Formal Analysis and Systematic Construction of Two-factor Authentication Scheme

Guomin Yang[1], Duncan S. Wong[1], Huaxiong Wang[2], and Xiaotie Deng[1]

[1] Department of Computer Science
City University of Hong Kong
Hong Kong, China
{csyanggm,duncan,deng}@cs.cityu.edu.hk
[2] Department of Computing
Macquarie University
Sydney, Australia
hwang@ics.mq.edu.au

Abstract. One of the most commonly used two-factor authentication mechanisms is based on smart card and user's password. Throughout the years, there have been many schemes proposed, but most of them have already been found flawed due to the lack of formal security analysis. On the cryptanalysis of this type of schemes, in this paper, we further review two recently proposed schemes and show that their security claims are invalid. To address the current issue, we propose a new and simplified property set and a formal adversarial model for analyzing the security of this type of schemes. We believe that the property set and the adversarial model themselves are of independent interest.
We then propose a new scheme and a generic construction framework. In particular, we show that a secure password based key exchange protocol can be transformed efficiently to a smartcard and password based two-factor authentication scheme provided that there exist pseudorandom functions and collision-resistant hash functions.

Keywords: Authentication, Password, Smart Card, Guessing Attack

1 Introduction

Password authentication with smart card is one of the most convenient and effective *two-factor authentication* mechanisms. This technology has been widely deployed for various kinds of authentication applications which include remote host login, online banking, access control of restricted vaults, activation of security devices, and many more. Although some smart-card-based password authentication systems have already been in use, many of them are having issues on both security and performance aspects.

A smart-card-based password authentication scheme involves a server S and a client A with identity ID_A. At the very beginning, S issues a smart card to A with the smart card being personalized with respect to ID_A and some initial password. This phase is called the *registration phase* and is carried out only once for each client in some secure way. After obtaining the smart card, A can access S in the *login-and-authentication phase*. This phase can be carried out as many times as needed. However, in this phase, there could have various kinds of passive and active adversaries in the communication channel between A and S. They can eavesdrop messages and even modify, remove or insert messages into the channel. The security goal of the scheme in this phase is to ensure mutual authentication between A and S in the presence of these adversaries. In particular, it is required to both *have* A's smart card and *know* A's password in order to carry out the smart-card-based password authentication scheme successfully with server S, that is, maintaining *two-factor security* that the scheme should provide. There are also some other desirable properties people would like the scheme to possess. We will discuss these properties shortly.

Besides registration phase and login-and-authentication phase, A may want to *change password* from time to time. Conventionally, this activity usually has S involved and requires S to maintain a database for storing the passwords or some derived values of the passwords of its clients. In this paper, we promote the idea of letting A change the password at will without interacting with or notifying S (while ensuring two-factor security), and also eliminating any password database at the server side. Below are the reasons. Most of the current systems require the server to maintain a database for the passwords or derived values of the passwords of its clients. The derived values of the passwords can be obtained by using a password-based KDF (key derivation function) which takes a password and a known random value called salt and apply a hash function or a block cipher for a number of iterations (standardized in PKCS [16] and IETF [6]). However, this approach not only introduces scalability problem to the server but also makes the systems suffer from disastrous loss when the server is compromised and the password database is stolen by adversaries.

Current systems also suffer from other potential security vulnerabilities. One prominent issue is security against *offline guessing attack* (also known as offline dictionary attack). The purpose of offline guessing attack is to compromise a client's password through exhaustive search of all possible password values. In the context of a password-based cryptosystem, we consider that passwords are short in the sense that they are human memorizable. In other words, we assume that the password space is so small that an adversary is able to enumerate all possible values in the space within some reasonable amount of time. For example, most of the ATM deployments use PINs (personal identification numbers) of only 4 or 6 digits long, that means the password space has no more than one million possible values. As another example, in many web-based email implementations, the email server authenticates a client by sending a random challenge in clear and having the client send back a response which is the hashed value of the password and the challenge. Although the password is not transmitted in clear, an adversary can launch offline guessing attack for finding out the password after eavesdropping the challenge and the response.

A stronger notion of security against offline guessing attack is to require that compromising a client's smart card does not help the adversary launch offline guessing attack against the client's password. In practice, the adversary may steal the smart card and extract all the information stored in it through reverse engineering. This notion is reminiscent of password-based key exchange protocols [12]. The difference is that for password-based key exchange protocols, the focus is on preventing adversaries from getting any useful information about the password mainly from the transcripts of protocol runs, while for smart-card-based password authentication schemes, in addition to thwarting related attacks against password-based key exchange protocols, we also need to protect the password from being known even after the client's smart card is compromised.

1.1 Our Results

In this paper, we contribute on three areas:

1. We propose a new and simplified set of desirable security properties for a smart-card-based password authentication scheme. We also propose an adversarial model for formal analysis of the security of this type of schemes.
2. We show that two recently proposed schemes are insecure with respect to their claimed security properties which have also been captured in our desirable property set.
3. We propose a generic construction framework and show that a secure smart-card-based password authentication scheme can be constructed by *transforming* a proven secure password based key exchange protocol (under some appropriate security model which will be specified) provided that there exist pseudorandom unctions and collision-resistant hash functions. The transformation is very efficient. It essentially adds in only two additional hash evaluations and one pseudorandom function evaluation.

Paper Organization: In Sec. 2, we review some previous work. In Sec. 3, we propose a set of desirable properties and an adversarial model for smart-card-based password authentication schemes and compare them with a recently proposed property set. In Sec. 4, we review a scheme proposed by Liao et al. in [19] and show that the scheme is insecure against their security claims. In Sec. 5, we propose a new scheme and a generic construction framework that can be used to build efficient schemes of this type. We conclude the paper in Sec. 6. The security analysis against another scheme proposed by Yoon and Yoo [29] is given in Appendix A.

2 Related Work

Since Lamport [17] introduced a remote user authentication scheme in 1981, there have been many smart-card-based password authentication schemes proposed (some recent ones are [5,28,29,19]). These schemes are aimed for different security goals and properties, and noticeably, there is no common set of desirable security properties that has been widely adopted for the construction of this type of schemes. Although the construction and security analysis of this type of schemes have a long history, recently proposed schemes are still having various security weaknesses being overlooked, and we can find many of these schemes broken shortly after they were first proposed [10,11,26,23,29].

In [19], Liao et al. made an attempt to consolidate a set of desirable properties for smart-card-based password authentication schemes. Although we will see later that many of the properties in their proposed set are redundant, the set indeed comes close to capture the *two-factor security* of smart-card-based password authentication schemes. One of the properties requires that offline guessing attack should not be feasible even after a client's smart card is stolen and compromised. Their attempt is to enforce the two-factor security in the sense that the client must have both the smart card and the password to gain access to the server. As noted in [19], it does not seem to have any protocol available that satisfies all of their properties. However, we find that Scott has a working paper [22] which has a smart-card-based password "key exchange" protocol that can be extended to provide explicit mutual authentication and satisfy all the properties given in [19]. However, Scott's construction requires some special non-supersingular elliptic curves to defend against offline guessing attack and may not be suitable for systems where Tate pairing operations are considered to be too expensive or infeasible to implement. In the next section, we will define a refined set of properties that not only captures all the security properties specified in [19], but also removes redundancies from the properties. We also point out that the property set proposed in [19] does not capture the exact two-factor security that a smart-card-based password authentication scheme should provide.

