A NEW ARCHITECTURE FOR USER AUTHENTICATION AND KEY EXCHANGE USING PASSWORD FOR FEDERATED ENTERPRISES

Yanjiang Yang[1, 2], Feng Bao[1] and Robert H. Deng[2]
[1]Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613; [2]School of information Systems, Singapore Management University, Singapore 259756

Abstract: The rapid rise of federated enterprises entails a new way of trust management by the fact that an enterprise can account for partial trust of its affiliating organizations. On the other hand, password has historically been used as a main means for user authentication because of operational simplicity. We are thus motivated to explore the use of short password for user authentication and key exchange in the context of federated enterprises. Exploiting the special structure of a federated enterprise, our proposed new architecture comprises an *external server* managed by each affiliating organization and a *central server* managed by the enterprise headquarter. We are concerned with the development of an efficient authentication and key exchange protocol using password, built over the new architecture. The architecture together with the protocol well addresses off-line dictionary attacks initiated at the server side, a problem rarely considered in prior effort.

Key words: federated enterprise; password authentication; dictionary attack; key exchange; public key cryptosystem.

1. INTRODUCTION

Driven by the promise of cost saving, expansion of market share and quality improvement of service provision through consolidation and cooperation, industry has seen a rapid rise of federated enterprises. Specifically, a federated enterprise consolidates under one corporate umbrella multiple divisions, branches and affiliations serving different aspects of business continuum and service coverage. For example, in the

banking sector, a central bank has numerous branches distributed across a city, a region. Another example is in the healthcare area, where a federated hospital integrates many inside and outside units, e.g., clinical laboratories, departments, outpatient clinics, managed care organizations, pharmacies and so on. In a federated enterprise, each affiliating organization has its own business interest, providing service to a distinct group of users.

Following conventional ways, each affiliating organization has to work independently on trust management, maintaining by itself account information of its users. However, this may not be optimal in practice. First, affiliating organizations may lack sufficient expertise and funds for a secure maintenance of user account. This situation deteriorates with the trend that organizations are becoming increasingly fond of outsourcing IT management to some specialized service providers. In such circumstances, system administrators may present themselves as a big threat to system security [1]. Second, from the users' perspective, a user apparently prefers assuming the higher credit of the entire enterprise rather than that of an individual affiliating organization. For these reasons, new paradigm of trust management that well takes advantage of the special structure of federated enterprises is of interest and urgency.

On the other hand, human memorable password has historically been used as a main means for user authentication, due to operational simplicity. In particular, no dedicated device is required for storing password, which is deemed of particular importance as users are becoming increasingly roaming nowadays. We are thus interested in exploring the use of short passwords for user authentication and key exchange in the context of federated enterprises. Towards this, we are faced to first address the weaknesses inherent in password-enabled systems: because of a limited *dictionary* space, password is susceptible to brute-force *dictionary attack,* and more precisely *off-line dictionary attack*[8] [2]. Specifically, in off-line dictionary attack, an attacker records the transcript of a successful login between a user and the server, and then enumerates and checks every possible password against the transcript, until eventually determine a correct password. Tremendous effort has been dedicated to resisting off-line dictionary attack in password-enabled protocols (e.g., [3, 4, 5, 6, 7, 8, 9, 10]). An assumption common to these methods is that the server is completely reliable, so users share with the server clear passwords or some easily-derived values of passwords for subsequent user authentication. As such, while these protocols are sufficiently robust to off-line dictionary attacks by the outside attackers, they

[8] In contrast to off-line attack is on-line dictionary attack, which turns out to be easily thwarted by restricting the number of unsuccessful login attempts made by a user.

are not intended to defend against the server, e.g., in the event of penetration by outside attackers.

However, for the reasons discussed earlier (limited expertise and funds, outsourcing, etc.), threats posed by the server become clearer in the setting of federated enterprises. As a consequence, servers maintained by the affiliating organizations of an enterprise are no longer deemed fully trusted. Adapting password-enabled systems to federated enterprises has to additionally mitigate the concern of *unreliable* servers, in addition to outside attackers. To this end, we propose a new architecture for user authentication and key exchange using password, geared to the needs of federated enterprises. In its simplest configuration, the architecture consists of an *external server* managed by each affiliating organization and a *central server* administrated by the enterprise; each server only keeps partial information on a user password, such that no single server can recover the password by means of off-line dictionary attacks user authentication is accomplished together by the two servers. Our attention is given to the development of an efficient authentication protocol for the new architecture, rather than formal provable security. The proposed architecture together with the protocol enjoys several attractive features

The rest of the paper is organized as follows. In Section 2, we review related work. We then present our new architecture and discuss extension in Section 3. In Section 4, we propose an efficient user authentication and key exchange protocol well attuned to the new architecture, and security of the protocol is examined. Finally, Section sec5 concludes the paper.

