2016

Vulnerabililty Analysis of Multi-Factor Authentication Protocols

Keith Garrett

VULNERABILITY ANALYSIS OF MULTI-FACTOR AUTHENTICATION PROTOCOLS

by

Keith A. Garrett

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

June, 2016

The thesis “Vulnerability Analysis of Multi-Factor Authentication Protocols” submitted by Keith Garrett 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. Asai Asaithambi

_____________________________ ____________

Dr. Sandeep Reddivari

_____________________________ ____________

Dr. Anand Seetharam

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

I wish to thank all the giants on whose shoulders I am privileged to stand; my teachers, instructors, and professors; and my family and friends who prayed for me, picked me up, dusted me off, and threw me back into the battle. A special thanks to Dr. Roy as all of the above and as a mentor and guide. Thanks to my thesis committee members, Dr. Swapnoneel Roy, Dr. Asai Asaithambi, Dr. Anand Seetharam, and Dr. Sandeep Reddivari.

CONTENTS

List of Figures viii

List of Tables ix

Abstract x

Chapter 1 Introduction 1

1.1 Multi-factor authentication protocols 1

1.2 Contributions of this thesis 3

1.3 Organization of this thesis 3

Chapter 2 Review of the Literature 4

2.1 Research on authentication protocols 4

2.1.1 A research snapshot of security protocols 4

2.1.2 Various kinds of attacks on the protocols 5

2.1.3 Approaches to develop secured protocols 7

2.2 Methodology adopted in this thesis 8

Chapter 3 A General Algorithm for Clogging Attack 9

3.1 The algorithm 9

3.2 The clogging conditions 10

3.2.1 Analysis 12

3.3 Choice of protocols 12

Chapter 4 Clogging Attack on the Protocols 13

4.1 Yang's protocol 13

4.1.1 Review of the protocol 13
4.1.2 Attack on Yang's protocol 14
4.1.3 Proposed countermeasures to the attack 16
4.1.3.1 Security analysis of the solution. 17
4.1.3.2 Another solution 17
4.2 Islam's protocol 17
4.2.1 Review of Islam's protocol 18
4.2.2 Attack on Islam's protocol 19
4.2.3 Proposed countermeasures to the attack 20
4.3 Jiang's password based protocol 20
4.3.1 Review of the protocol 20
4.3.2 Attack on Jiang et al.'s protocol 21
4.3.3 Clogging attack performed on other similar schemes 23
4.3.4 Proposed countermeasures to the attack 24
4.3.4.1 Steps to avoid the clogging attack 24
4.3.4.2 Security analysis of the solution 25
4.4 Wang's smart card based protocol 25
4.4.1 Review of the protocol 25
4.4.2 Replay Attack on Wang's protocol 27
4.4.3 Proposed countermeasures to the attack 28
4.4.3.1 The steps to avoid replay attack resulting in clogging attack 28
4.4.3.2 Yet another way to prevent a clogging attack 29

Chapter 5 Conclusion 30
5.1 Summary of results 30
5.2 Conclusion 32
5.3 Directions for future research 32

References 33

Appendix A Modular Exponentiation 36

A.1 Background on modular exponentiation 36

A.1.1 Exponentiation 37

A.1.2 Factoring 37

A.1.3 Modular Exponentiation 37

A.1.4 Discrete Logarithm Problem and Security 37

A.2 Applications in computer security 38

Publication 42

Vita 43

FIGURES

Figure 2.1 A snapshot of authentication protocols (Wang et al., 2013). 5

Figure 3.1 Energy consumption by modular exponentiation with exponents of various sizes (Harish and Roy, 2014) 10

Figure 3.2 Comparison of normalized energy consumption of different operations (on log scale) (Harish and Roy, 2014) 11

TABLES

Table 3.1 Summary of the Differences in the Protocols 12

Table 5.1 Summary of the Results 31

ABSTRACT

In this thesis, the author hypothesizes that the use of computationally intensive mathematical operations in password authentication protocols can lead to security vulnerabilities in those protocols. In order to test this hypothesis:

1. A generalized algorithm for cryptanalysis was formulated to perform a clogging attack (a form of denial of service) on protocols that use computationally intensive modular exponentiation to guarantee security.
2. This technique was then applied to cryptanalyze four recent password authentication protocols, to determine their susceptibility to the clogging attack.

The protocols analyzed in this thesis differ in their usage of factors (smart cards, memory drives, *etc.*) or their method of communication (encryption, nonces, timestamps, *etc.*). Their similarity lies in their use of computationally intensive modular exponentiation as a medium of authentication.

It is concluded that the strengths of all the protocols studied in this thesis can be combined to make each of the protocols secure from the clogging attack. The conclusion is supported by designing countermeasures for each protocol against the clogging attack.

Chapter 1

INTRODUCTION

1.1 Multi-factor authentication protocols

User authentication can enable a perimeter device (*e.g.*, a firewall, proxy server, VPN server, or remote access server) to decide whether or not to approve a specific access request to gain entry to a computer network. It is necessary to be able to identify and authenticate any user with a high level of certainty, so that the user may be held accountable should his/her actions threaten the security and productivity of the network. The more confidence a network administrator has regarding the user's identity, the more confidence the administrator will have in allowing that user specific privileges, and the more faith the administrator will have in the internal records regarding that user.

Multi-factor authentication is an approach to cyber-security in which the user is required to provide more than one form of verification in order to prove his/her identity and gain access to the system. It takes advantage of a combination of several authentication factors. Commonly used factors include 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 the use of biometrics) (Stallings and Brown, 2008). Due to their increased complexity, multi-factor authentication systems are harder to breach than those using any single factor.

Smart cards are widely used in multi-factor authentication due to their relatively low cost, robust security, versatility and variety. Smart card based password authentication is one of the most convenient and effective two-factor (smart cards and passwords) authentication mechanisms for remote systems (Chen et al., 2014; Islam, 2014; Jiang et al., 2015; Li and Lee, 2012; Li et al., 2013; Yang et al., 2014). This technique has been widely applied to various authentication applications, including remote host login, online banking, e-commerce and e-health. Also, it constitutes the basis of three-factor authentication. Challenges remain in both security and performance aspects of smart card authentication due to the stringent security requirements and resource constraints of the clients.

Many multi-factor authentication protocols (Beller et al., 1993; Huang et al., 2003; Kocher and Jaffe, 2001; Liao and Wang, 2009) use computationally intensive modular exponentiation, a one-way function (see Appendix A), to guarantee security. The security provided by modular exponentiation is due to its one-way nature, in other words, the hardness of computing its inverse called the discrete logarithm problem. But due to the computationally intensive nature of modular exponentiation, its use leaves a security loophole in protocols. An attacker can force the server or the client to waste resources by repeatedly performing unnecessary computations (due to modular exponentiation), resulting in clogging (a form of denial of service).

In this thesis, four recent multi-factor authentication protocols are analyzed for vulnerability. The first two, which are smart card based, differ in the fact that one (Yang et al., 2014) uses encryption and a nonce (a random number) for verification, while the other uses timestamps (Islam, 2014). The third multi-factor authentication protocol considered here is a memory device aided password authentication protocol (Jiang et al., 2013). In this kind of protocol, the authentication information (issued by a server) is stored in a memory device such as a universal serial bus (USB) stick, portable hard disk drive, mobile phone, PDA, PC, or, most recently, a software protection dongle (Jiang et al., 2013; Chen et al., 2012; Rhee et al., 2009). The last protocol considered is also smart card based and uses modular exponentiation for its security (Wang et al., 2013).

1.2 Contributions of this thesis

In order to demonstrate the vulnerabilities of the four protocols considered, the author designed a generalized cryptanalysis algorithm to perform clogging attacks on protocols that use modular exponentiation for authentication purposes being forced to perform useless modular exponentiation operations results in wastage of time and resources on the part of the server, and thus denial of service for legitimate users.

1.3 Organization of this thesis

The rest of this thesis is organized as follows. Chapter 2 gives a brief overview of the research done in this area prior to this work and discusses the techniques employed in this work. Chapter 3 describes the generic algorithm that was developed to launch the clogging attack on the various protocols analyzed in this work. Chapter 4 describes the protocols in detail, the clogging attack on them, and solutions for preventing the attack. Finally, Chapter 5 summarizes the work, discusses its significance, and proposes possible future research directions.

Chapter 2

REVIEW OF THE LITERATURE

2.1 Research on authentication protocols

This chapter summarizes the research done in the field of multi-factor authentication protocols. The various kinds of attacks that are generally performed against these protocols are discussed, followed by two approaches to dealing with these attacks. Finally the approach taken in this work is briefly described.

2.1.1 A research snapshot of security protocols

The design and security analysis of authentication protocols has been an active area of research in recent years. Generally, the security vulnerability analysis of one or more protocols leads to the design of new protocols. However, in most of these studies, the authors present attacks on previous schemes and propose new protocols with assertions of the superior aspects of their schemes, while ignoring benefits that their schemes don't attempt (or fail) to provide, thus overlooking dimensions on which they perform poorly.