Liao et al. also proposed a new scheme in [19] and claimed that the scheme satisfies all the security properties specified in their paper. However, in this paper, we show that the scheme does not satisfy some of their properties. In particular, we describe an offline guessing attack to compromise a client's password once after the client's smart card is compromised. We also show that their scheme may not provide server authentication even their proposed improvement is in place. Details are given in Sec. 4. In [29], Yoon and Yoo proposed another scheme and claimed its security against offline guessing attack and some other attacks. However, we find that their scheme is actually vulnerable to the offline guessing attack which is similar to the one described in Sec. 4. Details of our attack against Yoon-Yoo scheme are given in Appendix A.

In [24], Shoup and Rubin proposed an extension of the Bellare and Rogaway's model [2] for three-party key distribution protocols where smart cards are used to store the long-term keys. In their system, each user is considered to have a smart card storing the long-term key of the user. When using the system, the smart card is attached to a hosting machine and the machine is communicating with another machine which has another user's smart card attached. A key distribution server is called in for distributing a session key to the two machines in a similar sense to that of [2]. The smart cards are used to prevent adversaries from getting the long-term keys, for example, by compromising the hosting machines. However, it is assumed in their model that once an adversary has compromised the smart card of a user, the adversary has compromised the user. Therefore, the model still falls in the category of one-factor security. That is, once the smart card is compromised, the adversary can impersonate the corresponding user. In practice, we can do better than this when using smart cards. In this paper, instead of storing the plain long-term keys in the smart card, we "mask" the long-term key using some password-based mechanism and store the masked key in the smart card instead. We then propose a scheme which does not allow an adversary to impersonate a user unless both the smart card (i.e. the masked long-term key) and the password of the user are known to the adversary. In addition, knowing the masked long-term key does not help the adversary get the password and vice versa. This should be the core feature of two-factor authentication, where in our studies, the user has to show that he *has* the smart card and he *knows* the password. Compromising the smart card the user *has* does not help obtain the password the user *knows* and vice versa.

3 Security Properties

As introduced in Sec. 1, there are two phases and one activity in a smart-card-based password authentication system. The two phases are *registration phase* and *login-and-authentication phase*, and the activity is called *password-changing activity*.

In the registration phase, an authenticated and secure environment is assumed to present, and all parties are assumed to be honest and perform exactly according to the scheme specification. In the real world, this stage may require the client who is requesting for registration to show up in person at the server's office and then have a smart card initialized and personalized using a secure and isolated machine. The smart card is finally issued to the client at the end of the stage. After this phase is completed, the client is said to be *registered*. In the login-and-authentication phase, the communication channel between server S and a registered client A is no longer considered to be secure. Both passive and active adversaries are present and their objective is to compromise the scheme's primary security goal, that is, mutual authentication between S and A. During the password-changing activity, a registered client A change the password and updates the smart card accordingly. A may need to interact with S for changing the password. However, this is undesirable due to the scalability issue described before and the concern of user friendliness. It will be better if A can change the password freely without the help or notification of S.

Desirable Properties. We propose a list of desirable properties for evaluating a smart-card-based password authentication scheme.

1. (*Client Authentication*) The server is sure that the communicating party is indeed the registered client that claims to be at the end of the protocol.

2. (*Server Authentication*) The client is sure that the communicating party is indeed the server S at the end of the protocol.
3. (*Server Knows No Password*) S should not get any information of a registered client's password.
4. (*Freedom of Password Change*) A client's password can freely be changed by the client without any interaction with server S. S can be totally unaware of the change of the client's password.
5. (*Short Password*) The password space is small enough so that the underlying adversary can enumerate all the possible values of the space in a reasonable amount of time. We consider a human-memorizable password to be a value in this password space.

Adversarial Model. Consider an adversary $\mathcal{A}$ who has the full control of the communication channel between the server S and any of the *registered* clients. $\mathcal{A}$ can obtain all the messages transmitted between the server S and a registered client; $\mathcal{A}$ can also modify or block those transmitted messages; and $\mathcal{A}$ can even make up fake messages and send to any entity in the system while claiming that the messages are from another entity in the system (i.e. impersonation). To simulate insider attack [3], we also allow $\mathcal{A}$ to know the passwords and all information stored in the smart cards of all the clients except those of a client who is under attack from $\mathcal{A}$. In addition, we also allow $\mathcal{A}$ to *either* compromise the password *or* the smart card of the client under attack, but not both. However, $\mathcal{A}$ is not allowed to compromise S.

Discussions. In the list of desirable properties above, the first two constitute the primary security requirement of a secure smart-card-based password authentication scheme, that is, mutual authentication between the server S and a registered client A. The third property helps solve the scalability problem at the server side. In addition, since there is no information about clients' passwords stored at the server side, the property also alleviate damage entailed to the clients if the server is compromised. The fourth property will help improve the user friendliness of the system as there is no additional communication overhead when a client changes her password. One should note that property 3 does not imply property 4. It is always possible to construct a scheme such that the server does not have any information of a client's password while the client cannot change the password either once after registration. The fifth property means that we always consider that if an adversary launches an attack which needs to search through the password space (for example, an offline guessing attack), the adversary can always evaluate all the possible values in the space within the running time of the adversary. To prevent an adversary from launching offline guessing attack, we therefore need to make sure that the scheme is not going to leak any information useful about the client's password to the adversary, even though the password is considered to be weak and low-entropy.

Note that the adversary can always launch the *online* guessing attack. In this attack, the adversary impersonates one of the communicating parties and sends messages based on a trial password chosen by the adversary. If the trial password is guessed incorrectly, the other party will reject the connection. If so, the adversary will try another password and repeat the steps until a trial password leads to an acceptance of connection. Online guessing attack is easy to defend against in practice. Conventionally, a system can set up a policy mandating that if the password of a client is entered incorrectly for three times in a row, then the client will be blocked and refused to connect any further. This policy works well in practice and can effectively defend against online guessing attack if the attack only allows the adversary to try one password in each impersonation attack. However, we should also note that a secure scheme should not allow the adversary to test two passwords or more in each of this impersonation attack.