2. RELATED WORK

Resistance to off-line dictionary attack has long been in the core of research on password-enable systems. It is a proven fact that *public key techniques* are absolutely necessary so as to resist off-line dictionary attacks, whereas the involvement of *public key cryptosystems* is not essential [7]. Accordingly, two separate lines of research have been seen in the literature: combined use of password and public key cryptosystem, and password only approach. For the former, asymmetry of capacity between the users and the server is considered, so that a user only uses a password while the server has a public/private key pair at its disposal. Examples of such public key-assisted password authentication include [7, 8, 11]. With no surprise, the use of public keys entails the deployment of PKI for certification, adding to the users the burden of checking key validity. To eliminate this drawback, password-only authenticated key exchange (PAKE) protocols have been

extensively explored (e.g., [3, 4, 5, 6, 9, 10]). The PAKE protocols do not involve any public key cryptosystem whatsoever.

What common to the above methods is the assumption that the server is totally trustful, so a user shares a password or some password-derivative value with the server. In other words, these methods are by no means resilient to the off-line dictionary attack initiated by the server, e.g., in the event of break-ins by outside attackers. To address this problem, [12] first proposed the so-called *password hardening* technique by transforming a weak password into a strong one through several servers, thereby eliminating a single point of vulnerability. Afterwards, [13] improved this multi-server model. Further and more rigorous extensions were due to [14] and [15], where the former built a t-out-of-n threshold PAKE protocol and gave formal security proof under the Random Oracle Model [16], and the latter presented another two provably secure threshold PAKE protocols under the standard model. A limitation of the protocols in [14] and [15] is their low efficiency, so they may not be practical to resource-constrained users, e.g., mobile phones. Moreover, notice that in the above multi-server setting, password is still susceptible to a *weaker* form of a single point of vulnerability, in the sense that passwords are eventually reconstructed by a *dealer* at the time of user authentication.

By contrast, in our architecture no trust exists between the central server and the external server, thus a single point of vulnerability is completely eliminated. This however adds substantial challenges to the design of the underlying authentication protocol. Our basic architecture (one central server is involved) is similar to a recent novel two-server model in [17], which was to overcome the deficiency of complex and computation extensive protocols in [14, 15]. The protocol in [17] assumes that servers use SSL to establish secure communication channel with users. Distinctions between our work and [17] include: (a) we achieve mutual (bilateral) authentication as well as key exchange, whereas [17] considered merely unilateral authentication of the user to the servers, and no key exchange; (b) we develop a different protocol for testing the equality of two numbers under our presumed adversary model. Without a similar explicitly specified adversary model, the protocol in [17] may cause trouble in case that one of the two servers deliberately disrupts the protocol, attempting to gain advantages over the other; (c) our architecture is tailored to federated enterprises, whereas the model in [17] was suggested for an organization outsoucing a part of its trust management to a security service provider; (d) we also suggest extending the basic architecture to an architecture including a group of central servers, solving the possible bottleneck caused by one single central server.

3. A NEW AUTHENTICATION ARCHITECTURE FOR FEDERATED ENTERPRISES

Our basic architecture is shown in Figure 1: at the server side, an *external server* and a *central server* coexist; the external server is the actual one providing service to the users, thereby standing front-end; the central server is to assist the external server for user authentication, staying transparent to the users. A main objective of this architecture is to "harden" a user's short password into two long secrets and each is hosted by a server, so that neither of the two servers can launch off-line dictionary attacks. As discussed earlier, a uniqueness of this architecture is represented by the fact that no trust exists between the two servers. This on the one hand totally eliminates a single point of vulnerability, while on the other hand makes the design of the underlying password-enabled protocol particularly challenging. In particular, the two servers together validate user passwords, whereas no extra information should be leaked to each other in facilitating off-line dictionary attack.

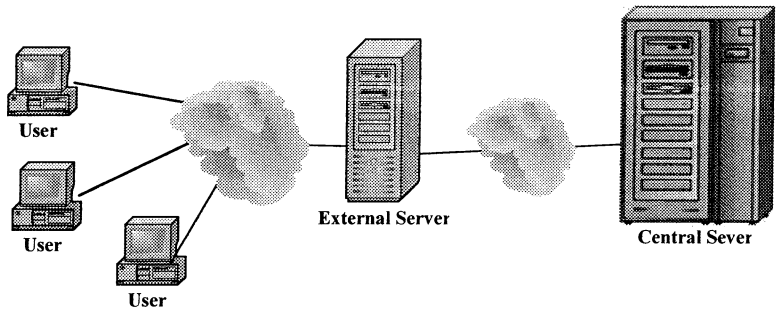

Figure 1. Basic two-server architecture for federated enterprises

Figure 2 shows the scenario when applying this basic structure to a federated enterprise: the headquarter of the enterprise manages the central server, and each affiliating organization operates an external server, providing service to a group of users of its own. This architecture offers several benefits:

1. First of all, of particular advantage is that neither the central server nor the external servers can compromise user passwords by means of off-line dictionary attack.
2. Affiliating organizations are relieved from strict trust management to some extent, so they can dedicate their limited expertise and resources to enhancing service provision to the users.

