Enabling Use of Single Password Over Multiple Servers in Two-Server Model

Yanjiang Yang  
Institute for Infocomm Research, Singapore  
yyang@i2r.a-star.edu.sg

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

Abstract—The two-server model is quite promising for password based authentication, well suited for the setting of federated enterprises. However, none of the existing two-server password based authentication schemes enables a user to use the same password over multiple service servers, which is deemed an important feature of the two-server model. In this paper, we propose a new scheme, enabling this prominent functionality. Our proposed scheme is password-only, and slightly more efficient than the latest two-server password based authentication scheme.

Keywords—password authentication; guessing attacks; two-server model;

I. INTRODUCTION

Password based authentication is the most commonly used entity authentication technique, due to the fact that no secure storage is required, and a user only needs to memorize his password and then can authenticate anywhere, anytime. Most of the existing password based authentication schemes assume the single-server model where a single server exists in a system. The major drawback of the single-server model is that the server may result in single point of failure, in the sense that compromise of the server reveals all user passwords held by the server. While in principle the multi-server model employing multiple servers solves this issue, it inevitably increases the operational cost to a great extent [18]. The two-server model, originally proposed by Brainard et al. [3] and subsequently strengthened by Yang et al. [19], [20], strikes a good balance between the single-server model and the multi-server model, solving the single point of failure problem, while without incurring too much operational cost.

Specifically, the two-server model comprises a front-end server, called Service Server (SS), and a back-end server, called Control Server (CS). The SS is the actual one providing certain services to users, and it is thus the one users communicate with; the CS stays behind the scene, and its sole responsibility is to help secret-share user passwords so as to avoid single point of failure, as well as to assist the SS for user authentication; users do not contact the CS during authentication, and they are even not necessarily aware of the existence of the CS. In practice, the two-server model is usually deployed in the form of one control server supporting multiple service servers, as shown in Figure 1. Notably, the one-CS-multi-SS architecture well suits the trend that enterprises increasingly now become federated. In particular, a federated enterprise consolidates under one corporate umbrella multiple divisions/branches/affiliations that serve different aspects of the business continuum. Applying the one-CS-multi-SS architecture to a federated enterprise is straightforward: the headquarter of the enterprise manages the CS, and each affiliated organization operates a SS, providing services to its own group of users.

![Figure 1. The One-CS-Multi-SS Architecture](image-url)
user if the user uses the same password over the two servers. We know that a big inconvenience in password based authentication is that a user has to use different passwords for distinct servers; otherwise, a server can impersonate a user to another server or a server can impersonate another server to the user. It is thus of great importance to enable the functionality of using the same password over multiple service servers in the two-server model.

Our Contribution. We are motivated to propose a new scheme that enables a user to use the same password over multiple service servers in the two-server model. Our scheme is password-only, and efficient, achieving slightly better efficiency than the latest two-server password based authentication scheme in [12], which is the most round efficient of its kind in the literature. The rest of the paper is organized as follows. In Section II, we review the related work and the background on password based authentication. Our new scheme is presented in Section III together with relevant discussions, and Section IV concludes this work.

II. BACKGROUND AND RELATED WORK

Using password for authentication has intrinsic weaknesses. In particular, passwords are short (to be memorizable), normally drawn from a relatively small space; thus they are low entropy in nature, and brute-force guessing attacks are a threat. 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 a victim user by trying a different password each time until finds the correct one. In the off-line guessing attack, the attacker needs not interact with the server; rather, it gleams the protocol transcript of a login session, and then enumerates all possible passwords against the login transcript to determine the actual one. It should be clear that on-line guessing attacks are unavoidable in password based authentication, but can be easily thwarted at the system level by limiting the number of unsuccessful login attempts made by a user. In contrast, off-line guessing attacks are notoriously harder to tackle, and they must be addressed at the protocol level.

It is now a known fact that public key operations, e.g., exponentiations in a multiplicative group, are essential in making password based authentication secure against off-line guessing attacks [11]. But be noted that 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 approaches exist for password based authentication: 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 [6], [10], [11]. 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 (Public Key Infrastructure) for certification. In contrast, the password-only approach does not involve any public key primitive, thereby eliminating the dependence on PKI. The password-only approach, or password authenticated key exchange (PAKE), has been extensively studied in the literature, e.g., [1], [2], [4], [5], [7], [8], [14], [16].