Regarding the adversarial model above, obviously, if both the client's smart card and password are compromised by an adversary, the client will not have any information advantage over the adversary. Hence there is no way to prevent the adversary from masquerading as the client in this case. However, in cases where only the client's smart card or password is compromised, but not both, we should make sure that the adversary cannot compromise the mutual authentication between the client and the server S. This security property is called *two-factor security*. One remark is that in this paper, we do not make assumption on the existence of any special security features supported by the smart cards. Instead, we simply consider a smart card to be a memory card with an embedded microcontroller for performing required operations specified in a scheme. As Kocher et al. [15] and Messerges et al. [20] pointed out, all existing smart cards cannot prevent the information stored in them from being extracted, for example, by monitoring their power consumption. Some other reverse engineering techniques are also available for extracting information from smart cards. Hence we put aside any special security features that could be supported by a smart card in this paper, and simply assume that once a smart card is stolen by an adversary, all information stored in the smart card will then be known to the adversary.

Another remark about the smart card is that after a smart card is compromised, the card holder may not be aware of it in practice. For example, the adversary simply *steals* all the information stored in the card using a fraudulent card reader without being noticed by the card holder rather than stealing the card itself. Hence in our model, we still let the card holder to use the card and for a secure protocol, we require that the adversary cannot compromise mutual authentication even when this case happens, that is, when the smart card of the client under attack is compromised but not the password.

3.1 Comparison with the Ten Properties Proposed by Liao et al.

In [19], Liao et al. proposed a set of requirements for smart-card-based password authentication schemes. In their set, there are ten requirements and all of them are claimed to be independent. In the following, we review the ten requirements and show that all of them have already been included in the original set of properties and the adversarial model we defined above. This also shows that their ten requirements contain a certain number of redundancies.

R1. *The passwords or verification tables are not stored inside the computer.*

This is included in our property 3 (Server Knows No Password). In property

3, we require that there should have no information of a registered client's password be obtained by the server. For requirement **R1**, however, there may still be some password information stored at the server side. For example, a server may keep the information of the range of the passwords of the registered clients. The server may also keep information about which clients are sharing a common password.

R2. *The password can be chosen and changed freely by the owner.*
This is included in our property 4 (Freedom of Password Change).

R3. *The password cannot be revealed by the administrator of the server.*
This is included in our property 3 (Server Knows No Password).

R4. *The passwords are not transmitted in plaintext on network.*
This is also included in our property 3 (Server Knows No Password).

R5. *No one can impersonate a legal user to login the server.*
This is included in our property 1 (Client Authentication).

R6. *The scheme must resist the replay attack, guessing attack, modification attack, and stolen-verifier attack.*
The sole purpose of these attacks is either to compromise client authentication or server authentication. Also note that besides these attacks, there are many other attacks [3] that may be able to sabotage the original security goals of a scheme. As our adversarial model captures the capabilities of the adversary for launching all these mentioned attacks, if a scheme satisfies our property 1 (Client Authentication) and 2 (Server Authentication), then the scheme must satisfy requirement **R6**.

R7. *The length of a password must be appropriate for memorization.*
This is included in our property 5 (Short Password).

R8. *The scheme must be efficient and practical.*
This is a performance issue. Furthermore, the efficiency of a scheme depends on the implementation environment; while practicality is related to the target applications. A scheme can be efficient and practical in some scenarios while may not be the case in some other scenarios. It does not seem to be measurable without referring to any other comparable schemes. We therefore leave the performance evaluation until a concrete construction is given and some implementation environment is specified.

R9. *The scheme can achieve mutual authentication. Not only can server verify the legal users, but users can verify the legal server.*
This is included in our property 1 (Client Authentication) and 2 (Server Authentication).

R10. *The password cannot be broken by guessing attack even if the smart card is lost.*
This is already captured by our adversarial model. However, we should also note that this set of ten properties is not strong enough to capture the exact security requirements of two-factor security. Note that we can construct a protocol satisfying all these ten properties but whose security relies solely on the smart card and is independent of what password the client chooses. For example, we store a plain long-term symmetric key in the smart card and use the key to conduct mutual authentication with the server. The password

is not used at all. Obviously, it will be insecure once the client's smart card is compromised. However, property **R10** is still satisfied. In other words, the exact security goal of the notion of two-factor security is not captured.
In order to capture this notion, in our model, we boost the capability of the adversary $\mathcal{A}$ by allowing $\mathcal{A}$ to either compromise the password or the smart card of the client under attack, but not both. To see why our model is stronger, first we note that property **R10** is captured as $\mathcal{A}$ is allowed to compromise the smart card of the client under attack. Second, a protocol whose security relies only on the smart card will no longer be secure in the new model. Third, it also captures another aspect of two-factor security, that is, knowing the password alone should not allow the adversary to compromise the mutual authentication between the client and the server.

4 Offline Guessing Attack Against a Smart-card-based Password Authentication Scheme

On the construction of smart-card-based password authentication schemes, although there have been quite a number of schemes proposed, most of the old schemes have already been found to be insecure. For some recently proposed ones, in this section, we show that the scheme proposed by Liao et al. [19] is insecure with respect to their claimed security properties. In Appendix A, we show that another scheme recently proposed by Yoon and Yoo [29] is also insecure.

Let p be a 1024-bit prime. Let g be a generator of $\mathbb{Z}_p^*$. The server S chooses a long secret key x. In [19], the authors did not specify the length of x, however, in order to prevent brute-force search, we assume x to be a random string of at least 160 bits long. Let h be a hash function (e.g. SHA-256) and $a \| b$ denote the concatenation of a and b.

Registration phase: Server S issues a smart card to a client A as follows.

1. A arbitrarily chooses a *unique* identity ID_A and password PW_A. PW_A is a short password that is appropriate for memorization. A then calculates $h(PW_A)$ and sends $(ID_A, h(PW_A))$ to S.
2. S calculates $B = g^{h(x \| ID_A)+h(PW_A)} \bmod p$ and issues A a smart card which has (ID_A, B, p, g) in it.

Login-and-authentication phase[3]: A attaches the smart card to an input device and keys in ID_A and PW_A. Afterwards, S and A (the smart card) carry out the following steps.

1. A sends a login request to S.
2. On receiving the login request, S calculates $B'' = g^{h(x \| ID_A)R} \bmod p$ where $R \in \mathbb{Z}_p^*$ is a random number, and sends $h(B'')$ and R to A.

[3] In [19], the authors divide the *login-and-authentication phase* into two parts: *login phase* and *authentication phase*. However, we put them together as other protocols do in the literature.

3. Upon receiving the message from S, A calculates $B' = (Bg^{-h(PW_A)})^R \bmod p$ and checks if $h(B'') = h(B')$. If they are not equal, S is rejected. Otherwise, A calculates $C = h(T \| B')$ where T is a timestamp, and sends (ID_A, C, T) to S.
4. Let T' be the time when S receives (ID_A, C, T). S validates A using the following steps.
   (a) S checks if ID_A is in the correct format[4]. If it is incorrect, S rejects.
   (b) Otherwise, S compares T with T'. If $T' - T \geq \Delta T$, S rejects, where ΔT is the legal time interval for transmission delay.
   (c) S then computes $C' = h(T \| B'')$ and checks if $C = C'$. If they are not equal, S rejects. Otherwise, S accepts.