3. Users are afforded to assume the higher credit of the enterprise, while engaging business with individual affiliating organizations.
4. The enterprise is provided a way to monitor the affiliating organizations, deterring them from cheating.
5. As the central sever is hidden from the public, the chance for it under attacks is substantially minimized, thereby increasing the overall security of the architecture.

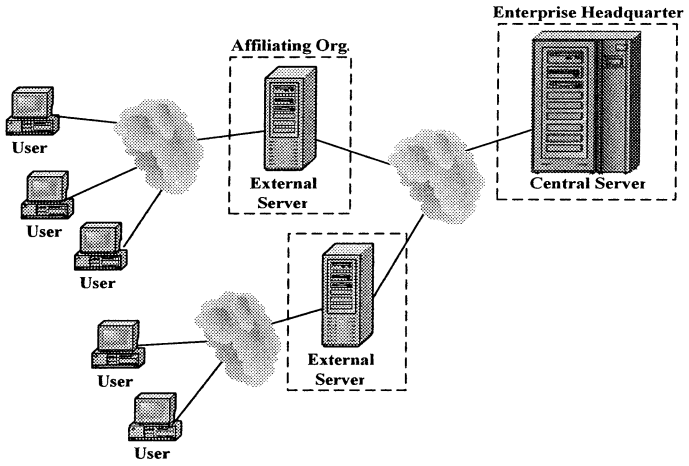

Figure 2. Typical application scenarios

3.1 Extension

In the basic architecture, a single central server is to collaborate with numerous external servers in a federated enterprise. It is thus possible that the central server becomes a system bottleneck. To mitigate this concern, we suggest extending to include several central servers as a group as shown in Figure 3. The group of central servers work under a t-out-of-n threshold secret sharing scheme (e.g., [18]), so that each holds a share of the secret that would be otherwise kept by a single central server (n is the total number of the central servers). When an external server requests for user authentication, one of the servers volunteers to manage the reconstruction of the secret among the group. This voluntary central server is the only one interacting with the requesting external server, thus the external server feels nothing different as with a single central server. This extension not only solves the potential bottleneck problem, but also addresses the issue of failures or break downs of a single central server.

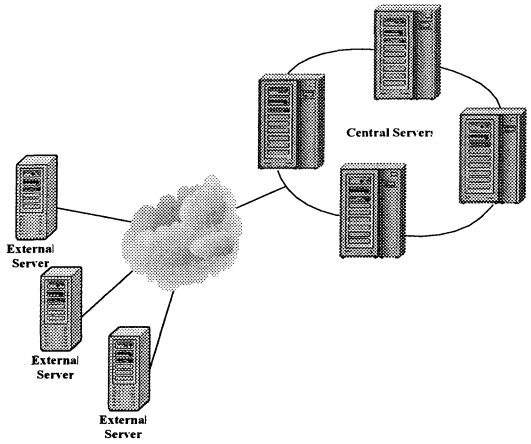

Figure 3. Extended architecture including a group of central servers

4. AN AUTHENTICATION AND KEY EXCHANGE PROTOCOL USING PASSWORD

In this section, we shall detail the authentication and key exchange protocol using password, which is geared to the setting of federated enterprises. The protocol is built over the basic architecture in Figure 1. Extension to the extended architecture in Figure 3 is also discussed. A central building block of our protocol is a sub-protocol for testing the equality of two numbers. For ease of reference, we start with listing the notations that are used in the sequel.

Table 1. Notations

p, q	two large primes such that $p = 2q+1$.
g	a generator of a subgroup of Z_p^* of order q.
π	a user's password.
$h(.), H(.)$	cryptographic hash functions which are modelled as random oracle [16].
U, ES, CS	identity of a user, the external sever and the central server, respectively.
$E_X(.), D_X(.)$	encryption and decryption functions (of a semantic secure public key cryptosystem) by entity X's public key and private key, respectively.

