel R_{1y} \parallel R_{3x} \parallel R_{3y} \parallel ID_i \parallel S \parallel B_2 \parallel K_{ux} \parallel K_{uy})$. Abort if not true, continue if true.
3. **Step A18.** Compute session key $sk_s = h_2(R_{1x} \parallel R_{1y} \parallel R_{3x} \parallel R_{3y} \parallel S \parallel ID_i \parallel B_2 \parallel K_{ux} \parallel K_{uy})$.

5.3.2 Clogging Attack on Xu et al.'s Protocol

The adversary $\mathscr{A}$ has the same power as assumed by Xu et al's (Xu and Wu, 2015) while exposing the flaws of Li et. al's protocol (Li, 2013). $\mathscr{A}$ needs to be able to read and modify the contents of messages over an insecure channel during the Authentication phase of the protocol (Khatwani and Roy, 2015).

1. $\mathscr{A}$ intercepts a valid login request $(\{CID_i, B_1, R_1\})$ from step **Step A7**.
2. $\mathscr{A}$ simply replays this login message to S.

Lemma 5.3.1. *The server S is bound to perform steps A8 through A15 if* $\{CID_i, B_1, R_1\}$ *is valid.*

Proof. The only step that could prevent the server S from executing further steps is **Step A9**. We claim $B_1 = h_0(k_i \parallel h_0(R_{1x} \parallel R_{2x}' \parallel R_{1y} \parallel R_{2y}'))$ would always be true if B_1 and R_1 are legit. To verify our claim we note $B_1 = h_0((W_i \oplus HPW_i) \parallel h_0(R_{1x} \parallel R_{2x} \parallel R_{1y} \parallel R_{2y}))$. Plugging in the value of $W_i = k_i \oplus HPW_i$, we have $B_1 = h_0(k_i \parallel h_0(R_{1x} \parallel R_{2x} \parallel R_{1y} \parallel R_{2y}))$. Now, $R_1 = r_u \times P = (R_{1x}, R_{1y})$, $R_2 = r_u \times P_{pub} = (R_{2x}, R_{2y})$ and $R_2' = X_s \times R_1$. Therefore, if we have a valid B_1 and R_1, the values of R_2 and R_2' always match, which verifies the claim.

The adversary $\mathscr{A}$ replays the message $\{CID_i, B_1, R_1\}$ many times and utilizes the server S compute steps A8 through A15 (by Lemma 5.3.1). The steps contain several scalar multiplications, and map-to-point conversations. $\mathscr{A}$ could probably save all the incoming login request messages from an authorized user to S for future replay. As the mathematical operations in the steps are computationally intensive, the victimized server spends ample computing resources performing worthless calculations. Thus $\mathscr{A}$ clogs S with useless work and hence rejects any legitimate user any service. $\mathscr{A}$ needs only one message from a single authentic user to perform the clogging attack repeatedly (Khatwani and Roy, 2015).

However, as (Xu and Wu, 2015) noted, the attacker $\mathscr{A}$ will not get a authorized access by this replay attack. As r_u will differ every time, the validation $B_2' = h_0((W_i \oplus HPW_i) \parallel h_0(R_{2x} \parallel R_{2y}))$ at the user's side would fail. However, the objective of $\mathscr{A}$ here is not to attain unauthorized access, but to crash the server by implementing a clogging attack (Khatwani and Roy, 2015).

5.3.2.1 Clogging Attack Performed on Other Similar Schemes

The clogging attack performed on (Xu and Wu, 2015) can also be performed on Li's protocol (Li, 2013). The clogging attack on Li's protocol is more effective because Li's protocol (Li, 2013) utilizes the utmost computationally intensive bilinear pairing operation. The protocols by (Xu and Wu, 2015) and (Li, 2013) are vulnerable as the user's smart card does not encrypt the message it sends to the server for login and authentication. Thus the attacker has the opportunity to change the context of message.

5.3.2.2 Proposed Countermeasures from Clogging Attack

Replay attacks on utmost smart card based protocols are possible as their security heavily depend on computationally intensive operations (in this case ECC), and by default the messages are unencrypted. This vulnerability is usually unnoticed, as the common result of a replay is not a DoS. An approach to minimizing these attacks on Xu and Wu's protocol (and smart card based protocols in general) would be

1. U_i uses a time stamp T in **Step A7**, and S validates it in **Step A8**. The time stamp should also be encrypted in a manner such that $\mathscr{A}$ cannot tamper with it.

2. S validates if multiple login requests regularly directs from the same user. This *minimizes* the opportunities of a reply.

The chances of a replay are minimized but not eliminated as $\mathscr{A}$ can gain many authentic user IDs and send invalid login requests repeatedly from various IDs. Alternatively, $\mathscr{A}$ can save different (authentic) login requests over a period of time, and replay them repeatedly (Khatwani and Roy, 2015).

Another way to avoid clogging attack: The mathematical basis of the protocols' vulnerability to clogging attacks is modular exponentiation. An approach to altogether bypass clogging attacks would be to *encrypt* all the messages between U_i and S. Carrying out would call for a key exchange step, where every user has a private key and a public key (Khatwani and Roy, 2015). The server learns the public key, and can decrypt a message encrypted by a user's private key. Hence the server makes sure that the message is from a authentic user before it computes the expensive ECC operations. This approach comes with a cost and is based on the layer of security desired. This countermeasure works for all protocols (whether or not they are smart card based) (Khatwani and Roy, 2015).

5.3.3 Weak Authentication and SQL Injection Attack

The proposed scheme in this paper does not mention how the ID and N are stored in the account table during the registration phase. This paper does not mention if an encryption is present at the database level. Under the assumption that the account table is accessible to an untrusted source, ID and N can be dynamically constructed in the query thus leading to a SQL injection attack. Also, under the assumption that ID and N are unprotected, weak authentication attack is possible as well. These attacks lead to loss in confidentiality, authentication, authorization and integrity. Hence, we conclude that the database layer authentication is not strong enough to protect unknown users from gaining access to the database.

