12-2009

A new approach for anonymous password authentication

Yanjiang YANG

Jianying ZHOU

Jian Weng
Singapore Management University, jianweng@smu.edu.sg

Feng BAO

DOI: https://doi.org/10.1109/ACSAC.2009.26

A New Approach for Anonymous Password Authentication

Conference Paper · December 2009

DOI: 10.1109/ACSAC.2009.26 · Source: DBLP

CITATIONS
13

READS
116

4 authors, including:

Yanjiang Yang
Agency for Science, Technology and Research (A*STAR)
72 PUBLICATIONS **1,269** CITATIONS
SEE PROFILE

Jian Weng
Jinan University (Guangzhou, China)
100 PUBLICATIONS **1,597** CITATIONS
SEE PROFILE

Feng Bao
Agency for Science, Technology and Research (A*STAR)
165 PUBLICATIONS **2,674** CITATIONS
SEE PROFILE

A New Approach for Anonymous Password Authentication

Yanjiang Yang, Jianying Zhou
Institute for Infocomm Research
Singapore 138632, Singapore
Email: {yyang,jyzhou}@i2r.a-star.edu.sg

Jian Weng
Jinan University, Guangzhou, China
& Singapore Management University
Email: cryptjweng@gmail.com

Feng Bao
Institute for Infocomm Research
Singapore 138632, Singapore
Email: baofeng@i2r.a-star.edu.sg

***Abstract*—Anonymous password authentication reinforces password authentication with the protection of user privacy. Considering the increasing concern of individual privacy nowadays, anonymous password authentication represents a promising privacy-preserving authentication primitive. However, anonymous password authentication in the standard setting has several inherent weaknesses, making its practicality questionable. In this paper, we propose a new and efficient approach for anonymous password authentication. Our approach assumes a different setting where users do not register their passwords to the server; rather, they use passwords to protect their authentication credentials. We present a concrete scheme, and get over a number of challenges in securing password-protected credentials against off-line guessing attacks. Our experimental results confirm that conventional anonymous password authentication does not scale well, while our new scheme demonstrates very good performance.**

***Keywords*-anonymous password authentication; guessing attack; unlinkability; scalability;**

I. Introduction

Inputting one's "user ID" and "password" has been the most common practice for authentication since the advent of computers, and is still gaining popularity. Every day, there are probably billions of instances of password usage in cyberspace. The reason for the wide employment of password authentication is straightforward: password authentication requires no dedicated devices, and a user only needs to memorize his password and then can authenticate anywhere, anytime. As users are becoming increasingly roaming nowadays, its independence of the supporting infrastructure makes password authentication even more essential.

However, password authentication has intrinsic weaknesses. In particular, passwords are short (to be memorizable), normally drawn from a relatively small space, thus they have a low entropy in nature, and are susceptible to brute-force guessing attacks. Guessing attacks can be *on-line* or *off-line*. In the on-line guessing attack, the attacker attempts to login to the (authentication) server in the name of the victim user by trying a different password each time until finding the correct one. In the off-line guessing attack, the attacker does not need to interact with the server; instead, it gleans the protocol transcript of a login session between a user and the server, and then checks all possible passwords against the login transcript to determine the actual one. On-line guessing attacks can be easily thwarted at the system level by limiting the number of repetitive unsuccessful login attempts made by a user. In contrast, off-line guessing attacks are notoriously harder to deal with, and they must be addressed at the protocol level.

User's activities in the digital world can be easily logged and profiled. Abuses of individual information may cause serious consequences to users, e.g., financial/credit losses. For this reason, users are becoming increasingly privacy-aware, reluctant to disclose individual information when accessing online services. However, password authentication in general does not protect user privacy. In the standard setting of password authentication, the server maintains a password file with each entry being of the form $\langle userID, passw \rangle$, where $userID$ is the user's identification, and $passw$ is either the user's password or a password-derived value. To login to the server, a user needs to provide his $userID$ to the server, who then uses the corresponding $passw$ to engage in the authentication protocol, where the two authenticate each other and/or establish a shared session key between them.

To meet the growing need of privacy protection, it is desirable to reinforce password authentication with the protection of user privacy. Recently, a few schemes for anonymous password authentication [30], [32], [33] have been proposed. In particular, anonymous password authentication promises *unlinkability*, i.e., the server should not be able to link user accesses, such that the logins from the same user cannot be recognized as such. However, anonymous password authentication in the standard setting (described above) has inherent weaknesses. Among others, anonymous password authentication needs to implement PIR (Private Information Retrieval), thus the computation cost upon the server is no better than $\mathcal{O}(N)$, where N is the total number of users registered to the server. This makes the server a bottleneck in large systems having a large number of users.

A. Our Contributions

In this paper, we propose a new approach for anonymous password authentication, solving the weaknesses in the standard setting[1]. In particular, our contributions are three-fold.

[1]To distinguish from our approach, wherever needed we will refer to anonymous password authentication in the standard setting as conventional anonymous password authentication.

First, we analyze the weaknesses of conventional anonymous password authentication. To make our analysis concrete, we present a generic construction for conventional anonymous password authentication that covers all the existing schemes, and we base our analysis on this generic construction. The first weakness is that server computation is no better than $\mathcal{O}(N)$. The second is that unlinkability can be achieved only if the server is passive. We also show that existing anonymous password authentication schemes may be subject to undetectable on-line guessing attacks [17], where the server does not realize that it is experiencing on-line guessing attacks.

Second, we propose a new approach for anonymous password authentication, to address the weaknesses in the standard setting. Notably, server computation in our approach is independent of the number of users in the system, thus breaking the bound of $\mathcal{O}(N)$ in the standard setting. Our approach assumes a different setting where users do not register their passwords to the server, and the server thus does not hold any password file. This attributes to the success of getting over the barrier of $\mathcal{O}(N)$. Another advantage resulting from the password-file-free server is that there is no concern of immediate exposure of all user passwords in case the server is compromised.

The main idea of our approach is as follows. The server issues to each user a credential to be used for authentication, and the users protect their credentials by passwords. Each time a user logins to the server, he recovers his credential using password, and demonstrates to the server his possession of a valid credential. A notable feature is that the password-protected credentials can be public, and no secure device (e.g., smartcard) is needed to store the credentials. This solves a main issue in PKI (Public Key Infrastructure), i.e., safe management of the long secrets.

Third, we experiment on the generic anonymous password authentication construction, and the results empirically confirm that conventional anonymous password authentication has limited scalability. We also implement a prototype of our proposed scheme, which demonstrates very good performance.

B. Organization

In Section II, we review the related work, followed by Section III, an overview of the main cryptographic primitives to be used. In Section IV, we analyze the weaknesses of conventional anonymous password authentication. Our new approach is presented in Section V. We report the implementation results in Section VI, and Section VII concludes the paper.