New Protocol → Broken → Improved Protocol → Broken Again
→ Further Improved Protocol → ⋯

In addition to the lack of evaluation criteria, another common feature of these studies is that there is no proper security justification (or even an explicit security model) presented, which explains why these protocols previously claimed to be secure turn out to be vulnerable. Such an approach has generated a lot of literature, yet as far as we know, little attention has been paid to systematic design and analysis. Fig. 2.1 summarizes the relationships of the research studies done in this area to date.

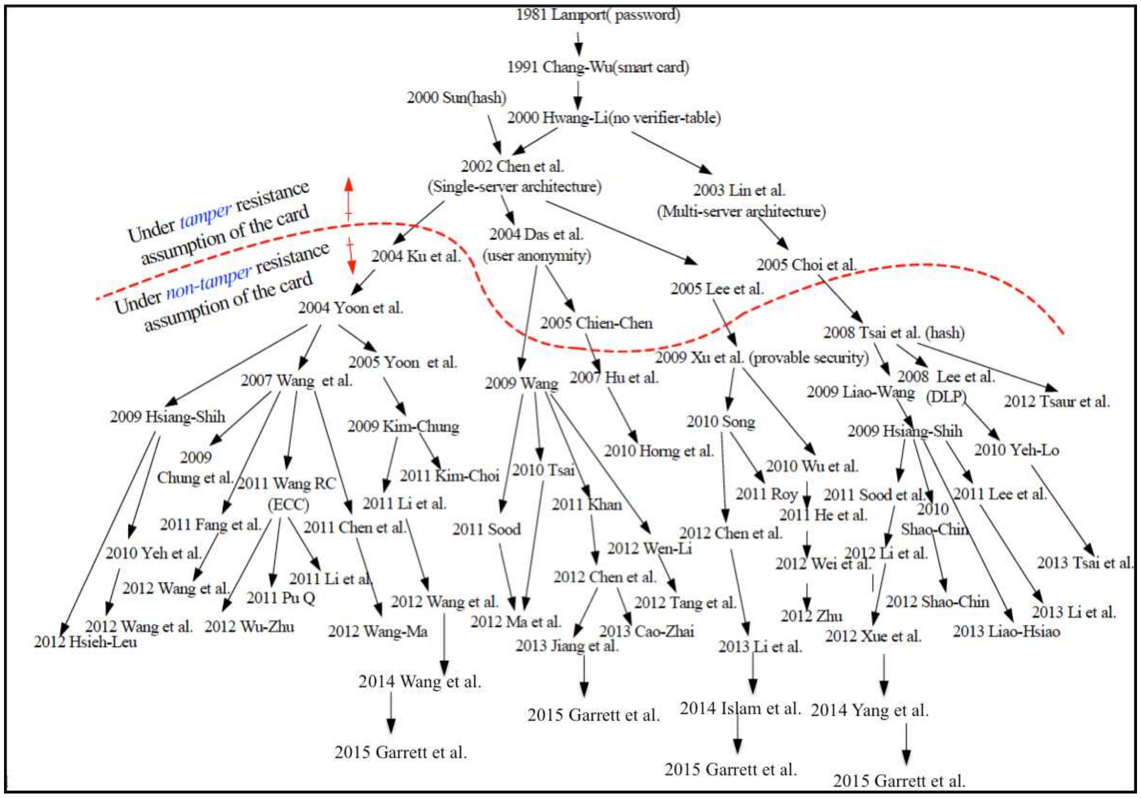

Figure 2.1: A snapshot of authentication protocols (Wang et al., 2013).

2.1.2 Various kinds of attacks on the protocols

Most multi-factor authentication protocols have the following phases:

- *Registration Phase.* In this phase, the user registers with the server with an identity and password. The password is stored by the server and remains a secret between the user and the server.

- *Login and Authentication Phase.* In this phase, the user wants a service from the server. It sends its identity and password to the server to gain a service. The server then determines whether the user is legitimate by comparing the received values of identity and password to the stored values. The server extends the desired service to the legitimate user (Garrett et al., 2015).

Multi-factor authentication protocols are prone to various kinds of attacks. Some common kinds of attacks include:

- *Brute Force Attack.* In this kind of attack, the adversary simply tries out all possible combinations of alphabets to obtain the correct password. Alternatively, the adversary could be a bit more intelligent to try to guess the password by obtaining some information about the user. Usually, users choose passwords which are easy for them to remember. Hence, passwords contain things like the year of birth, or a part of the user name or user ID. The adversary could gather such information to simplify his efforts to break the password (Knudsen and Robshaw, 2011).
- *Impersonation Attack.* In this kind of attack, the adversary pretends to be a legitimate user and gains access to the services meant for the legitimate user. The adversary might spoof the IP address of the legitimate user, or could intercept the messages sent by the user and use them to gain access to the server (Wei-Chi and Chang, 2005).
- *Replay Attack.* In this kind of attack, the adversary intercepts a message with the identity and the password of the legitimate user. The adversary then waits till the legitimate user logs out of the service. Then he replays the message with the identity and the password to the server and gains access. Replay attacks are possible even when the user identity and password are encrypted (Syverson, 1994).

- *Stolen smart card attack.* In a password authentication scheme employing smart cards, the smart card contains the identity and password of the legitimate user. The stealing of the smart card by an adversary is equivalent to the stealing of a credit card. The adversary could enjoy service until all the servers to which the legitimate user has access are notified about the stolen smart card (Chen et al., 2014).

- *Stolen server database attack.* In this kind of attack, the adversary steals the database maintained by the server that contains the user identities and passwords of the legitimate clients. The adversary now has enormous power to do anything at his will (Wei-Chi et al., 2004).

- *Insider attack.* A legitimate user of one server could try to hack the credentials of another user. The victim might have access to some other service which the adversary does not. The adversary thus gains access to that service (Wood, 2000).

- *Denial of service attack.* The adversary tries to prevent legitimate users from obtaining the designated service. Various techniques are adopted by the adversary to achieve this goal. This attack prevents the availability of a service rather than compromising any information (confidentiality), or modifying unauthorized information (integrity) (Long and Thomas, 2001).

2.1.3 Approaches to develop secured protocols

Security has become an integral part of software design. Nowadays security is considered during the design and implementation of software rather than as an add on. There are two approaches to analyzing software for security:

- *Static analysis.* In this approach, the algorithm and the code after implementation are analyzed to find security vulnerabilities (Chess and McGraw, 2004; Garrett et al., 2015; Harish and Roy, 2014; Roy et al., 2011; Talluri and Roy, 2014). The advantage of static analysis is that it can find potential security violations without executing the application or, in the case of an algorithm analysis, even before the implementation is done.
- *Dynamic analysis.* In this approach, the software is executed and tested for potential vulnerabilities (Rahimi et al., 1993; Russo and Sabelfeld, 2010; Sobajic and Pao, 1989; Yasinsac, 2001). One well known technique is *penetration testing* in which the tester tries to be a hacker to break into a system (Petukhov and Kozlov, 2008).

2.2 Methodology adopted in this thesis

In this thesis, static vulnerability analysis is employed on four multi-factor authentication protocols. Specifically, static vulnerability analysis is performed to conclude if these protocols are vulnerable to clogging attack, a form of denial of service. The protocols analyzed and modified to produce new secured protocols have been listed in Chapter 1.

Chapter 3

A GENERAL ALGORITHM FOR CLOGGING ATTACK

3.1 The algorithm

Most of the illustrations in this section is a repeat of an already published work by the author in (Garrett et al., 2015). The mechanism of most password authentication protocols is that the client (usually a memory stick or a smart card reader) sends its credentials to the server which then performs certain mathematical operations to verify those credentials. The protocols usually work in different phases. All the phases will be discussed when each protocol is considered individually. For now, the focus is on the login and authentication phases, in which the client is authenticated and the server authenticates itself to the client.

In general, the attacker intercepts the message from the client to server, which contains the login credentials. This message is plaintext in some protocols and encrypted in others. It might or might not contain a timestamp. These differences will be noted in the protocols considered later. In either case, the attacker replays the intercepted message several times to force the server to perform computationally intensive operations (in this case modular exponentiation), thus forcing the server to waste its time and resources. Legitimate users are thus denied services. Algorithm 1 shows the line of attack developed by the author (Garrett et al., 2015).

Algorithm 1 The general algorithm for a clogging attack (Garrett et al., 2015).

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

3.2 The clogging conditions

The clogging conditions we analyze in this work are based on the computational and resource intensiveness of the operation of *modular exponentiation*. The energy consumption of the modular exponentiation computation $y \equiv g^x \pmod{n}$ has been measured in (Harish and Roy, 2014). In practice, to guarantee security, the modulus n is a very large prime, and the exponent x, which is often a randomly chosen secret value, should be very large as well. The researchers computed the energy consumed by modular exponentiation on two different platforms.