5.3.3.1 Proposed Security Countermeasures for Weak Authentication Attack

SQL Injection attacks can be overcome by parsing and authenticating SQL communications to ensure they are not corrupted. If the protocol shows how the ID and N are stored and protected in a table, then the attack will be minimized. The weak authentication attack can be protected by adding more access layers and by enforcing strict user privileges.

5.3.4 Unauthorized Access Attack

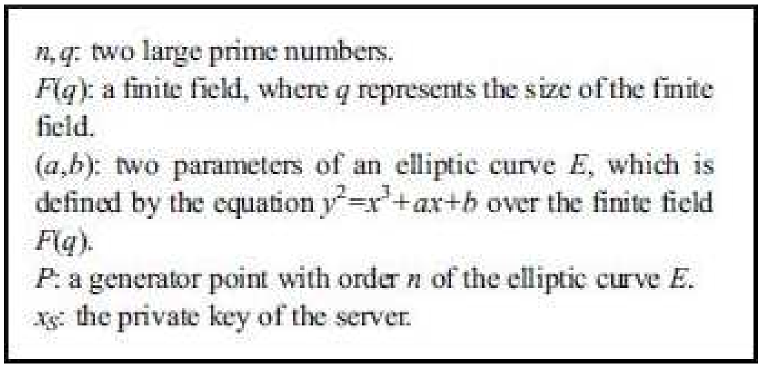
In (Xu and Wu, 2015), sections 4.4 card revocation and 4.5 user eviction talk about a smart card being stolen and verified after being stolen. This can lead to a potential risk. If the smart card is seized by the attacker, the user ID and the password will be accessible to the attacker. The attacker can enter malicious input and potentially hack the database server. Under the assumption, that the smart card re-registration phase is unprotected, unauthorized access attack is possible on this protocol. For example, an unauthorized client can steal the smart card or eavesdrop on the data exchanged between an authorized client and server. This critical information such as user ID and password can be used to cause other major attacks.

5.3.4.1 Proposed Security Countermeasures Against Unauthorized Access Attack

To counter unauthorized access, card revocation should be handled by more efficient algorithms. Validation methods, access control mechanisms and password policies granted by the server provide accomplished ways of avoiding unauthorized access. A network firewall, consisting of a software program, hardware device, or a consolidation of the two, safeguards an internal network against harmful access from the outside. Network firewalls can also be contructed to block access from internal users to the outside.

5.4 He et al.'s RFID Based Authentication Protocol

The next protocol considered in this work is due to (He et al., 2014). This is an RFID based authentication protocol to sustain identity privacy. He et al.'s protocol is an improvement over (Liao and Hsiao, 2014), which claimed to be vulnerable towards insider, and impersonation attacks. We briefly show the protocol in Algorithm 6. Figure 5.3 shows the notations used for the protocol. This protocol has been claimed to be resistant from different attacks (He et al., 2014). We demonstrate a clogging attack on the protocol and notice that it is inherently vulnerable to clogging, weak authentication and SQL injection attacks. The proposed ECC-based RFID authentication scheme consists of two phases, i.e., the setup phase and the authentication. We present the protocol in Algorithm 6.



n, q: two large prime numbers.
$F(q)$: a finite field, where q represents the size of the finite field.
(a,b): two parameters of an elliptic curve E, which is defined by the equation $y^2=x^3+ax+b$ over the finite field $F(q)$.
P: a generator point with order n of the elliptic curve E.
x_S: the private key of the server.

Figure 5.3: Notations for He et al.'s Protocol (He et al., 2014).

5.4.1 Clogging Attack on He.et.al's Protocol

The adversary $\mathscr{A}$ has an equal power as supposed by (He et al., 2014) while showing the flaws of Liao's protocol (Liao and Hsiao, 2014). $\mathscr{A}$ needs only to be able to read and modify the contexts of messages over an insecure channel while in the authentication phase. Steps taken by A to execute the clogging attack are as follows:

- $\mathscr{A}$ intercepts a authentic login request $m_2 = R_2, Auth_T$ from step Step A4.
- As the message is not encrypted, $\mathscr{A}$ changes $Auth_T$ to a arbitrary garbage value $Auth_{\mathscr{A}}$.
- $\mathscr{A}$ then sends $\{R_2, Auth_{\mathscr{A}}\}$ to the server S.

The following steps are carried out by the server S:

1. The server S calculates $TK_{S1} = X_S R_2$, $TK_{S2} = r_1 R_2$, and $X'_T = (Auth\mathscr{A}_{\oplus} TK_{S2}) - TK_{S1}$.
2. The server search its database for X'_T. It is not found, and therefore the server terminates the session.

Algorithm 6 He et al.'s Scheme

Set Up Phase

1. **Step S1.** The server S chooses a random number $x_S \in Z_n^*$ such that $P_S = x_S P$.
2. **Step S2.** The server S chooses a random point X_T on the elliptic curve E for each tag.
3. **Step S3.** S stores the ID-verifier X_T and parameters into the tag's memory. The server also keeps x_S as its private key, and stores X_T into its database.

Authentication Phase

Server S

1. **Step A1.** Choose a random number $r_1 \in Z_n^*$ then computes $R_1 = r_1 P$.
2. **Step A2.** $S \rightarrow T$: $m_1 = R_1$.

Tag T

1. **Step A3.** Choose a new random number $r_2 \in Z_n^*$ then calculates $R_2 = r_2 P$, $TK_{T1} = r_2 P_S$, $TK_{T2} = r_2 R_1$, and $Auth_T = (X_T + TK_{T1}) \oplus TK_{T2}$.
2. **Step A4.** $T \rightarrow S$: $m_2 = R_2, Auth_T$.