Password-changing activity: To change a password, the client A carries out the following steps.

1. Select a new password PW'_A.
2. Compute $Y = g^{h(PW'_A)} \bmod p$.
3. Compute $\beta = Bg^{-h(PW_A)}Y \bmod p$, where PW_A is the original password of A.
4. Replace B with β in the smart card.

In [19], it is claimed that the scheme above satisfies all of their ten properties reviewed in Sec. 3.1. However, in the following, we show that the scheme is vulnerable to offline guessing attack once the client's smart card is compromised. It is also vulnerable to impersonation attack. That is, the scheme does not satisfy requirements **R6**, **R9** and **R10** and potentially requirement **R5** as well.

4.1 Offline Guessing Attack

Malicious user offline guessing attack. In [19], the scheme above is claimed to be secure against offline guessing attack even if the client's smart card is compromised. In the following, we show that this is not true. Suppose client A's smart card is compromised by an adversary $\mathcal{A}$. $\mathcal{A}$ can carry out the offline guessing attack as follows.

1. $\mathcal{A}$ impersonates A and sends a login request to S.
2. S calculates $B'' = g^{h(x \| ID_A)R} \bmod p$ and sends back $(h(B''), R)$.
3. $\mathcal{A}$ then carries out offline guessing attack by checking if
$$h(B'') = h((Bg^{-h(PW^*_A)})^R \bmod p)$$
   for each trial password PW^*_A (i.e. $\mathcal{A}$'s guess of PW_A).

[4] In [19], the format of identity ID_A was not given. We hereby assume that there is some pre-defined format for all the identities used in their system.

Note that after $\mathcal{A}$ receives the message from S in step (2), $\mathcal{A}$ does not need to provide any response to S and therefore S does not know whether the communicating party is launching an attack or simply the message sent by S is lost during transmission. This makes the guessing attack described above difficult to detect. Also notice that if $\mathcal{A}$ possesses a past communication transcript Trans between A and S, $\mathcal{A}$ can perform the offline guessing attack directly without interacting with S. In case the current password is not the old one involved in Trans, by performing this attack, $\mathcal{A}$ will retrieve the current password instead of the old one.

Malicious server offline guessing attack. The scheme is also vulnerable to malicious server offline guessing attack. Since in the registration phase, user A will pass the value $h(PW_A)$ to the server. It is obvious that the server can retrieve PW_A easily by performing offline guessing attack. Although the authors of [19] provided some arguments for this issue, we still believe this is undesirable. It also violates property 3 in our property set.

4.2 Impersonation Attack

The scheme cannot provide server authentication either. An adversary can impersonate the server S by simply replaying a previously intercepted message $(h(B''), R)$. In [19], the authors have already realized this and suggested to thwart this replay attack by having S add another timestamp without providing a concrete construction. However, if this timestamp is added improperly, reflection attack will work and an adversary can impersonate a client without knowing the client's password. Consider the additional timestamp T' is added as below in the login-and-authentication phase.

...

2. On receiving the login request, S calculates $B'' = g^{h(x\|ID_A)R} \bmod p$ where $R \in \mathbb{Z}_p^*$ is a random number, and sends $C' = h(T'\|B'')$, T' and R to A.

...

If T' is placed in C' as shown above, an adversary $\mathcal{A}$ can impersonate A by simply sending ID_A, C', T' back to S in step 3.

5 A New Scheme and Its Extensions

In this section, we propose a new smart-card-based password authentication scheme which is proven secure and also satisfies all the properties we described in Sec. 3. This new scheme can also be extended to a generic construction framework which allows us to convert most of the proven secure password-based key exchange protocols [12] to smart-card-based versions. The significance of this framework is that we can now be able to design provably secure smart-card-based password authentication scheme in a systematic way and make use of those proven secure password-based key exchange protocols as the main building blocks. The schemes constructed in this framework will also have session keys established that are generally useful for target applications.

Before describing our scheme, we first review some characteristics (i.e. the two-factor security) of a smart-card-based password authentication scheme. In the two-factor security, we do not consider the case that both the password and the smart card are compromised, but we need to consider the other three cases: (1) neither the password nor the smart card is compromised; (2) the password is leaked while the smart card remains secure; (3) the smart card is compromised but the password remains secure. It is obvious that security under case (1) can be ensured if security under either case (2) or case (3) is guaranteed. And our goal is to achieve security under both case (2) and case (3). In other words, compromising one factor should not affect the other. However, the linkage between the two factors may not be obvious, below is an example.

First we relax our security requirements so that the server S can have a password table and we assume that S will never be compromised. S also maintains a long-term k-bit (high-entropy) cryptographic key x for some security parameter k, while each client possesses a password PW and a smart card issued by S. Suppose there is some pseudorandom function [7] f_x which takes a k-bit secret key x, produces a k-bit output and allows inputs of up to k bits. Each client with a unique k-bit identity ID also has a long-term cryptographic key $K = f_x(ID)$ issued by S, and the client can retrieve K from the smart card using the password PW. For example, a value $B = K \oplus h(PW)$ is stored in the smart card of the client where h is a cryptographic hash function. When the client connects to S, the client carries out a secure password-based authentication protocol π_1 with S using PW, and then runs a symmetric-key based authentication protocol π_2 with S using K. The authentication is said to be successful if both π_1 and π_2 succeed. We assume that π_1 is secure against offline guessing attacks (e.g. [9]) and π_2 is a conventional symmetric-key based authentication protocol proven secure under some appropriate model (e.g. [4]).

At the first glance, the scheme seems providing two-factor security. However, this is not true. Suppose case (3) happens (i.e. the smart card is compromised), then the effective key space of K for π_2 shrinks to the size of the password space. This is because the adversary can enumerate all the possible passwords and find out all the corresponding K's by calculating $K = B \oplus h(PW)$. Note that now π_2 may be vulnerable to offline guessing attack. The reason is that π_2 is designed for cryptographic keys, there is no security guarantee on what will happen if the key space is significantly reduced. And once K is compromised, PW is also known. As a result, the password may be compromised not because of π_1 but due to the break of the smart card.

Although the scheme above fails and is also inefficient (as it consists of two protocols), it gives us some inspiration on designing a secure smart-card based password authentication scheme. The above example fails because π_2 may not be secure against offline guessing attack if the key space is small. One simple solution is to replace π_2 by π_1 but use K as the "password". And in this case, running protocol π_1 using PW can be omitted. Intuitively, the security of case (3) can be ensured since now K plays essentially the same role as PW. And the security of case (2) is somehow "upgraded" from "password" level to "cryptographic key" level. In the following, we will analyze the security of this scheme under a password model defined by Halevi and Krawczyk [9].