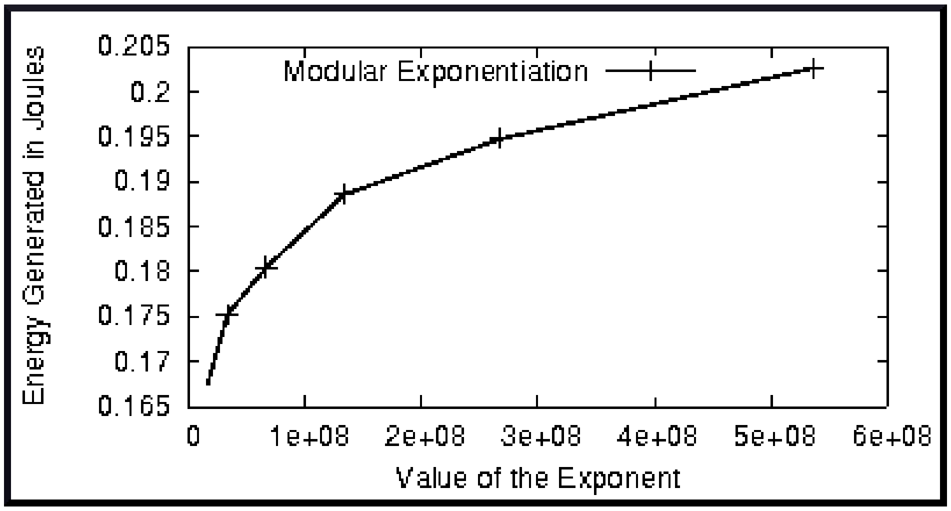

Figure 3.1: Energy consumption by modular exponentiation with exponents of various sizes (Harish and Roy, 2014)

Fig. 3.1, plots the energy consumed in Joules by the modular exponentiation operation with a (randomly) fixed base $g = 1024$, modulus $n = 32416187567$, and the exponent x ranging from values $x = 16777216$ to $x = 536870912$ on two different platforms. In practice, the modulus and the exponent have very large values to guarantee security. The energy generated due to modular exponentiation was observed to steadily increase for large values of x.

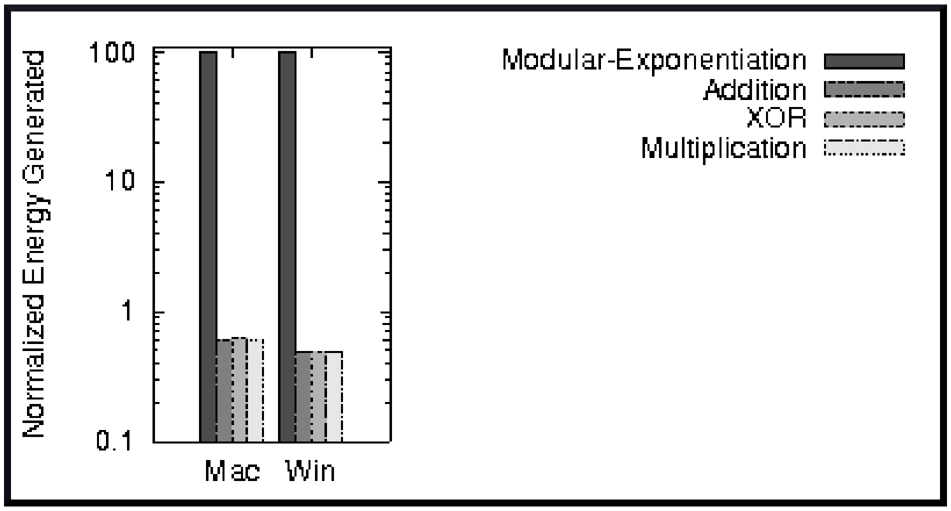

Figure 3.2: Comparison of normalized energy consumption of different operations (on log scale) (Harish and Roy, 2014)

To see how much energy is consumed by modular exponentiation in comparison to common mathematical operations like addition, XOR, and multiplication, Harish and Roy (2014) further plotted the normalized energy required by all four operations. They measured energy consumed for the operations $g^x \mod n$, $g + x + n$, $g \oplus x \oplus n$, and $g \times x \times n$. Fig. 3.2 shows this comparison. It is evident from Fig. 3.2 that the energy consumed by modular exponentiation is two orders of magnitude more than that of the other operations.

3.2.1 Analysis

In summary, any security protocol that uses modular exponentiation to guarantee security comes with high energy consumption and computational costs. Therefore, a protocol that uses modular exponentiation is vulnerable to the clogging attack, a form of denial of service in which the attacker exploits the computational and energy intensive nature of modular exponentiation (Garrett et al., 2015).

3.3 Choice of protocols

The protocols chosen for analysis in this thesis fall in the broad category of multifactor authentication protocols. All of them use a password as one of the factors of authentication. Selection of these protocols is based on their differences in the choice of the second factor (smart cards, memory drives, *etc.*), and the tools to provide confidentiality (encryption, nonces, timestamps, *etc.*).

Protocol	*Factors Used*	*Confidentiality*
Yang	Password, Smart Card	Usage of Nonces, Encryption
Islam	Password, Smart Card	Usage of Timestamps
Jiang	Password, Memory Stick	Usage of Timestamps
Wang	Password, Smart Card	Usage of Random Numbers

Table 3.1: Summary of the Differences in the Protocols

Table 3.1 shows how the protocols the author has chosen to analyze are different from each other. Their similarity lies in their use of modular exponentiation as a medium for authentication.

Chapter 4

CLOGGING ATTACK ON THE PROTOCOLS

In this chapter are described each of the protocols, the attacks on them, and the solutions to foil the attacks. The following notations are used to describe the protocols.

- $\mathbf{Z}_q^*$ 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.

4.1 Yang's protocol

The first protocol considered in this work was developed by Yang et al. (2014). It is a smart card based password authentication protocol to preserve identity privacy. Yang's protocol is an improvement over Song's protocol (Song, 2010) in that Song's protocol proved to be vulnerable against impersonation and insider attacks (Yang et al., 2014).

4.1.1 Review of the protocol

Yang's protocol is presented in Algorithm 2. It works in four phases: Initialization, Registration, Login, and Authentication. Yang et al. (2014) showed this protocol to be immune from various attacks.

4.1.2 Attack on Yang's protocol

An adversary $\mathscr{A}$ has the same capabilities as assumed by Yang et al. (Yang et al., 2014) while they exposed the weaknesses of Song's protocol (Song, 2010). $\mathscr{A}$ needs only to be able to read and modify the contents of messages over an insecure channel during the Login and Authentication phase of the protocol in order to launch a clogging attack.

1. $\mathscr{A}$ intercepts a valid login request ($\{W_i, C_i\}$) from **Step L3** (Algorithm 2).
2. Since the message is unencrypted, $\mathscr{A}$ changes C_i to any random garbage value $C_{\mathscr{A}}$.
3. $\mathscr{A}$ then sends $\{W_i, C_{\mathscr{A}}\}$ to the server S.

The following is performed by the server S:

1. S decrypts W_i to retrieve ID_i, $(R_i \otimes K_i)$ and, N_i. Verifies that ID_i is valid from its stored value. Here it is valid. S then computes $K_i = h(ID_i)^x \mod (p \otimes N_i)$ and $R_i' = (R_i \otimes K_i) \otimes K_i$, and compares $C_{\mathscr{A}}$ with $h(N_i \| R_i' \| ID_i \| W_i \| K_i)$. This fails, so the request gets rejected.

Adversary $\mathscr{A}$ would now repeat the steps several times and make the server S compute the modular exponentiation step several times. $\mathscr{A}$ can potentially change all the incoming login request messages from any legitimate user to S. Since modular exponentiation is computationally intensive, the victimized server spends considerable computing resources doing useless modular exponentiation rather than any real work. Thus $\mathscr{A}$ clogs

Algorithm 2 Yang et. al.'s (2014) protocol for password authentication

Initialization Phase

Server S

1. **Step I1.** Choose large prime numbers p and q such that $p = 2q + 1$.
2. **Step I2.** Choose a private key $x \in \mathbf{Z}_q^*$

Registration Phase

User U_i

1. **Step R1.** Choose identity ID_i, password PW_i, and random number b.
2. **Step R2.** Compute $h(b \otimes PW_i)$.
3. **Step R3.** $U_i \rightarrow S$: $\{ID_i, h(b \otimes PW_i)\}$ through a *secure channel.*

Server S

1. **Step R4.** On receiving the registration message from U_i, S creates an entry for U_i in the account-database and stores ID_i in this entry. Next, S computes $B_i = h(ID_i)^x \otimes h(b \otimes PW_i) \mod p$.
2. **Step R5.** S stores $\{ID_i, B_i, h(\cdot), p, q, F_S(\cdot)\}$ in a smart card and issues it to the user.