II. Related Work

A. Password Authentication

As mentioned earlier, a major challenge in password authentication is to counter against off-line guessing attacks. To achieve this objective, it has been proven that *public key operations*, e.g., exponentiations in a multiplicative group, are essential in designing secure password authentication protocols [20]. But public key operations are not equivalent to public key primitives such as public key encryption and digital signature. Depending on whether or not public key primitives are involved, two distinct password authentication approaches exist: *public-key-assisted approach*, and *password-only approach*.

The public-key-assisted approach enlists a combined use of password and public key primitives, such that the users use passwords while the server has a public/private key pair (for encryption or signature) at its disposal. Examples of public-key-assisted password authentication schemes include [5], [19], [20]. The employment of a public key primitive by the server on the one hand simplifies protocol design, while on the other hand entails the deployment of PKI for certification. In contrast, the password-only approach does not involve any public key primitive, thereby eliminating the reliance on PKI. The password-only approach, or password authenticated key exchange (PAKE), has been extensively studied in the literature, e.g., [3], [4], [6], [7], [24], [26].

For either public-key-assisted schemes or password-only schemes, they assume the standard setting, where the server holds a password file that contains all users' password information. A security concern is that compromise of the server immediately reveals all passwords. A natural solution is to deploy multiple servers to secret-share the passwords [18], [25]. Multi-server password authentication schemes, however, not only downgrade operational quality [29], but also cause inconvenience for users to update passwords. The smartcard based authentication schemes [22], [8] enforce two-factor authentication: a user's authentication credential is stored in smartcard, and the smartcard is protected by password. These two-factor authentication schemes do not require the server to keep a password file, offering a solution to the drawbacks of multi-server password authentication. Our proposed approach does not require any smartcard, while enjoying the advantage of password-file-free server.

B. Password-Enabled PKI

A prerequisite for the use of public key primitives is the safe storage of the private keys. In principle, smartcards can be used to store private keys. However, the use of smartcards is not convenient, as they need the supporting infrastructure (e.g, smartcard reader) to operate. To solve this problem, password-enabled PKI has been proposed [29]. The idea of password-enabled PKI is to enable the use of public key primitives, with the private keys being protected by passwords. There exist two general approaches to realize password-enabled PKI. The first is to store a user' private key on a trusted server, and when needed the owner retrieves the private key from the server after authenticating to the server using password [27], [31]. The second is a key split

approach: a private key is split into two parts; the owner holds a part generated from his password, and a trusted server holds the other; use of the private key requires the two to cooperate. A concern of both approaches is that the storage server must be honest, as it learns users' private keys.

The software smartcard technique [21] can be viewed as a special case of password-enabled PKI, without requiring the presence of a trusted server. The idea of software smartcard is encrypting a private key with password, and the encrypted private key does not need further protection. To be secure against off-line guessing attacks, the public key must not be publicly known. Otherwise, anyone can recover the password and in turn the private key, based on the relationship between the public key and the private key. However, this contradicts the main advantage of PKI that the public keys are public.

Our approach using password-protected credentials is quite similar to the software smartcard technique. The reason why off-line guessing attacks do not ruin the usage of our password-protected credentials is that credentials are to be used to the authentication server only, and this allows us to conceal the structure of the credentials from anyone other than the server. In contrast, private keys in PKI are assumed to be used universally, without any restriction. We thus believe that the software smartcard technique is unlikely to succeed in the general PKI setting.

C. Anonymous Password Authentication

Anonymous password authentication is a recent primitive, first proposed in [32]. The construction in [32] combines a password-only protocol with a PIR (Private Information Retrieval) protocol, where the former generates a shared key between the user and the server, and the latter achieves user privacy protection. Subsequently, new anonymous password authentication schemes were proposed in [30]. These new schemes also rely on PIR to preserve user privacy, but the PIR protocol they use is a trivial construction, i.e., the server passes a whole database to the user. The scheme proposed in [33] uses the trivial PIR solution as well. [2] considered three-party (i.e., user-gateway-server) anonymous password authentication, and the proposed protocol also uses PIR to attain user privacy. All these anonymous password authentication schemes assume the standard setting. As a matter of fact, we will show shortly that the use of PIR is essential in conventional anonymous password authentication. We find out that these existing anonymous password authentication schemes [2], [30], [32], [33] do not provide explicit authentication of the user to the server, which may lead to undetectable on-line guessing attacks in some applications.

D. Other Privacy-Preserving Authentication Primitives

There are a lot of privacy-preserving authentication techniques proposed in the literature, among which anonymous credential [11], [13] and group signature [1], [14] are two important primitives, aiming to achieve unlinkability among the whole user population. The techniques that are used to construct anonymous credentials and group signatures have some similarities. Compared to anonymous password authentication, they offer a higher level of security, as they use long secrets (i.e., credentials in anonymous credential and group signing keys in group signature). They thus also have the problem of safe management of long secrets. The credentials to be protected by passwords in our approach are precisely simplified anonymous credentials without anonymity revocation property. However, while our approach essentially uses long secrets for authentication, the security it offers actually depends on the strength of passwords with respect to on-line guessing attacks, thereby weaker than anonymous credential and group signature.

III. Cryptographic Primitives

For ease of understanding, we review the main cryptographic primitives to be used in our constructions.

Homomorphic Encryption. Homomorphic encryption is a public-key encryption scheme, $E(.)$, satisfying $E(m_1).E(m_2) = E(m_1 + m_2)$ for any m_1, m_2. The Paillier encryption [28] is a typical homomorphic encryption scheme. The Paillier homomorphic encryption works in a multiplicative group $Z^*_{n^2}$, where n is a RSA-type modulus. To distinguish from regular public key encryption, we use $\texttt{Hom\_Enc}(m)$ to denote the homeomorphic encryption of m, and $\texttt{Hom\_Dec}(c)$ the decryption of a ciphertext c.

Zero-knowledge Proof of Knowledge. A Zero-knowledge Proof of Knowledge protocol (we call it zero-knowledge proof for short) is a two-party three-round protocol, where a prover proves to a verifier the knowledge of a secret without disclosing any information on the secret. The three-round is "commit-challenge-response". To be specific, we show a simple zero-knowledge proof as an example, proving the knowledge of x with respect to y such that $y = g^x \pmod p$, where $p = 2q + 1$ (both p, q are primes), g is a generator of group Z^*_p:

- *Commit*: the prover chooses a random number $t \in Z_q$, and gives a *commitment* $r = g^t \pmod p$ to the verifier.
- *Challenge*: the verifier sends back a *challenge* c to the prover.
- *Response*: the prover computes and returns a *response* $s = t - cx \pmod q$ to the verifier. The verifier accepts as long as $g^s y^c = r \pmod p$ holds.