Server S

1. **Step A5.** The server S calculates $TK_{S1} = X_S R_2$, $TK_{S2} = r_1 R_2$, and $X_T = (Auth_T \oplus TK_{S2}) - TK_{S1}$.
2. **Step A6.**The server search its database for X_T. If not located, the server terminates the session. Otherwise, the server calculates $Auth_S = (X_T + 2TK_{S1}) \oplus (2TK_{S2})$.
3. **Step A7.** $S \rightarrow T$: $m3 = \{Auth_S\}$.

Tag T

1. **Step A8.** The tag checks whether $(X_T + 2TK_{T1}) \oplus (2TK_{T2})$ and $Auth_S$ are same. If they are different, the tag terminates the session, else, the server is authenticated.

$\mathscr{A}$ would now play over the steps many times and cause the server S calculate the elliptic curve operations several times in Steps A5 and A6. $\mathscr{A}$ can possibly modify the incoming requests from an authorized Tag to S. Thus, $\mathscr{A}$ clogs S with worthless task (ECC operations) and hence denies an authorized user any service. $\mathscr{A}$ only requires the value of X_T of one authorized user to perform the clogging attack constantly.

5.4.1.1 Proposed Countermeasures for Clogging Attack

One reason for the vulnerability of this protocol is that the messages m_1, m_2, m_3 are not encrypted; if the adversary gains access to the message m_2, then repeated transmission of this message can clog the server. Also, the timestamp checking is not used between the tag and the server. Although, random numbers are generated in every step and used for real time calculation, when messages are not encrypted and timestamps are not used, the messages are open to the adversary to modify and replay. The solution could be to add a timestamp T_i to R_i or to m_2. The server can validate the authenticity of the timestamp before it performs the elliptic curve operations.

5.4.2 Weak Authentication Attack

(He et al., 2014) paper does not mention how X_T(ID identifier) is stored in the database. Weak authentication schemas grant attackers to acquire the identity of authorized database users. Attack schemes include social engineering, brute force attacks, etc (Higgins, 2008). Under the assumption, that X_T is unprotected, the adversary can gain access to X_T resulting in illegitimate data access, data availability or corruption. Hence, weak authentication attack is possible assuming the database layer authentication is not strong enough to protect unknown users from gaining access to the database.

5.4.2.1 Proposed Security Countermeasures for Weak Authentication Attack

To prevent weak authentication attack, enforcement of two-factor authentication or password is a must (Higgins, 2008). Had the protocol (He et al., 2014) included security and protection for X_T, the weak authentication attack could have been avoided. Weak authentication attack can also be prevented by adding more access layers and by enforcing stricter user privileges.

5.4.3 Desynchronization Attack

A desynchronization attack is a typical RFID related threat in which a tag's key stored in the back-end tables and the tag's memory would be different, because an attacker blocks the communication between the parties. (He et al., 2014) paper mentions that X_T is not performed (X_T is an ID identifier). Their paper claims that the proposed scheme provides scalability, availability and DoS resistance by not updating X_T. To provide privacy protection, many RFID authentication schemes refresh the tag's secret information, in the back-end tables and in the tag, after a successful protocol run. Hence, integration of private information between the database and the tag is important for consecutive validations. The most serious threat to which an RFID tag is vulnerable to is the desynchronization attack. During the past years, RFID (Radio Frequency Identification) technology became ubiquitous as they are low cost, it is used in every field and hence this vulnerability raises a threat in the area of data protection.

5.4.3.1 Proposed Security Countermeasures for Desynchronization Attack

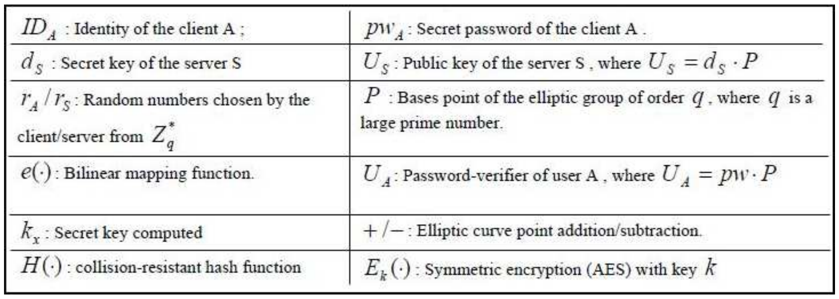
One of the countermeasures is offcourse to update the X_T after each run. It is an intractable task to design the lightweight RFID authentication protocol, because the security engineer must deal with the establishment amid cost, performance and security. In the future, security engineers can be made aware of the trade offs so as to build an efficient protocol.

5.5 Hui et al.'s Protocol

The next protocol in consideration is due to Hui et al. (Shao-hui et al., 2012). In that paper, after pointing out the weakness of the password change phase of Islam et al. (Islam and Biswas, 2013) and after evaluation several other password authentication schemes, (Shao-hui et al., 2012) have demonstrated a new password-based authentication and update scheme using ECC and showed that it can withstand different attacks. But, in this work the protocol is shown to be primarily susceptible to clogging, weak authentication and SQL injection attacks.

5.5.1 Review of the Protocol

Hui et al.'s protocol works in four stages: registration phase, password authentication phase, session key distribution phase and password change phase. We present the protocol in Algorithm 7. Figure 5.4 shows the notations used for the protocol.



ID_A : Identity of the client A ;	pw_A : Secret password of the client A .
d_S : Secret key of the server S	U_S : Public key of the server S , where $U_S = d_S \cdot P$
r_A / r_S : Random numbers chosen by the client/server from Z_q^*	P : Bases point of the elliptic group of order q , where q is a large prime number.
$e(\cdot)$: Bilinear mapping function.	U_A : Password-verifier of user A , where $U_A = pw \cdot P$
k_x : Secret key computed	$+/-$: Elliptic curve point addition/subtraction.
$H(\cdot)$: collision-resistant hash function	$E_k(\cdot)$: Symmetric encryption (AES) with key k

Figure 5.4: Notations for Hui et al.'s Protocol (Shao-hui et al., 2012)

The final two phases are omitted since they are insignificant to the clogging attack demonstration. The protocol claims to be immune against replay attacks; however, we find that it is vulnerable against other attacks as well.