User U_i

1. **Step R6.** Upon receiving the smart card, the user enters random number b in it.

Login and Authentication

User U_i

1. **Step L1.** Smart card generates a random number R_i, and nonce N_i.
2. **Step L2.** Smart card computes $K_i = B_i \otimes h(b \otimes PW_i) \otimes N_i$, $W_i = F_S(ID_i, R_i \otimes K_i, N_i)$, and $C_i = h(N_i \| R_i \| ID_i \| W_i \| K_i)$.
3. **Step L3.** $U_i \rightarrow S$: $\{W_i, C_i\}$.

Server S

1. **Step V1.** S decrypts W_i to retrieve ID_i, $(R_i \otimes K_i)$ and, N_i. Verifies whether ID_i is valid from its stored value. If not, the request is dropped, and the session is terminated. Otherwise, S computes $K_i = h(ID_i)^x \mod (p \otimes N_i)$ and $R_i' = (R_i \otimes K_i) \otimes K_i$, and compares C_i with $h(N_i \| R_i' \| ID_i \| W_i \| K_i)$. If they are not equal the session is terminated. Otherwise S authenticates U_i and the login request is accepted. S generates a nonce N_j, and computes session key $sk = h(ID_i \| N_i \| N_j \| R_i')$, and message authentication code $C_S = h(sk \| ID_i \| R_i' \| N_j)$.
2. **Step V2.** $S \rightarrow U_i$: $\{C_S, N_j\}$.

Yang et. al.'s (2014) scheme (cont'd.)

User U_i

1. **Step V3.** On receiving the reply message from the server S, smart card computes session key $sk = h(ID_i \| N_i \| N_j \| R_i)$, and message authentication code $C'_S = h(ID_i \| R_i \| N_j)$, and verifies whether C_S and C'_S are equal. This equivalency authenticates the legitimacy of the server S, and mutual authentication between S and U_i is achieved. Otherwise S is not authenticated.

S with useless work and denies service to legitimate users. $\mathscr{A}$ needs only the ID of a single valid user to perform the clogging attack repeatedly.

4.1.3 Proposed countermeasures to the attack

The vulnerability of this protocol is due to the fact that no timestamp checking is used in order to work around clock synchronization problems between the user and server. Nonces are used, but not to prevent a replay by the adversary (which clogs the server).

At the beginning of the authentication phase, the server could check whether the network address of the user is valid. To do this, it has to know the network addresses of all the registered legitimate users. In spite of that, adversary $\mathscr{A}$ could spoof the network address of a legitimate user and replay the login message. To prevent this, a cookie exchange step may be added at the beginning of the login phase of Yang's scheme. This step was designed in the well-known Oakley key exchange protocol (Orman, 1998), as follows:

1. User U_i chooses a pseudo-random number n_1 and sends it along with the message $\{W_i, C_i\}$.

2. Server S, upon receiving the message, acknowledges the message and sends its own cookie n_2 to U_i.

3. The next message from U_i must contain n_2, else S rejects the message and the login request.

4.1.3.1 Security analysis of the solution.

Had $\mathscr{A}$ spoofed the U_i's IP address, $\mathscr{A}$ would not get n_2 back from S. $\mathscr{A}$ only succeeds in having S send back an acknowledgement, but S doesn't perform the computationally intensive modular exponentiation. Therefore the clogging attack is avoided by these additional steps. However, this process does not completely prevent the clogging attack but only thwarts it to some extent. This fix can fully work if n_1 and n_2 are encrypted, respectively, by the U_i's and S's private keys for secure communication.

4.1.3.2 Another solution

Another fix would be of course to add a timestamp T_i to W_i. Since W_i is encrypted, the adversary cannot change the timestamp to do the replay. The server can check the validity of the timestamp before it does the modular exponentiation step.

4.2 Islam's protocol

The second protocol is that of Islam (2014). It is again a smart card based remote user password authentication protocol. Islam's protocol is an improvement over Li's protocol (Li et al., 2013), which Islam (2014) showed to be vulnerable against attacks such as stolen smart card, offline password guessing, and insider attack. Islam's protocol has been shown to be immune to various attacks (Islam, 2014).

4.2.1 Review of Islam's protocol

Islam's protocol works in five phases: Registration, Login, Authentication, Password Change, and Stolen Smart card Revocation (Islam, 2014). The protocol, presented in Algorithm 3, omits the last two phases since they are not important in the clogging attack demonstration.

Algorithm 3 Islam's scheme of password authentication

Registration Phase

User U_i

1. **Step R1.** Choose identity ID_i.
2. **Step R2.** $U_i \rightarrow S$: ID_i through a *secure channel.*

Server S

1. **Step R3.** On receiving the registration message from U_i, S chooses a new smart card and extracts SID_i as its identity. Next, S computes $C_i = h(ID_i \| x \| SID_i)$.
2. **Step R4.** S stores $\{C_i, p, q, h(\cdot)\}$ in a smart card and issues it to the user. S stores (ID_i, SID_i) for U_i in its database.

User U_i

1. **Step R5.** Upon receiving the smart card, the user enters password PW_i in it. The smart card computes $B_i = C_i \otimes h(PW_i) = h(ID_i \| x \| SID_i) \otimes h(PW_i)$ and $A_i = C_i^{PW_i} \mod p = h(ID_i \| x \| SID_i) \mod p$. Now, the smartcard replaces C_i by B_i and stores A_i . Finally, the smart card contains the information $\{A_i, B_i, h(\cdot), p, q\}$.

Login and Authentication

User U_i

1. **Step L1.** Ui attaches the smart card to an input device and keys his/her (ID_i, PW_i) into the smart card.
2. **Step L2.** The smart card computes $C_i = B_i \otimes h(PW_i)$ and $A_i^* = C^{PW_i} \mod p$. The smart card then verifies $A_i^* = A_i$. If they are not equal, the smart card rejects U_i's login request, otherwise goes to the next step.
3. **Step L3.** The smart card picks the current timestamp T_i , $\alpha \in \mathbf{Z}_p$ and computes $D_i = C_i^{\alpha} \mod p = h(ID_i \| x \| kSID_i)^{\alpha} \mod p$, $Mi = h(ID_i \| C_i \| D_i \| T_i)$.
4. **Step L4.** $U_i \rightarrow S$: $\{ID_i, D_i, M_i, T_i\}$.

Islam's protocol (contd.)

`Server` S

1. **Step V1.** S checks if ID_i is valid from its stored value, and $T_S - T_i \leq \Delta T_i$. If not, the request is dropped, and the session is terminated. Otherwise, S computes $C'_i = h(ID_i \| x \| SID_i)$ and $M'_i = h(ID_i \| C'_i \| D_i \| T_i)$, and compares M_i with M'_i. If they are not equal the session is terminated. Otherwise S authenticates U_i and the login request is accepted.
2. **Step V2.** S chooses $\beta \in \mathbf{Z}_p^*$, computes $V_i = (C'_i)^{\beta} \mod p$ and the session key $sk = D_i^{\beta} \mod p$.
3. **Step V3.** S selects a current timestamp T_s , computes $M_s = h(ID_i \| C'_i \| V_i \| sk \| T_s)$.
4. **Step V4.** $S \rightarrow U_i$: $\{ID_i, V_i, M_s, T_s\}$.

`User` U_i

1. **Step V5.** On receiving the reply message from the server S, smart card checks the timestamp. If valid, computes session key $sk' = V_i^{\alpha} \mod p$, and $M'_s = h(ID_i \| C'_i \| V_i \| sk' \| T_s)$, and verifies whether M_s and M'_s are equal. This equivalency authenticates the legitimacy of the server S, and mutual authentication between S and U_i is achieved. Otherwise S is not authenticated.

4.2.2 Attack on Islam's protocol

Again, $\mathscr{A}$ needs only to be able to read and modify the contents of messages over an insecure channel (during the Login and Authentication phase of the protocol).

1. $\mathscr{A}$ intercepts a valid login request ($\{ID_i, D_i, M_i, T_i\}$) from step **Step L4**.
2. Since the message is unencrypted, $\mathscr{A}$ changes T_i to $T_{\mathscr{A}}$ to pass the timestamp check.
3. $\mathscr{A}$ then sends $\{ID_i, D_i, M_i, T_{\mathscr{A}}\}$ to the server S.

The server S performs steps **V1** through **V4** since the timestamp check of **V1** passes. The adversary $\mathscr{A}$ would now repeat the steps several times and make the server S compute the computationally intensive modular exponentiation of **Step V2** several times. Thus, as in the previous case, the server gets clogged doing unnecessary computations.

4.2.3 Proposed countermeasures to the attack

The vulnerability of this protocol is due to the fact the message $\{ID_i, D_i, M_i, T_i\}$ is not encrypted. We could use the strength of Yang's protocol (Yang et al., 2014) here and design a function F_S to encrypt T_i. This prevents $\mathscr{A}$ from changing T_i, and hence also prevents the replay, which leads to clogging attack.

