DoS-resistant ID-based Password Authentication Scheme Using Smart Cards*

Min-Shiang Hwang† Song-Kong Chong‡ Te-Yu Chen§

Department of Management Information Systems†
National Chung Hsing University
250 Kuo Kuang Road, Taichung 402, Taiwan, R.O.C.
Email:mshwang@nchu.edu.tw
Fax: 886-4-22857173

Department of Computer Science and Information Engineering‡
National Cheng-Kung University
1 University Road, Tainan 701, Taiwan, R.O.C.

Department of Computer Science and Engineering §
National Chung Hsing University
250 Kuo Kuang Road, Taichung 402, Taiwan, R.O.C.

September 19, 2012

*This work was supported in part by Taiwan Information Security Center (TWISC), National Science Council under the grants NSC 96-2219-E-001-001, and NSC96-2219-E-009-013.

†Corresponding Author: Prof. Min-Shiang Hwang

DoS-resistant ID-based Password Authentication Scheme Using Smart Cards

Abstract

In this paper, we provide a defense mechanism to Kim-Lee-Yoo's ID-based password authentication scheme, which is vulnerable to impersonation attacks and resource exhaustion attacks. Mutual authentication and communication privacy are regarded as essential requirements in today's client/server-based architecture; therefore, a lightweight but secure mutual authentication method is introduced in the proposed scheme. Once the mutual authentication is successful, the session key will be established without any further computation. The proposed defense mechanism not only accomplishes the mutual authentication and the session key establishment, but also inherits the security advantages of Kim-Lee-Yoo's scheme, e.g. it is secure against password guessing attacks and message replay attacks.

Keywords: ID-based scheme, client-puzzles, fingerprint, password authentication, resources exhaustion attack, smart card.

1 Introduction

The client/server-based service architecture has become the major service mode for today's Internet. It enables a single computer to serve a huge amount of clients which are dispersed over different regions around the world [9]. Providing services for qualified clients and secure communication between client and server are the main topics of interest for the business environment. The authentication for legality is the first and most important step in the environment; hence, many researchers focus on this issue [8, 13, 18, 19, 20, 22, 23, 34, 35, 36, 38, 39, 42].

In general, the approaches for designing an authentication schemes can be categorized as the following types: the password-based approaches, the symmetric encryption approaches, the public-key encryption approaches, the ID-based approaches, and the hybrid approaches. In a password-based approach, a client has to submit his/her identity and password to the server, then the server validates the authenticity by comparing the data it received with the records it stored at the password table. If the password table is stored in plain-text form, it should be protected carefully in order not being disclosed. Consequently, the password table could be encrypted into a verification table for withstanding the threat of disclosing [16]. As a result, it is guaranteed that the passwords are still secure even if the verification table is disclosed. Nevertheless, the presence of the verification table still results in paying for maintenance cost and suffering from the password guessing attack once it is revealed. Accordingly, some approaches without employing the verification table are proposed [7, 11, 18, 34].

The public-key cryptography can be applied to the authentication scheme. [8, 24, 40], and the X.509 authentication service [1] are some examples in which the public-key cryptography is employed. In general, to verify clients, a server must retain key directories or adopt the services of a third party such as public key directories or certificates. These approaches demand extra protection and management mechanisms. The server must assure the robustness of key directories, which an intruder cannot access arbitrarily, and the trusted third party must ensure that the public key directories will not be altered by an attacker.

The concept of ID-based cryptosystems was first introduced by Shamir [33]. The major advantage of an ID-based scheme is the key management will be simplified. [11, 37] are some examples of authentication schemes in which the ID-based approaches are adopted.

For some specific purposes, a majority of authentication schemes are designed by combining different approaches. These schemes try to benefit from different approaches while avoiding their limitations. [7, 8, 11, 18, 23, 25, 43] are some examples of this type.

Owing to the Internet's openness and lack of security concern, the information transmitted between the authenticated client and the server should be protected from eavesdropping. In order to provide a secure environment for data transmission after the authentication process is completed, session key establishment has been incorporated within an authentication procedure. Hence, the qualified authentication schemes should not only authenticate the client but establish the session key in their authentication processes.

A user authentication scheme is usually employed by a server to verify the legitimacy of a user before he/she can access the resource or service provided by a server. A denial of service (DoS) attack or a distributed denial of service (DDoS) attack aims to thwart the legitimate user from accessing the resource or service provided by a server. In order to help the authentication scheme to work smoothly and let the legitimate user access service he/she desires successfully, the DoS/DDoS attack should be took into consideration.

A DoS/DDoS attack achieves its goal by exploiting the weakness existing in protocols. These attacks exhaust a victim's resources by sending large amounts of packets or requests. The major difference between DoS and DDoS is the amount of attackers. That is, there will be only one or few attackers in a DoS attack, while a large number of attackers will exist in a DDoS attack. Due to DoS/DDoS attacks can be easily implemented but difficultly prevented [2, 14, 27, 30], in the latest decades, these attacks have been one of the most crucial types of attacks on the Internet; these attacks make heavy losses of property [31]. Therefore, a lot of attention has been paid to these attacks and many defense mechanisms have been proposed [6, 29]. Generally, there are four types of defense mechanisms against DoS/DDoS attacks [29]: attack prevention, attack detection, attack source identification, and attack reaction. Almost all the proposed schemes work in the network layer and analyze the information existing in the packets. To resist the DoS/DDoS attack, some additional network devices such as firewalls, IDS (Intrusion detection system), or IPS (Intrusion prevention system) should be installed. Unfortunately, because of the weakness existing in the TCP/IP protocol, the DoS/DDoS attack cannot be defeated completely [29].

Furthermore, many cryptographic protocols operating on the application layer, such as authentication schemes, could suffer the DoS/DDoS attack, too. The general solution to the DoS/DDoS attack can hardly be applied to these protocols. Moreover, the weakness of each protocol suffering from a DoS/DDoS attack varies individually. This situation becomes a thorny problem.

The idea of a client puzzle [3, 4, 9, 17, 26, 41] is another feasible solution to defeat DoS/DDoS attacks. The concept of the client puzzle was first proposed by Merkle in 1978 [26]. It begins with how to share a key secretly over insecure channels. The later development of the client puzzles pays particular attention to DoS/DDoS attacks [3, 4, 9, 17, 41].

To protect the server's resources from being exhausted by clients in a DoS/DDoS attack, the design principle of client puzzles advocates that the client should always commit his/her resources before the server allocates its resources [3]. Only after the client has demonstrated his/her sincerity will the server commit its memory and computational resources to perform expensive cryptographic operations. The cost is tolerable for normal clients but high for malicious attackers [3].

In general, a puzzle could be a challenge for finding the missing part of a pre-image of a hash function whose output is given. That is, given y and z, find x such that $z = h(x, y)$ [17]. The server has to perform only one hashing in order to issue a puzzle. While the client should apply the method of brute-force in order to find the solution. Performing the brute-force search will take the client a lot of time. The client puzzles are particular suitable to be embedded in a scheme which the computation loading between two parties are seriously disequilibrium.