Algorithm 7 Hui et al.'s Scheme

Registration Phase

Client C

1. **Step C1.** Client C chooses identity ID_C, pw_C.
2. **Step C2.** $U_C = pw_C \cdot P$.
3. **Step C3.** Client C sends ID_C and U_C to the server S.

Server S

1. **Step C4.** Server S stores ID_C and U_C in a write protected file.

Password Authentication Phase

Client C

1. **Step C5.** Enter ID_C, pw_C.
2. **Step C6.** Choose random number $r_C \in Z_n^*$.
3. **Step C7.** Compute $W_C = r_C \cdot pW_C \cdot U_S$, $R_C = r_C \cdot U_C = (k_x, k_y)$, $Y_C = r_C \cdot P$.
4. **Step C8.** Compute $M_1 = E_{kx}(ID_C, Y_C)$.
5. **Step C9.** Client C sends message (ID_C, W_C, M_1) to server S.

Server S

1. **Step C10.** Upon receiving the messages, compute $R'_C = W_A \cdot d_s^{-1} = (k'_x, k'_y)$.
2. **Step C11.** Check if $ID'_C = ID_C$, $e(Y'_C, U_C) = e(R'_C, P)$ hold. If the equations do not hold, then stop the session.
3. **Step C12.** Choose $r_s \in Z_q^*$.
4. **Step C13.** Server S sends message $M_2 = R'_C + W_S$, $M_3 = H(W_S)$ to Client C.

Client A

1. **Step C14.** Compute $W'_S = M_2 - R_C$.
2. **Step C15.** Check if $H(W'_S) = M_3$ holds. If yes, calculate $M4 = H(R'_C, W'_S)$ and send it to Server S, else abort.

5.5.2 Clogging Attack on Hui et al.'s Protocol

As before, we essentially need attacker $\mathscr{A}$ to be able to read and modify the contexts of messages over an not so secure channel.

- $\mathscr{A}$ intercepts a authentic login request (ID_C, W_C, M_1) from Step C9

- Since the message is unencrypted, $\mathscr{A}$ changes ID_C to ID_B
- $\mathscr{A}$ then sends (ID_B, W_C, M_1) to the server S

The server S carry out steps C10 through C15 since the ID matches. $\mathscr{A}$ would now rerun the steps many times and cause the server S calculate the intensive elliptic curve bilinear mapping function of Step C11 and collision-resistant hash function of Step C13, C15 many times. Hence, as in the earlier case, the server gets clogged performing useless calculations.

5.5.2.1 Proposed Security Countermeasures against Clogging Attack

The vulnerability of the mentioned protocol arises as the message (ID_C, W_C, M_1) is not encrypted and the timestamp is not used. The protocol can be strengthened by using strong encryption of the messages M_1, M_2 and M_3. Also, adding timestamp to the messages at the time of run will ensure privacy and thus prevent replay attacks which leads to clogging.

5.5.3 Unauthorized Access Attack

Unauthorised access is the method of attaining access to a resource, network, system or any other application without approval. Unauthorised access might occur if a user tries to access a domain which they are not allowed to access. Unauthorised access may also be a result of unchanged default policies or absence of prescribed access policy documentation (Telelink, 2016). In the paper, (Shao-hui et al., 2012), ID_C and U_C are stored in a write protected file in the step $C4$ of the registration phase. Their paper (Shao-hui et al., 2012) does not mention how the file is stored and also if it is protected using standard security measures. Under the assumption, that the ID_C and U_C are unprotected, an unauthorized client could potentially steal the write protected file or eavesdrop on the data interchanged between an authorized client and the directory server. When the file containing the ID and password becomes accessible to an unauthorized user, the protocol is compromised under this attack.

5.5.3.1 Proposed Security Countermeasures Against Unauthorized Access Attack

Unauthorized access can occur from outside if the organization is connected to an extranet or internet or, from inside the organization. The server directory provides experienced ways of preventing illegitimate access. Some of the methods include utilizing authentication methods, access control mechanisms and password policies. A network firewall safeguards an internal network against harmful access from the outside. Network firewalls can also be contructed to block access from internal users to the outside (Telelink, 2016).

5.5.4 Unauthorized Tampering Attack

Tampering is the unauthorized modification of data by an unauthorized user. When an unauthorized user gains access to the write protected file in which ID_C and U_C are stored or if they block communication between client and a server, they have the means to change the server data. These illegitimate modifications may contain:

- Illegitimate modification of data
- Illegitimate change of configuration data
- Alter or cancellation of client's request to the server

- Alteration of the server's response to the client

Adversary $\mathscr{A}$ can cause all other kinds of attacks if the write protected file is accessible to them as it contains all the user identity's and passwords.

5.5.4.1 Proposed Security Countermeasures Against Unauthorized Tampering Attack

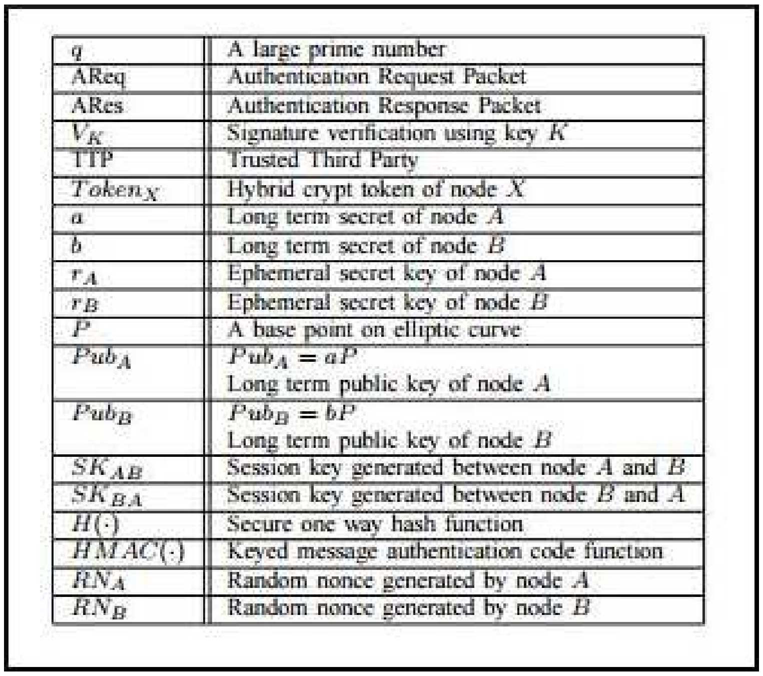
An adversary $\mathscr{A}$ may modify a client's request to the server, deny the request, or modify the server's response to the client. The countermeasures include using Secure Socket Layer (SSL) protocol to solve this issue by authorizing information at either end of the established connection. Another solution could be to store the ID_A and U_A in the cloud instead of a physical device.

