

# Role Sharing in Password-Enabled PKI

Xunhua Wang<sup>†</sup>, Samuel Redwine  
Commonwealth Information Security Center &  
Department of Computer Science  
James Madison University  
Harrisonburg, VA 22807 USA  
{wangxx,redwinst}@jmu.edu

## Abstract

Password-enabled PKI schemes simplify the management of end users' private keys by storing them in password-protected form on a centralized on-line server. Under such schemes an end user needs only remember his password and can access his private key from anywhere the centralized server is available. Existing password-enabled PKI schemes are based on the single-user model where a private key is owned by one user. In this article, we present mechanisms to support role sharing in password-enabled PKI. In our schemes, using passwords only, a group of users share the privileges of a role through sharing the private key of that role. We first develop a hybrid password-enabled PKI scheme, which supports both easy password change and misuse monitoring. Then, based on this hybrid and existing password-enabled PKI schemes, we give password-enabled role sharing schemes for both threshold access structures that require a threshold number of these users to execute the shared role and more general access structures that allow more flexible role sharing policies.

**Keywords:** Role sharing, Password-enabled PKI, PAKE

## 1 Introduction

In a password-enabled PKI scheme [22, 17], the private key of an end user is not stored on a smart-card or on the user's laptop. Instead, it is protected by a password chosen by the user and stored on a centralized online server. Compared to the conventional smartcard-based PKI approach, password-enabled PKI is a lightweight solution and enjoys high usability: no smartcard reader is required; an end user needs only remember his password and can roam anywhere the centralized server is available.

There are two different approaches for password-enabled PKI, *virtual soft token* [20, 17] and *virtual smartcard* [22]. In the virtual soft token PKI [20, 17], a password is used to encrypt the private key of a public/private key pair and the encrypted private key is stored on a centralized server. With his password, a user can remotely authenticate himself to the server, establish an authenticated and cryptographically strong session key (thus, a secure connection) with the server, download the encrypted private key via the secure channel, decrypt it and use the private key as in the conventional PKI activities. The first step of this approach authenticates a user before he can download a password-encrypted private key and the second step establishes a session key to protect the subsequent downloading of the password-encrypted private key from the off-line

---

<sup>†</sup>A part of this work has been supported by a Cisco CIAG grant.

dictionary attack [19]. These two steps can be accomplished by a *password-authenticated key exchange (PAKE)* protocol [2, 16, 30].

In the virtual smartcard PKI [22], an end user's private key is split into two parts, a *human memorizable password* and a *server component*. The end user holds the password and the server component is stored on a server. Like in the virtual soft token, to use his private key, a user of the virtual smartcard PKI first runs the PAKE protocol with the server to have mutual authentication and establish a secure channel. Then, the user applies his password to a message (either a message to be digitally signed or a ciphertext to be decrypted) and sends the partial result to the server over the secure channel. The server combines its own partial result, computed from the corresponding server component and the message, with the user's partial result to generate the final result. The major difference between virtual soft token schemes and virtual smartcard schemes is that, in virtual smartcard schemes, every cryptographic operation (such as digital signature and decryption) requires the cooperation of the centralized server while in virtual soft token schemes an end user can do many cryptographic operations as he wants after securely downloading his private key.

**The problem.** All existing password-enabled PKI schemes [20, 17, 22] are based on the one-user-one-private-key model and are essentially *single user-oriented*. That is, they do not support multiple-users-one-private-key and thus do not support *role sharing*.

A *role*, in the access control community, is defined as a basic semantic unit to describe the authority and responsibility that users of that role assume [24]. A good organization-wide access control decision is often based on roles (for example, president of AOL), instead of any specific individual user, as users may change over time while roles change less frequently. In many role-based access control models [24, 23], a user assigned to a specific role is implicitly granted all the privileges of that role. However, as pointed out in [6], within an organization, the responsibility of a role is not always assumed by any single individual but sometimes is shared among a group of users of that organization. To execute the role privileges, a subgroup of these users are required to agree on the action. In this way, power abuse by a single user or a small coalition of these users can be prevented and the principle of separation of duty can be guaranteed. We consider the case where a public/private key pair, called *role public/private key pair*, is affiliated with the role. The role public key is used by external users to encrypt messages intended for this role or verify messages digitally signed by the role private key. For users external to the role, what they see is the role itself and the users assigned to this role are invisible to them. Possibly, when collectively executing the role privileges, the users sharing a role do not necessarily trust each other.

Based on the above observation, in this article, we explore password-enabled PKI schemes to support role sharing, which are called *password-enabled role sharing PKI*.