4.1 The setting

Three types of entities are involved in our system, i.e., the users, the external servers and the central server. For the purpose of authentication, each user U has a short password π, and π is transformed into two long secrets, each of which is held by the external server ES that U belongs to and

the central server *CS*, respectively. *CS* stays transparent to the public, and *ES* acts as the relaying party between *U* and *CS*. In such a scenario, it is reasonable to assume authentic communications between *ES* and *CS*. Definitely, this assumption can be readily accomplished by the two parties sharing a secret, which is used to key a MAC. Considering the close tie between the two parties, it is also convenient for them to periodically (e.g., once a week) update this secret, e.g., the headquarter (*CS*) sends a new secret to the affiliating organization (*ES*) by normal mail weekly. *CS* has an authentic key pair that corresponds to a semantically secure public key cryptosystem, with the encryption function (resp. decryption function) $E_{CS}(.)$ (resp. $D_{CS}(.)$). As discussed earlier, in a federated enterprise *CS* clearly assumes more trust than *ES* because of sufficient expertise, funds, and the fact that *CS* is not directly exposed to the public. Considering such asymmetry in terms of trust upon *CS* and *ES*, adversary model in our protocol is that *CS* is *semi-honest* and *ES* is *malicious*, with respect to their desire for off-line dictionary attack, and they do not collude. More specifically, *CS* is honest-but-curious [19], i.e., it follows the protocol, with the exception that it may try to derive extra information by analyzing the protocol transcript; on the contrary, *ES* may act arbitrarily for uncovering user passwords.

4.2 High level description

Central to our protocol is to fight against off-line dictionary attack by the servers (Note that outside attackers are clearly no more powerful than the external server in this regard). The intuition behind our authentication and key exchange protocol is as follows: in an out-of-band *registration* phase, a user *U* "hardens" his password π into two random long secrets s and $\pi + s$, and registers them to the external server *ES* and the central server *CS*, respectively, where s is a random number. In *authentication* phase, *U* picks another long random number r and sends r and $\pi + r$ to the two servers, respectively. Upon receiving the messages, *ES* computes $a = r - s$, and *CS* computes $b = (\pi + r) - (\pi + s) = r - s$. Afterwards, the two servers engage into an interactive protocol to test $a \stackrel{?}{=} b$. Note that $a = b$ holds if and only if user *U* knows π. Upon the servers validating the user, *ES* and *U* negotiate a common session key for subsequent data exchanges. Clearly, from s and r (resp. $\pi + s$ and $\pi + r$), *ES* (resp. *CS*) is unable to gain anything useful on π. It is thus of crucial importance to ensure the protocol for testing $a \stackrel{?}{=} b$ could not facilitate the servers for off-line dictionary attack. In what follows, we first propose a protocol allowing two parties to test $a \stackrel{?}{=} b$, which will be invoked by our final authentication and key exchange protocol that follows.

4.3 A protocol testing *a ?= b*

A protocol for simply testing *a ?= b* by two parties *A* (possessing *a*) and *B* (possessing *b*), while without disclosing *a and b* may be quite straightforward. See a simple example: *A* sends $h(a)$ to *B* and *B* sends $h(b)$ to *A*, where $h(.)$ is a hash function as defined in Table 1; each party checks $h(a)$ *?=* $h(b)$. A variant is that *A* sends $G_a = g^a \bmod p$ to *B* and *B* sends $G_b = g^b \bmod p$ to *A*, where *p* and *g* are defined as in Table 1; each party then checks G_a *?=* G_b. Both methods however cannot avert off-line dictionary attack by the two parties in our case. Take the first example and *A* for instance, *A* chooses a random number *r* and computes $a = r - s$ for himself, and simply sends *r* to *B*. *B* will return $h(r - s - \pi)$ to *A*. It is easy to see that *A* can enumerate every possible password until find π', such that $h(a - \pi') = h(r - s - \pi)$. In a same way, the attack applies to the variant example, although in normal cases, it is hard to get *a* (resp. *b*) given G_a (resp. G_b) according to *discrete logarithm assumption.* These examples convey to some extent the subtleties in designing a protocol in our case of withstanding off-line dictionary attack.

Let QR_p denote the group of quadratic residues modulo *p*, and a hash function be defined as h: $\{0, 1\}^{|p|} \rightarrow QR_p$, where *p*, *q*, $h(.)$ are public parameters and as defined in Table 1. Note that in practice *h* can be achieved by squaring a one-way hash function, e.g., SHA1. We outline our proposed protocol for testing *a ?= b* in Figure 4 (all arithmetic operations are calculated modulo *p*).