The fore-mentioned password based authentication schemes are designed over the single-server model, where a single server holds a password file that contains all users’ password information. A security concern is that compromise of the server reveals all passwords. A natural solution is to deploy multiple servers to secret-share the passwords [9], [15], [17]. However, the protection to user passwords in the multi-server model comes at the price of downgraded operational quality [18].

The two-server model, initially proposed by Brainard et al. [3], strikes a good balance between the single-server model and the multi-server model: it on the one hand solves the single point of failure issue, while on the other hand incurs moderate extra operational cost. The password based authentication scheme proposed by Brainard et al., however, is not password-only. Subsequently, Yang et al. [19] strengthened the two-server model, and proposed password-only two-server authentication schemes [20]. A follow-up scheme by Jin et al. [12] has better round efficiency than Yang et al.’s schemes. We, however, notice that none of these existing two-server password based authentication schemes [3], [19], [20], [12] satisfies an important feature of the two-server model, i.e., a user uses the same password over multiple service server. Our focus in this work is thus to enable this functionality in the two-server model. As a final note, the two-server system considered in [13] is a special case of the multi-server model of two servers, rather than the two-server model we are considering.

III. ENABLING USE OF SINGLE PASSWORD OVER MULTIPLE SERVICE SERVERS

In this section, we present a new two-server password-only authenticated key exchange scheme that enables the use of single password over multiple service servers. Our scheme bears much similarities with Jin et al.’s [12], especially in the authentication of the user to the server. The major difference is that \( g_1, g_2 \) are generated real time by the user and the service server in our scheme, while they are pre-set system parameters in Jin et al.’s.

A. Global System Parameters

Let \( G \) be a multiplicative group with order \( q \), where \( q \) is a big prime. Let \( h_0(\cdot) : \{0, 1\}^* \rightarrow G \), and \( h_1(\cdot), h_2(\cdot), h_3(\cdot) : \{0, 1\}^* \rightarrow G \).
{0, 1}^* \rightarrow \{0, 1\}^l$ be cryptographic hash functions, where $l$ is a system parameter.

**B. User Registration**

Each user needs to register in advance his password to every service server he wants to communicate with, as well as the control server CS. To register to a particular service server SS and CS, a user U (supposing his password is $\pi$) generates two random password shares $\pi_1, \pi_2$ such that $\pi = \pi_1 + \pi_2 \pmod{q}$. U then registers $\pi_1$ to SS and $\pi_2$ to CS through some out-of-band channels. Note that CS needs to associate $\pi_2$ not only with U, but also with that particular SS, since we allow U to register with different service servers using the same password $\pi$.

**C. Protocol Details**

Suppose a user U has password $\pi$, and the corresponding shares at a service server SS and the control server CS are $\pi_1$ and $\pi_2$, respectively. The authentication and key exchange protocol is depicted in Figure 2, which ends up allowing U and SS to authenticate each other and establish a common session key $SK$. Note that in the protocol, $g_1, g_2$ are generated in real time by each party as follows. U generates $g_1 = h_0(U, SS, 1)$, $g_2 = h_0(U, SS, 2)$ using his own identity and that of the service server he wants to communicate with; following the same format, SS generates $g_1$ and $g_2$ using the identity of the requesting SS and the user. Given that each party generates $g_1$ and $g_2$ as above, the description in Figure 2 is already clear, so we do not repeat the protocol step by step here. But for ease of understanding, we briefly explain the intuitions behind the protocol.

Intuitively, the protocol can be viewed to be composed of two parts: one part are the messages associated with $X, Y_0, A_1, A_2$ which enable U and SS for mutual authentication and key exchange; the other part are the messages associated with $B_2, B_1, A_0$ and $S_3$, which enable CS to authenticate U and SS. We point out that the latter part is essential, since otherwise it may allow SS to launch undetectable on-line guessing attacks against CS. Let us next look at the details of the two parts. For the first part,

- U authenticates SS (and CS): U commits his password and a random $x$ in $X$, and expects Diffie-Hellman key exchange (with $x$ being his secret component), which is possible only if the valid SS together with CS can remove the password portion. On the servers’ side, the sessional secrets committed by SS and CS for this Diffie-Hellman key exchange is $y_1$ and $y_2$, respectively. Note that $y_0$ used by SS in $Y_0$ is simply to conceal $\pi_1$. As a result, $A_1$ and $S_1$ indeed allow U to perform Diffie-Hellman key exchange as expected.
- SS authenticates U and CS: apparently, SS authenticates U and CS following the same rationale, in that only
the valid U and CS can remove \( \pi_1 \) it committed in \( Y_0 \), and in turn make the Diffie-Hellman key exchange successful.