5.1 Password-based Authentication: Model and Protocol

In this section, we review the security model for password-based authentication protocols and a one-way password authentication protocol. Both of them are due to Halevi and Krawczyk [9]. Here we only consider user authentication and the user only possesses a password. The definition of security in this model essentially says that the "best" possible strategy for the adversary to compromise user authentication is online guessing attack, which can be thwarted in practice by limiting the number of consecutive authentication failures that each user is allowed.

The security model is defined by a game, the players in the game are client A, server S and adversary $\mathcal{A}$. The game is parameterized by a security parameter k and a public dictionary $\mathcal{D}$ which contains a set of possible passwords. The game proceeds as follows.

Set-up phase: S chooses its cryptographic keys and publishes its public keys. A uniformly[5] picks a password PW from $\mathcal{D}$ and gives it to S while keeping it secret from $\mathcal{A}$. $\mathcal{A}$ can also register additional clients with S at any time (before, during, or after the set-up phase) by picking any pair of identity A' and password PW' (provided that $A' \neq A$ and $PW' \in \mathcal{D}$) and giving PW' to S.

Game running phase: $\mathcal{A}$ has full control over all the clients it created, as well as the communication between A and S. That is, A and S can only communicate through $\mathcal{A}$, every message that A and S send goes to $\mathcal{A}$, and every message they receive comes from $\mathcal{A}$. $\mathcal{A}$ may choose to forward messages faithfully, but may also insert, drop, or modify (including replay of previous messages) messages. $\mathcal{A}$ can send special "prompt" messages to the parties at any time, causing them to start new authentication sessions (in particular, several simultaneous sessions by the same or different parties are possible). Each session will have a unique session identifier. This game is run until $\mathcal{A}$ decides to halt.

Outputs of parties: For capturing the security requirements, A and S will record events related to the security of the authentication by giving some special outputs. A outputs a pair (S, sid) whenever it authenticates itself to server S under session identifier *sid*. S outputs (A, sid) whenever a successful authentication by A is completed during session *sid*. If an attempt to authenticate (alleged) A in session *sid* fails, S outputs $(A, sid, \perp)$. The latter is needed so that the "number of failed authentication attempts" can be counted, after which the password must expire.

[5] Security analysis can be extended to the non-uniform case if Definition 1 is modified accordingly.

Security: Some terminologies are defined as follows.

– An authentication protocol π is said to be *syntactically correct* if whenever all the messages between A and S in session sid are passed unchanged, then S and A output (A, sid) and (S, sid), respectively.
– An event in which S outputs (A, sid) but A has never output a pair (S, sid) is called a *successful impersonation* (here we assume that A sends the last message and outputs (A, sid) only after the last message is sent and S outputs (S, sid) only after receiving the last message sent by A). An event in which S outputs $(A, sid, \perp)$ is called an *authentication failure.* An event in which S outputs a pair (A', sid) after already outputting some other pair (A'', sid) in the past is called a *successful replay.* (Here A' and A'' are arbitrary clients, and sid is the same in both pairs.) All the events above are referred to as *active impersonation attempts.*
– An (ℓ, m)-run of the game is a run with at most m active impersonation attempts, and in which the legitimate user outputs at most ℓ pairs of (S, sid). This can be treated as the situation that the legitimate client should change the password once every ℓ times of login attempt. An (ℓ, m)-win for the adversary is an (ℓ, m)-run which contains at least one successful impersonation or replay event.

Definition 1. *Let $\epsilon(\cdot,\cdot,\cdot)$ be a positive real function and let π be a syntactically correct authentication protocol. We say that π ensures one-way password authentication up to ϵ, if for every feasible adversary $\mathcal{A}$, every finite dictionary $\mathcal{D}$, every security parameter k, and every ℓ and m, we have*

$$|\Pr[(\ell, m)\text{-}\mathit{win\ for}\ \mathcal{A}] \leq \frac{m}{|\mathcal{D}|} + \epsilon(k, \ell, m)$$

where the probability is taken over the random coins of S, A and $\mathcal{A}$ in an execution of the game with dictionary $\mathcal{D}$ and security parameter k.

The target of our protocol is to have $\epsilon(k, \ell, m)$ be a negligible function in k when ℓ and m are appropriately chosen.

In [9], Halevi and Krawczyk presented an Encrypted Challenge-Response one-way password authentication protocol and showed its security under the model described above. They also used a technique called "Public Password" which acts as a "hand-held certificate" for certifying the server's public key. In our application, since each user has a smart card on hand and the smart card is assumed to be issued by the server, we therefore can simply assume that the server's public key is written into the non-volatile memory of the smart card when the smart card is initialized. Therefore, in the remaining part of the paper, we assume that the server's public key is always available to the user.

Halevi-Krawczyk One-way Password Authentication Protocol

1. A sends a request with (A, sid) to S.
2. Upon receipt of the request, S picks a random value n and sends (S, sid, n) to A.

3. On receiving the reply, A computes $c = \mathrm{ENC}_{\mathrm{PK}_S}(PW, A, S, sid, n)$, where PK_S denotes the public key of S. A then sends (A, sid, c) to S.
4. Upon receipt of (A, sid, c), S decrypts c and do the verification.

In practice, one usually generates the session identifier sid as a pair (sid_1, sid_2) where sid_1 is a value chosen by A and different (with very high probability) from all such values chosen by A in A's other sessions with server S. Similarly, sid_2 is chosen by S with an analogous uniqueness property. And sid_1 (sid_2) is included in the first (second) message of the protocol.

It is proved by Halevi and Krawczyk that the successful probability of the adversary to break the protocol above is bounded by $\frac{m}{|\mathcal{D}|} + m \cdot \ell \cdot \epsilon'(k)$ where $\epsilon'(k)$ is a negligible function in k if the public key encryption scheme in use is semantically secure against chosen ciphertext attack [21].

5.2 A Password-based Authenticated Key Exchange protocol

In [1], Bellare et al. developed a framework for the design and analysis of conventional authenticated key exchange protocols. The notion of an "authenticator" was defined as a protocol that "compiles" any protocol in an eavesdropper-only world (in which authenticity is directly ensured) to a protocol secure (with respect to authenticity) against active adversaries. Later, Canetti and Krawczyk [4] extended this work and proved that this compilation will not affect the security of a key exchange protocol. In other words, if a key exchange protocol is secure (with respect to session key security) in the eavesdropper-only world, the compiled protocol remains secure against active adversaries. And they also pointed out (Sec. 5.4 of the full paper) that the Halevi-Krawczyk one-way password authentication protocol reviewed above is actually a password-based authenticator. Below is another authenticator based on digital signature.

A Signature Based Authenticator [1].

$$\begin{aligned}
&P_i \rightarrow P_j : msg, sid \\
&P_i \leftarrow P_j : msg, sid,\ N_j \\
&P_i \rightarrow P_j : msg, sid,\ \mathrm{SIG}_{P_i}(msg, sid, N_j, P_j)
\end{aligned}$$

The figure above illustrates a signature based authenticator for party P_i to send a message m to party P_j in an authenticated way. Let k be a security parameter, $N_j \in_R \{0,1\}^k$ be a random challenge and SIG_{P_i} be the signature generation function of P_i. The signature scheme is assumed to be secure against chosen message attack [8].