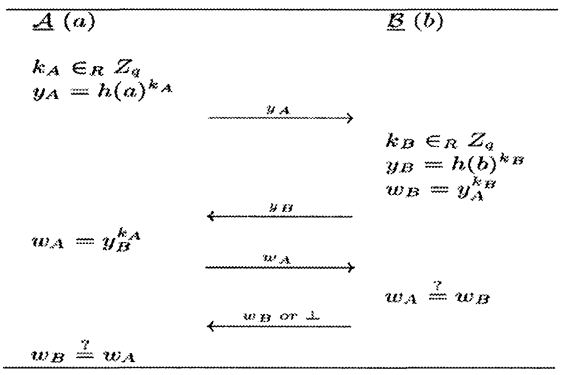

Figure 4. A protocol testing *a ?= b*

Specifically, *A* picks $k_A \in_R Z_q$ on the fly and computes $y_A = h(a)^{k_A} \bmod p$. *A* initiates the protocol by sending y_A to *B*. Upon receiving the message, *B* chooses $k_B \in_R Z_q$, in turn computes $y_B = h(b)^{k_B} \bmod p$ and

$w_B = y_A^{k_B} \bmod p$, respectively. B then sends y_B to A. After receiving y_B, A computes $w_A = y_B^{k_A} \bmod p$ and sends w_A to B. With w_A, B tests $w_A \stackrel{?}{=} w_B$. B then sends w_B to A if $w_A = w_B$, and a special label $\perp$ otherwise. A then tests $w_B \stackrel{?}{=} w_A$ if w_B is received.

4.3.1 Security analysis

In line with the adversary model defined in our final protocol, we assume A a malicious adversary while B an honest-but-curious adversary. In addition, for the moment we assume A does not replay messages, and the communication between A and B is authentic. These assumptions will be clear when we come to our final protocol.

We start by by claiming that *upon completion of the protocol, either* (1) *A and B learn that $a = b$ or* (2) *A and B learn $a \neq b$ but nothing more on the opposite side's secret.*

Clearly in the first case, if $w_A = w_B$ holds, A and B learn that $a = b$. We next show if $a \neq b$ (this may be due to that A cheats by intentionally using a, which is different from his original input), both parties learn nothing more. Consider A first: intuitively, A gets $y_B = h(b)^{k_B} \bmod p$ at the end of the protocol. It is easy to see that without knowing k_B, A is unable to obtain anything on b in an information theoretic sense. Next, consider the case of B: when the protocol terminates, of relevance to B is ($y_A = h(a)^{k_A} \bmod p$, $w_A = y_B^{k_A} \bmod p$). Notice intuitively that (w_A, $y_B = w_A^{k_A^{-1}} \bmod p$, y_A, $h(a) = y_A^{k_A^{-1}} \bmod p$) is indistinguishable from (w_A, y_B, y_A, z) under the decisional Diffie-Hellman assumption [20], where z is random and $k_A k_A^{-1} = 1 \bmod q$. Therefore B cannot learn anything more on a from executing the protocol.

Our next claim is that *if A aborts the protocol before completion, he is unable to gain more advantages over B.* To see this, the only place A is likely to abort is after receiving y_B from B. But as we have discussed, y_B does not leak anything on b. Our claim thus holds. We stress that as an honest-but-curious adversary, B is not interested in deviating from the protocol, e.g., deliberately aborting the protocol or sending $\perp$ in the case of $w_A = w_B$. In this sense, our protocol achieves "fairness".

4.4 Authentication and key exchange using password

We now present an efficient authentication and key exchange protocol using password, built over the basic architecture in Figure 1. The earlier protocol for testing $a \stackrel{?}{=} b$ is invoked in this protocol as a building block, where *ES* plays the role of A and *CS* takes the role of B. In the sequel, we

occasionally omit "modulo p" in stating arithmetic operations as long as the context is clear.

System parameters are defined as follows: p and q are as defined in Table 1, and QR_p is the group of quadratic residues modulo p. A hash function $H(.)$ (e.g., SHA1) is employed. Moreover, g is picked from QR_p as $g \in_R QR_p \backslash \{1\}$. Clearly, g is of order of q. $H(.)$, p, q and g are public parameters.

To enrol as a legitimate user in a service, it is natural that at the beginning, a user must authenticate to the service provider and in turn establish a password with the organization for subsequent service access. In our case, U needs to register to not only the actual service provider ES but also the enterprise CS that ES is affiliated to.

4.4.1 Registration

Suppose U has already successfully authenticated to ES, e.g., by showing his identity card, U picks his password π and selects a random number $s \in_R QR_p$. U then registers in a secure way s and $\pi + s \bmod p$ to ES and CS, respectively. Here for purely simplicity reasons, we assume $(\pi + s \bmod p) \in QR_p$[9]. Consequently, ES stores the account information (U, s) to its secret database, and CS stores $(U, \pi + s \bmod p)$ to its secret database. Someone may wonder how U registers $\pi + s$ to CS, as CS is supposed hidden from the public. This is in fact not a problem in practice: U can contact CS by normal mail, etc. Indeed, imagine that a user enrols in a branching bank, it is not strange at all that the user still needs to submit a secret to a higher authority of the bank so as to activate his account.

Upon completion of the registration, U can request service from ES, by exploiting the protocol in Figure 5 for authentication and establishment of a common session key.