For simplicity, we denote the procedure by $PoK\{(\chi) : y = g^\chi\}$, which stands for "zero-knowledge Proof of Knowledge of a value χ such that $y = g^\chi$". The convention here is that Greek letters denote the items to be proved, while all other parameters are known to the verifier. Generalizing this basic protocol, more complex relations among elements within a group or across multiple groups can be proved, e.g., $PoK\{(\chi_1, \chi_2, ..., \chi_l) : y = g_1^{\chi_1} g_2^{\chi_2} ... g_l^{\chi_l}\}$, and $PoK\{(\chi) : y_1 = g_1^\chi \wedge y_2 = g_2^\chi\}$.

A zero-knowledge proof protocol can be made non-interactive by applying the Fiat-Shamir heuristic, where the prover himself generates the challenge by applying a collision-free hash function to the commitment. We denote $NPoK\{.\}$ the non-interactive version of $PoK\{.\}$. Furthermore, $NPoK\{.\}$ is a signature on a message m if the challenge is generated from the commitment together with m, which is denoted $NPoK\{.\}[m]$.

Pederson Commitment. A data commitment scheme allows a prover to submit a commitment to a verifier, and prove certain algebraic properties of the data committed by the commitment. A commitment scheme has two properties: *hiding* and *binding*. The hiding property refers to the ability of a commitment concealing the committed value from the verifier, and the binding property is the ability to prevent the prover from changing the committed value, once the commitment is released. The Pederson commitment [16] (on a message m) takes the form of $C_m = g_1^m g_2^r \pmod p$, where $p = 2q + 1$ is defined as above, $g_1, g_2 \in QR_p$ with QR_p denoting the subgroup of quadratic residues modulo p, and $r \in Z_q$ is a random number. The Pederson commitment scheme is unconditionally hiding but computationally binding.

CL Signature [12]. Camenisch and Lysyanskaya [12] proposed an interesting signature scheme, which allows a signer to sign a message, while without necessarily seeing the actual message; and allows a prover to prove the possession of a signature on a message to the verifier who again does not know the message. For simplicity, we call it CL signature. Specifically, the CL signature works as follows. Let $n = pq$ be a RSA-type modulus, and $a, b, c \in QR_n$ be random elements. The public key is then $pk = (n,\ a,\ b,\ c)$ and the private key is $sk = (p,\ q)$. Both Signing and Signature Verification can be interactive:

- *Signing*: to get a signature on a message m, the user sends C_m, a Pederson commitment on m, to the signer. The signer returns a signature (v, k, s) satisfying $v^k = a^m b^s c \pmod n$.
- *Signature Verification*: the user who has (v, k, s) proves the possession of the signature as follows: the user first sends to the verifier C'_m, another Pederson commitment on m. Then the user, by a set of zero-knowledge proofs, proves to the verifier that he knows (v, k, s), such that $v^k = a^m b^s c$ and C'_m is a commitment to m. To avoid delving into the details, we use $PoK\{(\upsilon, \kappa, \varsigma, \varpi) : \upsilon^\kappa = a^\varpi b^\varsigma c)\}$ to denote the set of zero-knowledge proofs proving the possession of the signature.

IV. Weaknesses of Anonymous Password Authentication

Recall that in the standard setting of password authentication, the server holds all users' password information in a password file, and uses the corresponding user's information to authenticate the user. In this section, we analyze the limitations of anonymous password authentication in the standard setting. To make our analysis concrete, we present a generic construction. We also show that existing anonymous password authentication schemes [30], [32], [33], [2] may be subject to undetectable on-line guessing attacks.

The tools we use in this generic construction are homomorphic encryption and PIR (Private Information Retrieval). PIR is a cryptographic primitive allowing a user to retrieve a string from a N-string database, without disclosing anything on the index of the retrieved string to the server(s) holding the database [10], [23]. A single-server PIR protocol (where the database is held by a single server) is aimed to achieve better communication performance than $\mathcal{O}(N)$, which occurs in the trivial PIR solution where the server passes the entire database to the user. For computation performance, in the single-server PIR the server has to "touch" every string so as to answer a request; thus the computation overhead upon the server is at least $\mathcal{O}(N)$.

A. Generic Construction

Let p, q be large primes and $p = 2q + 1$, $g \in QR_p$, and $h(.)$ be a cryptographic hash function. Suppose the password information contained in the password file is a list of user passwords, i.e., $pw_1, pw_2, \ldots, pw_N$, corresponding to users $\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_N$, respectively. The generic protocol between user $\mathcal{U}_i$ and Server $\mathcal{S}$ works as follows.

Step 1. $\mathcal{U}_i$ generates a public/private key pair for a homomorphic encryption scheme; picks a random $x \in Z_q$ and computes $X = g^x \pmod p$; computes $\mathsf{Hom\_Enc}(pw_i)$. Finally $\mathcal{U}_i$ sends $\mathsf{Hom\_Enc}(pw_i)$ and X to $\mathcal{S}$.

Step 2. Upon reception of the login request, $\mathcal{S}$ first picks a random $y \in Z_q$ and computes $Y = g^y \pmod p, \mathsf{AuthS} = h(Y, X)$; for $j = 1..N$, $\mathcal{S}$ chooses a random r_j, and computes $e_j = (\mathsf{Hom\_Enc}(pw_i).\mathsf{Hom\_Enc}(-pw_j))^{r_j}.\mathsf{Hom\_Enc}(Y) = \mathsf{Hom\_Enc}((pw_i - pw_j)r_j + Y)$. Finally, $\mathcal{S}$ constructs a temporary N-entry database $\mathcal{D} = \{D_j\}_{j=1..N}$, where $D_j = \langle e_j, \mathsf{AuthS}\rangle$.

Step 3. $\mathcal{U}_i$ engages in a PIR protocol with $\mathcal{S}$ to get $D_i = \langle e_i, \mathsf{AuthS}\rangle$. Then $\mathcal{U}_i$ computes $\mathsf{Hom\_Dec}(e_i) = Y$, and tests whether $h(Y, X) = \mathsf{AuthS}$. If the test passes, $\mathcal{U}_i$ computes a session key $sk = h(X, Y, Y^x)$. Otherwise, $\mathcal{U}_i$ aborts.

Step 4. $\mathcal{U}_i$ computes and sends $\mathsf{AuthU} = h(X, Y)$ to $\mathcal{S}$, who then tests whether $\mathsf{AuthU} = h(X, Y)$. If the test passes, $\mathcal{S}$ accepts and computes $sk = h(X, Y, X^y)$; otherwise, $\mathcal{S}$ aborts.

It is not hard to understand the correctness and the security of the protocol. Note that all the existing anonymous password authentication schemes [2], [30], [32], [33] can be viewed as special cases of this generic construction.