Armed with these tools, we can directly build a mutual authentication and key exchange protocol from the basic Diffie-Hellman protocol which is proved to provide session key security in the eavesdropper-only world [4]. We will use the Halevi-Krawczyk password authenticator to achieve user authentication and the signature based authenticator to achieve server authentication.

Let G be a subgroup of prime order q of a multiplicative group $\mathbb{Z}_p^*$. Let g be a generator of G. Let $(\mathrm{PK}_S, \mathrm{SK}_S)$ denote a public/private key pair of the server S. User A has a password PW_A which is shared with S.

A Password-based Authenticated Key Exchange / Mutual Authentication Protocol

$$\begin{aligned}
&A \rightarrow S : A, sid, g^x \\
&A \leftarrow S : S, sid, g^y, \mathrm{SIG}_{\mathrm{SK}_S}(S, A, sid, g^x, g^y) \\
&A \rightarrow S : A, sid, c = \mathrm{ENC}_{\mathrm{PK}_S}(PW_A, A, S, sid, g^x, g^y)
\end{aligned}$$

The session key is calculated as $\sigma = g^{xy}$. Here the values of g^x and g^y also play the role of random challenges. Hence if session key is not needed and only the mutual authentication is required, one can replace g^x and g^y with two random numbers. The final protocol will then be a password-based mutual authentication protocol. In [9], the authors used an encryption based authenticator to achieve server authentication. However, recent research shows that this authenticator is insecure against session-corruption attack unless some additional restrictions on the security model of [4] are made [25].

In the protocol above, server authentication is guaranteed by the signature-based authenticator. Readers can refer to [4] for details. Hence in the following, our security analysis will focus on user authentication only.

5.3 Upgrading the Password-based Protocol by using a Smart Card

In this section, we "upgrade" the password-based mutual authentication protocol above by using a smart card.

Let p, G, g, q be the group parameters defined as above. Besides a public/private key pair $(\mathrm{PK}_S, \mathrm{SK}_S)$, the server S also maintains a long-term secret x which is a random string of length k. Let $H : \{0,1\}^* \rightarrow \{0,1\}^k$ denote a collision resistant hash function and $PRF_K : \{0,1\}^k \rightarrow \{0,1\}^k$ a pseudorandom function keyed by K.

Registration phase: Server S issues a client A as follows.

1. A arbitrarily chooses a unique identity ID_A and sends it to S.
2. S calculates $B = PRF_x(H(ID_A)) \oplus H(PW_0)$ where PW_0 is the initial password (e.g. a default such as a string of all '0').
3. S issues A a smart card which contains $\mathrm{PK}_S, ID_A, B, p, g, q$. In practice, we can have all these parameters except B be "burned" in the read-only memory of the smart card when the smart card is manufactured.
4. On receiving the smart card, A changes the password immediately by performing the password-changing activity (described below).

Login-and-authentication phase: A attaches the smart card to an input device, and then keys in ID_A and PW_A. The smart card checks if the identity is equal to the value stored in it. If not, the smart card will refuse carrying out any further operation. Otherwise, the smart card retrieves the value $LPW = B \oplus H(PW_A)$. A (actually performed by the client's smart card) and S then use LPW as the password to perform the password-based mutual authentication protocol described in Sec. 5.2 above.

Password-changing activity: If A wants to change the password, A carries out the following steps.

1. Select a new password PW'_A.
2. Compute $Z = B \oplus H(PW_A) \oplus H(PW'_A)$, where PW_A is the old password.
3. Replace B with Z in the smart card.

Remarks: The "password" used in the login-and-authentication phase is LPW, instead of the real password PW_A. Note that S can compute the value of LPW once after receiving ID_A. Hence it does not violate property 3 (Server Knows No Password) in Sec. 3. From the password-changing activity above, it is obvious that the scheme also satisfies property 4 (Freedom of Password Change).

Case (2) Security. If the smart card is not compromised (even when the password is leaked), our proposed scheme deduces the success probability of the adversary to a negligible level by assuming that pseudo-random functions exist.

Lemma 1. *If the smart card is not compromised, and $PRF_K(\cdot)$ is replaced by a random function, then the adversary has only a negligible success probability in the Halevi-Krawczyk model.*

The proof of Lemma. 1 is straightforward, by applying the result of Halevi and Krawczyk [9]. The success probability of the adversary is bounded by $\frac{m}{2^k} + m \cdot \ell \cdot \epsilon'(k)$ which is negligible.

Theorem 1. *If the smart card is not compromised, and $PRF_K(\cdot)$ is a pseudo-random function, then the adversary has only a negligible success probability in the Halevi-Krawczyk model.*

Proof. The proof is by contradiction. Suppose there exists a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ which has a non-negligible probability to win in the Halevi-Krawczyk game. We construct another PPT adversary $\mathcal{B}$ which wins the pseudorandom function game with a non-negligible probability.

Adversary $\mathcal{B}$ is given access to an oracle $\mathcal{O}$ which is either $PRF_K(\cdot)$ (with probability 1/2) or a random function $RAND$ (with probability 1/2). $\mathcal{B}$ can adaptively query an arbitrarily chosen string $x \in \{0,1\}^k$ to $\mathcal{O}$ and get the output which is either $PRF_K(x)$ or a random string uniformly selected from $\{0,1\}^k$. After performing polynomially many queries, $\mathcal{B}$ finally makes a decision whether the oracle $\mathcal{O}$ is the function $PRF_K(\cdot)$ or a random function $RAND$. $\mathcal{B}$ wins the game if the decision is correct.

To do this, $\mathcal{B}$ runs a simulation of the Halevi-Krawczyk game and plays the role of the server S. Suppose the public/private key pair generated by $\mathcal{B}$ is $(\mathrm{PK}_S, \mathrm{SK}_S)$. $\mathcal{B}$ invokes adversary $\mathcal{A}$ in the game. In the registration phase for client A with identity ID_A, $\mathcal{B}$ calculates $h = H(ID_A)$ and enquiries oracle $\mathcal{O}$ with input h. $\mathcal{B}$ sets the return value from $\mathcal{O}$ to LPW and passes it to A (in the form of $LPW \oplus H(PW_0)$ where PW_0 is a k-bit default value.