4.4.2 The protocol

Let us follow the protocol (in Figure 5) step by step. To initiate the protocol, U picks x as $x \in_R Z_q$ and computes $e_x = g^x \bmod p$, which will be used for (Diffie-Hellman) key exchange. U also selects r as $r \in_R Z_q$, and encrypts e_x, $\pi + r \bmod p$ and T using CS's authentic public key as $e_0 = E_{CS}(e_x, \pi + r, T)$, where T is the current timestamp. U then sends in $M1$ the message of (U, e_x, r, e_0, T) to ES. Upon receipt of the message, ES first checks whether T is within a pre-defined time window: if T expires, ES simply

[9] In our protocol, we require $(\pi + s \bmod p) \in QR_p$. Indeed, if $(\pi + s \bmod p) \notin QR_p$, then it must hold that $(p - \pi - s \bmod p) \in QR_p$.

returns *reject* to *U* and aborts the protocol; otherwise, *ES* proceeds ahead. *ES* searches his secret database for *U*'s account. If no such an account is found, *ES* returns *reject* to *U* and stops the protocol; otherwise, *ES* fetches the secret s and computes $a = r - s \bmod p$; *ES* also keeps e_x in his *live* buffer. Afterwards, *ES* relays (U, e_0, T) to *CS* in *M*2.

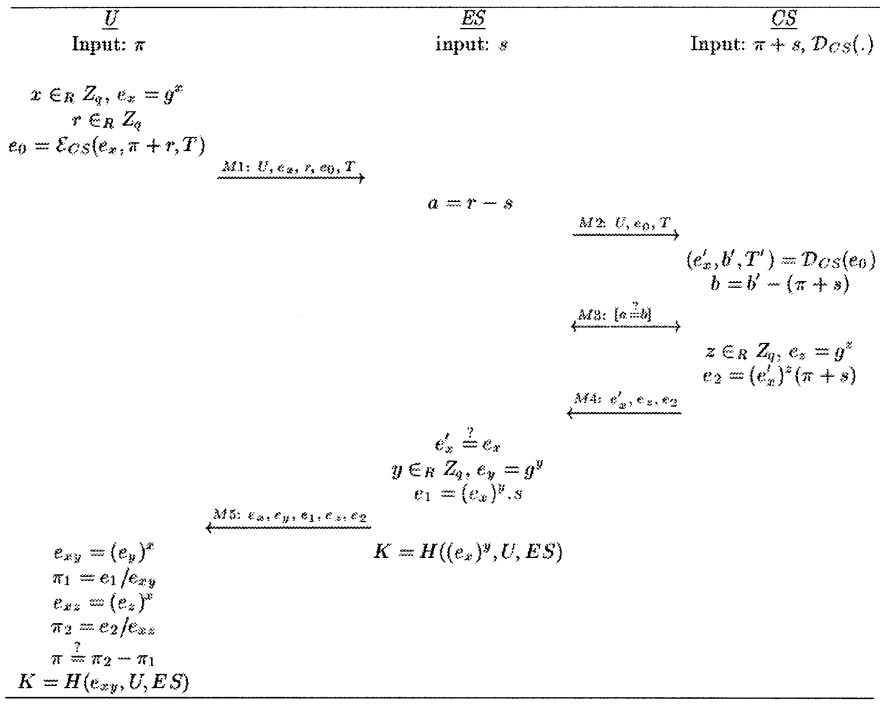

Figure 5. An authentication and key exchange protocol using password

In a similar way, *CS* checks the freshness of T and the account of U in his secret database. If both are correct, *CS* decrypts e_0 to get $(e_0', b', T') = D_{CS}(e_0)$. *CS* then checks whether $T = T'$: if not, *CS* rejects; otherwise, *CS* continues. *CS* takes out $\pi + s$ and computes $b = b' - (\pi + s) \bmod p$. Next, in *M*3, *CS* and *ES* engage in the protocol of Figure 4 to test $a \stackrel{?}{=} b$. If $a \neq b$, *CS* rejects and *ES* in turn replies *reject* to *U*. Otherwise, *CS* chooses $z \in_R Z_q$ and computes $e_z = g^z \bmod p$, which is in turn used to "encrypt" $\pi + s$ as $e_2 = e_x'^z(\pi + s) \bmod p$. *CS* then sends in *M*4 the message of (e_x', e_z, e_2) to *ES*. Upon receiving the message, *ES* checks whether $e_x' = e_x$ (e_x is being kept alive in the buffer) to ensure that e_x received in *M*1 has not been replaced by outside attackers. Interestingly, here e_x and e_x' are serving an extra purpose of "freshness nonce". If $e_x' \neq e_x$, *ES* notifies *CS* and sends *reject* to *U*. Otherwise, *ES* picks