4.3 Jiang's password based protocol

The next protocol considered is that of Jiang (Jiang et al., 2013). It is a remote authentication protocol which does not involve smart cards. However, Jiang et al. also had another version of the protocol which works with smart cards (Jiang et al., 2015). Both protocols are vulnerable to clogging attacks. Jiang's protocol (Jiang et al., 2013) is an improvement over Chen's protocol (Chen et al., 2012) which they proved to be vulnerable against the off-line dictionary attack.

4.3.1 Review of the protocol

Jiang's protocol works in five phases: Initialization, Registration, Login, Authentication, and Password Change. Jiang's group showed that their remote authentication protocol was immune to various attacks (Jiang et al., 2013). The protocol is presented in Algorithm 4. The password change phase is omitted since it is not required to demonstrate the effect of the clogging attack on the protocol.

4.3.2 Attack on Jiang et al.'s protocol

Adversary $\mathscr{A}$ has the same capabilities and knowledge as assumed by Jiang (Jiang et al., 2013) while exposing the weaknesses of Chen's protocol. $\mathscr{A}$ needs only to be able to read and modify the contents of messages over an insecure channel (during Login and Authentication phase of the protocol) in order for the protocol to be susceptible to attack.

1. $\mathscr{A}$ intercepts a valid login request ($\{ID_i, C_i, V_i, T_1\}$) from step **Step L3**.
2. Since the message is unencrypted, $\mathscr{A}$ can change the timestamp T_1 to some $T_{\mathscr{A}}$ so that it meets the criterion $(T_2 - T_{\mathscr{A}}) < \Delta T$.
3. $\mathscr{A}$ changes C_i to any random garbage value $C_{\mathscr{A}}$.
4. $\mathscr{A}$ then sends $\{ID_i, C_{\mathscr{A}}, V_i, T_{\mathscr{A}}\}$ to the server S.

The following is performed by the server S:

1. Check whether ID_i is valid. Here it is valid.
2. Check whether the difference between $(T_2 - T_{\mathscr{A}}) < \Delta T$. This step passes as well.
3. Compute $Y_i'' = \mathscr{H}(ID_i \| x)$ and $D_i' = C_{\mathscr{A}}^x \mod p$, and compare V_i with $\mathscr{H}(ID_i \| Y_i'' \| C_{\mathscr{A}} \| D_i' \| T_{\mathscr{A}})$. This fails, so the request gets rejected.

Adversary $\mathscr{A}$ would now repeat the steps several times and make the server S compute the modular exponentiation step several times. $\mathscr{A}$ can potentially change all the incoming login request messages from any legitimate user to S. Since modular exponentiation

Algorithm 4 Jiang's scheme of password authentication

Initialization Phase

Server S

1. **Step I1.** Choose large prime numbers p and q such that $p = 2q + 1$.
2. **Step I2.** Choose a generator g of $\mathbf{Z}_q^*$, secret key $x \in \mathbf{Z}_q^*$, and secure one way hash $\mathcal{H}$.
3. **Step I3.** Compute public key $X = g^x \mod p$.

Registration Phase

User U_i

1. **Step R1.** Choose identity ID_i, password PW_i.
2. **Step R2.** $U_i \rightarrow S$: $\{ID_i, PW_i\}$ through a *secure channel.*

Server S

1. **Step R3.** On receiving the registration message from U_i, S creates an entry for U_i in the account-database and stores ID_i in this entry. Next, S computes $Y_i = \mathcal{H}(ID_i \| x)) \otimes \mathcal{H}(PW_i)$.
2. **Step R4.** $S \rightarrow U_i$: $\{X, Y_i, \mathcal{H}, p, q\}$.

User U_i

1. **Step R5.** Upon receiving $\{X, Y_i, \mathcal{H}, p, q\}$ from S, U_i enters it locally in his/her memory device (e.g. USB stick).

Login and Authentication

User U_i

1. **Step L1.** U_i chooses a random number $\alpha \in \mathbf{Z}_q^*$.
2. **Step L2.** U_i computes $Y_i' = Y_i \otimes \mathcal{H}(PW_i)$, $C_i = g^\alpha \mod p$, $D_i = x^\alpha \mod p$, and $V_i = \mathcal{H}(ID_i \| Y_i' \| C_i \| D_i \| T_1)$, where T_1 is the current system time of U_i.
3. **Step L3.** $U_i \rightarrow S$: $\{ID_i, C_i, V_i, T_1\}$.

Server S

1. **Step V1.** S checks whether ID_i is valid from its stored value, and $(T_2 - T_1) < \Delta T$, where T_2 is the current system time for S. If either does not hold, the request is dropped, and the session is terminated. Otherwise, S computes $Y_i'' = \mathcal{H}(ID_i \| x)$ and $D_i' = C_i^x \mod p = g^{x\alpha} \mod p = X^\alpha \mod p = D_i$, and compares V_i with $\mathcal{H}(ID_i \| Y_i'' \| C_i \| D_i' \| T_1)$. If they are not equal the session is terminated. Otherwise S authenticates U_i and the login request is accepted. S computes $M_i = \mathcal{H}(ID_i \| D_i' \| T_3)$, where T_3 is the current system time of S.
2. **Step V2.** $S \rightarrow U_i$: $\{M_i, t_3\}$.

Jiang's scheme (contd.)

User U_i

1. **Step V3.** On receiving the reply message from the server S, U_i checks whether T_3 is valid, and M_i is equal to $\mathscr{H}(ID_i \| D_i \| T_3)$. This equivalency authenticates the legitimacy of the server S, and mutual authentication between S and U_i is achieved. Otherwise S is not authenticated.

Compute Session Key

User U_i

$$sk_U = \mathscr{H}(D_i)$$

Server S

$$sk_S = \mathscr{H}(D'_i)$$

is computationally intensive, the victimized server spends considerable computing resources doing useless modular exponentiation rather than any real work. Thus $\mathscr{A}$ clogs S with useless work and denies service to legitimate users. $\mathscr{A}$ needs just an ID of a single valid user to perform the clogging attack repeatedly.

4.3.3 Clogging attack performed on other similar schemes

Jiang's group devised a smart card based password authentication protocol (Jiang et al., 2015). This work was an improvement over another such scheme by Chen et al. (2014b). This researcher observes that the clogging attack performed on the current protocol under consideration is also effective against both Jiang's (Jiang et al., 2015) and Chen's (Chen et al., 2014) smart card protocols. Both protocols are vulnerable because the user's smart card does not encrypt the message it sends over to the server for login and authentication. This gives an adversary the chance to manipulate the message.

4.3.4 Proposed countermeasures to the attack

The proposed countermeasures to the attack on Jiang's protocol are on similar lines to those already seen for the two previous protocols. They are illustrated along with their security analysis in the next sections.

4.3.4.1 Steps to avoid the clogging attack

At the beginning of the authentication phase, the server could check whether the network address of the user is valid. It has to know the network addresses of all the registered legitimate users. In spite of that, adversary $\mathscr{A}$ could spoof the network address of a legitimate user and replay the login message. A countermeasure might be the addition of a cookie exchange step at the beginning of the login phase of Jiang's scheme. This step has been designed as in the well known Oakley key exchange protocol (Orman, 1998).

1. User U_i chooses a pseudo-random number n_1 and sends it along with the message $\{ID_i, C_i, V_i, T_1\}$.
2. Server S upon receiving the message, acknowledges the message and sends its own cookie n_2 to U_i.
3. The next message from U_i must contain n_2, else S rejects the message and the login request.

4.3.4.2 Security analysis of the solution

Had $\mathscr{A}$ spoofed the U_i's IP address, $\mathscr{A}$ would not get n_2 back from S. Hence $\mathscr{A}$ succeeds only in having S send back an acknowledgement, but not in launching the computationally intensive modular exponentiation. Hence the clogging attack is avoided. However, this process does not prevent the clogging attack but only thwarts it to some extent. This fix can fully work if n_1, and n_2 are encrypted respectively by the U_i's and S's private keys for secure communication.

4.4 Wang's smart card based protocol

Wang's group (Wang et al., 2013) has reported a smart card based authentication protocol. They claim their protocol to be immune to denial of service attacks. To illustrate this, they assume a situation of a stolen smart card.

4.4.1 Review of the protocol

Wang's protocol, like most smart card based protocols, has the Registration, Login, and Verification phases. The protocol is presented in Algorithm 5. To perform a clogging attack on their protocol, the attacker would not have to steal the smart card. The clogging attack on the protocol presented here is done via replay. A replay is not necessary, but a replay step by the attacker makes the clogging attack more effective. Most of the smart card based protocols cited by Wang et al. (2013) are vulnerable to this attack.

Algorithm 5 Wang's protocol for password authentication

Registration Phase

User U_i

1. **Step R1.** Choose identity ID_i, password PW_i and a random number b.
2. **Step R2.** $U_i \rightarrow S$: $ID_i, \mathscr{H}_0(b \| PW_i)$.
3. **Step R3.** Upon receiving the smart card SC, U_i enters b into SC.