$\mathcal{B}$ then runs the game until $\mathcal{A}$ halts, thus $\mathcal{B}$ is in polynomial time. If $\mathcal{A}$ wins in the game, $\mathcal{B}$ makes a decision that the oracle is $PRF_K(\cdot)$. Otherwise $\mathcal{B}$ chooses the random function $RAND$ as its decision.

$$
\begin{aligned}
&\Pr[\mathcal{B} \text{ wins the game }] \\
&= \Pr[\mathcal{B} \text{ outputs } PRF_K | \mathcal{O} = PRF_K] \Pr[\mathcal{O} = PRF_K] \\
&\quad + \Pr[\mathcal{B} \text{ outputs } RAND | \mathcal{O} = RAND] \Pr[\mathcal{O} = RAND] \\
&= \frac{1}{2} \Pr[\mathcal{B} \text{ outputs } PRF_K | \mathcal{O} = PRF_K] + \frac{1}{2} \Pr[\mathcal{B} \text{ outputs } RAND | \mathcal{O} = RAND] \\
&= \frac{1}{2} \Pr[\mathcal{B} \text{ outputs } PRF_K | \mathcal{O} = PRF_K] + \frac{1}{2} (1 - \Pr[\mathcal{B} \text{ outputs } PRF_K | \mathcal{O} = RAND]) \\
&= \frac{1}{2} (\Pr[\mathcal{A} \text{ wins the game } | \mathcal{O} = PRF_K]) + \frac{1}{2} (1 - \Pr[\mathcal{A} \text{ wins the game } | \mathcal{O} = RAND]) \\
&= \frac{1}{2} + \frac{1}{2} (\mathcal{A} \text{ wins the game } | \mathcal{O} = PRF_K] - \Pr[\mathcal{A} \text{ wins the game } | \mathcal{O} = RAND])
\end{aligned}
$$

According to Lemma. 1, if $\mathcal{O}$ is a random function, then $\mathcal{A}$ has only a negligible probability to win. If $\mathcal{A}$'s winning probability becomes non-negligible when changing the random function to a pseudorandom function, $\mathcal{B}$ also gets a non-negligible probability to win the pseudorandom function game over random guess. □

Case (3) Security. If the smart card is compromised while the password remains secure, there is no security "upgrade" when compared with a password-based protocol. It is easy to see that if $PRF_K(\cdot)$ is replaced by a random function, then the protocol provides the same security as the password protocol. And by using the same approach as in the proof of Theorem. 1, we can show that our scheme provides almost the same security level (with at most a negligible gap) when compared with the password-based protocol.

5.4 A Generic Construction Framework for Smart-card-based Password Authentication Schemes

Up to this point, readers will realize that the upgrading technique described above can essentially be applied to any password-based authentication protocol rather than restricted to the one proposed in Sec. 5.2. In particular, we can even take a protocol which does not require either of the communicating parties to generate and share a public key in addition to sharing a password, and transform it to a smart-card-based scheme using our upgrading technique. The resulting scheme will then be secure under a model similar to the security model for the original password-based authentication protocol, but extended according to the discussions in Sec. 5.3.

For example, we may start with a provable secure and efficient password-based authentication (and key exchange) protocol, such as [14,13], then we "upgrade" it using the technique described in Sec. 5.3. Interestingly, both of the protocols in [14,13] are proven secure without random oracle. Our upgrading technique does not rely on random oracle either. The "upgraded" scheme for smart-card-based password authentication will then be secure with security statements similar to that of Theorem 1 (but now in the corresponding model of the original password-based authentication protocol) and also with respect to Case (2) as well as Case (3) Security.

Efficiency. The "upgrading" technique proposed in Sec. 5.3 is very efficient. During the login-and-authentication phase, the smart card only needs to carry out one pseudorandom function evaluation and two hashes in addition to the operations incurred by the underlying password-based protocol. The generic construction framework allows us to choose a password-based protocol which is efficient enough when implemented on smart cards.

A Practical Issue. In the description above, we consider the server S to maintain one single long-term secret x for communication with all the clients. As a result, the secrecy of x is utmost important because the security of the entire system essentially relies on the security of x. In practice, we can alleviate the damage caused to a system by using multiple values of x to partition the system, and in each partition, a randomly generated x is used by a disjoint set of clients. Each partition is to be handled by a distinct and independent server. Compromising one server will therefore only affect the security of the corresponding partition of clients rather than the entire system. Note that this partitioning method does not affect the fulfillment of any of the desirable properties for a secure smart-card based password authentication scheme proposed in Sec. 3. Another mechanism which can be used in conjunction with the mechanism above is to set each long-term secret x with a validity period. Usually, smart cards are used such that they are valid only for a period of time. Hence for a different period of time, a fresh long-term secret x can be used.

6 Conclusion

Smart-card-based password authentication is one of the most convenient means to provide authentication for the communication between a client and a server. In this paper, we defined a set of desirable properties for secure smart-card-based password authentication schemes. We provided evidence to support the need of each of the properties and discussed in detail on why they are important. We showed that a recently proposed scheme of this type due to Liao et al. [19] does not satisfy some of the properties and some of their security claims are incorrect. In particular, we showed that their protocol is vulnerable to offline guessing attack once the client's smart card is compromised. Also, we showed that their protocol may not provide server authentication even the protocol is modified according to the specification of their rectified protocol. Moreover, we showed that another scheme proposed by Yoon and Yoo recently is also vulnerable to our offline guessing attack. We further proposed a new protocol and showed that it satisfies all the desirable properties specified in this paper. Finally, we hope that the set of properties proposed in this paper can be used in the future for analyzing new protocols of this type. This is important for evaluating comparable protocols in terms of both performance and security, and will also be useful for separating the studies of security models and protocol design.

References

1. M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proc. 30th ACM Symp. on Theory of Computing*, pages 419–428. ACM, May 1998.
2. M. Bellare and P. Rogaway. Provably secure session key distribution – the three party case. In *Proc. 27th ACM Symp. on Theory of Computing*, pages 57–66, Las Vegas, 1995. ACM.
3. C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment.* Springer-Verlag, 2003.
4. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Proc. EUROCRYPT 2001*, pages 453–474. Springer-Verlag, 2001. LNCS 2045. Full paper available at `http://eprint.iacr.org/2001/040/`.
5. 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.
6. T. Dierks and C. Allen. *The TLS Protocol Version 1.0. IETF RFC 2246*, January 1999.
7. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):210–217, 1986.
8. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attack. *SIAM J. Computing*, 17(2):281–308, April 1988.
9. S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. *ACM Trans. Inf. Syst. Secur.*, 2(3):230–268, 1999.
10. M.-S. Hwang. Cryptanalysis of remote login authentication scheme. *Computer Communications*, 22(8):742–744, 1999.
11. M.-S. Hwang, C.-C. Lee, and Y.-L. Tang. An improvement of SPLICE/AS in WIDE against guessing attack. *Internat. J. Inform.*, 12(2):297–302, 2001.
12. IEEE. *P1363.2 / D23: Standard Specifications for Password-based Public Key Cryptographic Techniques*, March 2006. Available at `http://grouper.ieee.org/groups/1363/passwdPK/draft.html`.
13. S. Jiang and G. Gong. Password based key exchange with mutual authentication. In *11th International Workshop on Selected Areas in Cryptography (SAC 2004)*, pages 267–279. Springer-Verlag, 2005. LNCS 3357.
14. J. Katz, R. Ostrovsky, and M. Yung. Efficient and secure authenticated key exchange using weak passwords. *Journal of the ACM, to appear*, 2006.
15. P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Proc. CRYPTO 99*, pages 388–397. Springer-Verlag, 1999.
16. RSA Laboratories. *PKCS #5 v2.0: Password-Based Cryptography Standard*, 1999.
17. L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–771, November 1981.
18. C. C. Lee, C. H. Lin, and C. C. Chang. An improved low computation cost user authentication scheme for mobile communication. In *Proc. 19th Advanced Information Networking and Applications (IEEE AINA'05)*, pages 249–252, 2005.
19. I.-E. Liao, C.-C. Lee, and M.-S. Hwang. A password authentication scheme over insecure networks. *Journal of Computer and System Sciences*, 2005. To be published, currently available online at the journal's website under "Articles in Press".
20. T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*, 51(5):541–552, May 2002.