**Our contribution.** We first propose a new password-enabled PKI scheme, called *hybrid password-enabled PKI*, which is later extended to support role sharing. Compared to existing password-enabled PKI schemes [20, 22, 17], this hybrid scheme allows server administrators to perform instant revocation of a user's public key and to monitor user PKI activities for misuse detection and, at the same time, it supports user password change very well. (Previous password-enabled PKI schemes support only one of these two features.) Then, we propose *password-enabled role sharing PKI* schemes for both threshold access structure and general access structure. In the scheme for threshold access structure, which is called *threshold password-enabled role sharing PKI*, a group of (say  $n$ ) users are assigned to a role (and thus share the role private key), each with his favorite password and nothing else, and a threshold (say,  $t$ ,  $t \leq n$ ) of them are required to cooperate to execute the role privileges without reconstructing the shared role private key at any single location. Like the traditional single user-oriented password-enabled PKI schemes, our architecture adopts a central server, where password-protected credentials are stored. A subset of users fewer than  $t$ , together with the centralized server, will not be able to use the role private key directly. In

Table 1: Comparison of our work with previous research

Single User-oriented Password-enabled PKI		Password-enabled Role Sharing PKI	
Name	Property	Name	Property
Virtual soft token	easy password change	Threshold virtual soft token <sup>‡</sup>	easy
		Type-1 password-enabled role sharing PKI for general access structure <sup>‡</sup>	password change
Virtual smartcard	misuse detection		
Hybrid password-enabled PKI <sup>‡</sup>	easy password change & misuse detection	Threshold hybrid password-enabled PKI <sup>‡</sup>	easy password change & misuse detection
		Type-2 password-enabled role sharing PKI for general access structure <sup>‡</sup>	

this article we also propose password-enabled role sharing schemes to support more general access structure, in which more flexible role sharing policies are allowed.

It should be noted that our role-sharing schemes are different from the voting-based role sharing approach, where a *fully trusted* centralized server checks the votes from a subgroup of users and, if a certain condition is met, executes the role privilege. In our schemes, the centralized server is not fully trusted and the server itself alone cannot directly execute the role. Thus, neither the central server administrator nor an attacker who has successfully compromised the server can assume the role directly. Table 1 gives comparison between this work (marked with <sup>‡</sup>) and previous research. The first two columns of table 1 give the single user-oriented password-enabled PKI schemes, including the hybrid password-enabled PKI proposed in this paper, and the last two columns of the table list their corresponding extensions for role sharing.

The remainder of this article is organized as follows. Section 2 gives the related work. Section 3 presents a hybrid password-enabled PKI scheme. Section 4 discusses some principles for designing role sharing password-enabled PKI schemes. Section 5 presents our role sharing password-enabled PKI schemes for threshold access structure and 6 gives our role sharing password-enabled PKI schemes for general access structures. In Section 7 we discuss some operational and performance issues. Concluding remarks are given in Section 8.

## 2 Related Work

Desmedt [6] first proposed the concept of group-oriented cryptography to allow a threshold number of users sharing a group private key. In all the threshold cryptography schemes, including those threshold RSA [10, 8, 9, 21, 26, 14] and threshold DSS [12, 13, 18] schemes, each user of the role is assigned one or more *long* random secret shares of the role private key. Since most human being are not good at memorizing long random secrets and smart-cards have not been widely used yet, so far these threshold cryptography schemes have only been used in machine-oriented applications [31, 1, 29], not people-oriented systems. In contrast, the schemes explored in this article are password-based and thus, people-oriented.

Ganesan [11] first introduced passwords into the 2-out-of-2 threshold RSA [5] and used it to enhance the Kerberos system. We notice that this enhancement, like [22], is still single-user oriented and does not support role sharing.

Using a 2-out-of-2 threshold RSA scheme Boneh et al. [4] proposed an architecture for fast public key revocation. In their architecture is a semi-trusted mediator (SEM) who can monitor a user's

PKI activities. Compared to this scheme, our hybrid password-enabled PKI scheme (presented in Section 3) is password-enabled and thus enjoys better usability.

### 3 A New Password-Enabled PKI Scheme

The virtual soft token PKI scheme proposed in [20] allows its users to change their password easily. However, the administrators of its centralized server cannot monitor users' PKI activities as a user can perform many PKI operations after downloading his private key. On the other hand, the virtual smartcard PKI scheme proposed in [22] allows the administrators of the centralized server to monitor user PKI activities and supports instant public key revocation. However, as observed in [28], user password change is not supported very well as it is computation intensive.

