Cryptanalysis and Security Enhancement of Two Advanced Authentication Protocols[⋆]

S. Raghu Talluri and Swapnoneel Roy

School of Computing
University of North Florida, USA
`https://www.unf.edu/ccec/computing/`

Abstract. In this work we consider two protocols for performing cryptanalysis and security enhancement. The first one by Jiang et al., is a password-based authentication scheme[1] which does not use smart cards. We note that this scheme is an improvement over Chen et al.'s scheme shown vulnerable to the off-line dictionary attack by Jiang et al. We perform a cryptanalysis on Jiang at al.'s improved protocol and observe that it is prone to the clogging attack, a kind of denial of service (DoS) attack. We then suggest an improvement on the protocol to prevent the clogging attack.
The other protocol we consider for analysis is by Wang et al. This is a smart card based authentication protocol. We again perform the clogging (DoS) attack on this protocol via replay. We observe that all smart card based authentication protocols which precede the one by Wang et al., and require the server to compute the computationally intensive modular exponentiation are prone to the clogging attack. We suggest (another) improvement on the protocol to prevent the clogging attack, which also applies to the protocol by Jiang et. al.

Keywords: Authentication Protocols, Smart Cards, DoS, Replay Attacks, Clogging Attack.

1 Introduction

In a cyber environment, user authentication can enable a perimeter device (a firewall, proxy server, VPN server, remote access server, etc.) to decide whether or not to approve a specific user's request to gain entry to the network.

It is necessary to be able to identify and authenticate users with a high level of certainty, so that they may be held accountable should their actions threaten the security and productivity of the network. The more confidence network administrators have that a user is who they say they are, the more confidence they will have in allowing those users specific privileges; and the more faith they will have in their network devices' internal records regarding that user. Reliable user authentication can help achieve what are necessary elements

[⋆] Accepted at ICACNI 2014.

[1] We use the terms *scheme* and *protocol* interchangeably in this work.

in basic network security positively identifying someone; allowing them specific rights; and holding them accountable for their actions should they compromise the security and productivity of the network for other users on the network.

Multi-factor authentication is as an approach to cyber-security authentication, in which the user of a system is required to provide more than one form of verification in order to prove their identity and allowed access to the system. It takes advantage of a combination of several factors of authentication; three major 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). Due to their increased complexity, authentication systems using a multi-factor configuration are harder to break than ones using a single factors.

The first such multi-factor authentication protocol we consider here is by Jiang et al. [1]. It is a *memory device aided* password authentication protocol. In this kind of protocols (e.g. [1], [4], [6]), the authentication information (issued by a server) is stored in a memory device such as universal serial bus (USB) sticks, portable HDDs, mobile phones, PDAs, PCs etc. A very common example is a software protection dongle that is used frequently now-a-days for various purposes.

The other protocol we consider by Wang et al. [3] is a smart card based authentication protocol. Smart card based password authentication (e.g. [2], [3], [7], [8], [9], [10], [11]) is one of the most convenient and effective two-factor authentication mechanisms for remote systems. This technique has been widely deployed for various kinds of authentication applications, such as remote host login, online banking, shopping on the internet, e-commerce and e-health. Also, it constitutes the basis of three-factor authentication. However, there still exists challenges in both security and performance aspects due to the stringent security requirements and resource strained characteristics of the clients.

1.1 Our Results

We first analyze the protocol by Jiang et al. Their protocol is an improvement over Chen et al.'s [4] protocol which they show to be insecure against the offline password guessing attack. There are protocols in the literature which came before Chen et al.'s protocol e.g. [6], that have been shown to be vulnerable against some form of attacks. We find Jiang et al.'s protocol to be insecure against the *clogging attack*, a form of denial of service (DoS). The inherent vulnerability lies in the usage of the computationally intensive modular exponentiation by the server in the authentication process. We then present a fix to prevent an attacker to perform such an attack on the protocol. We note there has been another recent protocol by the same authors which is smart card based [2]. We observe that protocol also to be insecure against the clogging attack.

The second protocol we analyze is a smart card based protocol by Wang et. al [3]. They have actually claimed their protocol to be secure against DoS. But we however find the protocol to be insecure against the clogging attack. We show an attacker can exploit the fact their protocol uses multiple modular

exponentiations for authentication. A *replay attack* can be launched on their protocol to achieve a bigger clogging attack. However clogging attack can be done on this protocol in the classical way (without replays). We propose a way of making the protocol secure against this attack. This fix also works for the protocol by Jiang et al. to make it immune against clogging attacks.
Our observation is modular exponentiation is a technique which guarantees a level of security. But it might lead to an easy insecurity just in case it is used without an additional level of protection. Most of the multi-factor authentication protocols in the literature either smart card based, or memory device aided rely on the usage of modular exponentiation for their security. Hence some level of protection should be added to them to guarantee total security against the clogging attack.