In 2000, Hwang-Li proposed a remote user authentication scheme using smart cards based on the E1Gamal public key cryptosystem [8]. The scheme can perform remote user authentication in a novel method, that is, it does not need any verification or password table in the authentication procedures. Later, Sun proposed an authentication scheme to improve the efficiency of Hwang-Li's scheme [35]; the security of this scheme is based on a one-way function. However, these two schemes do not allow users to choose and change their password freely. Therefore, Lee-Hwang-Yang proposed a simple and efficient remote user authentication scheme [18] to remedy these weaknesses. Without significantly increasing the computation cost, Lee-Hwang-Yang's scheme allows users to freely choose and change their passwords. Like Sun's scheme, the security of Lee-Hwang-Yang's scheme is also based on a one-way function. In 2003, Shen-Lin-Hwang proposed an enhanced scheme [34] to withstand the masquerade attack in [8], which was shown by Chan-Cheng [5]. Shen-Lin-Hwang also showed a different but easier and simpler attack in [8].

In 2002, Lee, Ryn and Yoo proposed a remote user authentication scheme [21] based on Hwang-Li's scheme; however, the scheme was found to be vulnerable to impersonation attack [12, 23]. In addition, Kim-Lee-Yoo discovered that the scheme does not allow users to change their password freely if the password is compromised [11]. Therefore, Kim-Lee-Yoo proposed two ID-based password authentication schemes using smart cards and fingerprints, i.e., a time-stamp based scheme and a nonce based scheme. To maximize the security, the schemes combine three traditional authentication methods: user knowledge (the password), possession (the uncloneable smart card) and biometrics (the fingerprint). Kim-Lee-Yoo fully employs the concept of ID-based cryptosystem proposed by Shamir in 1984 [33]. Their ID-based authentication schemes try to enable two parties to communicate securely without exchanging private or public keys or maintaining key directories and without the assistances of a third party. Kim-Lee-Yoo improved [33] by allowing all users to change their passwords freely in their schemes.

In 2004, Scott presented an impersonation attack to Kim-Lee-Yoo's schemes [32] where an attacker is able to masquerade as any identity in order to login to a server without accessing to any smart cards, passwords or fingerprints. We amend Kim-Lee-Yoo's security weakness in this article through some simple but secure modifications. The security of Kim-Lee-Yoo's schemes is based on the difficulty of the discrete logarithm problems. Accordingly, such computation intensive authentication schemes are vulnerable to DoS/DDoS attacks, especially in current client/server environment where a single server serves many clients simultaneously. Kim-Lee-Yoo did not address these problems or provide any mechanisms to prevent such attacks. To overcome this flaw, we propose a simple and efficient defense mechanism to defeat server resources exhaustion attacks.

Since the mutual authentication and confidential communication become two main security focal points in the Internet, therefore the corresponding improvements are appended to Kim-Lee-Yoo's scheme. With the proposed mutual authentication and session key establishment methods, the clients can be confident of the follow-up transaction; in addition, the rights and interests of the clients are protected.

The remaining sections of this paper are organized as follows. Section 2 reviews one of Kim-Lee-Yoo's schemes and describes its weaknsses. In Section 3, we propose the improved scheme. The resources exhaustion attacks, impersonation attacks, unilateral authentication problems and communication privacy will be conquered in this scheme. We analyze our scheme in detail in Section 4 and compare it with Kim-Lee-Yoo's scheme in Section 5. Finally, Section 6 concludes the paper.

2 The Weaknesses of Kim-Lee-Yoo's scheme

Kim-Lee-Yoo proposed two schemes in [11]. One is based on the time stamp and the other is based on the nonce. However, they consider the timestamp based authentication scheme may experience potential replay attack. The main reason causing Kim-Lee-Yoo misgivings about such an attack is the unpredictable transmission delay in the network environment. Therefore, we only briefly describe Kim-Lee-Yoo's nonce based scheme.

Before describing the weaknesses of Kim-Lee-Yoo's scheme, we first review this scheme. Kim-Lee-Yoo's scheme is composed of three phases: registration, login, and verification.

Registration phase: In this phase, the main task of a server Ser is to issue a smart card to each registered client. A client U_i who wishes to get any service must register with Ser by submitting his/her identity ID_i and chosen password PW_i to Ser through a secure channel. Ser then computes as follows:

1. Generate smart card's identifier CID_i.
2. Compute $S_i = ID_i^{SK} \bmod n$.
3. Compute $h_i = g^{PW_i \cdot SK} \bmod n$.

Here SK is a secret key kept by Ser, n is a large prime number, and g is a generator of Z_n^*. Ser stores the values of $n, g, ID_i, CID_i, S_i, h_i$ and f into the smart card of U_i. f represents a one-way function herein. g and n are public elements known to U_i and Ser.

Login phase: When U_i wants to access to Ser, he/she first inserts the smart card into the card reader and then enters his/her ID_i, PW_i and fingerprint into the input device. If verified successfully, the following tasks will be performed.

1. The smart card sends ID_i and CID_i to Ser to initiate the login request.
2. After the verification on ID_i and CID_i, Ser sends $N = f(CID_i, r_s)$ to the smart card, where r_s is a random number generated by Ser.
3. The smart card computes $X_i = g^{r_i \cdot PW_i} \bmod n$ and $Y_i = S_i \cdot h_i^{r_i \cdot N} \bmod n$, where r_i is a random number using co-ordinates of minutia of input fingerprint. The smart card sends X_i and Y_i to Ser.

Verification phase: Ser verifies U_i through the following equation:

$$Y_i^{SK^{-1}} \equiv ID_i \cdot X_i^N \bmod n.$$

There are four security weaknesses that exist in both the nonce-based and timestamp-based schemes. For simplicity, we will focus on the nonce-based scheme.

2.1 Impersonation Attacks

The carelessness design of the login phase in Kim-Lee-Yoo's scheme allows for successful impersonation attacks. There are no source identification procedures before Ser takes the received parameters X_i and Y_i to authenticate U_i by performing a complex discrete logarithm computation $Y_i^{SK^{-1}} \equiv ID_i \cdot X_i^N \bmod n$. The same weaknesses can be found in the Secure Sockets Layer (SSL) protocol [9]. The impersonation attacks shown by Scott take advantage of this security weakness [32].

In Scott's attacks [32], an attacker firstly eavesdrops to obtain values of ID_j, X_j, Y_j and N from a legitimate login of client U_j. The attacker then calculates $G = ID_j \cdot X_j^N \bmod n$ and gets $G^{SK} = Y_j \bmod n$. To login as user U_i, the attacker starts the login procedure by sending the eavesdropped values of ID_i and CID_i. When the attacker receives a nonce N sent back by Ser, he/she responds by submitting $X_i = G/ID_i^{N^{-1}} \bmod n$ and $Y_i = (G^{SK})^N \bmod n$. X_i and Y_i will satisfy the verification of Ser. Thus, with the values G and G^{SK}, an attacker is able to impersonate any clients he/she likes.

In this article, we propose a defense mechanism to withstand such attacks. The defense mechanism allows Ser to ascertain whether the parameters X_i and Y_i are coming from a legitimate client before it launches heavy authentication computing. There is no way for an attacker to masquerade as a legitimate client and forge X_i and Y_i in the proposed scheme. This means that the threat of impersonation attacks and server resource exhaustion attacks no longer exists. See Sections 4.1.1 and 4.1.6 for detailed discussion.

2.2 Server Resource Exhaustion Attacks

Kim-Lee-Yoo's authentication scheme makes full use of the property of discrete logarithms over finite fields; however the heavy computation cost makes it vulnerable to resources exhaustion attacks. These valuable resources include memory and CPU cycles. An attacker can arbitrary send a large number of counterfeit values for X_i and Y_i to Ser in Step 3 of the login phase and then request a heavy authentication scheme. As a result, the system becomes paralyzed by the attacker.