B. Undetectable On-Line Guessing Attacks

We first show that existing anonymous password authentication schemes [2], [30], [32], [33] may suffer from undetectable on-line guessing attacks [17], where the server is not aware of the presence of on-line guessing attacks.

We notice that all the existing anonymous password authentication schemes [2], [30], [32], [33] stop at Step 3, without Step 4 (which enables the explicit authentication of the user to the server). To be fair, this is not an issue from the key establishment point of view, because of the implicit authentication that the user is not able to compute the shared key unless he uses a valid password. However, without Step 4, they may succumb to undetectable on-line guessing attacks. To see this, there are two cases to be considered, depending on the usage of the shared session key in the subsequent communication between the user and the server:

- In many applications, the server simply needs to "push" data to the user, e.g., a user downloads data from a FTP server. In such a case, the session key is only needed to protect the channel from the server to the user. Undetectable on-line guessing attacks work in these applications.
- In some other applications, the shared key will be used by the user to interact with the server. In this case, undetectable on-line guessing attacks are avoided, because the server can learn in retrospection whether the user has established the correct key.

The advantage of our generic construction is that it eliminates undetectable on-line guessing attacks at the authentication stage, independent of the underlying applications.

C. Weaknesses

We now analyze the limitations of the generic construction. These weaknesses are inherent to conventional anonymous password authentication, making it questionable whether conventional anonymous password authentication is practically useful.

Weakness 1. Server Computation $\mathcal{O}(N)$: It is clear that the computation overhead upon the server is $\mathcal{O}(N)$, linear with the total number of users. This in principle causes the scalability problem in large systems having a large number of users. In fact, in the standard (single-server) setting, $\mathcal{O}(N)$ is the lower bound of server computation for anonymous password authentication that achieves unlinkability. The reason is that the server's computation has to involve all user passwords; otherwise, those "un-touched" entries by the server must not be the requesting user.

We can also show that anonymous password authentication has to implements PIR. In particular, PIR can be constructed from anonymous password authentication as follows. Associate a password with each string in the N-string database. To retrieve a string, the requesting user uses the corresponding password (note that the passwords are not necessarily secret) to engage in anonymous password authentication with the server. The server sends back to the user every string, whose associated password has been "touched" during anonymous password authentication. It is clear that if the user succeeds in password authentication, then he clearly already gets the requested string. This corroborates the fact that server computation in anonymous password authentication is no better than $\mathcal{O}(N)$, which is the lower bound for (single-server) PIR.

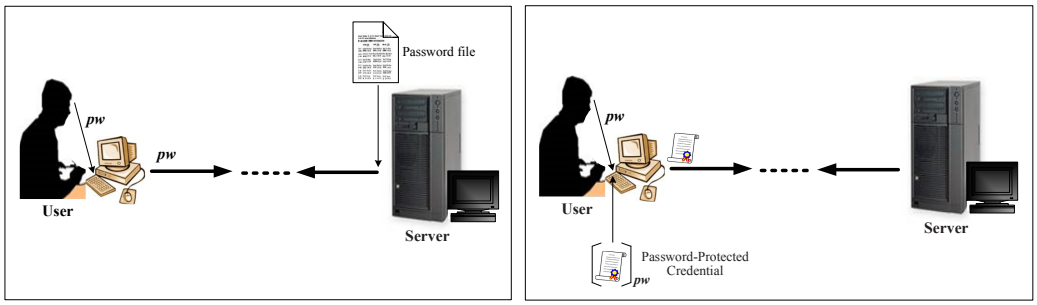
Weakness 2. Passive Server: Anonymous password authentication must be secure against undetectable on-line guessing attacks, and it should assume that the server is passive[2]; otherwise, unlinkability cannot be achieved. To see this, if the server is malicious: in Step 2 of the generic construction, for different passwords the server picks different y's in computing Y, AuthS, and e_j. Then in Step 4, from AuthU the server can determine which password the user uses, thereby breaking unlinkability. In turns out that this attack applies to any anonymous password authentication scheme, because for each password, the server can always use a distinct data in negotiating the shared key with the user (of course, there may exist countermeasures allowing the user to detect). Passive server is a quite strong assumption, and it may not be easy to find such a server in practice.

V. A New and Efficient Approach

We next present a new and efficient approach, solving the above weaknesses in conventional anonymous password authentication. Our approach assumes a different setting where users do not register their passwords to the server, who thus does not hold any password file. In particular, each user is issued a credential to be used for anonymous authentication, and the user protects his credential using password; each time to login to the server, the user first recovers his credential with password, and then uses the credential for authentication with the server. Figure 1 shows the conceptual difference between our approach and conventional anonymous password authentication.

A crucial feature of the password-protected credentials is that they can be made public, requiring no further protection. A user can store his password-protected credential in any portable devices, e.g., handphone, PDA, USB flash memory, or even in a public directory. With such portability of password-protected credentials, what a user essentially needs at the point of login is indeed his password (this is the reason why our approach still belongs to password authentication). However, it is not trivial to construct password-protect credentials preserving user privacy and secure against off-line guessing attacks. To show the challenges, we first briefly introduce the intuitions underlying our construction.

[2]A passive entity is honest, but tries to find out more useful information from the data it is supposed to get. In contrast, a malicious entity can behave arbitrarily in order to achieve its objective.



Conventional Anonymous Password Authentication

Our New Approach

Figure 1. Conceptual Comparison Between Conventional Approach and Ours

A. Overview

First Try. Since the credentials must protect user privacy, a natural choice is using blind signatures (e.g., [9]) as credentials. In particular, we adopt a "use-then-issue" strategy, i.e., the user uses a credential (a blind signature) for authentication, and at the end of each login, the server issues to the user a new blind signature to be used for next login. Since the server cannot link different blind signatures, it is expected that this can achieve unlinkability. Unfortunately, this is not true. To see this, not only the server but any outsiders can recover the password from a password-protected credential by off-line guessing attacks: anyone can use different passwords to "undo" a password-protected blind signature, and clearly only the right password generates a valid blind signature.

Lesson 1: *The failure of the first try is because blind signatures have known structure, publicly verifiable. Likewise, using passwords to protect other privacy-preserving primitives, e.g., anonymous credentials, has the same vulnerability. The lesson we learned is that the credentials to be protected by passwords should not be publicly verifiable.*

Second Try. It should be clear that credentials must be verifiable to the server, thus it is unavoidable for the server to recover passwords. This actually is not an issue in password authentication, since the server (in the standard setting) holds all users' password information. Our second try is thus to restrict verifiability of credentials to the server only. Continuing with the first try, one way to achieve restricted verifiability is that the server does not publicize the public key of the blind signature scheme, such that no one other than the server can verify blind signatures. This solves the issue of off-line guessing attacks by outsiders. However, since credentials are issued real time, users also need to check the validity of the credentials issued to them. Therefore, this method is not acceptable.