In this section, we propose a hybrid password-enabled PKI scheme that supports both PKI activity monitoring and simple password change. The essential idea behind the hybrid password-enabled PKI scheme is that an additive 2-out-of-2 secret sharing is performed on the user's private key first and one of the two resulting shares, called the *server component*, is assigned to the centralized server. The other share, called the *client component*, is assigned to the user and is encrypted with the user's password and stored on the centralized server. When the user needs to use his private key, he securely downloads the password-encrypted key share, as done in virtual soft token, decrypts the key share and uses it to compute a partial result. To get the final result, the user needs the cooperation of the centralized server, which uses the server component as in the virtual smartcard scheme. Thus, this hybrid password-enabled PKI scheme is similar to the virtual smartcard scheme [22] in that the centralized server is also assigned a component of the user's private key; on the other hand, it is also similar to the virtual soft token [20] in that a user needs to download a password-encrypted credential to perform the client-side computation, which makes password change simpler.

Below we give the details of the RSA-type hybrid password-enabled PKI scheme. The same idea can be used to build DSA-type hybrid password-enabled PKI scheme but it is more complicated [18].

**RSA-type hybrid password-enabled PKI** Assume that Alice is a user of the hybrid password-enabled PKI scheme and her RSA public key is  $(N, e)$ , where  $N = p \times q$ ,  $p$  and  $q$  are two primes.  $d$  is Alice's corresponding private key.

- **Component generation.** In our hybrid password-enabled PKI scheme, the centralized server picks a random  $r$ ,  $1 \leq r \leq \phi(N)$  where  $\phi(N) = (p - 1) \times (q - 1)$ , and computes  $r' = d - r \bmod \phi(N)$ .  $r$  is the server component and  $r'$  is the client component. Alice picks her favorite password  $\hat{p}$  and uses it to encrypt  $r'$ . The password-encrypted result,  $y = E_{\hat{p}}(r')$ , is stored on the server. For Alice, the centralized server also stores a password verification data which is a value derived from  $\hat{p}$  and is used by the server to run a PAKE protocol with Alice.
- **Private key use.** Armed with her password,  $\hat{p}$ , Alice runs a PAKE protocol with the centralized server and establishes a secure channel. She then securely downloads  $y$  and decrypts at the client side to recover  $r'$ . To use her private key to perform a cryptographic operation on a message  $m$ , Alice first applies  $r'$  to  $m$  to get a partial result  $c_1 = m^{r'} \bmod N$ .  $c_1$  is sent to the centralized server via the secure channel and the server applies its  $r$  to  $m$  to get  $c_2 = m^r \bmod N$ . The final result is  $c_1 \times c_2 \bmod N$ , which is equivalent to applying Alice's private key on message  $m$ .

In the above process, the centralized server is required to participate in the computation, which allows the centralized server administrator to monitor users' PKI activities and do instant public key revocation. On the other hand, Alice can change her password  $\hat{p}$  to another password  $\bar{p}$  by downloading  $y$ , recovering  $r'$ , computing  $y' = E_{\bar{p}}(r)$  and sending it to the centralized server. None of these steps is computation intensive and can be simply performed.

It is worth mentioning that in this hybrid scheme, every cryptographic operation related to the private key requires interactions with the centralized server. In contrast, with a virtual soft token, a user can load his private key onto the laptop and work offline, decrypting emails and signing new messages with no further interactions with the server.

## 4 Design Principles for Password-enabled Role Sharing PKI

In the remainder of this paper,  $PU_{role}$  and  $PK_{role}$  are used to denote the role public key and the role private key respectively.  $n$  is the size of the group of users to share the role and we use  $\mathcal{P} = \{U_1, U_2, \dots, U_n\}$  to denote the set of the users. An *authorized subset* is defined as a subset of  $\mathcal{P}$  whose users are allowed to collectively execute the role privileges and an *access structure*,  $\Gamma$ , for the role is the set of authorized subsets [27, pages 331].

### 4.1 Centralized server

Besides the users to share a role, in our architecture, there is a centralized on-line server, as in the traditional single user-oriented password-enabled PKI schemes [20, 17, 22]. It is this on-line server that makes password-enabling possible. On the other hand, this on-line server is not fully trusted in the sense that role-related credentials are *not* stored in the clear on it, but protected by passwords, and the private credentials are never exposed on the server. This distinction differentiates both the traditional password-enabled PKI and our schemes from the voting-based approach where the server is fully trusted.

For each user sharing a role, after he picks a password, the centralized server also stores the corresponding password verification data (PVD) for that user.

### 4.2 Design principles

There are several design principles for our password-enabled role sharing schemes. Some are straightforward while others are not.