Kim-Lee-Yoo did not address this weakness nor provide any defense mechanism. We propose a simple and efficient defense mechanism in which the server can master its loadings to prevent resource starvation. The mechanism fully takes the limited computation capability of smart cards into consideration. There are no unequal treatments in the proposed mechanism when the clients request for services. We believe that such moderate hard client puzzles [3, 4, 9, 41] are a good fit for the smart card environment.

2.3 Absence of Mutual Authentication

The astonishing market growth of the Internet has led to increased concerns over authentication and privacy [19, 20]. Authentication ensures that only the validated client has the right to access the server, while privacy ensures that communication messages are accessible only by authorized parties. However, Kim-Lee-Yoo's scheme fails to achieve these security requirements.

Due to the unilateral authentication of Kim-Lee-Yoo's scheme, the server can authenticate the client, but the client cannot verify the server. The unsoundness scheme makes U_i unable to confirm who is over the other end. To fool U_i, an attacker can respond to the login request of U_i by sending a randomly generated value for N. Upon receiving X_i and Y_i from U_i, Ser sends back an *authentication success* message to U_i. U_i cannot trust the replied message without mutual authentication.

To remedy this weakness, an efficient mutual authentication method is proposed. The computational costs of the method are very low; only hash operations and exclusive-or calculations are employed to achieve this purpose.

2.4 Absence of Session Key Establishment

To maintain the conversations privacy after authentication, messages encryption is imperative. There are no session keys established in Kim-Lee-Yoo's scheme; therefore, privacy cannot be ensured. The authors might argue that the session key establishment is not the focus of the scheme or that the related parties can run another protocol to establish a session key for messages encryption; however, we disagree with these contentions. Firstly, we deem that the session key establishment is an essential requirement when designing an authentication scheme. The scheme designers should not neglect it. Secondly, employing another protocol to establish a session key is inefficient due to the increased calculation and communications costs.

In the proposed mechanism, we achieve this requirement in a novel way. The mechanism is very efficient, with no added calculations and communications needed to establish a session key after successful mutual authentication. In the next section, a new defense mechanism is proposed to remedy the weaknesses described.

Table 1: The notations used in the proposed scheme

Notations	Description
v_s	Solution of the puzzle, decided by Ser
N_s	Nonce of Ser
N_i	Nonce of U_i's smart card
q_i	Random number generated by U_i's smart card
$h()$	128 bits one-way hash function
T_i	Signature of U_i on values X_i and Y_i
SK	Secret key of Ser, used for clients registration and mutual authentication
sk_s	Secret key of Ser, used for puzzle verification
$token_i$	Message authentication code issues from Ser to U_i
$puzzle(p, x_1, x_2, ..., x_n)$	Given $p, x_1, x_2, ..., x_n$, find v such that $h(x_1, x_2, ..., x_n, v) = p$

3 The Proposed Defense Mechanism

The proposed authentication mechanism contains three phases: registration, login and verification. Since some related notations have been described in Section 2, for simplicity, we ignored the duplicate description here. Table 1 shows the rest of notations used through this article. Below, we shall depict how the defense mechanism comes about in our scheme. Figure 1 demonstrates the proposed defense mechanism.

Registration phase: The registration phase is similar to that of Kim-Lee-Yoo's scheme. After client U_i submits his/her identity ID_i and password PW_i, Ser then computes as follows:

1. Compute $S_i = ID_i^{SK} \bmod n$.
2. Compute $h_i = g^{PW_i \cdot SK} \bmod n$.
3. Compute $W_i = h(ID_i, SK)$.

In the above descriptions, n is a large prime number, g is a generator of Z_n^*, and SK is a secret key kept by Ser. Ser then writes n, g, ID_i, CID_i, S_i, h_i and W_i into a smart card and issues the smart card to the user U_i. Subsequently, the user U_i enrolls his/her fingerprint which is written to the smart card as a tem-

Client		Server
(1) $M_1 = \{ID_i, CID_i, N_i\}$	$\xrightarrow{M_1}$	(2) Verify ID_i, CID_i and N_i $p = h(ID_i, N_i, N_s, v_s)$ $token_i = h(p, ID_i, N_i, N_s, v_s, sk_s)$ $M_2 = \{p, N_s, token_i\}$
(3) $h(ID_i, N_i, N_s, v'_s) \stackrel{?}{=} p$ $X_i = g^{r_i \cdot PW_i} \bmod n$ $Y_i = S_i \cdot h_i^{r_i \cdot token_i} \bmod n$ $Z_i = W_i \oplus q_i$ $T_i = h(X_i, Y_i, token_i, q_i)$ $M_3 = \{ID_i, X_i, Y_i, Z_i, T_i, v'_s, N_i, N_s\}$	$\xleftarrow{M_2}$ $\xrightarrow{M_3}$	
		(4) Checks N_s, ID_i and N_i $token_i \stackrel{?}{=} h(p, ID_i, N_i, N_s, v'_s, sk_s)$ Extracts $q'_i = Z_i \oplus h(ID_i, SK)$ Verifies $T_i \stackrel{?}{=} h(X_i, Y_i, token_i, q'_i)$ $Y_i^{SK^{-1}} \stackrel{?}{=} ID_i \cdot X_i^{token_i} \bmod n$ $M_4 = h(q'_i)$
(5) $h(q_i) \stackrel{?}{=} h_(q'_i)$	$\xleftarrow{M_4}$	

Figure 1: The proposed login and verification mechanism

plate by a fingerprint input device. With adopting match-on-card technology [15, 28], the fingerprint information is stored and matched in the tamper proof environment of the smart card. There are different access modes for the data stored in the smart card. ID_i and CID_i could be readable only after successful user authentication through password and fingerprint. While, S_i, h_i, and W_i should never be read from the smart card. The previous procedures are done when U_i registers at Ser for the first time. The submission of PW_i should be carried out through a secure channel. The secure channel could be achieved when the client and the server manager meet each other physically. For these protocols in which smart cards are adopted, the client identification process is necessary before the smart card is issued to the identified client. Moreover, the client identification process is generally carried out by a physical manner in practice. If U_i's password is compromised, the registration phase is completed again and U_i can choose a new password he/she likes at that time.

Login phase: To access the server Ser, the client U_i must first insert the smart card into the card reader. Then U_i enters his/her password PW_i and imprints his/her fingerprint through a fingerprint input device. If the password PW_i and fingerprint are verified by the smart card, the following actions are taken:

1. The off-card software on the client side reads ID_i, CID_i, and N_i from the smart card and initiates the login request by forwarding $M_1 = \{ID_i, CID_i, N_i\}$ to Ser.
2. Ser verifies ID_i, CID_i and N_i. If the client has already used N_i, then Ser ignores the remaining computation; otherwise, Ser determines v_s based on its loading, then computes $p = h(ID_i, N_i, N_s, v_s)$ and $token_i = h(p, ID_i, N_i, N_s, v_s, sk_s)$. Afterward Ser sends $M_2 = \{p, N_s, token_i\}$ to U_i. p is the output of the hash function which is included in M_2, while v_s is one of inputs of the hash function which is not included in M_2. Hence, the puzzle issued by the Ser can be regarded as the problem that given p, ID_i, N_i, and N_s, find v_s such that $h(ID_i, N_i, N_s, v_s) = p$. Note that N_s and v_s are intertwined; Ser will change these values periodically.
3. The smart card tries to seek out the solution v'_s of $puzzle(p, ID_i, N_i, N_s)$. In order to find a valid v'_s such that $h(ID_i, N_i, N_s, v'_s) = p$, the smart card, without any knowledge of the solution, should apply a brute-force method to try each possible value of v'_s until a valid one is found. Further, the searching for the solution is requested to start from 0, 1, 10, 11, ... respectively, so as to control the server's loading and resist the opportunistic users which will be discussed in Sections 4.1.7 and 4.1.9. Once the solution of the puzzle has been found, the smart card computes $X_i = g^{r_i \cdot PW_i} \bmod n$, $Y_i = S_i \cdot h_i^{r_i \cdot token_i} \bmod n$, $Z_i = W_i \oplus q_i$, and $T_i = h(X_i, Y_i, token_i, q_i)$. Note that the value r_i is used to compute X_i and Y_i is a random number which is generated by U_i's smart card using a secure random number generator. And q_i is chosen and stored by the smart card for the future mutual authentication and session key establishment. Finally, the smart card sends $M_3 = \{ID_i, X_i, Y_i, Z_i, T_i, v'_s, N_i, N_s\}$ to

Ser.