2 Jiang et al.'s password based protocol

The first protocol we look at in this work is due to Jiang et al. [1] It is a remote authentication protocol, which does not involve smart cards. We however note that they had another version of the protocol which works with smart card in[2]. Once we demonstrate the vulnerability in [1] against the clogging attack, the vulnerability is easily observed to work for [2] as well. Jiang et al.'s protocol in [1] is an improvement over Chen at al.'s protocol [4] which they proved to be vulnerable against the off-line dictionary attack.

We briefly present Jiang et. al.. They prove their protocol to be immune from various attacks in [1]. However we see their protocol to be inherently vulnerable to the clogging attack (a form of the classical DoS). We present a clogging attack on the protocol. We observe their smart card based version of the protocol of [2] also to be insecure against this attack. Most of the protocols they cite in their papers [1] and [2] are vulnerable to clogging attack. We identify the mathematical basis which make the protocols vulnerable to this attack, and suggest a possible fix for them.

2.1 Review of the protocol

Jiang et. al's protocol works in five phases: *Initialization*, *Registration*, *Login*, *Authentication*, and *Passoword Change*. We present the protocol in Algorithm 1. We omit the password change phase since it is not required to demonstrate the clogging attack on the protocol.

2.2 Attack on Jiang el. al.'s protocol

The adversary $\mathcal{A}$ has the same power as assumed by Jiang et al's [1] while exposing the weaknesses of Chen et. al's protocol. We only need $\mathcal{A}$ to be able to read and modify the contents of messages over an insecure channel (during Login and Authentication phase of the protocol).

Algorithm 1 Jiang et. al.'s scheme of password authentication

```
1:
                         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. $S \rightarrow U_i$: $\{M_i, t_3\}$.
                              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 $\mathcal{H}(ID_i \| D_i \| T_3)$. 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 $U_i$
                          $sk_U = \mathcal{H}(D_i)$
                              Server $S$
                          $sk_S = \mathcal{H}(D_i')$
```

1. $\mathcal{A}$ intercepts a valid login request ($\{ID_i, C_i, V_i, T_1\}$) from step **Step L3**.
2. Since the message is unencrypted, $\mathcal{A}$ can change the timestamp T_1 to some $T_\mathcal{A}$ so that it meets the criterion $(T_2 - T_\mathcal{A}) < \Delta T$.
3. $\mathcal{A}$ changes C_i to any random garbage value $C_\mathcal{A}$.
4. $\mathcal{A}$ then sends $\{ID_i, C_\mathcal{A}, V_i, T_\mathcal{A}\}$ 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 $(T_2 - T_\mathcal{A}) < \Delta T$. This step passes as well.
3. Compute $Y_i'' = \mathcal{H}(ID_i \| x)$ and $D_i' = C_\mathcal{A}^x \mod p$, and compare V_i with $\mathcal{H}(ID_i \| Y_i'' \| C_\mathcal{A} \| D_i' \| T_\mathcal{A})$. This fails, so the request gets rejected.

The point here is the adversary $\mathcal{A}$ would now repeat the steps several times and make the server S compute the modular exponentiation step several times. Basically $\mathcal{A}$ 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 $\mathcal{A}$ clogs S with useless work and therefore denies any legitimate user any service. $\mathcal{A}$ just needs an ID of a single valid user to perform the clogging attack repeatedly.

2.3 Clogging attack performed on other similar schemes

Jiang et al., devised another smart card based password authentication protocol in [2]. This work was an improvement over another such scheme by Chen et al. [5]. We observe, that the clogging attack performed on the current protocol under consideration can also be performed on both the protocols [2] and [5]. Both the protocols are vulnerable because, the users 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 this message.

2.4 Proposed countermeasures from the attack

The 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 $\mathcal{A}$ could spoof the network address of a legitimate user and replay the login message. To prevent it, we might add a cookie exchange step at the beginning of the login phase of Jiang et al.s scheme. This step has been designed as in the well known Oakley key exchange protocol [12].

1. The 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. The 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.

Security analysis of the fix. Had $\mathcal{A}$ spoofed the U_i's IP address, $\mathcal{A}$ would not get n_2 back from S. Hence $\mathcal{A}$ only succeeds to have the S send back an acknowledgement, but not to compute the computationally intensive modular exponentiation. Hence the clogging attack is avoided by these additional steps. Saying this, we would note that 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_is and Ss private keys for a secure communication.