1. The role public/private key pair does not change as often as the users assigned to the role. This is the rationale for role-based access control and is also true in our role sharing password-enabled PKI schemes.
2. A user revoked from a role should *not* know the shared role private key. Nor do a small coalition of users who have been revoked from the same role and who are not in the access structure anymore. Obviously, virtual soft token [20] does not meet this principle.
3. Users sharing a role possess passwords only and nothing more. All operations related to the role need the explicit permission of users from an authorized subset.
4. No full trust is placed on the centralized server. The server should not know the role private key. Thus, its administrators or a hacker who has compromised the server cannot execute

the role privilege in a simple way. On the other hand, using what’s stored on the server, the server administrators can mount off-line dictionary attacks. This characteristics is common to all password-based schemes and can be mitigated using multiple servers [28]. We notice that *not* all passwords are vulnerable to off-line dictionary attacks. Moreover, compared to the traditional single user-oriented password-enabled PKI schemes, in our role sharing schemes, it is harder for a malicious server administrator or a hacker who has taken control the server to mount off-line dictionary attacks as multiple, instead of a single, passwords are involved.

5. Both threshold access structure and general access structure should be supported. In a threshold access structure, any subset of size not less than the threshold is an authorized subset and this access structure is commonly used. On the other hand, threshold access structure is not always applicable and sometimes more general access structure is used.

## 5 Threshold Password-enabled PKI

In this section, we shall present *threshold password-enabled PKI* schemes. In the following discussion,  $t$ ,  $t \leq n$ , is the threshold. We first discuss how to add role sharing support to the virtual soft token scheme [20]. We then extend the hybrid password-enabled PKI proposed in Section 3 to support role sharing.

### 5.1 Threshold virtual soft token

Threshold virtual soft token is the role sharing extension of the virtual soft token scheme [20]. Threshold cryptography schemes [7, 12, 26] are used to for this purpose. We have two types of threshold virtual soft tokens, the *threshold virtual RSA soft token* for RSA-type role public/private key pair and the *threshold virtual DSA soft token* for DSA-type role key.

In a threshold virtual RSA (DSA) soft token, a role RSA (DSA) public/private key pair is first generated and then shares of the role private key,  $PK_{role}$ , are generated through a  $(t, n)$  Shamir secret sharing [25],  $(s_1, s_2, \dots, s_n) \xrightarrow{(t,n)} PK_{role} \bmod \phi(N)^*$  where  $s_i$  are the shares. Each user  $U_i$ ,  $1 \leq i \leq n$ , picks his password,  $\hat{p}_i$ , and his corresponding password verification data,  $PVD_i$ , is generated and stored on the centralized server. For each user, also stored on the centralized server is  $y_i$ , the encryption of  $s_i$  by  $\hat{p}_i$  (that is,  $y_i = E_{\hat{p}_i}(s_i)$ ).

When the role privilege needs to be executed, depending on the threshold cryptography scheme employed, users’ steps vary. In our following discussions we use the threshold RSA given in [26], called Sho00, and the threshold DSA scheme given in [12, 13], called GJKR96.

**Threshold virtual RSA soft token** To authorize a role-related operation,  $t$  or more of the  $n$  users are required. Let  $m$  be the message to be processed by the role private key. Each participating user first uses his password to run a PAKE protocol with the centralized server and establishes a secure connection; he then securely downloads the password-encrypted key share  $s_i$ , decrypts it and computes a partial result as  $c_i = m^{2\Delta s_i} \bmod N$  where  $\Delta = n!$  [26];  $c_i$  is sent back to the centralized server over the secure channel. After collecting enough partial results, the centralized server combines them into the final result. The Sho00 threshold RSA is non-interactive and thus, in the role execution, users do not need to interact with others.

---

\*For DSA, the modulus is the DSA system parameter  $q$ .

**Threshold virtual DSA soft token** When  $t$  or more users want to collectively authorize a role-related operation on message  $m$ , each of them first runs a PAKE protocol to log onto the centralized server, securely downloads his  $y_i$  and decrypts it as  $s_i$ . They then use the GJKR96 threshold DSA to collectively generate a DSA signature on  $m$ . The GJKR96 threshold DSA is an interactive scheme while, in our applications, interactions between users are not desirable. Fortunately, we observe that the interactive computation (all the steps until the computation of  $r$  [13, pages 70]) of the GJKR96 threshold DSA scheme are message-independent and can be pre-computed. Based on this observation, in our threshold virtual DSA soft token, we can avoid user interactions by performing the message-independent interactive computations in a *partially-protected* store-and-forward way: all the broadcast messages by user  $U_i$  are sent to the centralized server in the clear, which will be forwarded to other participating users by the server, and all the *intermediate* private messages of  $U_i$  are encrypted by  $U_i$ 's password before they are sent to and stored on the server (for future use). These pre-computations need *no* input from users and can be performed, without user  $U_i$ 's interventions and notices, after  $U_i$  logs into the system.