Verification phase: In this phase, the mutual authentication is achieved between Ser and U_i. Ser first verifies U_i in Step 4, and then U_i verifies Ser in Step 5. This mutual authentication allows U_i to believe that he/she is communicating with the legitimate Ser and provides confidence in the remaining transaction. We describe this process as follows.

4. Upon receiving the message M_3 from U_i, Ser checks the value of N_s. If N_s is fresh, and U_i did not use the same N_i under this N_s before, Ser performs the remaining verification; otherwise, it discards M_3 silently. Ser checks if the equation $token_i \stackrel{?}{=} h(p, ID_i, N_i, N_s, v'_s, sk_s)$ holds. The approval of the equation means that U_i has solved the puzzle. That is, U_i has shown his/her sincerity to Ser. Hence, Ser will proceed to authenticate U_i further. Afterward, Ser stores N_s, ID_i and N_i to prevent the solution from being reused. Note that N_s-ID_i-N_i will only be recorded for a short time, they are deleted after the lifetime of N_s and v_s.
   Ser continues its verification by extracting the random number q'_i from $Z_i \oplus h(ID_i, SK)$. Ser checks the T_i to determine if X_i and Y_i are coming from the true client. Ser performs further verification by checking the password PW_i of U_i through the equation $Y_i^{SK^{-1}} \equiv ID_i \cdot X_i^{token_i} \bmod n$. If the equation is correct, then Ser verifies U_i and sets q_i as a session key for future confidential communications. Ser then sends $M_4 = h_(q'_i)$ to U_i for the mutual authentication.
5. The smart card verifies M_4 using its pre-stored value q_i. If the equation $h(q_i) \stackrel{?}{=} h_(q'_i)$ holds, then U_i verifies Ser and sets the session key as q_i.

In the proposed mechanism, we maintain the merit of Kim-Lee-Yoo's scheme, which allows users to choose their password in the registration phase. More importantly, a defense mechanism is proposed to resist resource exhaustion attacks and impersonation attacks. Moreover, the mutual authentication and session key establishment are also achieved. In the next section, we shall briefly analyze the robustness of the proposed defense mechanism.

4 Analysis

In this section, we evaluate and discuss several aspects of the proposed scheme. First, the security of our scheme is analyzed in detail. Several important parameters related to the security are also discussed. Finally, we evaluate the performance of our scheme.

4.1 Security Analysis

In the following, we deliberate on the security of our scheme. We demonstrate that the proposed scheme is secure against some well-known attacks. Then we examine how mutual authentication and session key establishment can be achieved. In addition, some security-related parameters are investigated.

4.1.1 Resistance to Impersonation Attacks

Generally, impersonation attacks can be classified as internal or external.

1. Internal impersonation attacks: a legitimate but malicious client tries to impersonate the other clients to get the services of a server.

2. External impersonation attacks: A non-client attacker tries to impersonate some clients to obtain the services of a server.

According to Scott's attacks [32] on Kim-Lee-Yoo's scheme, an attacker is able to impersonate any clients in order to successfully login to the server. The key point is that the values of X_i and Y_i can be forged without the detection of the server. The proposed mechanism defeats these types of impersonation attacks using T_i as a signature of X_i and Y_i, and sending them to Ser in chorus. Below, we discuss the attacks in detail.

For internal impersonation attacks, assume that there is a legitimate but malicious client U_m intends to impersonate client U_i. Consider the scenario in which U_m has eavesdropped login messages ID_j, X_j, Y_j and $token_j$ of a legitimate user U_j first, and he/she attempts to login the server by masquerading as the user U_i . To login as client U_i, U_m starts the login procedure by sending the values of ID_i, CID_i and a self-chosen nonce value N_m to Ser. When the values of $M_2 = \{p, N_s, token_i\}$ sent from Ser are received by U_m, U_m seeks out the puzzle solution and responds by submitting

$$\begin{aligned} X_i &= \frac{ID_j \cdot X_j^{token_j}}{ID_i^{token_i^{-1}}}, \\ Y_i &= [Y_j]^{token_i} (= [(ID_j \cdot X_j^{token_j})^{SK}]^{token_i}). \end{aligned}$$

The above X_i and Y_i appear to be passing the identity verification $Y_i^{SK^{-1}} \equiv ID_i \cdot X_i^{token_i} \bmod n$ of Ser. Since

$$\begin{aligned} ID_i \cdot (X_i)^{token_i} = ID_i \cdot \left(\frac{ID_j \cdot X_j^{token_j}}{ID_i^{token_i^{-1}}} \right)^{token_i} &= (ID_j \cdot X_j^{token_j})^{token_i}, \\ (Y_i)^{SK^{-1}} = \{[(ID_j \cdot X_j^{token_j})^{SK}]^{token_i}\}^{SK^{-1}} &= (ID_j \cdot X_j^{token_j})^{token_i}. \end{aligned}$$

However, verification of T_i fails. In the proposed defense mechanism, every client creates a signature $T_i = h(X_i, Y_i, token_i, q_i)$ using a random number q_i as a secret key. Ser checks T_i before performing the further verification. To impersonate U_i successfully, U_m needs to pass the verification of Ser on T_i.

Since the secret key q_i used to generate T_i is a random number chosen by U_i arbitrarily. It is unrelated to the identity of any client and is protected by performing the exclusive-or operation with $W_i = h(ID_i, SK)$ which is a secret stored in U_i's smart card. Therefore, if U_m persists in impersonating U_i then U_m employs his/her own $W_m = h(ID_m, SK)$ and a self-chosen q_m to generate $Z_m = W_m \oplus q_m$, X_i', Y_i', and $T_i' = h(X_i', Y_i', token_i, q_m)$. However, he/she will be defeated by Ser, even though he/she solves the puzzle because Ser has been told that U_i has started the login procedure. Ser will use the identity of U_i to generate $h(ID_i, SK)$ and extract q_m from $Z_m \oplus h(ID_i, SK)$. This extraction will fail because $q_m \neq h(ID_m, SK) \oplus q_m \oplus h(ID_i, SK)$; as a result, the verification of T_i' will fail, too.

To generate a proper q_i that will pass the verification of T_i , an attacker A_t must obtain the values of W_i stored in U_i's smart card. Since the smart card is tamper-proof and the stored values cannot be retrieved directly, A_t will be defeated. A similar situation is confronted by an attacker A_t who launches external impersonation attack.