Another way to attain restricted verifiability is to encrypt the blind signatures with the server's public key before applying password protection (we assume that the server provides public key encryption). It is easy to see that off-line guessing attacks by outsiders are addressed. However, this method cannot achieve unlinkability with respect to the server. On the one hand, the user needs to surrender the encrypted blind signatures to the server for authentication purposes. On the other hand, by assumption the server also knows the corresponding password-protected credentials (i.e., encrypted blind signatures protected by password). By combining the two, the server clearly can recover the password, and thus link the encrypted blind signatures protected by the same password.

Lesson 2: *The reason for the failure is that users directly submit the items protected by passwords (i.e., encrypted blind signatures) to the server. The lesson is thus that the server should be prevented from seeing the items protected by passwords. This further means that users should not directly submit the credentials to the server.*

Third Try. Without direct submission of credentials, proving the possession of credentials by zero-knowledge proofs seems the only feasible choice. The CL signature [12] is a primitive that meets this need. However, even we have decided the strategy and the tool, there are still more to be considered. Recall that the CL signature on message m is (v, k, s) satisfying $v^k = a^m b^s c \pmod n$. Without loss of generality, let us define a user credential as (v, k, s) such that $v^k = a^{\mathcal{U}} b^s c \pmod n$, where $\mathcal{U}$ is the user's identity. To achieve restricted verifiability, (v, k, s) should be encrypted by the server's public key, as discussed above. Nevertheless, if the entire credential is encrypted, the user himself is unable to use the credential, because he needs to know v, k, s in order to perform zero-knowledge proofs.

Furthermore, partial encryption of some elements of the credential does not work either. Suppose s is encrypted[3]. Then, every time to use the credential, the user needs to pass the encrypted s to the server and then perform $PoK\{(\nu, \kappa, \mu) : \nu^\kappa a^\mu = b^s c\}$ (note that the zero-knowledge proofs are to prove $v^k a^{-\mathcal{U}} = b^s c \pmod n$). The server clearly can link different uses of the credential simply from s, regardless of the zero-knowledge proofs.

A remedy is to submit a distinct encrypted item for each

[3]In fact, it seems to us that encrypting v or k would make it harder for the user to perform zero-knowledge proofs.

use of the credential. Specifically, the user partitions s into two random shares s_1, s_2 such that $s_1 + s_2 = s$; encrypts s_1 using the server's public key, denoted as $E(s_1)$, and protects (v, k, s_2) using password, denoted as $[v, k, s_2]_{pw}$. The entire password-protected credential is thus $\langle E(s_1), [v, k, s_2]_{pw} \rangle$. Note that the encryption of s_1 successfully breaks the known structure of the credential, and no one other than the server can verify the validity of (v, k, s_2). Hence, off-line guessing attacks by outsiders are prevented. To use the credential for login, the user submits $C = b^{s_2} g^r$ (i.e., a Pederson commitment of s_2) together with $E(s_1)$ to the server. The server decrypts to get s_1 and computes $Cb^{s_1}c = b^{s_1+s_2} g^r c = b^s g^r c$, and then the user executes $PoK\{(\nu, \kappa, \mu, \gamma) : \nu^\kappa a^\mu g^\gamma = Cb^{s_1}c\}$, where the zero-knowledge proofs are to prove $v^k a^{-\mathcal{U}} g^r = b^s g^r c \pmod n$. At the end of the login, the server sends back s_1 to the user, who then restores s and re-partitions it into two new shares. In this way, the user is entitled to submit a distinct s_1 each time to the server. Does this solve the problem? Unfortunately, the server can still link uses of the credential. The situation is similar to that in the second try: the server can recover the password used to protect (v, k, s_2), and associate it with s_1; therefore, the server can link different s_1 associated with the same password, regardless of the zero-knowledge proofs.

Lesson 3: *The lesson we learned is that the user should never directly submit the data in storage (i.e., $E(s_1)$) to the server, whether it is an entire credential or a part thereof.*

Final Try. It is now clear that the encrypted item submitted to the server has to be different from that in storage. To achieve this, the user needs to manipulate $E(s_1)$ before submission, and render the resulting item in encryption distinct from s_1. The actual method we use is that the user further partitions s_2 into two random shares $s_2^{\langle 1 \rangle}, s_2^{\langle 2 \rangle}$ such that $s_2 = s_2^{\langle 1 \rangle} + s_2^{\langle 2 \rangle}$, and adds $s_2^{\langle 1 \rangle}$ to s_1 to generate $E(s_1') = E(s_1 + s_2^{\langle 1 \rangle})$. Here $s_2^{\langle 1 \rangle}$ serves as a blinding factor to blind s_1. Then the user submits $E(s_1')$ and $C = b^{s_2^{\langle 2 \rangle}} g^r$ to the server; the construction of zero-knowledge proofs remain unchanged. Since the manipulation is performed upon ciphertexts, the public key encryption possessed by the server should be homomorphic.

B. Details of the Scheme

Setup: The server $\mathcal{S}$ sets up $pk_{CL} = (n = p'q', a, b, c), sk_{CL} = (p', q')$ for the CL signature; and picks $g, h \in QR_n$. $\mathcal{S}$ also has a public/private key pair $(pk_{\mathcal{S}}, sk_{\mathcal{S}})$ for homomorphic encryption, and we use $\mathtt{Hom\_Enc}_{\mathcal{S}}(.), \mathtt{Hom\_Dec}_{\mathcal{S}}(.)$ to denote the encryption function under $pk_{\mathcal{S}}$, and the decryption function under $sk_{\mathcal{S}}$, respectively. $\mathcal{S}$ decides a cryptographic hash function $H(.)$, and a symmetric key encryption $enc(.)$. The public system parameters include $(pk_{CL}, g, h, pk_{\mathcal{S}}, H(.), enc(.))$.

Registration: Users need to register to the server in advance, getting a credential to be used for authentication. The server issues each user $\mathcal{U}_i$ a credential (v_i, k_i, s_i) using the CL signature scheme, satisfying $v_i^{k_i} = a^{\mathcal{U}_i} b^{s_i} c \pmod n$, where $\mathcal{U}_i$ is the user's identity. Upon reception of his credential, $\mathcal{U}_i$ partitions s_i into two random shares $s_{i,1}, s_{i,2}$ such that $s_i = s_{i,1} + s_{i,2}$; encrypts $s_{i,1}$ using the server's public key, i.e., $E_{i,1} = \mathtt{Hom\_Enc}_{\mathcal{S}}(s_{i,1})$; protects $(v_i, s_{i,2})$ using his password pw_i, i.e, $E_{i,2} = [v_i, s_{i,2}]_{pw_i}$, where $[.]_{pw}$ denotes, e.g., symmetric key encryption with a key derived from pw. Finally, $\mathcal{U}_i$ puts $\langle E_{i,1}, E_{i,2}, k_i \rangle$ to his preferred storage, e.g., handphone, USB flash memory, or a public directory.