In GJKR96,  $b$ , the number of users required for a threshold DSA signature, is  $(2t - 1)$ , not  $t$ . That is,  $t$  should satisfy that  $t < \frac{n}{2}$ . Therefore, in our threshold virtual DSA soft token scheme,  $(2t - 1)$  users are required to collectively execute the shared role.

Both the threshold virtual RSA soft token and threshold virtual DSA soft token allow a user to change his password while keeping his role private key share unchanged. To change his password,  $U_i$  uses his old password to run a PAKE protocol with the server, securely downloads the key share protected by the old password, decrypts it, re-encrypts it with his new password, and securely uploads it to the server. To change his password, the user should also notify, via the secure connection, the server of his new PVD.

In the above threshold virtual soft token schemes, although the centralized server is used as a working platform, it is not assigned a share of the role private key and does not contribute to the final result.

## 5.2 Threshold hybrid password-enabled PKI

In a virtual smartcard scheme [22], the centralized server is also assigned a share of the user's private key and is required to participate in the computation when the user's private key is used. This allows an administrator of the central server to monitor the use of the user's private key and to instantly disable the user's private key if his public key is revoked. (In contrast, the virtual soft token [20] scheme does not offer this monitoring granularity since the private key is recovered and used on the user's machine.)

It is not immediately obvious on how to extend the virtual smartcard scheme given in [22] to support password-enabled role sharing. In a  $(t, n)$  Shamir secret sharing scheme [25], to share a secret, at most  $(t - 1)$  shares can be passwords. This fact prevents us from simply extending the virtual smartcard scheme for password-enabled role sharing since, ideally, in a password-enabled role sharing scheme, all the  $n$ , not just  $(t - 1)$ , users hold their favorite passwords only and nothing else. One might think to apply the following extension to the virtual smartcard scheme: for each combination  $U_{i_1}, U_{i_2}, \dots, U_{i_t}$ , where  $\{i_1, i_2, \dots, i_t\} \subset \{1, 2, \dots, n\}$ , we can compute  $d_{\{i_1, i_2, \dots, i_t\}} = d - \hat{p}_{i_1} - \hat{p}_{i_2} - \dots - \hat{p}_{i_t} \bmod \phi(N)$ , where  $\hat{p}_{i_j}$  is the password of  $U_{i_j}$ ,  $1 \leq j \leq t$ , and store  $d_{\{i_1, i_2, \dots, i_t\}}$  on the server. In this way, any  $t$  users can cooperate with the server to collectively apply the shared role private key on a message. However, this extension has a security flaw: it stores  $\binom{n}{t}$  such  $d_{\{i_1, i_2, \dots, i_t\}}$  values on the server and in some cases the server will be able to restore the role private key from them, which contradicts with our design principle 4 (see Section 4).

On the other hand, the hybrid password-enabled PKI scheme proposed in Section 3 can be extended to support role sharing for threshold access structure, which allows both monitoring granularity and easy password change. The following details are based on the RSA-type hybrid password-enabled PKI give in Section 3.

- **Component generation.** After the role RSA public/private key pair  $(PU_{role}, PK_{role} = d)$  is generated, a random  $r$ ,  $1 \leq r \leq \phi(N)$ , is generated and  $r'$  is computed as  $r' = d - r \bmod \phi(N)$ .  $r$  is the server component and is stored on the centralized server. A  $(t, n)$  Shamir secret sharing is performed on the client component  $r'$ ,  $r' \xrightarrow{(t,n)} (s_1, s_2, \dots, s_n) \bmod \phi(N)$  and  $s_i$  is assigned to  $U_i$ ,  $1 \leq i \leq n$ . Each user  $U_i$  picks his password,  $\hat{p}_i$ , and it is used to encrypt  $s_i$  into  $y_i = E_{\hat{p}_i}(s_i)$ . For user  $U_i$ ,  $y_i$  and a password verification data derived from  $\hat{p}_i$  are stored on the centralized server.
- **Private key use.** When  $t$  or more users agree to apply the role's private key on a message  $m$ , each of them uses his  $\hat{p}_i$  to run a PAKE protocol with the centralized server and establish a secure channel. He then securely downloads  $y_i$  and decrypts at the client side to recover  $s_i$ . He then first applies  $s_i$  to  $m$  and gets a partial result  $c_{1i} = m^{s_i} \bmod N$ .  $c_{1i}$  is sent to the centralized server via the secure channel. The server also applies its  $r$  to  $m$  to get  $c_2 = m^r \bmod N$ . After collecting enough partial results, the centralized server combines all partial results,  $c_{1i}$  and  $c_2$  into the final result, which is exactly of the role's private key on message  $m$ .

