

Intrusion-Tolerant Password-Enabled PKI[§]

Xunhua Wang

Commonwealth Information Security Center &

Department of Computer Science

James Madison University

Harrisonburg, VA 22807 USA

wangxx@jmu.edu

Abstract

Password-enabled PKI facilitates the private key management by integrating easy-to-use passwords into PKI. In the first PKI research workshop, Sandhu *et al.* categorized password-enabled PKI schemes as virtual soft tokens and virtual smartcards [26]. Compared to the conventional PKI, password-enabled PKI introduces a security-critical server where large number of password-related credentials are stored. The compromise of this server will render these password-based credentials susceptible to the dictionary attack and, thus, damage the security of numerous private keys. In this article, using multiple servers, we propose an intrusion-tolerant virtual soft token scheme and an intrusion-tolerant virtual smartcard scheme. In our schemes, compromising up to a threshold number of these servers will not help an attacker mount a dictionary attack and, compared to previous work, our schemes can still function in the presence of some server failures. The multiple servers introduced in our intrusion-tolerant password-enabled PKI can be easily managed and PKI users can roam with human memorable passwords.

Keywords: Password-enabled PKI, Intrusion Tolerance, Virtual Soft Token, Virtual Smartcard

1 Introduction

In the conventional public key infrastructure (PKI), the private key of a public/private key pair is held

[§]This research is supported in part by a grant from the Virginia Commonwealth Technology Research Fund (SE 2001-1) through the Commonwealth Information Security Center, James Madison University.

by an end user (for digital signature or decryption) while the public key is certified as a digital certificate by a trusted third party. Ideally, the private key is stored in a smart card and should never leave the card when it is used. The user is capable of roaming easily with the smart card. However, PKI-based smartcards have not happened in the real world yet. Passwords, on the other hand, are commonly used for authentication in our daily lives and support user roaming very well. For instance, people have been using passwords for remote authentication over the Internet. To integrate passwords' convenience into PKI, two different approaches, called *virtual soft token* and *virtual smartcard*, have been proposed [25, 21, 26].

In the virtual soft token PKI [25, 21], a password is used to encrypt the private key of a public/private key pair and the encrypted private key is stored on a 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 since it is vulnerable to the dictionary attack [24]. These two steps can be accomplished by a *password-authenticated key exchange (PAKE)* protocol [2, 18, 31], in which a client (user) with a password and a server storing the related password verification data (PVD) can authenticate each other and establish a cryptographically strong session key to protect subsequent communication.

In the virtual smartcard PKI [26], an end user's pri-

private key is split into two parts, a *human memorable password* and a *secret component*. The end user holds the password and the secret component is stored on a server. Let (N, e) be a RSA public key and d is the corresponding private key ($d \times e = 1 \pmod{\phi(N)}$ and ϕ is the Euler function). In [26], d is split into a password-derived value d_1 and another value d_2 , $d = d_1 \times d_2 \pmod{\phi(N)}$, and d_2 is stored on a server. Note that $m^d = (m^{d_1})^{d_2} \pmod{N} = (m^{d_2})^{d_1} \pmod{N}$. To perform a cryptographic operation (digital signature or decryption) on a message m by the user's private key, the end user first authenticates himself to the server using the password and establishes an authenticated session key (thus, a secure connection) with the server. (Again, this can be accomplished by a PAKE protocol.) After securely receiving m from the user, the server applies the secret component, d_2 , to m to get a partial result $c_2 = m^{d_2} \pmod{N}$. c_2 is then passed back to the user through the secure channel. In the end, the user derives d_1 from his password and computes the final result as $c = c_2^{d_1} = m^{d_2 \times d_1} = m^d \pmod{N}$. Note that, in the above process, the overall value of the private key, d , is never reconstructed on the client nor on the server. Both the virtual soft token and virtual smartcard allow a user to *roam* with a memorable password solely and digitally sign a message (with a long-term private key) at a new location.

The problem. People tend to choose easily memorable passwords (from a dictionary) and thus, password-based systems are notoriously vulnerable to the *dictionary attack* [24], in which an attacker does not brute-force all possible passwords but rather work on a much smaller dictionary of likely passwords*. Compared to the conventional PKI, password-enabled PKI introduces a security-critical server where both password-verification data (PVD) and password-related credentials (password-encrypted private keys in the virtual soft token PKI and secret components in the virtual smartcard PKI) are stored. This makes it subject to the *server compromise-based dictionary attack*: after breaking into this server and stealing the password verification data and password-based credentials, an attacker can mount dictionary attacks to find the password and recover the private key.