Authentication: Suppose a user $\mathcal{U}$ already has his password-protected credential $\langle E_1 = \mathtt{Hom\_Enc}_{\mathcal{S}}(s_1), E_2 = [v, s_2]_{pw}, k \rangle$ available at the point of login. The authentication protocol between $\mathcal{U}$ and server $\mathcal{S}$ is as follows.

Step 1. $\mathcal{U}$ does the following computations.

(1). Recovers (v, s_2) by decrypting E_2 with his password pw.

(2) Partitions s_2 into two shares $s_2^{\langle 1 \rangle}, s_2^{\langle 2 \rangle}$, such that $s_2 = s_2^{\langle 1 \rangle} + s_2^{\langle 2 \rangle}$. Computes $E_1' = E_1.\mathtt{Hom\_Enc}_{\mathcal{S}}(s_2^{\langle 1 \rangle}) = \mathtt{Hom\_Enc}_{\mathcal{S}}(s_1 + s_2^{\langle 1 \rangle})$.

(3). Picks a random $r \in [0..n/4]$, computes $R = b^{s_2^{\langle 2 \rangle}} h^r \pmod n$ and $\Sigma(R) = NPoK\{(\varsigma, \gamma) : R = b^\varsigma h^\gamma\}$. Note that the zero-knowledge proof guarantees that R is well-formed.

(4). Computes $V = v^k a^{-\mathcal{U}} h^r \pmod n$; picks a random $x \in [0..n/4]$ and computes $X = g^x \pmod n, X^* = \mathtt{Hom\_Enc}_{\mathcal{S}}(X)$; constructs $\Sigma(V) = NPoK\{(\nu, \kappa, \mu, \gamma) : V = \nu^\kappa a^\mu h^\gamma\}[X^*, R]$.

(5). Finally, $\mathcal{U}$ sends $E_1', R, \Sigma(R), X^*, \Sigma(V)$ to $\mathcal{S}$ as a login request:

$$\mathcal{U} \longrightarrow \mathcal{S}\text{: } E_1', R, \Sigma(R), X^*, \Sigma(V)$$

Step 2. Upon reception of the login request, $\mathcal{S}$ does the following.

(1). Verifies the validity of $\Sigma(R)$, and aborts if not valid.

(2). Computes $\mathtt{Hom\_Dec}_{\mathcal{S}}(E_1') = s_1' = s_1 + s_2^{\langle 1 \rangle}$, and computes $V' = Rb^{s_1'}c = b^{s_1' + s_2^{\langle 2 \rangle}} h^r c = b^s h^r c \pmod n$. V' should be equal to V.

(3). With V', verifies the validity of $\Sigma(V)$, and aborts if not valid.

(4). Computes $\mathtt{Hom\_Dec}_{\mathcal{S}}(X^*) = X$. Chooses a random $y \in [0..n/4]$, computes $Y = g^y \pmod n$, and a temporary key $tk = H(X^y)$. Encrypts s_1' by symmetric key encryption as $enc_{tk}(s_1')$.

(5). Finally, computes the shared key $sk = H(X, Y, X^y)$, and returns $Y, enc_{tk}(s_1')$ to $\mathcal{U}$:

$$\mathcal{S} \longrightarrow \mathcal{U}\text{: } Y, enc_{tk}(s_1')$$

Step 3. $\mathcal{U}$ concludes the login process as follows.

(1). Computes $tk = H(Y^x)$, and decrypts $enc_{tk}(s_1')$ to get s_1'. Restores $s = s_1' + s_2^{\langle 2 \rangle}$, and checks whether $v^k = a^{\mathcal{U}} b^s c \pmod n$. Aborts if not valid.

(2). Computes a shared key $sk = H(X, Y, Y^x)$, and ends the authentication procedure. The password protected credential remains the same for the next login.

Note that the server sending back s'_1 to the user is to authenticate the server to the user, in that only the server can correctly decrypt E'_1, X^* to get s'_1, X, and in turn make the user accept. In fact, the correct computation of tk suffices authenticating the server, and sending back s'_1 is not absolutely necessary. Step 3 thus can be simplified such that the server authenticates to the user by tk, e.g., using MAC keyed by tk.

C. Security Analysis

Due to the limited space, the following definitions and analysis are informal.

1) Adversary Model: Either the server or outsiders could be the adversary in our system, with respect to different security objectives listed below. An outsider is defined to be anyone other than the server and the user who are engaging in the authentication protocol. The adversary is malicious, can do arbitrarily in order to violate the respective security objectives. In particular, we assume that the adversary acquires all users' password-protected credentials.

2) Security Objectives: We desire the following security objectives.

−*Authentication* [Outsiders]. The authentication objective requires that an outsider cannot impersonate a valid user to the server, and vice versa.

−*Secrecy of Session Key* [Outsiders]. It requires that an outsider should not learn the session key established between the server and the user.

−*Off-line Guessing Attacks* [Outsiders]. The resistance against off-line guessing attacks is with respect to the outsiders. It requires that an outsider should not be able to recover passwords used to protect credentials by off-line guessing attacks.

−*Unlinkability* [Server]. The user unlinkability is defined with respect to the server. It requires that the server cannot link different logins by the same user.

3) Security Analysis: We show the intuitions on how our scheme manages to satisfy the above security objectives.

Authentication. In our scheme, authentication of the user to the server is by $\Sigma(V)$. Without the knowledge of a valid CL signature (v, k, s), an outsider is not able to generate E'_1, which makes the server accept $\Sigma(V)$. This is the unforgeability of the CL signature [12]. We notice that the user's message contains no freshness data from the server, so replay is possible. But this is not a big issue, since using timestamp or an extra round of interaction suffices to solve the problem. Note also that $\Sigma(V)$ essentially asserts $v^k = a^{\mathcal{U}}c$, but again this is not an issue, since no one can compute such (v, k) without the help of the server; better yet, it is easy to attain the same strength as the original CL signature if the server uses a variant satisfying $v^k = a^m b_1^s b_2^{\hat{s}} c \pmod n$, with $(v, k, s, \hat{s})$ being the credential.

Authentication of the server to the user is by E'_1, X^*. An outsider clearly cannot decrypt E'_1, X^* to get correct s'_1 and X, and in turn $enc_{tk}(s'_1)$ that will be accepted by the user.