5.6 Ammayappan et al. Protocol

The final protocol considered in this work is that of Ammayappan et al. (Ammayappan et al., 2011). This is a key agreement protocol and works in a mobile ad hoc networks (MANET) based domain. Ammayappan et al. show their protocol to be immune to man in the middle (MITM) attack. They claim that the protocol's security is based on the ECC logarithm.

5.6.1 Review of the Protocol

The protocol by (Ammayappan et al., 2011)) works in two phases: registration phase and active phase. Figure 5.5 shows the notations used for the protocol.



q	A large prime number
AReq	Authentication Request Packet
ARes	Authentication Response Packet
V_K	Signature verification using key K
TTP	Trusted Third Party
$Token_X$	Hybrid crypt token of node X
a	Long term secret of node A
b	Long term secret of node B
r_A	Ephemeral secret key of node A
r_B	Ephemeral secret key of node B
P	A base point on elliptic curve
Pub_A	$Pub_A = aP$ Long term public key of node A
Pub_B	$Pub_B = bP$ Long term public key of node B
SK_{AB}	Session key generated between node A and B
SK_{BA}	Session key generated between node B and A
$H(\cdot)$	Secure one way hash function
$HMAC(\cdot)$	Keyed message authentication code function
RN_A	Random nonce generated by node A
RN_B	Random nonce generated by node B

Figure 5.5: Notations for Ammayappan et al.'s Protocol (Ammayappan et al., 2011)

This protocol uses trusted third party (TTP) as a certifying authority. The protocol is presented in Algorithm 8.

5.6.2 Man in the Middle Attack (MITM) on the Protocol

The adversary *E* has an equal power as supposed by Ammayappan et al. in (Ammayappan et al., 2011) while conducting the security analysis of the protocol. *E* is allowed to read and change contexts of messages over a not so secure medium (in the active phase of this protocol). The line of our attack is a man in the middle (MITM) attack where the objective of *E* would be to impersonate one party (e.g. *A*) while communicating with the other (e.g. *B*). E is allowed to read and change contexts of messages over a not so secure medium in the active phase of this protocol. Ammayappan et al. claim that their protocol ensures security through the elliptic curve discrete logarithm. However, the protocol may be compromised even with ECC being used in the protocol as a security

Algorithm 8 Ammayappan et al.'s Scheme

```
                         Node A
1. Step A1. Select a random number $r_A$ and computes $Q_A = r_A$Âů$P$ as its public key.
2. Step A2. Node $A$ sends $AReq(Token_A, RN_A, Q_A)$ to Node $B$.
                         Node B
1. Step B1. Validate $Token_A$.
2. Step B2. Generate a random nonce $RN_B$.
3. Step B3. Select a random integer $r_B$ and compute $Q_B = r_B \cdot P$.
4. Step B4. Compute $SK_{BA} = H((r_B + b) \cdot (Q_A + Pub_A) \| ID_A \| ID_B \| RN_A \| RN_B)$,
   $HMAC_B = H(SK_{BA} \| H((Q_A \cdot x + Q_B \cdot x) \| (Q_A \cdot y + Q_B \cdot y) \| ID_A \| ID_B \| RN_A \| RN_B)$.
5. Step B5. Construct message $m = RN_A \| RN_B \| Q_B \| HMAC_B$.
6. Step B6. Generate $Sig_B(m) = (r, s)$.
7. Step B7. Send $Arep(m, Sig_B(m))$ to Node $A$.
                         Node A
1. Step A3. Verify $B$'s signature $Sig_B(m)$.
2. Step A4. Verify received $RN_A$ with previous $RN_A$, if no match then
3. Step A5. Compute $SK_{AB} = H((r_A + a) \cdot (Q_B + Pub_B) \| ID_A \| ID_B \| RN_A \| RN_B)$,
   $HMAC_A = H(SK_{AB} \| H((Q_A \cdot x + Q_B \cdot x) \| (Q_A \cdot y + Q_B \cdot y) \| ID_A \| ID_B \| RN_A \| RN_B)$.
4. Step A6. Compare computed $HMAC_A$ with received $HMAC_B$ for integrity check.
5. Step A7. Send an acknowledgement $Sig_A(RN_B \| HMAC_B)$ to $B$.
```

measure.