For the virtual soft token, if the server is compromised and the password-encrypted private key is

*For a specific user, his password may not always fall within an attacker's dictionary. But an alarmingly high fraction of the actual passwords match passwords in a constructed dictionary [20].

stolen, an attacker can guess a password, use it to decrypt the stolen credential and verify the correctness of the guessing by checking the decrypting result with the corresponding public key ((N, e) for RSA). If an attacker also steals the password-verification data, the attack will be simpler: the attacker can simply guess a likely password, compute the corresponding PVD and compare it against the stolen PVD. If he observes a match, then he finds the password and it can be used to decrypt the stolen password-encrypted private keys.

As for the virtual smartcard PKI, if the server is compromised and d_2 is stolen, an attacker can simply guess a likely password, derive d'_1 from it, pick a random m ($1 < m \leq (N - 1)$), compute $c' = m^{d'_1 \times d_2} \pmod{N}$, and verify the correctness of the guessing by checking if $m = c'^e \pmod{N}$ holds. In this case, an attacker can mount dictionary attacks against thousands, if not millions, of users' password-protected private keys stored on the corrupted server. If an attacker also steals the password-verification data (PVD), the attack will be simpler: the attacker can simply guess a likely password, compute the corresponding PVD and compare it against the stolen PVD. If he observes a match, then he finds the password, which can be used with the stolen d_2 to recover the private key. It is worth noting that, compared to the virtual soft token PKI, the dictionary attack against the virtual smartcard PKI is more subtle. An attacker does not need d_2 to mount an off-line dictionary attack and m^{d_2} for any public message m will be sufficient: if an attacker obtains the value of $c_2 = m^{d_2} \pmod{N}$ for some m , he can simply guess a likely password, derive d'_1 from it, compute $c' = c_2^{d'_1} \pmod{N}$, and verify the correctness of the guessing by checking if $m = c'^e \pmod{N}$ holds.

Proactive password checking mitigates the dictionary attack problem but it does not fully solve it. Wu [32] showed that a proactive password checking system still allows about 8.28% of its passwords susceptible to the dictionary attack. Considering the large number of password-verification data and password-related credentials stored on the server, this server compromise-based dictionary attack could be large-scale and catastrophic. We argue that intense monitoring of the server may not be sufficient and server compromise (by outside attackers, inside attackers or through the mistakes of honest insiders) seems inevitable. For instance, an attacker might gain the `root` privilege of the server by exploiting bugs in server software (for instance,

through bugs [6, 7] in the Wu-FTP ftp server, bug [10] in the Apache web server, bug [8] in Microsoft IIS web server, and bug [9] in Kerberos server). It is our belief that, even though people have worked hard to fix these known bugs, root privilege-leaking bugs will not disappear since new bugs are being discovered continuously.

To avoid this large-scale dictionary attack, it makes sense to distribute the functionality of one server to multiple servers to tolerate intrusions [13]. Sandhu et al. [26] observed that a multiple-server scheme may degrade operational quality and is vulnerable to the common-mode failures — once an attacker knows how to break one server, likelihood of success on the other is quite significant in practice. In this paper, we believe that the common-mode failure can be significantly mitigated by the system diversity [14] — including both hardware diversity, operating system diversity and application software diversity — and breaking into one server will not necessarily increase an attacker’s chance to break another server with diverse systems (hardware, operating system and software). In this way, introducing multiple servers using different hardware and software and distributing a secret component among these servers, if done properly, will significantly improve the security against the server compromise-based dictionary attack.

The main results. The contribution of this paper includes an intrusion-tolerant virtual soft token PKI scheme and an intrusion-tolerant virtual smartcard PKI scheme.

In our virtual soft token scheme, a password-encrypted private key, together with the password-verification data, is shared among n servers ($n > 1$). Compared to the multiple-server virtual soft token scheme given in [16], our intrusion-tolerant virtual soft token PKI scheme is threshold: any t ($t \leq n$) or more of these servers can collectively authenticate a user (using the shared PVD) and let the user *securely* download his password-encrypted private key shares **without reconstructing the shared PVD at any single location and the shared password-encrypted private key on any single server**. Any subset of size less than t of these n servers can *not* reconstruct either the *shared* password-encrypted private key or the *shared* PVD, hence tolerating intrusions against the servers.

In our virtual smartcard PKI scheme, the secret component of a private key, together with the

password-verification data, is shared among the n servers. Any t ($t \leq n$) or more of these servers can collectively authenticate a user (via the shared PVD) and help the authenticated user securely perform a digital signature **without reconstructing the shared secret component and the shared PVD at any single location**. Corruption of any less than t of these servers will *not* help an attacker to get the secret component to mount a dictionary attack.