4.1.2 Resistance to Password Guessing Attacks

Because of the low entropy of human's memorizing passwords, the password guessing attack is a crucial concern in the scheme where passwords are employed. The password guessing attacks can be categorized as on-line and off-line password guessing attacks. An on-line password guessing attack happens when an adversary attempts to use a guessed password to pass the verification of a remote server in an on-line transaction. While an off-line password guessing attack happens when an adversary intercepts a valid login request, guesses the password and verifies his guessing locally in an off-line manner.

If an adversary attempts to perform an on-line password guessing attack to the proposed scheme. First of all, he/she must pay computational costs to solve the puzzle issued from the server for each on-line guessing. This will result in considerable computational costs and deter an adversary from performing an on-line password guessing attack. Furthermore, if the server records the times of continuous authentication failures for each client, it can directly reject all requests from the client when his/her times of the continuous failed trials have exceeded the predefined times.

If an adversary attempts to perform an off-line password guessing attack to the proposed scheme. It is noticeable that the proposed scheme is free from password/verification table. There are two possible cases to be considered. For the first case in which an adversary attempts to guess the password from h_i, ($h_i = g^{PW_i \cdot SK} \bmod n$), firstly, h_i is stored in the tamper-proof smart card and cannot be retrieved directly. Even if h_i is compromised, it is still infeasible for an adversary to verify the guessed password because of the unknown of SK. For the second case in which an adversary attempts to guess the password form X_i, ($X_i = g^{r_i \cdot PW_i} \bmod n$), similarly, it is infeasible for him/her to verify the guessed password. This is because that the adversary cannot get any information about the random number r_i.

Moreover, if a client reveals his/her password accidentally or he/she wants to change his/her password for some reason, the registration phase could be re-performed again. At this phase, the client can choose his/her passwords at his/her own will.

4.1.3 Resistance to Message Replay Attacks

The message replay attacks refers to the process that an adversary replays intercepted messages of a legal client and attempts to impersonate the client to pass through the authentication procedure. This attempt will be defeated by the adoption of the nonces, N_i and N_s, in the proposed scheme. As the proposed scheme proceeds, N_s should be changed periodically; during the period that N_s keeps the value unchanged, each N_i should be different from each other for a client. That is, the values of $token_i (token_i = h(p, ID_i, N_i, N_s, v_s, sk_s))$ will not be repeated in any individual authentication procedure. Therefore, if an adversary eavesdrops the messages from a legitimate authentication procedure and replays the messages, the intention will be discovered by the server. That is, if the adversary simply replays the eavesdropped messages, the solution to the puzzle will be incorrect. It will be detected when the server performs the following validation in the Verification phase:

$$token_i \stackrel{?}{=} h(p, ID_i, N_i, N_s, v_s, sk_s).$$

Even if the adversary has solved the puzzle and passed the above verification, the fraud will be detected by the server because of the following validation in the Verification phase:

$$T_i \stackrel{?}{=} h(X_i, Y_i, token_i, q_i).$$

Hence, the login request will be rejected and the attack will not succeed. At the end, the proposed scheme can resist the message replay attacks.

4.1.4 Evaluation of Mutual Authentication and Session Key Establishment

To achieve the mutual authentication and session key establishment, Ser stores the value of $W_i = h(ID_i, SK)$ in U_i's smart card at the registration phase. To verify Ser, U_i's smart card generates a random number q_i and computes $Z_i = W_i \oplus q_i$. Ser will return $h(q_i')$ to U_i in Step 4 for verification. Through this simple method, U_i can verify Ser and establishes a session key with Ser in Step 5. This session key establishment is very efficient. It does not require any extra calculations or transmissions. In the later communications, Ser and U_i can use the established session key q_i to encrypt all the messages transmitted between them.

We consider the verification to Ser should not be started in Step 1 (U_i sends M_1 and $Z_i = W_i \oplus q_i$ to Ser) and verified by U_i in Step 3 (through an equation $h(q_i) \stackrel{?}{=} h(q_i')$). It is too premature. The main consideration is that Ser is unable to assure the legitimate of U_i at the beginning, therefore it should not commit more resources to these calculations $q_i' = Z_i \oplus h(ID_i, SK)$ and $h(q_i')$. Only after verifying the identity of U_i should Ser commit its resources to compute q_i' and $h(q_i')$ to achieve the mutual authentication and session key establishment.

4.1.5 The Security of Session Key q_i

To compute a signature T_i, U_i's smart card uses a random number q_i as a secret key but not W_i. Since W_i is a long-term secret value in the smart card, its utilization should be minimized to avoid unknown attacks. Therefore, a one-time value for q_i is introduced. For security reasons, the length of q_i is recommended as 128 bits and should be generated through a secure random number generator.

Each time that U_i wants to login to Ser, his/her smart card will employ a new random number q_i as a secret key to compute T_i. This q_i will then be used to achieve mutual authentication and session key establishment as described in Section 4.1.4. This secure protection to q_i ($Z_i = W_i \oplus q_i$) requires low computation. If the length of W_i is long enough, e.g. 128 bits, then it is secure against compromise. There is only a 50 percent probability of guessing a correct bit. Thus, it should be quite difficult for an attacker to obtain the value of q_i.

4.1.6 Resistance to Server Resource Exhaustion Attacks

In a client/server environment, a single server must support many clients simultaneously. However, an authentication scheme without any protection mechanism is vulnerable to resource exhaustion attacks, especially for computation intensive authentication schemes. Attackers or malicious clients may launch DoS/DDoS attacks such as resource exhaustion attacks.

These problems occur when the computation loadings of the server heavily exceed that of its clients. In this situation, an attacker is able to storm Ser by sending various malevolent authentication requests. As a result, the server's resources will be exhausted by these highly concentrated verification requests.

Figure 2 represents the authentication scheme without the puzzle protection. For simplicity, we named it as ASwP scheme. As described in Sections 4.1.1, there is no ways for an attacker to mount the internal or external impersonation attacks. In other words, no forged X_i' and Y_i' can pass the verification of Ser on the signature T_i. However, the problem of resource exhaustion attacks still exists. We describe a scenario in which ASwP is vulnerable to such attacks.

When there are a large number of authentication requests, the client will send each login request represented as ID_i and CID_i to Ser. As soon as the client U_i who originates the request receives the respective N from Ser, he/she will randomly generate two fake values for X_{fi} and Y_{fi}, then U_i uses his/her own $W_i = h(ID_i, SK)$ and a self-chosen q_i to compute $Z_i = W_i \oplus q_i$ and $T_i = h(X_{fi}, Y_{fi}, N, q_i)$. Afterward, U_i sends X_{fi}, Y_{fi}, Z_i and T_i to Ser.

It is easy to see that T_i will satisfy the verification of Ser. Ser then checks $Y_i^{SK^{-1}} \equiv ID_i \cdot X_i^N \bmod n$. Of course the equation will fail. With enough malevolent authentication requests, the computational resources of Ser will be exhausted due to heavy exponential calculations. Consequentially, regular clients will be rejected by Ser.

Since X_{fi} and Y_{fi} are two randomly generated values, the computation of Z_i only involves one exclusive-or, and the signature T_i conducts only one hash operation; therefore, the cost to mount the resources exhaustion attack is low.

To defeat these attacks, the idea of a client puzzle [3, 4, 9, 26, 41] is employed into ASwP. The login and verification mechanism with puzzle protection is shown at Figure 1 on Section 3.