Secrecy of Session Key. Establishment of the shared session key sk is through the exchange of X and Y by DH key exchange protocol. Given the authentication property, the exchange of X, Y is authenticated, so an outsider cannot play main-in-the-middle. As such, an outsider observing communication between user and server can learn Y only. Thus the outsider is unable to compute g^{xy}. Note that even the outsider also learns X, he still cannot compute g^{xy} from X, Y, according to the (computational) DH assumption.

Off-line Guessing Attacks. Our scheme is a two-round protocol: user-requests-then-server-responds. Active adversarial behaviors such as impersonation do not gain an outsider more advantages in terms of off-line guessing attacks than passive interception of communication, because the server will not respond unless he is assured of the genuineness of the user. With this in mind, the data that may be helpful to a passive outsider in off-line guessing attacks include $\langle E_1, E_2\rangle, E'_1, \Sigma(V)$, and $[s'_1]_{tk}$. As we discussed in Section V-A, from E_1, E_2, an outsider cannot succeed in off-line guessing attacks, because the known structure of the CL signature has been broken. E'_1 and $\Sigma(V)$ do not help off-line guessing attacks either, as the zero-knowledge proofs do not reveal information on the items to be proved. $[s'_1]_{tk}$ clearly is of no use to the outsider for off-line guessing attacks, considering the secrecy of tk.

Unlinkability. As we have discussed, the user surrenders to the server a distinct s'_1 each time, and s'_1 is different from s_1, thus the server cannot link users from s'_1 and s_1. From $b^{s'_1}R = b^s h^r$, the server cannot link either, because r differs each time. It remains to examine whether $\Sigma(V)$ helps the server link users. It is important to know that by off-line guessing attacks, the server can learn all users' passwords, and in turn all credentials. For the analysis, let us consider a scenario, which is the most favorable to the server trying to break unlinkability: the server happens to use $\mathcal{U}$'s credential (v, k, s) to determine whether his counterpart indeed uses this credential. In the CL signature scheme, $\Sigma(V)$ consists of a set of zero-knowledge proofs, each of which is of the following form: to prove the knowledge of x, the user computes $C_x = g_1^x g_2^r$, a Pederson commitment on x, and then constructs $\Sigma(C_x) = NPoK\{(\chi, \gamma) : g_1^\chi g_2^\gamma = C_x\}$. Without loss of generality, it suffices to see whether the server (knowing x) can relate $C_x, \Sigma(C_x)$ to x.

We need to delve into the content of $\Sigma(C_x) = (C, S_1, S_2)$, which are computed as follows: $R = g_1^{x'} g_2^{r'}$, where x', r' are random numbers; then $C = h(R), S_1 = x' - xC, S_2 = r' - rC$. Here $h(.)$ is a collision-free hash function. Verification of $\Sigma(C_x)$ is to test whether $C = h(g_1^{S_1} g_2^{S_2} C_x^C)$. We claim that the server cannot associate $C_x, \Sigma(C_x)$ with x in the

information-theoretic sense. To see this, for any $\bar{x}$, there is a corresponding $\bar{r}$, satisfying $C_x = g_1^{\bar{x}} g_2^{\bar{r}}$. It can be easily verified that (C, S_1, S_2) is a valid zero-knowledge proof for any of such pair $\bar{x}, \bar{r}$.

VI. Implementation Results

To evaluate the performance of our proposed approach, we implemented a simple prototype, written in C/C++ and the Crypto++ libraries [15]. We also implemented the generic anonymous password authentication construction for experiments, and the experimental results empirically confirm that conventional anonymous password authentication is indeed not scalable. For both implementations, the client program runs on a Fujitsu notebook, Intel Core2 Duo CPU, 2.53GHz, OS Windows Vista, and the server program runs on a PC, Intel Pentium D CUP, 3.00GHz, OS Windows XP. We next report the implementation results.

A. Limited Scalability of Generic Construction

We have analyzed that the computational overhead upon the server in conventional anonymous password authentication is linear with the total number of users. This will cause scalability problem in large systems. To empirically determine how serious the problem is, we implement the generic construction proposed in Section IV. While specific schemes could be more efficient than the generic construction, the difference should not be drastic in principle.

In this implementation, we adopt the trivial PIR solution in Step 3, such that the server simply sends to the user all e_j's, together with $AuthS$; other steps remain unchanged. We use 2048-bit Paillier homomorphic encryption (i.e., $|n^2| = 2048$ bits), and $h(.)$ is instantiated by SHA-1. Figure 2 shows the experimental results with respect to differen number of passwords contained in the server's password file.

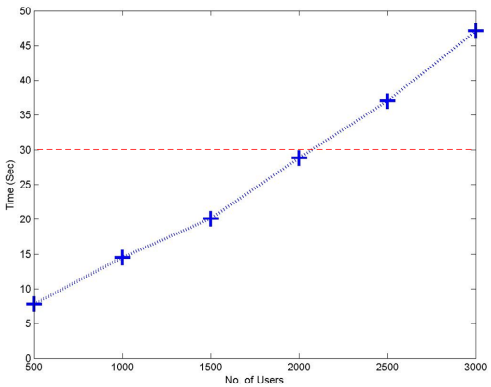


Figure 2. Experimental Results on Generic Construction

Suppose that users can tolerate up to 30 seconds of login latency, the generic construction can only support approximately 2000 users, according to Figure 2. The experimental results confirm that conventional anonymous password authentication is not acceptable to large systems having tens/hundreds of thousands of users.

B. Implementation Results of Our New Scheme

Below lists the instantiation of the cryptographic primitives and the parameters involved in the implementation of our new scheme:

CL signature	$\|n\| = 1024$
Pederson commitment	$\|p\| = 1024$
Homomorphic encryption	2048-bit Pederson encryption
Hash function $H(.)$	SHA-1
Symmetric key encryption $enc(.)$	128-bit AES
Authenticated symmetric key encryption [.]	128-bit AES + keyed SHA-1

Figure 3 shows the client program. To initiate a login, the user needs to input his password, and indicate the password-protected credential to be read. Our current implementation supports downloading the password-protected credential from a FTP server, or reading it from a USB external device (e.g., USB flash memory). Then the user clicks "OK" button, and the client program starts the proposed authentication protocol with the server. Our testing results show that it takes about 0.8 second for the user and the server to complete the authentication protocol. We obtained this average time from 200 runs of the protocol.

Figure 3. Client Program

VII. Conclusion

Anonymous password authentication enjoys the advantages of password authentication while providing user privacy protection. However, anonymous password authentication in the standard setting has intrinsic weaknesses. To solve these problems, we proposed a new approach for anonymous password authentication, assuming a different setting where users use password-protected credentials. We implemented a prototype of the proposed scheme, which demonstrated very good performance for real applications. There are issues unadressed in our scheme, e.g., user revocation. These will be our future work.