For the second part, CS authenticates SS and U using exactly the same approach as above, and this part is easier to understand. This is because in the first part, SS needs additional computation to remove the involvement of \( Y_0 \).

**REMARK.** We like to highlight how our scheme allows a user to use the same password over different service servers. The basic idea is that \( g_1, g_2 \) are generated by each party using the identities of the user and the service server (see above). As such, if SS impersonates another service server to CS after receiving \( M_1 \), then CS will compute the values of \( g_1, g_2 \) using the identity of the impersonated service server. This clearly cannot make the user accept \( S_1 = h_1(A_1^2) \).

**D. Efficiency**

We compare the performance of our scheme with the latest scheme proposed by Jin et al. [12] (the JWX scheme for short), which is also the most efficient in round efficiency. The comparison results are listed in Table I, where computation performance counts the number of exponentiations performed by each party, and \(|G|, |h|\) denote the bit length of a group element in \( G \), and the output length of cryptographic hash functions, respectively. It can be seen that our scheme has the same round efficiency as the JWX scheme, and is slightly better in terms of computation and communication in bits.

**E. Security Discussions**

So far, there is no formal security model for the two-server password based authentication. Undoubtedly, establishing a formal security model is important, but we are not intended to claim contributions to this respect in the current work, and we shall consider it to be our future target. In this paper, our focus is to strengthen the existing two-server password based authentication schemes to enable a user to use the same password over multiple service servers. We thus only provide the proposed scheme with some heuristic security discussions, as in [12].

Following the existing literature [12], [20], the main security requirement is that the scheme is secure with respect to off-line guessing attacks, when either SS or CS is controlled by an active adversary. To see the intuitions why our scheme satisfies the security requirement, notice that the following two blind techniques are always used whenever each party computes messages to be sent out using its secret. The first blind technique is in the form of \( g_1^{r_1} g_2^{r_2} \), where \( m \) is either the password or a password share, and \( r \) is a random number in \( Z_1^* \) (e.g., \( X \) in \( M_1 \)). From \( g_1^{r_1} g_2^{r_2} \), the adversary cannot get anything useful on \( m \) using off-line guessing attacks as \( g_2^{r_2} \) is not known. The second blind technique is in the form of \( m_1 = (D/g_1^m)^r, m_2 = g_2^r \), where \( D \) is the message received, \( m \) is the password or a share, and \( r \) is a random number (e.g., \( A_2, Y_2 \) in \( M_3 \)). Note that \( D \) can be generated by the adversary. This implies that the adversary is able to compute \((g_1^m)^r\) from \( m_1 \) and \( m_2 \). To see this, suppose the adversary sets \( D = g_2^r \) for some arbitrarily picked \( d \). The adversary can compute \((g_1^m)^r = m_1/m_2^d \). Since \( r \) is random and unknown to the adversary, knowing \( g_1^m \) and \( g_2^r \) does not help the adversary determine the value of \( m \) under the DDH assumption. By using these two blind techniques throughout the protocol, security against off-line guessing attacks is achieved.

### IV. Conclusion and Future Work

A big inconvenience in password based authentication is that a user needs to memorize and use different passwords upon different servers; otherwise, a server can impersonate a user to another server or a server can impersonate another server to the user. The two-server model offers the potential to solve this issue, allowing a user to use the same password over multiple service servers. However, none of the existing two-server password based authentication schemes has enabled and implemented this functionality. In this paper, we proposed a new two-server password authentication scheme, providing this prominent feature. Our scheme is password-only, and slightly more efficient than the latest two-server password based authentication scheme.

Establishing an appropriate formal model for two-server password based authentication, and in turn proving the security of our proposed scheme within the model are among our future work. Another future work is to empirically study the performance of our scheme (to complement the analysis in Section III-D) by implementing and experimenting the scheme in a real-world application scenario.
REFERENCES