Suppose in the same scenario, a large number of authentication requests come simultaneously. These mass authentication requests could be originated from either few malicious clients (DoS) or a group of malicious clients (DDoS). In the former case, the use of client puzzles is able to deter clients from sending a large number of requests simultaneously, since each malicious client will be requested to solve many client puzzles. Thus, there are two possibilities: first, if a malicious client actually solves these puzzles, this will result in spending too much time to perform the attack. Second, if a malicious client replies to these puzzles with fake solutions, the server will easily detect the fake solution by performing only one hash operation. Hence, with the help of the client puzzle, the server can resist the DoS attack efficiently.

In the latter case, if these mass authentication requests are originated from a group of malicious clients, the server will encounter more troubles with distinguishing sincere clients from malicious clients. Thus, it is almost impossible to resist this attack. Nevertheless, the incorporation of client puzzle allows "graceful degradation" in service. The difficulty of a client puzzle can be scaled based on the server's loadings. Hence, the server can control the amount of incoming requests on the basis of its loadings by adjusting the difficulty of the client puzzle. The more loadings the server takes, the more difficult puzzles will be issued. If there are a large numbers of clients arriving at the same time, the system's loading will go beyond the threshold. Accordingly, the new coming client puzzles issued by the server will be more difficult. This adjustment enables the system to provide a proper quality of service by alleviating the amount of incoming requests. Instead of obtaining a denial of service from the server, a legitimate client will accept the consequence of obtaining the requested service, even though this will result in some degradation in the performance. Note that, for the consideration of performance, once the server's loading is lower than some predefined threshold, the server will stop the mechanism of puzzles.

4.1.7 Evaluation of Puzzle Difficulty v_s

So far, none of the proposed protocols [3, 4, 9, 26, 41] can control the difficulty of the puzzle precisely. Consequentially, it is difficult for the server to control its loading, because it does not know how long the client will take to obtain the puzzle solution or when it should expect to perform the expensive cryptographic operations. In the previous protocols, the server usually decides the

<u>*Client*</u>		<u>*Server*</u>
(1) $M_1 = \{ID_i, CID_i\}$	$\xrightarrow{M_1}$	(2) Verify ID_i, CID_i $N = f(CID_i, r_s)$ $M_2 = \{N\}$
(3) $X_i = g^{r_i \cdot PW_i} \bmod n$ $Y_i = S_i \cdot h_i^{r_i \cdot N} \bmod n$ $Z_i = W_i \oplus q_i$ $T_i = h(X_i, Y_i, N, q_i)$ $M_3 = \{ID_i, X_i, Y_i, Z_i, T_i\}$	$\xleftarrow{M_2}$	
	$\xrightarrow{M_3}$	(4) Extracts $q_i' = Z_i \oplus h(ID_i, SK)$ Verifies T_i $Y_i^{SK^{-1}} \equiv ID_i \cdot X_i^N \bmod n$ $M_4 = h(q_i')$
(5) $h(q_i) \stackrel{?}{=} h_(q_i')$	$\xleftarrow{M_4}$	

Figure 2: The authentication scheme without puzzle protection

puzzle difficulty (refer to value k of [3, 4] and value m of [41]), and then estimates the time step (the interval of time required to perform a hash operation, refer to [9]) needed to solve a puzzle on "average".

Here, we use [3, 4] as an example. Let k represent a puzzle difficulty level. The expected time steps for a client to solve a puzzle is $O(2^{k-1})$ on average, $O(2^k)$ in the worst case, and $O(1)$ in the best case. We think that the server is difficult to anticipate where the solution will fall. Therefore, the number of time steps needed to solve a puzzle is hard to expect. If these client puzzle protocols are applied, not only the server is hard to control its loading, but each smart card will face a different puzzle difficulty. Due to limited computation capability, the smart cards which require $O(2^k)$ time steps to solve a puzzle are harmful to their general performance, and, it is unfair if compared to those who get simple puzzles. We consider all clients should have fair access to resources when connecting to the server.

Our defense mechanism eliminates flaws by allowing the server Ser to determine the puzzle difficulty v_s at the beginning. Note that the value of v_s is a set of binary numbers. For example, if $v_s = 110111010110$, then the most efficient way to search for the correct value of v_s is by brute force [3, 4, 41]. U_i

starts its guessing of values from $0, 1, 10, 11, \cdots$ on. Since Ser knows v_s before, it can estimate the number of time steps needed for U_i to solve the puzzle precisely. The defense mechanism assures that no matter what the value of v_s is, Ser can guarantee that the smart card will obtain the puzzle solution in an appropriate number of time steps.

Controlling the value of v_s is the key factor in the proposed defense mechanism. Ser should raise the difficulty of v_s if it is overloading, for example, under resource exhaustion attacks. When the loading becomes normal, the defense mechanism should be halted, and Ser should send a special message containing the value $token_i = h(CID_i, r_s)$ to those clients who are making connection requests to Ser, where r_s is a random value generated by Ser. This special message tells the clients to compute X_i, Y_i and Z_i only. The remaining verification phase in Ser site will remain valid.

Alternatively, with the same puzzle difficulty v_s for each U_i, we consider Ser will still alive when receiving these puzzle solutions from a different U_i. The main consideration is that the puzzle difficulty v_s is decided by Ser, each time it issues a puzzle to U_i, it can totalize the issued puzzles. Therefore, the loadings of Ser are easy to be estimated when receiving these puzzle solutions which under the same puzzle difficulty v_s. Ser can control its resource utilization elastically by adjusting the cost for U_i. The cost is negligible for normal clients but high for attackers.

Furthermore, for those attackers who send many forged messages M_1 and M_2 to Ser, attempt to paralyze the puzzle's generations and verifications, we make the following assumption:

Assumption 1: The puzzle generations and verifications will never be affected by server resource exhaustion attacks.

The above assumption is also true for [3, 4, 9, 41]. If attackers can paralyze the puzzle generations and its verifications, these protocols are not considered functional in such attack models. Therefore, such an assumption is required. From another point of view, the puzzle generations and its verifications perform inexpensive hash operations making it difficult for attackers to succeed in these types of attacks. Computational resource exhaustion attacks to hash operations are very rare.

4.1.8 Verification of N_s, N_i, ID_i and CID_i

Ser verifies ID_i, CID_i and N_i in Step 2 of the proposed mechanism as shown in Figure 1. The main purpose of this verification is to prevent duplicate puzzle generation and wasted resources as a result of receiving the same M_1 from client U_i due to network transmission delay or attackers. For example, the network transmission delay may cause U_i to believe that the related packets are lost and a retransmission mechanism to begin.

Moreover, to prevent duplicate puzzle verification, Ser will check N_s, ID_i and N_i in Step 4 of the proposed mechanism. Note that this is a weak verification; it cannot prevent the problem that ID_i or N_i are modified by an attacker in the transmission. Therefore, we make the following assumption:

Assumption 2: An attacker has only limited capability to interfere with the communication between U_i and Ser.

This assumption also exists in [9] and [41]. If an attacker can tamper any packets at will, then he/she can mount a DoS attack simply by destroying these packets. The similar discussion can be found in [9] and [41].

4.1.9 Resistance to Opportunists

In this section, we consider opportunistic clients who starts their guessing not from values $0, 1, 10, 11, \cdots$ gradually but from a greater value and wishes to access to server resources faster than others. To resist such opportunists, we adopt the idea of Bocan [4] and propose a new formula, which will be used in Step 4 of the proposed protocol shown in Figure 1.