5.6.2.1 Proposed Countermeasures for Man in the Middle Attack

The vulnerability in the mentioned protocol lies from the basis that the mechanism of public key distribution is not mentioned in it, e.g. how node A receives the public key (Pub_B) of node B is not mentioned. The attack in Algorithm 9 succeeds if the attacker E manages to make A believe that Pub_E is the public key of B. The countermeasure to

Algorithm 9 Ammayappan et al.'s Scheme under MITM Attack

```
                              Node A
1. Step A1. Select a random number $r_A$ and computes $Q_A = r_A$Âů$P$ as its public key.
2. Step A2. Node $A$ sends $AReq(Token_A, RN_A, Q_A)$ to Node $B$.
                              Adversary E
1. Step E1. Modify the message to $AReq(Token_A, RN_E, Q_E)$ and sends to Node $B$.
                              Node B
1. Step B1. Compute $SK_{BE} = H((r_B + b) \cdot (Q_E + Pub_E) \| ID_E \| ID_B \| RN_E \| RN_B)$,
   $HMAC_B = H(SK_{BE} \| H((Q_E \cdot x + Q_B \cdot x) \| (Q_E \cdot y + Q_B \cdot y) \| ID_E \| ID_B \| RN_E \| RN_B)$.
2. Step B2. Construct message $m_1 = RN_E \| RN_B \| Q_B \| HMAC_B$.
3. Step B3. Generate $Sig_B(m_1) = (r, s)$.
4. Step B4. Send $Arep(m_1, Sig_B(m_1))$ to Node $A$.
                              Adversary E
1. Step E2. Intercept $Arep(m_1, Sig_B(m_1))$ sent by $B$ (to $A$).
2. Step E3. Compute $SK_{AE} = H((r_E + e) \cdot (Q_A + Pub_A) \| ID_A \| ID_E \| RN_A \| RN_E)$,
   $HMAC_E = H(SK_{AE} \| H((Q_A \cdot x + Q_E \cdot x) \| (Q_A \cdot y + Q_E \cdot y) \| ID_A \| ID_E \| RN_A \| RN_E)$
3. Step E4. Construct message $m_2 = RN_A || RN_E || Q_E || HMAC_E$
4. Step E5. Generate $Sig_E(m_2) = (r, s)$.
5. Step E6. Send $Arep(m_2, Sig_E(m_2))$ to Node $A$.
                              Node B
1. Step B5. Compute $SK_{BE} = H((r_E + e).(Q_B + Pub_B) \| ID_E \| ID_B \| RN_E \| RN_B)$,
   $HMAC_E = H(SK_{BE} \| H((Q_E \cdot x + Q_B \cdot x) \| (Q_E \cdot y + Q_B \cdot y) \| ID_E \| ID_B \| RN_E \| RN_B)$.
                              Node A
1. Step A3. Compute $SK_{AE} = H((r_A + a) \cdot (Q_E + Pub_E) \| ID_A \| ID_E \| RN_A \| RN_E)$,
   $HMAC_A = H(SK_{AE} \| H((Q_A \cdot x + Q_E \cdot x) \| (Q_A \cdot y + Q_E \cdot y) \| ID_A \| ID_E \| RN_A \| RN_E)$.
```

this attack is to use an integrity check in the messages sent from the TTP to nodes. The following steps may be used to securely distribute (public) keys by the TTP. It is assumed, the TTP has a database storing public keys of all users.

1. When node A wants to communicate to B, it sends the following message to the TTP: $A \Rightarrow TTP : Sig_A(B \| T_A)$, where T_A is the current timestamp of A's system.

2. TTP upon receipt of the message, validates the timestamp.

3. TTP then sends the public key of B to A using the following message: $TTP \Rightarrow A$: $Sig_{TTP}(Pub_B \| T_{TTP} \| H(S))$, where T_{TTP} is the current timestamp of TTP's system, and S is a secret value shared between A and the TTP.

4. Upon receipt of the message A validates the timestamp, recomputes $H(S)$, using the hash and the secret value, and verifies it with the received $H(S)$.

5.6.2.2 Security Analysis of the Fix

As mentioned before, the intention of the attacker E, would be to somehow send its own public key Pub_E to A, making A believe it has the public key of B. To achieve this, E could do either of the following:

1.
   - Change A's request to the TTP to $Sig_A(E \| T_A)$.
   - Therefore have the TTP sends back $Sig_{TTP}(Pub_E \| T_{TTP} \| H(S))$ to A.

2. Try to change the message $Sig_{TTP}(Pub_B \| T_{TTP} \| H(S))$ to $Sig_{TTP}(Pub_E \| T_{TTP} \| H(S))$.

E would not be able to do the first because, E would be unaware of the private key of A needed to create a authentic (duplicate) signature of A. The timestamp provides protection against a possible replay of an old message by E. For the second, since E does not know the secret value S, it will not be able to modify the message.

Chapter 6

CONCLUSION

6.1 Summary of Results

In order to determine the vulnerabilities of the five protocols considered, the author designed generalized cryptanalysis algorithms to perform man in the middle (MITM), database, and clogging attacks on protocols that use ECC as a security measure. First, the protocols of Moosavi et al.(2014), Xu and Wu (2015), He et al. (2014) and Hui et al.(2012) were shown to be vulnerable to clogging attacks. The vulnerability rests in the use of ECC operations by the server in the validation phase. In this analysis, it was observed that a composition of timestamp, encryption, and a nonce will avoid clogging attack vulnerability in these three protocols. These protocols were also shown to be vulnerable to database attacks such as weak authentication, SQL injection, desynchronization, unauthorized access, unauthorized tampering and database protocol vulnerability. In this analysis, it was observed that using input validation, an updated ID process, a network firewall and encryption would prevent database attack vulnerability in these protocols. The key agreement protocol by Ammayappan et al. (2011) was next analyzed. Ammayappan et al. showed their protocol to be immune to MITM attack and they claimed that the protocol's security is based on the ECC algorithm. However, in this work their protocol was shown to be inherently vulnerable to MITM attack.

Table 6.1 summarizes the vulnerabilities found in the protocols analyzed. A Yes in a cell indicates we found the designated protocol vulnerable to the designated attack through a static analysis, and a No means that no vulnerability was found. It is evident that many of the protocols are vulnerable to clogging and database attacks.

Protocol	***Clogging***	***MITM***	***Weak Authentication***	***SQL Injection***	***Unauth. access***	***DB vulnerability***	***Desynch***	***Unauth. tampering***
Moosavi et al.	Yes	No	No	No	No	No	No	No
Xu et al.	Yes	No	Yes	Yes	Yes	No	No	No
He et al.	Yes	No	Yes	No	No	Yes	Yes	No
Hui et al.	Yes	No	No	No	Yes	No	No	Yes
Ammaya et al.	No	Yes	No	No	No	No	No	No

Table 6.1: Summary of the Vulnerabilities

The countermeasures are also summarized in Table 6.2. It must be emphasized that as everything is associated with costs, the amount of security needed will determine the nature of counter-measure.