3 Wang et al.'s smart card based protocol

We briefly present a very recent smart card based authentication protocol by Wang et. al. [3]. They claim their protocol to be immune from the DoS attack. They assume a situation of a stolen smart card to prove this. However we see their protocol to be inherently vulnerable to the clogging (DoS) attack. The attacker would not have to steal the smart card to perform a clogging attack on their protocol. We present a clogging attack on the protocol via *Replay.* We however note a replay is not necessary to perform this attack on this protocol. But a replay step by the attacker, makes the clogging attack more effective. Most of the smart card based protocols they cite in their paper [3] are vulnerable to this attack.

3.1 Review of the protocol

Wang et. al's protocol (as like most other smart card based protocols), has the *Registration*, *Login*, and the *Verification* phases. We present the protocol in Algorithm 2.

3.2 Replay Attack on Wang el. al.'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 DoS. 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 el. al.'s protocol [3] was claimed to be secured against replay attacks but as we see, we have been able to perform a replay attack on this protocol to achieve a DoS. We assume the protocol is known to $\mathcal{A}$ (i.e. not security under obscurity).

1. $\mathcal{A}$ intercepts a valid login request ($\{C_1, CID_i, M_i\}$) from step **Step L4**.
2. $\mathcal{A}$ replays $\{C_1, CID_i, M_i\}$ several times. That is, it performs $\mathcal{A} \rightarrow S$: $\{C_1, CID_i, M_i\}$ 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. $\mathcal{A}$ can intercept whatever replies S_i sends (**Step V1**) and discard them (they would anyway be lost since SC will not expect these replies).

Algorithm 2 Wang et. al.'s scheme of password authentication

1:

Registration Phase

`User` U_i

1. **Step R1.** Choose identity ID_i, password PW_i and a random number b.
2. **Step R2.** $U_i \rightarrow S$: $ID_i, \mathcal{H}_0(b \| PW_i)$.
3. **Step R5.** Upon receiving the smart card SC, U_i enters b into SC.

`Server` S

1. **Step R3.** 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_is 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 $N_i = \mathcal{H}_0(b \| PW_i)) \otimes \mathcal{H}_0(x \| ID_i \| T_{reg})$ and $A_i = \mathcal{H}_0((\mathcal{H}_0(ID_i) \otimes \mathcal{H}_0(b \| PW_i)) \mod n)$.
2. **Step R4.** $S \rightarrow U_i$: A smart card containing security parameters $\{N_i, A_i, q, g, y, n, \mathcal{H}_0(\cdot), \mathcal{H}_1(\cdot), \mathcal{H}_2(\cdot), \mathcal{H}_3(\cdot)\}$.

`User` U_i

1. **Step R5.** 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 $A_i^* = \mathcal{H}_0((\mathcal{H}_0(ID_i^*) \otimes \mathcal{H}_0(b \| PW_i^*)) \mod n)$ 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 $C_1 = g^u \mod p$, $Y_1 = y^u \mod p$, $k = \mathcal{H}_0(x \| ID_i \| T_{reg}) = N_i \otimes \mathcal{H}_0(b \| PW_i)$, $CIDi = ID_i \otimes \mathcal{H}_0(C_1 \| Y_1)$ and $M_i = \mathcal{H}_0(Y_1 \| k \| CID_i)$.
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 $ID_i = CID_i \otimes \mathcal{H}_0(C_1 \| Y_1)$ and checks whether ID_i is in the correct format. If ID_i is not valid, the session is terminated. Then, S computes $k = \mathcal{H}_0(x \| ID_i \| T_{reg})$ and $M_i^* = \mathcal{H}_0(Y_1 \| k \| CID_i)$, 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 $C_3 = \mathcal{H}_1(ID_i \| IDS \| Y_1 \| C_2 \| k \| KS)$.
2. **Step V2.** $S \rightarrow U_i$: $\{C_2, C_3\}$.

`User` U_i

1. **Step V3.** On receiving the reply message from the server S, SC computes $KU = (C_2)^u \mod p$, $C_3^* = \mathcal{H}_1(ID_i \| IDS \| Y_1 \| C_2 \| k \| KU)$, 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 $C4 = \mathcal{H}_2(ID_i \| IDS \| Y_1 \| C_2 \| k \| KU)$.
2. **Step V4.** $U_i \rightarrow S$: $\{C_4\}$

2: Wang et. al.'s scheme (contd.)