The key idea behind our intrusion-tolerant virtual soft token PKI is the application of an intrusion-tolerant password-authenticated key exchange (PAKE) protocol and the idea behind our intrusion-tolerant virtual smartcard PKI is the composition of an intrusion-tolerant PAKE with a password-adapted threshold cryptography scheme.

This article is organized as follows. Section 2 reviews some related work and Section 3 describes two building blocks for our intrusion-tolerant password-enabled PKI schemes. In Section 4 we present an intrusion-tolerant virtual soft token scheme and an intrusion-tolerant smartcard scheme, both of which are secure against the server compromise-based dictionary attack. Section 5 discusses some operational issues. Concluding remarks are given in Section 6.

2 Related Work

The concept of password-authenticated key exchange (PAKE) protocol was first developed in [2] and then studied in [3, 18, 31, 5, 1]. Perlman and Kaufman [25] applied the PAKE protocols and proposed the idea of virtual soft token. To resist the server compromise-based dictionary attack, Ford and Kaliski [16] proposed the first multiple-server approach for the virtual soft token PKI. However, it requires **all** of the multiple servers present when the user retrieves the distributively stored credential. This significantly degrades the availability of the resulting system — if one server goes down, the service provided will not be available. Jablon [19] improved the scheme of [16] but it still retains the all-server-present requirement. MacKenzie et al. [23] proposed the first *threshold* PAKE and, in our earlier work [30], we also proposed a threshold PAKE, which is used in this article to build our intrusion-tolerant password-enabled PKI.

Kwon [21] proposed a virtual soft token scheme where multiple servers are used. Compared to our intrusion-tolerant virtual soft token, the scheme of [21] used the *non-threshold* RSA given in [4] and thus, required *all* server to be available when a user retrieves its private keys.

To build the virtual smartcard scheme, Sandhu et al. [26] used a password-adapted 2-out-of-2 distributed RSA digital signature scheme given in [4], which is sequential and non-threshold. In contrast, in this article, we adapt the threshold RSA scheme proposed in [28] and use it to build the intrusion-tolerant virtual smartcard.

3 The Building Blocks

As stated earlier, we use an intrusion-tolerant password-authenticated key exchange (PAKE) protocol and a password-adapted threshold RSA as building blocks in our constructions of intrusion-tolerant password-enabled PKI. The intrusion-tolerant PAKE shares a PVD among multiple servers and is used in both the intrusion-tolerant virtual soft token scheme and the intrusion-tolerant virtual smartcard scheme. The password-adapted threshold RSA, on the other hand, shares a secret component among multiple servers and is used in the intrusion-tolerant virtual smartcard scheme. In this section, we will give the details of these two building blocks.

In the remainder of this paper, n is used to denote the number of the multiple servers. These n servers are numbered from 1 to n and are called server 1, 2, ..., n . We assume that there exist secure connections between these n servers, which can be implemented in Secure Socket Layer (SSL) [15]. Let \hat{N} be a safe prime, $\hat{N} = 2\hat{q} + 1$ where \hat{q} is also a prime. \hat{g} is an element of finite field $F_{\hat{N}}$ with order \hat{q} . $(\hat{N}, \hat{q}, \hat{g})$ are system parameters for a PAKE (see the Appendix). For a set S , $a \in_R S$ means that element a is randomly and uniformly selected from S . $|S|$ denotes the cardinality (the size) of S . For two integers a_1 and a_2 , $[a_1, a_2]$ denotes the set of integers x satisfying $a \leq x \leq b$. $\text{gcd}(a_1, a_2)$ denotes the greatest common divisor of a_1 and a_2 .

3.1 An intrusion-tolerant PAKE

In a PAKE, a user possesses a password and the server stores a related password verification data (PVD). Using what they have, the user and the server can perform a password-authenticated key exchange protocol and establish an authenticated (and cryptographically strong) session key, which can be used to protect subsequent communication between the user and the server[†].