6.2 Conclusion

In this thesis, clogging attacks and database attacks have been demonstrated on five recent ECC based authentication schemes. The goal was to bring to light the intricacies and challenges in designing such protocols. ECC schemes guarantee a high level of security . However, they could still contain an easily-exploitable vulnerability if they are

Protocol	***Mode of Attack***	***Countermeasure***
Moosavi et al.	Classical Clogging	Validity Checks
Xu et al.	Classical Clogging, Weak Authentication, SQL Injection, Unauthorized Access	Efficient Algorithm, input validation, Timestamp
He et al.	Classical Clogging, Weak Authentication, Database Protocol Vulnerability, Desynchronization	Input Validation, Updating of ID, Encryption, Timestamp
Hui et al.	Classical Clogging, Unauthorized Access, Unauthorized Tampering	Usage of SSL, Timestamp, Network firewalls
Ammayappan et al.	Man in the Middle	Securing Public Key Management

Table 6.2: Summary of the Results

applied without an additional level of protection. Hence, a layer of protection need to be added to assure complete security towards clogging attacks.

It is concluded that ECC assures a level of security, however it may create a vulnerability if it happens to be applied without an extra layer of protection. Many multi-factor authentication and key exchange protocols, whether smart card or RFID based, depend on ECC to provide security. Hence, an additional level of defense need be added to these protocols to assure increased security towards MITM, database and clogging attacks.

6.3 Directions of Future Research

Research on elliptic curve cryptography authentication protocols is ongoing. This research has been conducted using static analysis. An obvious next step is to test the dynamic vulnerability of these protocols. The documentation presented here will lay the groundwork for performing dynamic analysis on ECC based protocols. This documentation provides detailed steps and procedures to perform static analysis. Our suggested countermeasures on strengthening the security flaws in the above protocols will provide a strong basis to perform dynamic analysis. Other kinds of security flaws could potentially be discovered while performing dynamic analysis.

REFERENCES

Abidi, A., Bouallegue, B., and Kahri, F. (2014). Implementation of elliptic curve digital signature algorithm (ecdsa). In *Computer & Information Technology (GSCIT), 2014 Global Summit on*, pages 1–6. IEEE.

Ammayappan, K., Negi, A., Sastry, V., and Das, A. K. (2011). An ecc-based two-party authenticated key agreement protocol for mobile ad hoc networks. *Journal of Computers*, 6(11):2408–2416.

Bhargav-Spantzel, A., Squicciarini, A. C., Modi, S., Young, M., Bertino, E., and Elliott, S. J. (2007). Privacy preserving multi-factor authentication with biometrics. *Journal of Computer Security*, 15(5):529–560.

Blake, I. F., Seroussi, G., and Smart, N. (1999). *Elliptic curves in cryptography*, volume 265. Cambridge university press.

Bonneau, J., Just, M., and Matthews, G. (2010). What's in a name? In *International Conference on Financial Cryptography and Data Security*, pages 98–113. Springer.

Bos, J., Kaihara, M., Kleinjung, T., Lenstra, A. K., and Montgomery, P. L. (2009). On the security of 1024-bit rsa and 160-bit elliptic curve cryptography. Technical report.

Bos, J. W., Halderman, J. A., Heninger, N., Moore, J., Naehrig, M., and Wustrow, E. (2014). Elliptic curve cryptography in practice. In *International Conference on Financial Cryptography and Data Security*, pages 157–175. Springer.

Burr, J. (2016). elliptical curve cryptography (ECC). `http://searchsecurity.techtarget.com/definition/elliptical-curve-cryptography/`. [Online; accessed 19-July-2016].

Chang, R., Jiang, G., Ivancic, F., Sankaranarayanan, S., and Shmatikov, V. (2009). Inputs of coma: Static detection of denial-of-service vulnerabilities. In *2009 22nd IEEE Computer Security Foundations Symposium*, pages 186–199. IEEE.

Choi, Y., Lee, D., Kim, J., Jung, J., Nam, J., and Won, D. (2014). Security enhanced user authentication protocol for wireless sensor networks using elliptic curves cryptography. *Sensors*, 14(6):10081–10106.

Chuang, Y.-H., Hsu, C.-L., Shu, W., Hsu, K. C., and Liao, M.-W. (2015). A secure non-interactive deniable authentication protocol with certificates based on elliptic curve cryptography. In *New Trends in Intelligent Information and Database Systems*, pages 183–190. Springer.

Farash, M. S. and Ahmadian-Attari, M. (2014). A pairing-free id-based key agreement protocol with different pkgs. *IJ Network Security*, 16(2):143–148.

Garrett, K., Talluri, S. R., and Roy, S. (2015). On vulnerability analysis of several password authentication protocols. *Innovations in Systems and Software Engineering*, 11(3):167–176.

GlobalSign (2015). ECC 101: What is ECC and why would I want to use it? `https://www.globalsign.com/en/blog/elliptic-curve-cryptography/`. [Online; accessed 19-July-2016].

Graham, J., Olson, R., and Howard, R. (2016). *Cyber security essentials*. CRC Press.

Guide, R. (2014). elliptical curve cryptography (ECC). `http://www.sysax.com/ftblog/windows-ftp/elliptic-curve-cryptography-ecc/`. [Online; accessed 19-July-2016].

Gura, N., Patel, A., Wander, A., Eberle, H., and Shantz, S. C. (2004). Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 119–132. Springer.

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

He, D., Kumar, N., Chilamkurti, N., and Lee, J.-H. (2014). Lightweight ecc based rfid authentication integrated with an id verifier transfer protocol. *Journal of Medical Systems*, 38(10):1–6.

Higgins, K. J. (2008). Hacker's Choice: Top Six Database Attacks. `http://www.darkreading.com/risk/hackers-choice-top-six-database-attacks/d/d-id/1129481?/`. [Online; accessed 19-July-2016].