Server S

1. **Step R4.** On receiving the registration message from U_i at time T, S first checks whether U_i is a registered user. If it is U_i's initial registration, S creates an entry for U_i in the account-database and stores $(ID_i, T_{reg} = T)$ in this entry. Otherwise, S updates the value of T_{reg} with T in the existing entry for U_i. Next, S computes $N_i = \mathscr{H}_0(b \| PW_i)) \otimes \mathscr{H}_0(x \| ID_i \| T_{reg})$ and $A_i = \mathscr{H}_0((\mathscr{H}_0(ID_i) \otimes \mathscr{H}_0(b \| PW_i)) \mod n)$.
2. **Step R5.** $S \rightarrow U_i$: A smart card containing security parameters $\{N_i, A_i, q, g, y, n, \mathscr{H}_0(\cdot), \mathscr{H}_1(\cdot), \mathscr{H}_2(\cdot), \mathscr{H}_3(\cdot)\}$.

User U_i

1. **Step R6.** Upon receiving the smart card SC, U_i enters b into SC.

Login and Authentication

User U_i

1. **Step L1.** U_i inserts her smart card into the card reader and inputs ID_i^* , PW_i^*.
2. **Step L2.** SC computes $A_i^* = \mathscr{H}_0((\mathscr{H}_0(ID_i^*) \otimes \mathscr{H}_0(b \| PW_i^*)) \mod n)$ and verifies the validity of ID_i^* and PW_i^* by checking whether A_i^* equals the stored A_i. If the verification holds, it implies $ID_i^* = ID_i$ and $PW_i^* = PW_i$ with a probability of $\frac{n-1}{n} (\approx \frac{99.90}{100}$, when $n = 2^{10})$. Otherwise, the session is terminated.
3. **Step L3.** SC chooses a random number u and computes $C_1 = g^u \mod p$, $Y_1 = y^u \mod p$, $k = \mathscr{H}_0(x \| ID_i \| T_{reg}) = N_i \otimes \mathscr{H}_0(b \| PW_i)$, $CIDi = ID_i \otimes \mathscr{H}_0(C_1 \| Y_1)$ and $M_i = \mathscr{H}_0(Y_1 \| k \| CID_i)$.
4. **Step L4.** $U_i \rightarrow S$: $\{C_1, CID_i, M_i\}$.

Server S

1. **Step V1.** S computes $Y_1 = (C_1)^x \mod p$ using its private key x. Then, S derives $ID_i = CID_i \otimes \mathscr{H}_0(C_1 \| Y_1)$ and checks whether ID_i is in the correct format. If ID_i is not valid, the session is terminated. Then, S computes $k = \mathscr{H}_0(x \| ID_i \| T_{reg})$ and $M_i^* = \mathscr{H}_0(Y_1 \| k \| CID_i)$, where T_{reg} is extracted from the entry corresponding to ID_i. If M_i^* is not equal to the received M_i, the session is terminated. Otherwise, S generates a random number v and computes the temporary key $KS = (C_1)^v \mod p$, $C_2 = g^v \mod p$ and $C_3 = \mathscr{H}_1(ID_i \| IDS \| Y_1 \| C_2 \| k \| KS)$.
2. **Step V2.** $S \rightarrow U_i$: $\{C_2, C_3\}$.

Wang's scheme (cont'd.)

User U_i

1. **Step V3.** On receiving the reply message from the server S, SC computes $KU = (C_2)^u \mod p$, $C_3^* = \mathscr{H}_1(ID_i \| IDS \| Y_1 \| C_2 \| k \| KU)$, and compares C_3^* with the received C_3. This equivalency authenticates the legitimacy of the server S, and U_i goes on to compute $C4 = \mathscr{H}_2(ID_i \| IDS \| Y_1 \| C_2 \| k \| KU)$.
2. **Step V4.** $U_i \rightarrow S$: $\{C_4\}$

Server S

1. **Step V5.** Upon receiving $\{C_4\}$ from U_i, the server S first computes $C_4^* = \mathscr{H}_2(ID_i \| IDS \| Y_1 \| C_2 \| k \| KS)$ and then checks if C_4^* equals the received value of C_4. If this verification holds, S authenticates the user U_i and the login request is accepted else the connection is terminated.

Compute Session Key

User U_i

- $sk_U = \mathscr{H}_3(ID_i \| IDS \| Y_1 \| C_2 \| k \| KU)$

Server S

- $sk_S = \mathscr{H}_3(ID_i \| IDS \| Y_1 \| C_2 \| k \| KS)$

4.4.2 Replay Attack on Wang's protocol

A replay attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. Replays can be used to gain unauthorized access, or may be done simply to perform a denial of service. This is carried out either by the originator or by an adversary who intercepts the data and retransmits it, possibly as part of a masquerade attack by IP packet substitution (such as stream cipher attack).

Wang's protocol (Wang et al., 2013) was claimed to be secured against replay attacks, but as we see, this author has been able to perform a replay attack on Wang's protocol to achieve a denial of service. We assume the attacker $\mathscr{A}$ has complete knowledge of the protocol (*i.e.* not security under obscurity).

1. $\mathscr{A}$ intercepts a valid login request ($\{C_1, CID_i, M_i\}$) from **Step L4**.

2. $\mathscr{A}$ replays $\{C_1, CID_i, M_i\}$ several times. That is, it performs $\mathscr{A} \rightarrow S$: $\{C_1, CID_i, M_i\}$ a large number of times.

3. This will force S_i perform three modular exponentiations $Y_1 = (C_1)^x \mod p$, $KS = (C_1)^v \mod p$, and $C_2 = g^v \mod p$ of **Step V1**.

4. $\mathscr{A}$ can intercept whatever replies $Step_i$ sends (**Step V1**) and discard them (they would anyway be lost since SC will not expect these replies).

The attacker $\mathscr{A}$ can simply send fake login requests to the server S and could have launched the clogging attack by forcing S to perform $Y_1 = (C_1)^x \mod p$ on **Step V1**. But this replay attack results in a bigger DoS attack on S since it is forced to perform three modular exponentiations (in place of just one). $\mathscr{A}$ will need to send fewer messages to S to clog it. This replay attack is possible because, unlike Jiang's protocol, Wang's protocol does not have a timestamp check on incoming messages from the user(s) by S.

4.4.3 Proposed countermeasures to the attack

In the subsequent section a couple of countermeasures to the attack are illustrated. The author observes the usage of timestamps and their verifications can make this protocol secure against replays leading to denial of service.

4.4.3.1 The steps to avoid replay attack resulting in clogging attack

Replay attacks on most smart card based protocols might be possible because their security relies on computationally intensive modular exponentiation, and the messages are not by default encrypted. This vulnerability is often overlooked, since the natural result of a replay is not always denial of service. A few steps to avoid these attacks on Wang's protocol (and smart card based protocols in general) would be

1. U_i uses a time stamp T in **Step L4.**, and S verifies it in **Step V1.**. The time stamp also must be encrypted in some form so that $\mathscr{A}$ cannot tamper with it.

2. S checks whether multiple login requests frequently come from the same user. This step reduces but does not eliminate the chances of a replay, because $\mathscr{A}$ can obtain a lot of valid user IDs (they are public) and send fake login requests periodically from different IDs. Alternatively, $\mathscr{A}$ can store various (valid) login requests over a time period, and replay them periodically.

4.4.3.2 Yet another way to prevent a clogging attack

The mathematical basis which makes the protocols vulnerable to clogging attacks is modular exponentiation. The complete prevention of this vulnerability requires encryption of all the messages between U_i and S. But doing so would involve a key exchanging step, in which each user has a private key and a public key. The server knows the public key, and can decrypt a message encrypted by a user's private key. Thus, the server makes sure that the message is from a valid user before it launches the costly modular exponentiation. This comes with a cost and depends on the level of security desired (Al-Riyami and Paterson, 2003; Fujisaki and Okamoto, 1999; Wander et al., 2005). This countermeasure works for all protocols (smart card and non-smart card based).

Chapter 5

CONCLUSION

5.1 Summary of results

In this thesis, first, the protocols of Yang et al. (2014) and Islam (2014) are shown to be vulnerable to the clogging attack. The vulnerability lies in the use of computationally intensive modular exponentiation by the server in the authentication process. In this analysis it is observed that using a combination of encryption, a nonce, and a timestamp would prevent clogging attack vulnerability in these two protocols.

The protocol by Jiang (2013a) is next analyzed. Jiang's protocol is an improvement over the protocol of Chen et al. (2012), which they show to be insecure against offline password guessing attacks. Other protocols, for example that of Rhee et al. (2009), preceded Chen's protocol also have been shown to be vulnerable to attack. This author finds Jiang's protocol to be insecure against clogging attacks. The vulnerability again lies in the use of computationally intensive modular exponentiation by the server in the authentication process. Another recent protocol by the same researchers (Q. Jiang, 2013b), which is smart card based, is also observed to be insecure against clogging attacks. A solution is then presented by the author to prevent an attacker from carrying out such an attack on the protocol.