In the above process, the centralized server is required to participate, which allows the centralized server administrator to monitor the role's PKI activities and do instant public key revocation. On the other hand, each user can change his password  $\hat{p}_i$  to another password  $\bar{p}_i$  by downloading  $y_i$ , recovering  $s_i$ , computing  $y'_i = E_{\bar{p}_i}(s_i)$  and sending it to the centralized server. None of these steps is computation intensive and can be simply performed.

## 6 Password-enabled Role Sharing for General Access Structure

In our real world not all access structures are threshold-based and sometimes more general access structures are used. For example, four users,  $(U_1, U_2, U_3, U_4)$ , share a role and  $\Gamma = \{\{U_1, U_2\}, \{U_1, U_3, U_4\}\}$  is its access structure. This access structure is not threshold:  $\{U_1, U_2\}$  has two members and is allowed to execute the role while  $\{U_2, U_3, U_4\}$  is not allowed although its cardinality is 3.

In this section we discuss how to support password-enabled role sharing for general access structure. An access structure is said to be *monotone* if  $B \in \Gamma$  and  $B \subseteq C \subseteq \mathcal{P}$  implies  $C \in \Gamma$  [3]. We are only interested in monotone access structure here.

General access structure-oriented secret sharing — called *generalized secret sharing* — was first studied by Ito et al. [15]. Benaloh and Leichter [3] developed a simpler generalized secret sharing, which is called BL88 in the following discussion. It should be noted that password-enabled role sharing discussed here is more than secret sharing as we do not reconstruct a shared secret, as done in secret sharing schemes, since reconstruction leads to a single point of attack.

In the rest of this section we will give two types of password-enabled role sharing PKI schemes for general access structure. Our discussions are based on the RSA algorithm but can also be applied to DSA.

## 6.1 Type-1 password-enabled role sharing PKI

Type-1 password-enabled role sharing PKI for general access structure is the extension of the virtual soft token for role sharing.

- Component generation. Given a monotone access structure  $\Gamma$  for a role whose private key is  $PK_{role} = d$ , we first use the BL88 generalized secret sharing scheme to generate secret shares. Each of the  $n$  users will get one or more secret shares. Then, each of them picks his favorite password  $\hat{p}_i$  and uses it to encrypt all of his secret shares. The password-encrypted secret shares, together with a password verification data derived from  $\hat{p}_i$ , are stored on the centralized server.
- Private key use. When an authorized subset of users want to execute the role privilege on a message  $m$ , each of them,  $U_i$ , runs a PAKE protocol to log onto the centralized server and establishes a secure connection with it.  $U_i$  then securely downloads his secret shares and applies it to  $m$  to get a partial result. The partial result is securely sent to the centralized server who combines all partial results into a final result, which is equivalent to applying the role's private key on  $m$ .

Using the above  $\Gamma = \{\{U_1, U_2\}, \{U_1, U_3, U_4\}\}$  as an example, we have the following shares:  $d_1$  is assigned to  $U_1$ ,  $d_2$  is assigned to  $U_2$ ,  $d_3$  is assigned to  $U_3$ ,  $d_4$  is assigned to  $U_4$  where  $d_1 + d_2 = d \bmod \phi(N)$  and  $d_1 + d_3 + d_4 = d \bmod N$ . When  $U_1$  and  $U_2$  agree to apply the role private key to  $m$ ,  $U_1$  computes  $c_1 = m^{d_1} \bmod N$  and  $U_2$  computes  $c_2 = m^{d_2}$ . After receiving  $c_1$  and  $c_2$ , the centralized server combines them into the final result as  $c = c_1 \times c_2 \bmod N = m^d \bmod N$ , which is exactly the role private key on  $m$ .

In the above steps, the centralized server does not contribute to the final result and technically the step of combining partial results into the final result can be performed by any users. That is, type-1 password-enabled role sharing PKI does not provide a technical means to monitor role PKI activities on the centralized server.

## 6.2 Type-2 password-enabled role sharing PKI

Using the same idea of the hybrid password-enabled PKI, type-1 password-enabled role sharing PKI can be modified so that the centralized server is required to contribute for a role privilege execution. In the above component generation stage, instead of sharing  $d$ , we can first run a 2-out-of-2 additive secret sharing on  $d$  and get  $d'$  and  $d''$ , where  $d = d' + d'' \bmod \phi(N)$ .  $d'$  is the server component and is assigned to the centralized server. The client component,  $d''$ , is shared among the  $n$  users using the BL88 generalized secret sharing. Then each user uses his password to encrypt the shares assigned to him and stores the password-protected shares on the centralized server. When an authorized subset of users want to execute the role privilege, they compute their partial results. To get the final result, the centralized server is also required to participate and compute its own partial result. Thus, this modified scheme allows the centralized server administrators to monitor role PKI activities and is called *type-2 password-enabled role sharing PKI for general access structure*.