ACKNOWLEDGMENT

This work is supported by the A*STAR project SEDS-0721330047.

REFERENCES

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, *A Practical and Provably Secure Coalition-Resistant Group Signature Scheme*. Proc. Advances in Cryptology, Crypto'00, LNCS 1880, pp. 255-270, 2000.

[2] M. Abdalla, M. Izabachene, and D. Pointcheval, *Anonymous and Transparent Gateway-Based Password-Authenticated Key Exchange*. Proc. International Conference on Cryptology and Network Security, CANS'08, pp. 133-148, 2008.

[3] E. Bresson, O. Chevassut, and D. Pointcheval, *Security Proofs for an Efficient Password-Based Key Exchange*. Proc. ACM. Computer and Communication Security, pp. 241-250, 2003.

[4] S. Bellovin and M. Merritt, *Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks*. Proc. IEEE Symposium on Research in Security and Privacy, pp. 72-84, 1992.

[5] M. K. Boyarsky, *Public-key Cryptography and Password Protocols: The Multi-User Case*. Proc. ACM. Computer and Communication Security, pp. 63-72, 1999.

[6] V. Boyko, P. Mackenzie, and S. Patel, *Provably secure password-authenticated key exchange using Diffie-Hellman*. Proc. Advances in Cryptology, Eurocrypt'00, LNCS 1807, 2000.

[7] M. Bellare, D. Pointcheval, and P. Rogaway, *Authenticated Key Exchange Secure Against Dictionary Attacks*. Proc. Advances in Cryptology, Eurocrypt'00, pp. 139-155, 2000.

[8] H. Y. Chien, J. Jan, and Y. Tseng, *An Efficient and Practical Solution to Remote Authentication*. Computer & Security, 21(4), pp. 372-375, 2002.

[9] D. Chaum, *Blind Signatures for Untraceable Payments*. Proc. Avances in Cryptology, Crypto'82, pp. 199-203, 1982.

[10] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, *Private information retrieval*. Journal of the ACM, 1995.

[11] J. Camenisch, and A. Lysyanskaya, *Efficient Non-transferable Anonymous Multi-Show Credential System with Optional Anonymity Revocation*. Proc. Advances in Cryptology, Eurocrypt'01, LNCS 2045, pp. 93-118, 2001.

[12] J. Camenisch, and A. Lysyanskaya, *A Signature Scheme with Efficient Protocols*. Proc. Security and Cryptography for Networks, SCN'02, LNCS 2576, pp. 268-289, 2002.

[13] J. Camenisch, and A. Lysyanskaya, *Signature Schemes and Anonymous Credentials from Bilinear Maps*. Proc. Advances in Cryptology, Crypto'04, LNCS 3152, pp. 56-72, 2004

[14] J. Camenisch, and M. Stadler. *Efficient Group Signature Schemes for Large Groups*. Proc. Advances in Cryptology, Crypto'97, LNCS 1296, pp. 410-424, 1997.

[15] http://www.cryptopp.com/

[16] I. Damagard and E. Fujisaki, *An Integer Commitment Scheme Based on Groups with Hidden Order*. IACR Cryptology ePrint Archive 2001/064, 2001.

[17] Y. Ding and P. Horster, *Undetectable On-line Password Guessing Attacks*. ACM SIGOPS Operating Systems Review, Vol. 29(4), pp. 77-86, 1995.

[18] W. Ford and B. S. Kaliski Jr, *Sever-assisted Generation of a Strong Secret From a Password*. IEEE. 9^{th} International Workshop on Enabling Technologies, 2000.

[19] L. Gong, M. Lomas, R. Needham, and J. Saltzer, *Protecting Poorly Chosen Secrets from Guessing Attacks*. IEEE Journal on Seclected Areas in Communications, 11(5), pp. 648-656, 1993.

[20] S. Halevi and H. Krawczyk, *Public-key Cryptography and Password Protocols*. Proc. ACM. Computer and Communication Security, pp. 122-131, 1998.

[21] D. Hoover and B. Kausik, *Software Smart Cards via Cryptographic Camouflage*. Proc. IEEE Symposium on Security and Privacy, 1999.

[22] M. S. Hwang and L. H. Li, *A New Remote User Authentication Scheme using Smart Cards*. IEEE Transactions on Consumer Electronics 46(1), pp. 28-30, 2000.

[23] E. Kushilevitz and R. Ostrovsky, *Replication is not needed: single database, computationally private information retrieval*. Proc. 38th IEEE Symp. on Foundation of Computer Science, pp.364-373, 1997.

[24] J. Katz, R. Ostrovsky, and M. Yung, *Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords*. Proc. Advances in Cryptology, Eurocrypt'01, LNCS 2045, pp. 475-494, 2001.

[25] P. Mackenzie, T. Shrimpton, and M. Jakobsson, *Threshold Password-Authenticated Key Exchange*. Proc. Advances in Cryptology, Crypto'02, LNCS 2442, pp. 385-400, 2002.

[26] M. H. Nguyen and S. P. Vadhan, *Simpler Session-Key Generation from Short Random Passwords*. Proc. Theory of Cryptography, TCC'04, pp. 428-445, 2004.

[27] R. Perlman and C. Kaufman, *Secure Password-based Protocols for Downloading A Private Key*. Proc. Network and Distributed Systems Security Symposium, NDSS'99, 1999.

[28] P. Paillier, *Public-key Cryptosystems based on Composite Degree Residuosity Classes*. Proc. Advances in Cryptology, Eurocrypt'99, pp. 223-238, 1999.

[29] R. Sandhu, M. Bellar, and R. Ganesan, *Password Enabled PKI: Virtual Smartcards vs. Virtual Soft Tokens*. Proc. 1^{st} Annual PKI Research Workshop, pp. 89-96, 2002.

[30] S. Shin, K. Kobara, and H. Imai, *A Secure Construction for Threshold Anonymous Password-Authenticated Key Exchange*. IEICE Transactions on Fundamentals, Vol. E91-A, No. 11, pp. 3312-3323, 2008.

[31] J. Tardo and K. Alagappan, *SPX: Global Authenticaiton Using Public Key Certificate*, Proc. IEEE Sysmposium on Security and Privacy, pp. 232-244, 1991.

[32] D. Q. Viet, A. Yamamura, and T. Hidema, *Anonymous Password-Based Authenticated Key Exchange*. Proc. Advances in Cryptology, Indocrypt'05, LNCS 3797, pp. 233-257, 2005.

[33] J. Yang and Z. Zhang, *A New Anonymous Password-Based Authenticated Key Exchange Protocol*. Proc. Advances in Cryptology, Indocrypt'08, pp. 200-212, 2008.