$y \in_R Z_q$ and computes $e_y = g^y \bmod p$. e_y is then used to "encrypt" s as $e_1 = e_x^y.s \bmod p$. Afterwards, *ES* sends $(e_x, e_y, e_1, e_z, e_2)$ in *M*5 to *U*. Here, e_x acts as a freshness nonce. *ES* also computes a common session key K as $K = H((e_x)^y, U, ES)$. Upon receiving the message, *U* does the following calculations: computes $e_{xy} = (e_y)^x \bmod p$ and obtains $\pi_1 = e_1/e_{xy} \bmod p$; computes $e_{xz} = (e_z)^x \bmod p$ and gets $\pi_2 = e_2/e_{xz} \bmod p$; tests $\pi \stackrel{?}{=} \pi_2 - \pi_1 \bmod p$: if the equality holds, *U* is assured of the authenticity of *ES* and computes the common session K as $K = H(e_{xy}, U, ES)$.

4.4.3 Security analysis

Our architecture is different from either the standard client-server model (e.g., [7, 10]) or the multiple-server model (e.g., [14, 15]), so formal provable security may be quite involved. We thus give an informal security analysis for the moment, yet we believe our analysis still suffices to guarantee the security of the protocol. Recall that the primary goal of this protocol is to resist off-line dictionary attacks by the two servers, where *ES* is a malicious adversary and *CS* is an honest-but-curious adversary under the adversary model that represents different levels of trust upon *ES* and *CS*. It is easy to see that outside attackers are no more powerful than *ES* in terms of the capability to uncover *U*'s password. Admittedly, outside attackers can act as man-in-the-middle between *U* and ES, resulting in a legitimate user being deemed illegitimate by the servers. Note that such attacks are inevitable in any protocol, and discussion of them is beyond the scope of this work as they are not relevant to the password attacks.

Resistance to CS:

In the protocol, what relevant to *CS* for off-line dictionary attack is $(\pi + r \bmod p, \pi + s \bmod p)$, as well as the interactive protocol for testing $a \stackrel{?}{=} b$. Clearly, from $\pi + r \bmod p$ and $\pi + s \bmod p$, *CS* is unable to learn anything on π; as discussed earlier, the protocol for testing $a \stackrel{?}{=} b$ leaks nothing more on π. Consequently, as a passive semi-trusted adversary, *CS* cannot launch effective off-line dictionary attacks.

Resistance to ES:

Intuitively, if following the protocol, of help to *ES* regarding off-line dictionary attack are (r, e_0) and (s, e_2). However, $E_{CS}(.)$ is a semantic secure encryption, so the first pair does not help in dictionary attack; notice then that (e_z, e_2) is a standard ElGamal encryption. As widely known, it is also semantic secure when $g \in QR_p$ and $(\pi + s \bmod p) \in QR_p$ as in our protocol. Therefore, *ES* is not effective in off-line dictionary attack as long as he follows the protocol (behaving as a passive adversary).

As an active adversary, *ES* can modify or forge protocol transcript. To see this, *ES* may pick x of its choice and computes e_x, and in turn makes a dubious e_0, in an attempt to deceive *CS* into replying with e_2 under his e_x. This cheating however cannot succeed, due to the fact that without knowing π, *ES* is not able to convince *CS* of $a = b$. We do notice that in such a way, *ES* can launch *on-line* dictionary attack by repeatedly guessing passwords, and engaging in the protocol for testing $a \stackrel{?}{=} b$: each time *CS* returns $\perp$ (reject), *ES* is assured to exclude a password from the dictionary. However, it is clear that such attacks are *unavoidable* in any password-enabled system, but can be readily thwarted by limiting the number of unsuccessful authentication attempts (regarding a same user) made by *ES*.

Security to outside adversaries:

While no more effective than *ES* in terms of dictionary attack, an outside adversary could attempt to acquire the common key K established between U and *ES*. This is another common attack to authentication and key exchange protocols. In our protocol, as the adversary does not know π, he has no way to negotiate a dubious common session key with *ES* in the name of U. What remains to consider is the scenario that the adversary derives the session key K by watching the protocol transcript between U and *ES*. This in our case is clearly equivalent to breaking the Diffie-Hellman assumption: by given $g^x \bmod p$ and $g^y \bmod p$, to compute $g^{xy} \bmod p$ without knowing x and y.

4.4.4 Extension

We introduce briefly how to adapt the protocol to the extended architecture in Figure 3 that includes a group of central servers. The extension turns out to be straightforward. The central servers work under a t-out-of-n threshold secret sharing scheme [18], each keeping a share of $\pi + s$ that would be otherwise preserved by a single central server. At the time an external server requests for user authentication, one of the servers volunteers to be the *dealer*, managing the reconstruction of $\pi + s$. The voluntary dealer is the only one interacting with the requesting external server. While the dealer could become a single point of vulnerability, compromise of it actually affects solely those $\pi + s$ that had ever been reconstructed on it. Furthermore, as already discussed, the chance of compromising a central server is practically minor. After all, this extension at the central server side is actually aimed at solving possible bottleneck problems and break-downs due to a single central server.