For general access structure, a user is likely to be assigned more than one secret shares and, in deciding which share to use, he needs to know the identities of others users of the authorized subset. This might be undesirable sometimes as it needs coordinations between the participating users. A method for  $U_i$  to avoid this interaction is to apply all of his secret shares to  $m$  to get more partial results than necessary. When the centralized server combines the partial results, only those necessary partial results will be used.

## 7 More Discussions

In this section we discuss some performance and operational issues.

### 7.1 Performance considerations

Compared to the virtual soft token [20] and the virtual smartcard scheme [22], the hybrid password-enabled PKI scheme does not introduce any additional significant computational cost.

### 7.2 Operational considerations

**Password-based versus smartcard-based.** Passwords are commonly used for authentication in our daily lives and support user roaming very well. Password-enabled PKI schemes integrate the roaming capability and good usability of passwords into PKI. However, in some application cases, smartcard-based solution might still be preferred due to its high-level security. For these applications, password-enabled PKI can be used as a short-term solution and the migration from password-based to smartcard-based can be made smooth.

In both threshold virtual soft token scheme and threshold hybrid password-enabled PKI scheme, an end user is assigned one or more shares of the role private key and the password-encrypted key shares are stored on the centralized server. This structure makes it easy for password and smartcards to co-exist and makes it easy to migrate from password-based to smartcard-based: users who prefer passwords can still hold their passwords and store their password-encrypted shares on the server; users who like smartcards can download their password-encrypted key shares and feed them to smartcards. This is also true for both type-1 and type-2 password-enabled role sharing PKI schemes.

**Recovery from password loss.** In a password-based system, a user might inadvertently lose his password to somebody else. For example, a user might use an insecure computer on which a key logging program is installed to harvest passwords. For single user-oriented password-enabled PKI, this might be disastrous. In contrast, the password-enabled role sharing PKI schemes tolerate this type of mistakes to some extent: as long as an attacker does not steal more than  $(t - 1)$  passwords, he will not be able to assume the shared role. The recovery from such loss is also straightforward: after a user loses his password, his old key share is disabled and any  $t$  other users who share the same role can help him get a new key share.

## 8 Conclusion

Conventional password-enabled PKI schemes are based on the one-private-key-one-user model and do not support role sharing. In this article we developed schemes to add role sharing to password-enabled PKI schemes. We first presented a hybrid password-enabled PKI scheme, which supports both easy password change and misuse monitoring. Then, we extended our hybrid and existing password-enabled PKI schemes to support role sharing. Our password-enabled role sharing PKI schemes support both threshold access structures and general access structures. Compared to conventional password-enabled PKI schemes, from an end user's perspective, our password-enabled role sharing PKI schemes do not incur additional significant computational cost and also tolerate end user's operational mistakes.

## 9 Acknowledgement

We wish to thank the anonymous reviewers for pointing us to the related work in [4] and for other useful comments.