Server S

1. **Step V5.** Upon receiving $\{C_4\}$ from U_i, the server S first computes $C_4^* = \mathcal{H}_2(ID_i\|IDS\|Y_1\|C_2\|k\|KS)$ 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

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

Server S

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

We note that the attacker $\mathcal{A}$ can simply send fake login requests to the server S and could have launched the clogging (DoS) attack having forced 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). $\mathcal{A}$ will need to send much lesser number messages to S to clog it. This replay attack is possible because, unlike Jiang et. al's protocol, Wang et at.'s protocol does not have a timestamp check.

3.3 Proposed countermeasures from the attack

The steps to avoid replay attack resulting in clogging attack. As we observe replay attacks also might be possible on most Smart card based protocols because their security relies on the computationally intensive modular exponentiation, and the messages are not by default encrypted. This is very often overlooked, since the natural result of a replay is not a DoS. A few steps to avoid these attacks on Wang et. al's Protocol (and in all 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 $\mathcal{A}$ cannot tamper with it.
2. S checks whether multiple login requests frequently comes from the same user. This *reduces* the chances of a reply.

We say *reduces* because $\mathcal{A}$ can obtain a lot of valid user ids (they are public) and send fake login requests periodically from different ids. Or $\mathcal{A}$ can store various (valid) login requests over a time period, and reply them periodically.

Yet another way to prevent clogging attack. We identify the mathematical basis which make the protocols vulnerable to clogging attacks is the modular exponentiation. The complete removal of this attack again requires to *encrypt* all the messages between U_i and S. But this would involve a key exchanging step, where each user has a private key, and a public key. The server knows the public key, and can decrypt a message encrypted by a users private key. That way, the server makes sure that the message is from a valid user, before it computes the costly modular exponentiation. This comes with a cost and depends on the level of security we want to implement. This countermeasure works for all the protocols (smart card and non smart card based).

4 Conclusion

In this paper, we have demonstrated clogging attacks on two advanced password authentication schemes to uncover the subtleties and challenges in designing this type of protocols. We observe modular exponentiation to be a technique that guarantees a level of security. But it might lead to an easily-exploitable vulnerability just in case it is used without an additional level of protection. Most of the multi-factor authentication protocols in the literature either smart card based, or memory device aided rely on the usage of modular exponentiation for their security. Hence some level of protection should be added to them to guarantee total security against the clogging attack.

References

1. Q. Jiang, J. Ma, G. Li, Z. Ma: An Improved Password-Based Remote User Authentication Protocol without Smart Cards, Information Technology and Control, vol. 42(2), (2013).
2. Q. Jiang, J. Ma, G. Li, Z. Ma: Improvement of Robust Smart-card-based Password Authentication Scheme, International Journal of Communication Systems, (2013).
3. D. Wang and C. Ma: Robust smart card based password authentication scheme against smart card security breach. Cryptology ePrint Archive (2013), `http://eprint.iacr.org/2012/439`.
4. B. L. Chen, W. C. Kuo, L. C. Wuu: A Secure Password-Based Remote User Authentication Scheme Without Smart Cards, Information Technology and Control, vol 41(1), (2012).
5. B. L. Chen, W. C. Kuo, L. C. Wuu: Robust Smart Card Based Remote Password Authentication Scheme, International Journal of Communication Systems, (2012).
6. H. S. Rhee, J. O. Kwon, D. H. Lee: A remote user authentication scheme without using smart cards, Computer Standards & Interfaces archive Volume 31 (1), (2009).
7. T. Chen, H. Hsiang, W. Shih: Security enhancement on an improvement on two remote user authentication schemes using smart cards. Future Generation Computer Systems 27(4), 377380 (2011).
8. D. He, J. Chen, J. Hu: Improvement on a smart card based password authentication scheme. Journal of Internet Technology 13(3), 3842 (2012).
9. W.S. Juang, S.T. Chen, H.T. Liaw: Robust and efficient password-authenticated key agreement using smart cards. IEEE Transactions on Industrial Electronics 55(6), 25512556 (2008).
10. C.T. Li: A new password authentication and user anonymity scheme based on elliptic curve cryptography and smart card. IET Information Security 7(1), 310 (2013).
11. S.K. Sood: Secure dynamic identity-based authentication scheme using smart cards. Information Security Journal: A Global Perspective 20(2), 6777 (2011).
12. H. Orman. The Oakley Key Determination Protocol. University of Arizona. TR 97 02.