Islam, S. H. and Biswas, G. (2013). Design of improved password authentication and update scheme based on elliptic curve cryptography. *Mathematical and Computer Modelling*, 57(11):2703–2717.

Jin, C., Xu, C., Zhang, X., and Zhao, J. (2015). A secure rfid mutual authentication protocol for healthcare environments using elliptic curve cryptography. *Journal of medical systems*, 39(3):1–8.

Khatwani, C. and Roy, S. (2015). Security analysis of ecc based authentication protocols. In *Computational Intelligence and Communication Networks (CICN), 2015 International Conference on*, pages 1167–1172. IEEE.

Kildall, G. A. (1973). A unified approach to global program optimization. In *Proceedings of the 1st annual ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 194–206. ACM.

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209.

Koc, C. K. (2013). Elliptic Curve Cryptography. `https://koclab.cs.ucsb.edu/`. [Online; accessed 02-February-2017].

Krawczyk, H. (2005). Hmqv: A high-performance secure diffie-hellman protocol. In *Annual International Cryptology Conference*, pages 546–566. Springer.

LaMacchia, B. A. and Manferdelli, J. L. (2006). New vistas in elliptic curve cryptography. *information security technical report*, 11(4):186–192.

Lauter, K. (2004). The advantages of elliptic curve cryptography for wireless security. *IEEE Wireless communications*, 11(1):62–67.

Li, C.-T. (2013). A new password authentication and user anonymity scheme based on elliptic curve cryptography and smart card. *IET Information Security*, 7(1):3–10.

Liao, Y.-P. and Hsiao, C.-M. (2014). A secure ecc-based rfid authentication scheme integrated with id-verifier transfer protocol. *Ad Hoc Networks*, 18:133–146.

Moosavi, S. R., Nigussie, E., Virtanen, S., and Isoaho, J. (2014). An elliptic curve-based mutual authentication scheme for rfid implant systems. *Procedia Computer Science*, 32:198–206.

Orman, H. (1998). The oakley key determination protocol.

Owen, W. N. and Shoemaker, E. (2008). Multi-factor authentication system.

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

Sabzevar, A. P. and Stavrou, A. (2008). Universal multi-factor authentication using graphical passwords. In *Signal Image Technology and Internet Based Systems, 2008. SITIS'08. IEEE International Conference on*, pages 625–632. IEEE.

Shao-hui, W., Su-qin, C., Zhi-wei, W., and Guo-zi, S. (2012). A password authentication and update scheme based on elliptic curve cryptography. *International Journal of Advancements in Computing Technology*, 4(3).

Sherwood, T., Irvine, C., Huffmire, T., Levin, T., Valamehr, J., Kaya Koc, C., and Kastner, R. (2012). A qualitative security analysis of a new class of 3-d integrated crypto coprocessors.

Sullivan, N. (2013). A (relatively easy to understand) primer on elliptic curve cryptography. `https://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/`. [Online; accessed 24-October-2013].

Telelink (2016). Unauthorized Access Attack. `http://itsecurity.telelink.com/unauthorized-access-attack/`. [Online; accessed 19-July-2016].

Vanstone, S. A. (1997). Elliptic curve cryptosystemâĂŤthe answer to strong, fast public-key cryptography for securing constrained environments. *Information Security Technical Report*, 2(2):78–87.

Wang, D., Ma, C.-g., Wang, P., and Chen, Z. (2012). Robust smart card based password authentication scheme against smart card security breach. *Cryptology ePrint Archive*.

Wang, Y. (2012). Password protected smart card and memory stick authentication against off-line dictionary attacks. In *IFIP International Information Security Conference*, pages 489–500. Springer.

Wikipedia (2016a). Elliptic curve Diffie-Hellman. `http://blog.mgm-tp.com/2013/02/securing-your-password-database-using-bcrypt/`. [Online; accessed 19-July-2016].

Wikipedia (2016b). forward secrecy. `https://en.wikipedia.org/wiki/Forward_secrecy/`. [Online; accessed 19-July-2016].

Wikipedia (2016c). Trapdoor Function. `https://en.wikipedia.org/wiki/Trapdoor_function/`. [Online; accessed 09-October-2016].

Xu, L. and Wu, F. (2015). An improved and provable remote user authentication scheme based on elliptic curve cryptosystem with user anonymity. *Security and Communication Networks*, 8(2):245–260.

Yang, G., Wong, D. S., Wang, H., and Deng, X. (2006). Formal analysis and systematic construction of two-factor authentication scheme (short paper). In *International Conference on Information and Communications Security*, pages 82–91. Springer.

Yeh, H.-L., Chen, T.-H., and Shih, W.-K. (2014). Robust smart card secured authentication scheme on sip using elliptic curve cryptography. *Computer Standards & Interfaces*, 36(2):397–402.

Zhang, L., Tang, S., Chen, J., and Zhu, S. (2015). Two-factor remote authentication protocol with user anonymity based on elliptic curve cryptography. *Wireless Personal Communications*, 81(1):53–75.

LIST OF PUBLICATIONS FROM THE THESIS

1. C. Khatwani and S. Roy, "Security Analysis of ECC Based Authentication Protocols," 2015 International Conference on Computational Intelligence and Communication Networks (CICN), Jabalpur, India, 2015, pp. 1167-1172.

VITA

Chanchal Khatwani currently works for JP Morgan Chase as an IT Business Analyst. She earned her engineering degree B.E.(hons) at Birla Institute of Technology and Science, Pilani in Computer Science. At the University of North Florida, she is working towards her Master's degree in Software Engineering. She worked as a research assistant with the computing department and also as an IT Analyst with UNF physical facilities. She has participated in the Jax Startup weekend competition to build a search app. She is a member of Upsilon Pi Epsilon (UPE), the International Honor Society for the Computing Sciences. Previously, she has worked as a software engineer with Allscripts and Paramount. She has a background in Computer Science.