4.5 Discussions

The proposed protocol enjoys several advantages. Among others, first, the protocol is particularly efficient to the users in terms of both communication and computation. As to computation, a user only needs to compute 2 one-line exponentiations, and 1 off-line exponentiation and 1 off-line public key encryption. This is important when consider to support resource-constrained users, e.g., mobile phones. The communication to the users is optimal: only one round of interaction is involved. Second, a user can use the same password to register to different enterprises or to different affiliating organizations in a same enterprise (by varying s). This avoids a big inconvenience in traditional password-enabled systems (e.g., those reviewed in Section 2), where a user has to memorize different passwords for different applications.

We next clarify a possible argument why we do not simply rely on the central server(s) for full trust management of the affiliating organizations, a paradigm similar to Kerberos [2]. The reasons are as follows: first, each affiliating organization has its own business interest, so it has a stake to involve into the trust management of its own; second and more important, a main objective of our architecture is to avoid a single point of vulnerability.

Finally, while the assumption of *CS* being an honest-but-curious adversary well represents the different levels of trust upon an enterprise and its affiliating organization, it is a strong one. Design of an authentication and key exchange protocol in the case of *CS* being also a malicious adversary (e.g., allowed to wiretap the communication between *U* and *ES*) is an open problem.

5. CONCLUSIONS

We explored applying authentication and key exchange using password to federated enterprises. Taking advantage of the special structure of a federated enterprise, a new architecture comprising an external server and a central server was proposed. A user authentication and key exchange protocol using password that is geared to the architecture was presented. Attention was focused on resisting off-line dictionary attacks by the servers, a topic rarely considered in previous effort. Our proposed architecture and protocol enjoyed several attractive features.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] L. Bouganim, P. Pucheral, Chip-Secured Data Access: Confidential Data on Untrusted Servers, in: *Very Large Data Bases (VLDB)*, pp. 131-142, 2002.

[2] D. V. Klein, Foiling the Cracker - A Survey of, and Improvements to, Password Security, in: *2nd USENIX Security*, pp. 5-14, 1990

[3] E. Bresson, O. Chevassut, and D. Pointcheval, Security Proofs for an Efficient Password-Based Key Exchange, in: *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, in: *IEEE Symposium on Research in Security and Privacy*, pp. 72-84, 1992.

[5] S. Bellovin and M. Merritt, Augmented Encrypted Key Exchange: A Password-Based Protocol Secure Against Dictionary Attacks and Password File Compromise, in: *ACM. Computer and Communication Security*, pp. 244-250, 1993.

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

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

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

[9] J. Katz, R. Ostrovsky, and M. Yung, Forward Secrecy in Password-Only Key Exchange Protocols, in: *Security in Communication Networks*, 2002

[10] M. Bellare, D. Pointcheval, and P. Rogaway, Authenticated Key Exchange Secure against Dictionary Attacks, in: *Advance in cryptology, Eurocrypt'00*, pp. 139-155, 2000.

[11] 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.

[12] W. Ford, and B. S. Kaliski Jr, Sever-assisted Generation of a Strong Secret from a Password, in: *IEEE. 9th International Workshop on Enabling Technologies*, 2000.

[13] D. P. Jablon, Password Authentication Using Multiple Servers, in: *RSA Security Conference*, LNCS 2020, pp. 344-360, 2001.

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

[15] M. D. Raimondo, and R. Gennaro, Provably Secure Threshold Password-Authenticated Key Exchange, in: *Advances in Cryptology, Eurocrypt'03*, LNCS 2656, pp. 507-523, 2003.

[16] M. Bellare, P. Rogaway, Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, in: *ACM. Computer and Communication Security*, pp. 62-73, 1993.

[17] J. Brainard, A. Juels, and B. Kaliski, M. Szydlo, A New Two-Server Approach for Authentication with Short Secret, in: *USENIX Security*, 2003.

[18] A. Shamir, How To Share A Secret, *Communications of the ACM*, Volume 22, pp. 612-613, 1979.
[19] O. Goldreich, Secure Multi-party Computation, Working Draft, Version 1.3, June 2001.
[20] D. Boneh, The Decision Diffie-Hellman Problem, in: *3rd International Algorithmic Number Theory Symposium*, LNCS 1423, pp. 48-63, 1998.
[21] J. Kohl, and C. Neuman, RFC 1510: The Kerberos Network Authentication Service, 1993.