In 2004, Bocan proposed the "strong attack" which is regarded as a DoS attack in which an attacker can access massive computing power and launch attacks (refer to [4] for details). Bocan considers Aura et al.'s protocol [3] as vulnerable to strong attacks; therefore, a formula described below is proposed:

$$T_{estimated} = (2^k - 1) * T_{operation},$$

where $T_{estimated}$ represents the estimated time for solving the puzzle; k is the puzzle difficulty level as we described in Section 4.1.7; and $T_{operation}$ is the minimum time needed to perform a hash operation. Bocan used the $T_{estimated}$ as a reasonable response time for a client to solve the puzzle at difficulty level k. If the response time of some client is less than $T_{estimated}$, then a strong attack occurs, and connection with the client should cease.

There are some mistakes in the formula. Bocan argued that the formula estimates the average time needed to solve a puzzle, but we consider the formula is for the worst case. For an average case, the formula should be $T_{estimated} = (2^{k-1}) * T_{operation}$.

Bocan's formula inherits the weakness as we described in Section 4.1.7: it cannot estimate the number of time steps for a client U_i to solve a puzzle precisely. The main reason is the server Ser does not know where the solution will fall. Therefore the weakness of Bocan's threshold puzzles is unavoidable.

To resist a strong attack, we modify Bocan's formula as follows:

$$T_{estimated} = v_s * T_{operation}.$$

Since Ser decides the value of v_s, it can estimate the $T_{estimated}$ simply. This new formula can be placed in Step 4 of the proposed defense mechanism as demonstrated in Figure 1, that is, after Ser checks to the value of N_s, ID_i and N_i. The opportunists will be rejected by Ser if the response time is less than expected.

Table 2: The time complexity of our scheme

Phase	Client	Server
Registration	0	$1T_{hash} + 1T_{mul} + 2T_{exp}$
Login	$(n+1)T_{hash} + 1T_{xor} + 3T_{mul} + 2T_{exp}$	$2T_{hash}$
Verification	$1T_{hash}$	$4T_{hash} + 1T_{xor} + 1T_{mul} + 2T_{exp}$

4.2 Performance Analysis

The performance of the proposed scheme is analyzed in this section. The performance evaluation focuses on the computation and storage. Subsequently, we go deep into the time required for solving variant client puzzles on smart cards with respect to different speeds.

4.2.1 Performance Evaluation

In the following, we analyze the performance of the proposed scheme in terms of the number of cryptographic operations performed. In order to explain the computation cost, some notations are defined as below:

- T_{hash}: the time for a hash computation.
- T_{xor}: the time for an exclusive-or computation.
- T_{mul}: the time for a modular multiplication computation.
- T_{exp}: the time for a modular exponentiation computation.

The result of the computation cost is given in Table 2. In the registration phase, the server requires $1T_{hash} + 1T_{mul} + 2T_{exp}$ computations for each client's registration. To complete an authentication process, the server and the client require $6T_{hash} + 1T_{xor} + 1T_{mul} + 2T_{exp}$ and $(n+2)T_{hash} + 1T_{xor} + 3T_{mul} + 2T_{exp}$ computations respectively. Note that n is the number of hashing operations performed by the client to solve the puzzle. The number n can be adjusted by Ser according to its loading.

Table 3: The time (in seconds) required for solving puzzles on smart cards

Difficulty	Clock speed		
	3MHz	8MHz	16MHz
2^3	0.180592	0.067722	0.033861
2^5	0.722368	0.270888	0.135444
2^8	5.778944	2.167104	1.083552
2^{10}	23.115776	8.668416	4.334208

In the proposed scheme, Ser will keep N_s and v_s unchanged for a period. Before the client has been authenticated, Ser only stores ID_i and N_i for each login request during this period. Hence, the amount of information that Ser should store for the unauthenticated client in a period is very low. Moreover, Ser can adjust the duration of the period based on its memory utilization.

4.2.2 Evaluation of execution time for solving client puzzle on smart cards

The time needed for solving a client puzzle mainly depends on the difficulty v_s of the issued puzzle and the specification of the adopted smart card. The faster the smart card is or the easier the puzzle is, the less time it spends.

According to the provided information in [10], an 8-bit platform, the Motorola HC05 core, totally takes 67722 clock cycles to compute SHA once. 8-bit micro-controllers are common and widely used for smart cards nowadays [37]. The evaluations are made on the 8-bit processors varied at different clock speeds of 3MHz, 8MHz, and 16MHz; and SHA is chosen as the hash function in the client puzzle. For example, if a smart card equipped with an 8-bit processor running at clock speed of 3MHz, it takes 22.6 ms to compute SHA once. Thus, if the difficulty of the puzzle is 2^5, it takes up to 0.7 seconds for solving the client puzzle on this smart card. Table 3 shows the evaluation time required for solving variant client puzzles on smart cards at different cycle rates.

5 Comparisons

To further examine the proposed defense mechanism, we compare it with Kim-Lee-Yoo's scheme in Table 4. There are six and $n+2$ additional hash operations be appended to Ser and U_i, respectively, where n is the number of hash operations that should be performed to solve the puzzle. Moreover, one exclusive-or operation is needed for Ser and U_i, respectively. The computation costs are low to defeat impersonation and server resources exhaustion attacks as well as to achieve mutual authentication and session key establishment. The loadings of a server are very heavy in the client/server model, to protect the server against computation exhaustion attacks, it is necessary to minimize the computation utility rates. The proposed defense mechanism not only resists to computation exhaustion attacks, but also makes sure that all clients have a fair resource access chance when connecting to the server; therefore, the two hash operations cannot be avoided in Step 2 of Figure 1.

To defeat server memory exhaustion attacks, The amount of information stored in the server should be reduced as possible as it could be. Therefore, some related parameters must be transmitted over the communication links. This is a tradeoff. In order to prevent memory exhaustion attacks, the larger message size could not be avoided.

Furthermore, it is impossible to design a mutual authentication less than one round-trip transmission, for the security reasons described in Section 4.1.4, the proposed mechanism adopts two round-trip transmissions to achieved mutual authentication and session key establishment.

6 Conclusions

We propose a defense mechanism to Kim-Lee-Yoo's ID-based password authentication scheme, which is vulnerable to resource exhaustion attacks and impersonation attacks. Our scheme not only remedies the weaknesses shown

Table 4: Comparisons of our scheme and Kim-Lee-Yoo's scheme

Item	Kim-Lee-Yoo	Ours
Resistance to		
On-line password guessing attacks	Yes	Yes
Off-line password guessing attacks	Yes	Yes
Message replay attacks	Yes	Yes
Impersonation attacks	No	Yes
Server resources exhaustion attacks	No	Yes
Opportunists	–	Yes
Mutual authentication	No	Yes
Session key establishment	No	Yes
Number of hash operations		
In U_i site	0	$n+2^*$
In Ser site	1	6
Number of exclusive-or operations		
In U_i site	0	1
In Ser site	0	1
Number of modular multiplication operations		
In U_i site	3	3
In Ser site	1	1
Number of modular exponentiation operations		
In U_i site	2	2
In Ser site	2	2
Number of transmissions	3	4
Messages size	small	large

$^*n$ can be adjusted artificially by Ser according to its loading.

in the previous sections, but also introduces mutual authentication and session key establishment to Kim-Lee-Yoo's scheme. The proposed scheme greatly improves the robustness of Kim-Lee-Yoo's scheme through inexpensive hash and exclusive-or operations.

References

[1] *ITU-T Recommendation X.509: Information Technology - Open Systems Interconnection - The Directory: Authentication Framework*, 2005.

[2] Afrand Agah and Sajal K. Das. Preventing dos attacks in wireless sensor networks: A repeated game theory approach. *International Journal of*

Network Security, 5(2):145–153, 2007.