The final protocol analyzed is a smart card based protocol by Wang et al. (2013), which has been claimed to be secure against denial of service. However, the results here show that an attacker can exploit the fact that the protocol uses multiple modular exponentiations for authentication. A replay attack can be launched on the protocol to achieve a bigger clogging attack. A clogging attack can also be done on this protocol in the classical way (without replays). A strategy for making the protocol secure against this attack is proposed by the author; this fix also makes the protocol of Jiang (2013a) immune against clogging attacks.

It is observed that modular exponentiation guarantees a level of security, but it might create a vulnerability if it is used without an additional level of protection. Most of the multi-factor authentication protocols in the literature, whether smart card based or memory device aided, rely on modular exponentiation for their security. Hence, some level of protection should be added to them to guarantee increased security against clogging attacks.

Table 5.1 summarizes the results obtained in this work. It is evident that most of the protocols are vulnerable to the classical clogging attack.

Protocol	***Mode of Attack***	***Countermeasure***
Yang	Classical clogging	Timestamps
Islam	Classical clogging	Encryption
Jiang	Classical clogging	Cookie Exchanging
Wang	Replay attack[1]	Timestamps

Table 5.1: Summary of the Results

The last column of Table 5.1 suggests the strengths of the protocols could be combined to prevent the clogging attack on them. However, it must be emphasized that as ev-

[1]Also classical clogging.

erything has costs involved, the level of security needed will determine the nature of countermeasure.

5.2 Conclusion

In this thesis, clogging attacks on four advanced multi-factor authentication protocols have been demonstrated. The goal of this work is to uncover the subtleties and challenges in designing this type of protocol. While modular exponentiation guarantees a level of security, this work shows that modular exponentiation might lead to an easily-exploitable vulnerability if it is used without an additional level of protection. Most of the multi-factor authentication protocols in the literature, whether smart card based or memory device aided, rely on modular exponentiation for their security. Therefore, some level of protection should be added to them to guarantee total security against clogging attacks.

5.3 Directions for future research

Research on multi-factor authentication protocols is ongoing. An interesting direction would be to investigate whether protocols that use Elliptic Curve Cryptography (ECC) to guarantee security are less vulnerable to clogging attacks than those that rely on modular exponentiation.

REFERENCES

Al-Riyami, S. S. and Paterson, K. G. (2003). Certificateless public key cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 452–473. Springer.

Beller, M. J., Chang, L.-F., and Yacobi, Y. (1993). Privacy and authentication on a portable communications system. *IEEE Journal on Selected Areas in Communications*, 11(6):821–829.

Chen, B. L., Kuo, W. C., and Wuu, L. C. (2012). A secure password-based remote user authentication scheme without smart cards. *Information Technology and Control*, 41(1):53–59.

Chen, B.-L., Kuo, W.-C., and Wuu, L.-C. (2014). Robust smart-card-based remote user password authentication scheme. *International Journal of Communication Systems*, 27(2):377–389.

Chess, B. and McGraw, G. (2004). Static analysis for security. *IEEE Security & Privacy*, (6):76–79.

Fujisaki, E. and Okamoto, T. (1999). How to enhance the security of public-key encryption at minimum cost. In *International Workshop on Public Key Cryptography*, pages 53–68. Springer.

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.

Harish, P. D. and Roy, S. (2014). Energy oriented vulnerability analysis on authentication protocols for cps. In *Distributed Computing in Sensor Systems (DCOSS), 2014 IEEE International Conference on*, pages 367–371. IEEE.

Huang, Q., Cukier, J., Kobayashi, H., Liu, B., and Zhang, J. (2003). Fast authenticated key establishment protocols for self-organizing sensor networks. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 141–150. ACM.

Islam, S. (2014). Design and analysis of an improved smartcard-based remote user password authentication scheme. *International Journal of Communication Systems*.

Jiang, Q., Ma, J., Li, G., and Li, X. (2015). Improvement of robust smart-card-based password authentication scheme. *International Journal of Communication Systems*, 28(2):383–393.

Jiang, Q., Ma, J., Li, G., and Ma, Z. (2013). An improved password-based remote user authentication protocol without smart cards. *Information Technology and Control*, 42(2):113–123.

Knudsen, L. R. and Robshaw, M. J. (2011). Brute force attacks. In *The Block Cipher Companion*, pages 95–108. Springer.

Kocher, P. C. and Jaffe, J. M. (2001). Secure modular exponentiation with leak minimization for smartcards and other cryptosystems. US Patent 6,298,442.

Li, C.-T. and Lee, C.-C. (2012). A novel user authentication and privacy preserving scheme with smart cards for wireless communications. *Mathematical and Computer Modelling*, 55(1):35–44.

Li, X., Niu, J., Khan, M. K., and Liao, J. (2013). An enhanced smart card based remote user password authentication scheme. *Journal of Network and Computer Applications*, 36(5):1365 – 1371.

Liao, Y.-P. and Wang, S.-S. (2009). A secure dynamic id based remote user authentication scheme for multi-server environment. *Computer Standards & Interfaces*, 31(1):24–29.

Long, N. and Thomas, R. (2001). Trends in denial of service attack technology. *CERT Coordination Center, Summary.*

Messerschmitt, D. G. (1999). Rsa asymmetric encryption. [Online; accessed 20-July-2015].

Orman, H. (1998). *The OAKLEY Key Determination Protocol.* RFC Editor, United States.

Petukhov, A. and Kozlov, D. (2008). Detecting security vulnerabilities in web applications using dynamic analysis with penetration testing. *Computing Systems Lab, Department of Computer Science, Moscow State University.*

Rahimi, F. A., Lauby, M. O., Wrubel, J. N., and Lee, K. L. (1993). Evaluation of the transient energy function method for on-line dynamic security analysis. *Power Systems, IEEE Transactions on*, 8(2):497–507.

Rhee, H. S., Kwon, J. O., and Lee, D. H. (2009). A remote user authentication scheme without using smart cards. *Computer Standards & Interfaces*, 31(1):6 – 13.

Roy, S., Das, A. K., and Li, Y. (2011). Cryptanalysis and security enhancement of an advanced authentication scheme using smart cards, and a key agreement scheme for two-party communication. In *Performance Computing and Communications Conference (IPCCC), 2011 IEEE 30th International*, pages 1–7. IEEE.

Russo, A. and Sabelfeld, A. (2010). Dynamic vs. static flow-sensitive security analysis. In *Computer Security Foundations Symposium (CSF), 2010 23rd IEEE*, pages 186–199. IEEE.

Sobajic, D. J. and Pao, Y.-H. (1989). Artificial neural-net based dynamic security assessment for electric power systems. *Power Engineering Review, IEEE*, 9(2):55–55.

Song, R. (2010). Advanced smart card based password authentication protocol. *Comput. Stand. Interfaces*, 32(5-6):321–325.

Stallings, W. and Brown, L. (2008). Computer security. *Principles and Practice*.

Syverson, P. (1994). A taxonomy of replay attacks [cryptographic protocols]. In *Computer Security Foundations Workshop VII, 1994. CSFW 7. Proceedings*, pages 187–191. IEEE.

Talluri, S. R. and Roy, S. (2014). Cryptanalysis and security enhancement of two advanced authentication protocols. In *Advanced Computing, Networking and Informatics-Volume 2*, pages 307–316. Springer.

Wander, A. S., Gura, N., Eberle, H., Gupta, V., and Shantz, S. C. (2005). Energy analysis of public-key cryptography for wireless sensor networks. In *Third IEEE international conference on pervasive computing and communications*, pages 324–328. IEEE.

Wang, D., Ma, C., Zhang, Q.-M., and Zhao, S. (2013). Secure password-based remote user authentication scheme against smart card security breach. *JNW*, 8(1):148–155.

Wei-Chi, K. and Chang, S.-T. (2005). Impersonation attack on a dynamic id-based remote user authentication scheme using smart cards. *IEICE Transactions on Communications*, 88(5):2165–2167.

Wei-Chi, K., Hao-Chuan, T., and Tsaur, M.-J. (2004). Stolen-verifier attack on an efficient smartcard-based one-time password authentication scheme. *IEICE transactions on communications*, 87(8):2374–2376.

Wood, B. (2000). An insider threat model for adversary simulation. *SRI International, Research on Mitigating the Insider Threat to Information Systems*, 2:1–3.

Yang, F.-Y., Hsu, C.-W., and Chiu, S.-H. (2014). Password authentication scheme preserving identity privacy. In *Measuring Technology and Mechatronics Automation (ICMTMA), 2014 Sixth International Conference on*, pages 443–447.