21. C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Proc. CRYPTO 91*, pages 433–444. Springer, 1992. LNCS 576.
22. M. Scott. Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Cryptology ePrint Archive, Report 2002/164 (Revised Date: 10 Dec 2004), 2002. `http://eprint.iacr.org/2002/164`.
23. M. Scott. Cryptanalysis of an id-based password authentication scheme using smart cards and fingerprints. *SIGOPS Oper. Syst. Rev.*, 38(2):73–75, 2004.
24. V. Shoup and A. Rubin. Session key distribution using smart cards. In *Proc. EUROCRYPT 96*, pages 321–331. Springer, 1996. LNCS 1070.
25. X. Tian and D. S. Wong. Session corruption attack and improvements on encryption based MT-authenticators. In *RSA Conference 2006, Cryptographers' Track (CT-RSA 2006)*, pages 34–51. Springer-Verlag, 2006. LNCS 3860.
26. B. Wang, J. H. Li, and Z. P. Tong. Cryptanalysis of an enhanced timestamp-based password authentication scheme. *Comput. Secur.*, 22(7):643–645, 2003.
27. S. T. Wu and B. C. Chieu. A note on a user friendly remote authentication scheme with smart cards. *IEICE Trans. Fund*, E87-A(8):2180–2181, 2004.
28. E. J. Yoon, E. K. Ryu, and K. Y. Yoo. Efficient remote user authentication scheme based on generalized elgamal signature scheme. *IEEE Transactions on Consumer Electronics*, 50(2):568–570, May 2004.
29. E.-J. Yoon and K.-Y. Yoo. New authentication scheme based on a one-way hash function and Diffie-Hellman key exchange. In *4th International Conference of Cryptology and Network Security (CANS 2005)*, pages 147–160. Springer-Verlag, 2005. LNCS 3810.

A The Security Analysis of Yoon-Yoo Smart-card-based Password Authentication Scheme

In [29], Yoon and Yoo presented several attacks against Wu-Chieu [27] and Lee-Lin-Chang [18] smart-card-based password authentication schemes. They also proposed a new scheme and claimed its security against offline guessing attack. In the following, we show that their scheme is acutally vulnerable to an offline guessing attack which is similar to the one described in Sec. 4. We also show that their scheme does not satisfy another requirement in our property set described in Sec. 3.

The Yoon-Yoo scheme has the same group setup as that of Liao et al.'s reviewed in Sec. 4. Let x be the long-term secret key of the server S. In the following, we review the registration phase and login-and-authentication phase of their scheme. The login-and-authentication phase also has a session key generated.

Registration phase: S issues a smart card to a client A as follows.

1. A arbitrarily chooses a unique identity ID_A and password PW_A. A then sends them to S.
2. S calculates $T = h(ID_A \| x) \bmod p$ and $B = T \oplus PW_A$. S then issues A a smart card which has (ID_A, B, p, g) in it.

Login-and-authentication phase: A attaches the smart card to an input device and keys in ID_A and PW_A. Afterwards, S and A (the smart card) carry out the following steps.

1. A extracts T by computing $T = B \oplus PW_A$, randomly picks $c \in \mathbb{Z}_p^*$, and computes $C_1 = g^c \bmod p$. A then sends (ID_A, C_1) to S.
2. S checks if ID_A is in the correct format[6]. If not, S rejects. Otherwise, S computes $T = h(ID_A \| x)$, randomly pick $s \in \mathbb{Z}_p^*$, and computes $sk = C_1^s \bmod p$, $C_2 = g^s \bmod p$ and $C_3 = h(ID_A \| T \| sk \| C_1)$. Then S sends (C_2, C_3) back.
3. A computes $sk' = C_2^c \bmod p$ and $C_3' = h(ID_A \| T \| sk' \| C_1)$. Then A checks if $C_3' = C_3$. If they are not equal, S is rejected. Otherwise, A computes $C_4 = h(ID_A \| T \| sk' \| C_2)$ and sends it to S.
4. Upon receiving C_4, S computes $C_4' = h(ID_A \| T \| sk \| C_2)$ and checks if $C_4 = C_4'$. If they are equal, S accepts. Otherwise, S rejects.

A.1 Offline Guessing Attack

In [29], it is claimed that the scheme is secure against offline guessing attack even if A's smart card is compromised. In the following, we show that this is not true. Suppose an adversary $\mathcal{A}$ has compromised A's smart card. The following attack can be carried out by $\mathcal{A}$ for finding out A's password PW_A.

1. $\mathcal{A}$ impersonates A by choosing a random value $c \in \mathbb{Z}_p^*$ and then sending (ID_A, C_1) to S, where $C_1 = g^c \bmod p$.
2. On receiving (ID_A, C_1), S chooses a random $s \in \mathbb{Z}_p^*$ and computes $sk = C_1^s \bmod p$, $C_2 = g^s \bmod p$ and $C_3 = h(ID_A \| T \| sk \| C_1)$. Then S sends (C_2, C_3) back.
3. $\mathcal{A}$ then carries out offline guessing attack by checking if $C_3 = h(ID_A \| B \oplus PW_A^* \| C_2^c \bmod p \| C_1)$ for each trial password PW_A^* (i.e. $\mathcal{A}$'s guess of PW_A).

A.2 Other Problems of the Scheme

Besides the offline guessing attack described above, the Yoon-Yoo scheme has some other problems. By referring to the list the desirable properties in our property set described in Sec. 3, we notice that the Yoon-Yoo scheme does not achieve property 3 (Server Knows No Password). This is because in the registration phase, the client A sends the chosen password to S directly. Also, the length of passwords is not specified in [29], and the value of B is calculated as $h(ID, x) \oplus PW$, which may be modified to $h(ID, x) \oplus h(PW)$.

[6] The format of identity ID_A was not given in [29]. We hereby assume that there is some pre-defined format for all the identities used in their system.