[3] T. Aura, P. Nikander, and J. Leiwo. DoS-resistant authentication with client puzzles. In *The 8th International Workshop on Security Protocols*, LNCS 2133, pages 170–177, Cambridge, UK, April 2001. Springer-Verlag.

[4] V. Bocan. Threshold puzzles: The evolution of DoS-resistant authentication. *Transactions on Automatic Control and Computer Science*, 49(63):171–176, 2004.

[5] C. K. Chan and L. M. Cheng. Cryptanalysis of a remote user authentication scheme using smart cards. *IEEE Transaction on Consumer Electronics*, 46(4):992–993, 2000.

[6] Y. Chen, S. Das, P. Dhar, A. E. Saddik, and A. Nayak. Detecting and preventing IP-spoofed distributed DoS attacks. *International Journal of Network Security*, 7(1):69–80, 2008.

[7] H. Y. Chien, J. K. Jan, and Y. M. Tseng. An efficient and practical solution to remote authentication: Smart card. *Computers and Security*, 21(4):372–375, 2002.

[8] M. S. Hwang and L. H. Li. A new remote user authentication scheme using smart cards. *IEEE Transactions on Consumer Electronics*, 46(1):28–30, 2000.

[9] A. Juels and J. Brainard. Client puzzles: A cryptographic defense against connection depletion attacks. In *Proceedings of NDSS '99 (Networks and Distributed Security Systems)*, pages 151–165, Feb. 1999.

[10] G. Keating. Performance analysis of AES candidates on the 6805 CPU core. In *Second AES Candidate Conference(AES2)*, 1999.

[11] H. S. Kim, S. W. Lee, and K. Y. Yoo. Id-based password authentication scheme using smart cards and fingerprints. *ACM SIGOPS Operating Systems Review*, 37(4):32–41, Oct. 2003.

[12] W. C. Ku, S. T. Chang, and M. H. Chiang. Further cryptanalysis of fingerprint-based remote user authentication scheme using smartcards. *Electronics Letters*, 41(5):240–241, 2005.

[13] Manoj Kumar. An enhanced remote user authentication scheme with smart card. *International Journal of Network Security*, 10(3):175–184, 2010.

[14] Jinu Kurian and Kamil Sarac. Defending network-based services against denial of service attacks. *International Journal of Network Security*, 9(2):186–200, 2009.

[15] Guenther Lackner, Udo Payer, and Peter Teufl. Combating wireless lan mac-layer address spoofing with fingerprinting methods. *International Journal of Network Security*, 9(2):164–172, 2009.

[16] L. Lamport. Password authentication with insecure communication. *Communications of ACM*, 24:77–772, 1981.

[17] V. Laurens, A. E. Saddik, and A. Nayak. Requirements for client puzzles to defeat the denial of service and the distributed denial of service attacks. *International Arab Journal of Information Technology*, 3(4):326–333, 2006.

[18] C. C. Lee, M. S. Hwang, and W. P. Yang. A flexible remote user authentication scheme using smart cards. *ACM Operating Systems Review*, 36(3):46–52, 2002.

[19] C. C. Lee, M. S. Hwang, and W. P. Yang. Extension of authentication protocol for GSM. *IEE Proceedings-Communication*, 150(2):91–95, Apr 2003.

[20] C. H. Lee, M. S. Hwang, and W. P. Yang. Enhanced privacy and authentication for the global system of mobile communications. *Wireless Networks*, 5:231–243, July 1999.

[21] J. K. Lee, S. R. Ryu, and K. Y. Yoo. Fingerprint-based remote user authentication scheme using smart cards. *Electronic Letters*, 38(12):554–555, 2002.

[22] Chun-Ta Li and Yen-Ping Chu. Cryptanalysis of threshold password authentication against guessing attacks in ad hoc networks. *International Journal of Network Security*, 8(2):166–168, 2009.

[23] C. H. Lin and Y. Y. Lai. A flexible biometrics remote user authentication scheme. *Computer Standards and Interfaces*, 27(1):19–23, 2004.

[24] Y. Liu, W. Gao, H. Yao, and X. Yu. Elliptic curve cryptography based wireless authentication protocol. *International Journal of Network Security*, 5(3), 2007.

[25] Kumar V. Mangipudi and Rajendra S. Katti. A hash-based strong password authentication protocol with user anonymity. *International Journal of Network Security*, 2(3):205–209, 2006.

[26] R. C. Merkle. Secure communication over an insecure channel. *Comm. ACM*, 21(4):294–299, Apr. 1978.

[27] Yi-Jun He Moon-Chuen Lee and Zhaole Chen. Towards improving an algebraic marking scheme for tracing ddos attacks. *International Journal of Network Security*, 9(3):204–213, 2009.

[28] B. Nordin. Match-on-card technology white paper. Technical report. Precise Biometrics, 2004.

[29] T. Peng, C. Leckie, and K. Ramamohanarao. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Computing Surveys*, 39(1), 2007.

[30] Wei Ren. Pulsing roq ddos attacking and defense scheme in mobile ad hoc networks. *International Journal of Network Security*, 4(2), 2007.

[31] R. Richardson. 2007 CSI computer crime and security survey. In *http://i.cmpnet.com/v2.gocsi.com/pdf/CSISurvey2007.pdf*.

[32] M. Scott. Cryptanalysis of an id-based password authentication scheme using smart cards and fingerprints. *ACM SIGOPS Operating Systems Review*, 38(2):73–75, 2004.

[33] A. Shamir. Identity based cryptosystems & signature schemes. In *Advances in Cryptology, CRYPTO'84*, pages 47–53, Lecture Notes in Computer Science, 1984.

[34] J. J. Shen, C. W. Lin, and M. S. Hwang. A modified remote user authentication scheme using smart cards. *IEEE Transactions on Consumer Electronics*, 49(2):414–416, 2003.

[35] H. M. Sun. An efficient remote user authentication scheme using smart cards. *IEEE Transactions on Consumer Electronics*, 46(4):958–961, 2000.

[36] C. S. Tsai, C. C. Lee, and M. S. Hwang. Password authentication schemes: Current status and key issues. *International Journal of Network Security*, 3(2):101–115, 2006.

[37] Y. M. Tseng, T. Y. Wu, and J. D. Wu. A pairing-based user authentication scheme for wireless clients with smart cards. *INFORMATICA*, 19(2):285–302, 2008.

[38] Bin Wang and Zheng-Quan Li. A forward-secure user authentication scheme with smart cards. *International Journal of Network Security*, 3(2):116–119, 2006.

[39] Ren-Chiun Wang and Chou-Chen Yang. Cryptanalysis of two improved password authentication schemes using smart cards. *International Journal of Network Security*, 3(3):283–285, 2006.

[40] S. Wang, Z. Cao, and H. Bao. Efficient certificateless authentication and key agreement (CL-AK) for grid computing. *International Journal of Network Security*, 7(3):342–347, 2008.

[41] X. Wang and M. K. Reiter. Defending against denial-of-service attacks with puzzle auctions. In *In Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 78–92, Berkeley, CA, May 2003.

[42] Shuhua Wu and Yuefei Zhu. Proof of forward security for password-based authenticated key exchange. *International Journal of Network Security*, 7(3):335–341, 2008.

[43] Zeng Yong, Ma Jianfeng, and SangJae Moon. An improvement on a three-party password-based key exchange protocol using weil pairing. *International Journal of Network Security*, 10(3):188–193, 2010.