Yasinsac, A. (2001). Dynamic analysis of security protocols. In *Proceedings of the 2000 workshop on New security paradigms*, pages 77–87. ACM.

Appendix A

MODULAR EXPONENTIATION

A.1 Background on modular exponentiation

Modular arithmetic is also called clock arithmetic because like a clock, the hour that comes after 12 is 1 instead of 13. We count $1,2,3,\cdots 10,11,12,1,2,3,\cdots 10,11,12,1,2,3,\cdots$.
Example. $7+8 \mod 12 \equiv 3 \mod 12$. In this example we would say that "7 plus 8 is *congruent to* $3 \mod 12$", because 15 divided by 12 leaves a remainder of 3. The remainder is always a whole number less than the divisor or modulus n, $\{0,1,2,\cdots,n-2,n-1\}$. In the example, we could have used any counting number and multiplied it by 12 and added 3 to get a value that is congruent to $15 \mod 12$. Thus, 27, 39, and 51 are all congruent to $15 \mod 12$ because when they are divided by 12, they yield the same remainder.

A card game example. An example of modular arithmetic can be shown with a game of cards. When a standard 52 card deck is dealt among 5 players, 1 at a time, the dealer might count $1,2,3,4,5\cdots 1,2,3,4,5\cdots$ until all the cards are used except the last 2. That might be written as $2 \equiv 52 \mod 5$, that is 52 divided by 5 gives each player 10 cards with 2 left over. If there were 4 players instead of 5 then $0 \equiv 52 \mod 4$. Each player would receive 13 cards and there would be none left over.

A.1.1 Exponentiation

Exponentiation is raising a base number to a power. In this thesis we restrict those numbers to the set of counting numbers. For example $7^3 = 7 \times 7 \times 7$.

A.1.2 Factoring

Factoring is the process of taking a counting number and writing it as the product of its prime factors. Example: $30,030 = 2 \times 3 \times 5 \times 7 \times 11 \times 13$. 30029 is prime. Determining that a number is prime by factoring takes more time and energy or computer cycles than generating that prime.

A.1.3 Modular Exponentiation

Modular Exponentiation is used to generate a congruence with the input of a base, exponent and modulus. An example: With a base of 7, exponent of 3 and a divisor of 13, the congruence would equal 5 (since $5 \equiv 7^3 \mod 13$).

A.1.4 Discrete Logarithm Problem and Security

The discrete logarithm is why modular exponentiation is considered useful in cryptography. Discrete logarithm is the process of taking a known congruence, base and modulus and finding the exponent from modular exponentiation. For example, the problem $5 \equiv 7^x \mod 13$ is impractical to solve for x with current computer technology for large exponents that are generated using large primes. As the sizes of the prime numbers used to generate the exponent get larger so does the difficulty in factoring the product.

A.2 Applications in computer security

Why is modular exponentiation important to computer security? If a stored or transmitted password or other sensitive information is read by an unauthorized user, that information may be used to the detriment of a legitimate user or to compromise a system or network. A solution for that problem might be to encrypt the stored data or transmission in such a way that it is impractical for an unauthorized user to access it, but is easily accessible by an authorized user. A simple way to encrypt such data is to take a letter, or a numerical representation of it, and use that number as input in mathematical function to provide a result that is not equal to the input, but that can be converted back to that input value. This process can be extended to any numerical representation, such as a file that might contain a picture or text.

A symmetric key encryption scheme uses a single shared number to encrypt messages between users. A very simple example of this might be the substitution cipher that substitutes a 'B' for an 'A', a 'C' for a 'B' and so on. So a message such as "This is a test." with substitution only of the letters might come out "Uijt jt b uftu.". The reverse substitution would restore the message. In other words, add 1 to the letter to encrypt and subtract 1 to decrypt. This example of a fast and easy symmetric encryption scheme can be changed so that instead of adding or subtracting 1, you can do so with any counting number, but this does not add any security. Some of the problems with symmetric key encryption are that anyone who knows how it works and has the key can decipher any message, make his own encoded messages and imitate another user, or change the contents of a message without detection. Other protocols or schemes used to hide and reveal the data are without these faults.

An asymmetric encryption scheme uses a public key which can be used by anyone to hide data which can only be revealed by a complementary private key. The public key is also used to reveal the data which was hidden by using the private key. Among the mathematical functions available for asymmetric key generation, modular exponentiation is the most commonly used. One popular implementation using this one-way function is the RSA algorithm. Many descriptions of the RSA algorithm are available, but one simple example (Messerschmitt, 1999) will suffice here

The RSA algorithm is paraphrased here: Multiply two large prime numbers, a and b, to generate a modulus, M. Apply Euler's totient function to get a count of relatively prime counting numbers less than M. This is computed by taking the product of $a-1$ and $b-1$ and is called $\phi(M)$. Find a relatively prime counting number less than $\phi(M)$, and call it P. The public key is written as (M,P). The secret key may be determined with the function: $S \equiv P^{(\phi(\phi(M))-1)} \mod \phi(M)$ and is written (M,S).

An example of this algorithm follows: Take two prime numbers and multiply them together. $a = 19$, $b = 23$.
This product will be used for the modulus in a modular exponential. $M = a \times b = 437$.
The RSA algorithm uses Euler's totient function to get the number of relatively prime counting numbers less than M. $\phi(M) = (a-1)(b-1) =$. $(19-1)(23-1) = 18 \times 22 = (20-2)(20+2) = 400-4 = 396$. Which is the count of relatively prime natural numbers less than 437. Therefore $\phi(437) = 396$.

Choose a number that is relatively prime to 396, that is it has no factors in common. $396 = 2 \times 2 \times 3 \times 3 \times 11$. A relatively prime number might be $5 \times 13 = 65$, since it has no factors in common with 396 nor 437. We will use this number in our public key. Set the value of $P = 65$. The public key is then $(M, P) = (437, 65)$.

To find the private key use $S \equiv P^{(\phi(\phi(M))-1)} \mod \phi(M)$. See Euler's product formula to find $\phi(n)$. If the factors of a number can be written as $(F_1^{p_1})(F_2^{p_2})\cdots(F_n^{p_n})$ $\phi(n) = n(1/f_1)(1/f_2)\cdots(1/f_n)$ In this case the totient of 396 is $\phi(396) = (396)(1/2)(1/3)(1/11) = 2 \times 3 = 6$ and using that value to plug into the exponent part of the private key formula $\phi(\phi(437)) = \phi(396) = 6$ and $6 - 1 = 5$.

A private key then is $329 \equiv 65^5 \mod 396$. That is the private, secret, key is $(M, S) = (437, 329)$. Note that $1 \equiv 65 \times 329 \mod 396$, and in general $1 \equiv P \times S \mod \phi(M)$.

To demonstrate how these public/private keys work, choose a number to encrypt such as 42. Using the keys generated above $(M, P) = (437, 65)$ and $(M, S) = (437, 329)$. Since $238 \equiv 42^{65} \mod 437$, the encryption of 42 gives a result of 238. The decryption of 238 gives a result of 42, since $42 \equiv 238^{329} \mod 437$.

A user may encrypt a message with his private key and anyone can use the public key to decrypt it. Since that user is the only one that can use the private key, it follows that he is the one that sent the message, which is a guarantee of identity. If the message were tampered with, the decryption would fail, and so what is received is guaranteed to be what was sent. When the public key is given, attempting to determine the initial large prime numbers or private key is called the discrete logarithm problem, and it is impractical to solve with current computer resources given sufficiently large initial primes.

Asymmetric encryption using this private/public key method requires more time to encrypt and decrypt a message than a simpler symmetric key. However, a symmetric key allows anyone who knows it to decrypt or encrypt a message. A common scheme is to use an asymmetric key pair to send to the recipient a symmetric (session) key that is encrypted with his public key. The recipient then decrypts the symmetric (session) key and uses it in a much faster message transfer. No one else has that session key, so the messages are kept private between the sender and recipient. If another session is needed, another session key can be shared. That method gives both the sender and the receiver some of the security of asymmetric encryption and the speed of symmetric encryption.

LIST OF PUBLICATIONS FROM THE THESIS

Garrett, Keith, S. Raghu Talluri, and Swapnoneel Roy. "*On vulnerability analysis of several password authentication protocols.*" Innovations in Systems and Software Engineering 11.3 (2015): 167-176.

VITA

Keith A. Garrett works for the Florida Department of Health Bureau of Public Health Labs as a distributed computer systems specialist. He earned his A.A. from the Florida Junior College at Jacksonville in 1985, where he worked as a math tutor at the Kent campus lab and participated in the American Mathematical Association of Two-Year Colleges math team. At the University of North Florida he earned his undergraduate Computer Science degree, B.S.C.S., in 1990 with a minor in mathematics. He participated in the student chapter of the ACM at UNF and on the ACM programming team. He worked at the UNF computer lab as an assistant operator and a weekend operator. He is continuing his education at UNF where he intends to complete his master's degree in computer science.