## References

- [1] B. Barak, A. Herzberg, D. Naor, and E. Shai. The proactive security toolkit and applications. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 18–27, November 2–4 1999.
- [2] S. Bellare and M. Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 72–84, 1992.
- [3] J. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In S. Goldwasser, editor, *Advances in Cryptology - Crypto '88*, number 403 in *Lecture Notes in Computer Science*, pages 27–36, Berlin, 1988. Springer-Verlag.
- [4] D. Boneh, X. Ding, G. Tsudik, and C. M. Wong. A method for fast revocation of public key certificates and security capabilities. In *Proceedings of the 10th USENIX Security Symposium*, August 13–17 2001.
- [5] C. Boyd. Some applications of multiple key ciphers. In C. G. Gunther, editor, *Advances in Cryptology, Proc. of Eurocrypt '88*, volume 330 of *Lecture Notes in Computer Science*, pages 455–467, Davos, Switzerland, May 1988. Springer-Verlag.
- [6] Y. Desmedt. Society and group oriented cryptography: a new concept. In *Advances in Cryptology, Proc. of Crypto '87*, pages 120–127, August 16–20 1988.
- [7] Y. Desmedt. Some recent research aspects of threshold cryptography. In E. Okamoto, G. Davida, and M. Mambo, editors, *Information Security*, volume 1396 of *Lecture Notes in Computer Science*, pages 158–173, September 1997. URL <http://www.cs.fsu.edu/~desmedt/ISW.pdf>.
- [8] Y. Desmedt, G. Di Crescenzo, and M. Burmester. Multiplicative nonabelian sharing schemes and their application to threshold cryptography. In *Advances in Cryptology — Asiacrypt '94*, pages 21–32, November/December 1994.
- [9] Y. G. Desmedt and Y. Frankel. Homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM Journal on Discrete Mathematics*, 7(4):667–679, November 1994.
- [10] Y. Frankel and Y. Desmedt. Parallel reliable threshold multisignature. Tech. Report TR-92-04-02, Dept. of EE & CS, Univ. of Wisconsin-Milwaukee, April 1992. [ftp://ftp.cs.uwm.edu/pub/tech\\_reports/desmedt-rsa-threshold\\_92.ps](ftp://ftp.cs.uwm.edu/pub/tech_reports/desmedt-rsa-threshold_92.ps).
- [11] R. Ganesan. Yaksha: Augmenting Kerberos with public-key cryptography. In *Proceedings of the ISOC Network and Distributed Systems Security Symposium*, 1995.
- [12] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In *Advances in Cryptology — Eurocrypt '96*, pages 354–371, May 12–16 1996.

- [13] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. *Information and Computation*, 164(1):54–84, 2001.
- [14] N. Gilboa. Two party RSA key generation. In M. Wiener, editor, *Advances in Cryptology - CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 116–129, Santa Barbara, California, USA, August 1999.
- [15] M. Ito, A. Saito, and T. Nishizeki. Secret sharing schemes realizing general access structures. In *Proc. IEEE Global Telecommunications Conf., Globecom'87*, pages 99–102. IEEE Communications Soc. Press, 1987.
- [16] D. P. Jablon. Strong password-only authenticated key exchange. *ACM SIGCOMM Computer Communication Review*, 26(5):5–26, October 1996.
- [17] T. Kwon. Virtual software tokens - a practical way to secure PKI roaming. In G. Davida, Y. Frankel, and O. Rees, editors, *Proceedings of the Infrastructure Security (InfraSec)*, volume 2437 of *Lecture Notes in Computer Science*, pages 288–302. Springer-Verlag, 2002.
- [18] P. MacKenzie and M. Reiter. Two-party generation of DSA signatures (extended abstract). In J. Kilian, editor, *Advance in Cryptology - EUROCRYPT 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 137–154, Santa Barbara, CA, USA, August 2001. Springer.
- [19] R. Morris and K. Thompson. Password security: a case history. *Communications of the ACM*, 22(11):594–597, November 1979.
- [20] R. Perlman and C. Kaufman. Secure password-based protocol for downloading a private key. In *Proceedings of the ISOC Network and Distributed Systems Security Symposium*, 1999.
- [21] T. Rabin. A simplified approach to threshold and proactive RSA. In *Advances in Cryptology, Proc. of Crypto'98*, pages 89–104, August 23-27 1998.
- [22] R. Sandhu, M. Bellare, and R. Ganesan. Password enabled PKI: Virtual smartcards vs. virtual soft tokens. In *Proceedings of the 1st Annual PKI Research Workshop*, pages 89–96, April 2002.
- [23] R. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and Systems Security*, 2(1):105–135, February 1999.
- [24] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.
- [25] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.
- [26] V. Shoup. Practical threshold signatures. In *Advance in Cryptology - EUROCRYPT 2000*, pages 207–220, May 2000.
- [27] D. R. Stinson. *Cryptography: Theory and Practice*. CRC, Boca Raton, 1st edition, 1995.
- [28] X. Wang. Intrusion-tolerant password-enabled PKI. In *Proceedings of the 2nd Annual PKI Research Workshop*, pages 44–53, Gaithersburg, MD, USA, April 28–29 2003. Natl Inst. of Standards and Technology.

- [29] X. Wang, Y. Huang, Y. Desmedt, and D. Rine. Enabling secure on-line DNS dynamic update. In *Proceedings of the 16<sup>th</sup> Annual Computer Security Applications Conference*, pages 52–58, New Orleans, Louisiana, USA, December 11-15 2000. IEEE Computer Society Press.
- [30] T. Wu. The secure remote password protocol. In *Proceedings of the 1998 Network and Distributed System Security Symposium*, pages 97–111, 1998.
- [31] T. Wu, M. Malkin, and D. Boneh. Building intrusion tolerant applications. In *Proceedings of the 8th USENIX Security Symposium*, pages 79–91, 1999.