However, since the password-verification data stored on the server is derived from a password using a publicly known function, if an attacker manages to compromise the server and steal the PVD, he can still mount an off-line dictionary attack by just computing PVDs value with all likely passwords and comparing them with the stolen PVD. (If he observes a match, then the correct password is found.) The intrusion-tolerant PAKE developed in our earlier work [30] can be used to improve security against this attack, in which a PVD is shared among these multiple servers and is never reconstructed during a PAKE running. Each user of the intrusion tolerant PAKE registers himself with the servers in the user enrollment phase, during which the user's PVD, x , is shared, using a (t, n) -Shamir secret sharing [27][‡], among the n servers. Let $x \xrightarrow{(t,n)} (x_1, x_2, \dots, x_n)$ denote the secret sharing and each server i has PVD share x_i . Then, the user can remotely authenticate himself to Γ , a subset of these multiple servers, $|\Gamma| \geq t$, and establish a session with each of them *without reconstructing the shared PVD*. Any attacker who has compromised less than t servers will get no information about the shared PVD and, thus, cannot mount a dictionary attack. These servers can proactively update their PVD shares while keeping the shared PVD unchanged to further enhance their security. A user can also change his password as in normal password-based systems. The details of this intrusion-tolerant PAKE of [30]

[†] Just as passwords are always subject to the dictionary attack, a PAKE is subject to network-based dictionary attacks, including eavesdropping-based dictionary attack and active dictionary-based protocol attacks. Existing PAKE protocols such as EKE [2], SPEKE [18], SRP [31], provide either heuristic or provable security against network-based dictionary attacks.

[‡]A (t, n) -Shamir secret sharing splits a secret x into n secret shares x_i , $1 \leq i \leq n$, such that any t or more of these secret shares can be used to reconstruct x while any less than t secret shares could not. Shamir secret sharing is perfect in that any less than t secret shares leak no information about x .

is summarized in the appendix of this paper.

3.2 A password-adapted threshold RSA

Threshold cryptography researches on how to share a (cryptographically strong) private key among multiple parties and how a subset of these parties can perform a cryptographic computation without reconstructing the shared private key [11, 12]. Here, we integrate password into the threshold RSA given in [28] to obtain a password-adapted threshold RSA scheme and then, use it to build an intrusion-tolerant virtual smartcard. The password-adapted threshold RSA scheme is given below.

The key generation. A user picks his password \hat{p} and a value d_1 is derived from \hat{p} using a public function (the PBKDF2 function of [22] can be used for this purpose). Let $\Delta = 1 \times 2 \times \dots \times n = n!$ (n is the number of servers). (N, e) is the user's RSA public key where $N = p \times q$; p, q are two primes; e is a prime, $4\Delta < e < \phi(N)$, $\phi(N) = (p-1) \times (q-1)$. d is the user's overall RSA private key, $1 < d < \phi(N)$. $d \times e = 1 \pmod{\phi(N)}$. d is split into d_1 and d_2 and d_2 is computed as follows: $1 < d_2 < \phi(N)$, $d_1 + d_2 = d \pmod{\phi(N)}$. d_2 is further shared among the n servers as follows: let $a_0 = d_2$, $a_i \in_R [0, \phi(N)-1]$ for $1 \leq i \leq (t-1)$; define $f(x) = \sum_{i=0}^{t-1} a_i x^i \pmod{\phi(N)}$; then, one can compute $d_{2i} = f(i) \pmod{\phi(N)}$ for $1 \leq i \leq n$ and server i is assigned d_{2i} , $1 \leq i \leq n$.

Observation. In the above key generation process, p and q are ordinary primes, as opposed to the safe primes in [28]. d_2 is picked as $(d - d_1) \pmod{\phi(N)}$ for efficiency reasons. In this way, the user and the servers can perform the cryptographic computations in parallel. One can also compute d_2 as $d_1 \times d_2 = d \pmod{\phi(N)}$, as did in [26]. In this case, the computations of the user and the servers are sequential.

Digital signature. Let Γ be the subset of the servers who will help a user digitally sign a message m , $|\Gamma| \geq t$. Let a, b be integers satisfying $4\Delta a + eb = 1$, which can be computed by the extended GCD algorithm [29]. The user derives d_1 from his password and computes $c_1 = m^{4\Delta d_1} \pmod{N}$. In parallel, each server $j \in \Gamma$ computes $c_{2j} = m^{2d_{2j}} \pmod{N}$ and sends c_{2j} to the user.

After receiving all c_{2j} , the user computes $c_2 = \prod_{j \in \Gamma} c_{2j}^{2\lambda_{j,\Gamma}} \pmod{N}$, where $\lambda_{j,\Gamma} = \Delta \times \prod_{k \in \Gamma, k \neq j} \frac{k}{k-j}$. He then combines c_2 and c_1 into $\omega = c_2 \times c_1 \pmod{N}$ and computes y as $y = (\omega^a \times m^b) \pmod{N}$. Note that $\omega = m^{4\Delta d} \pmod{N}$ and $y^e \pmod{N} = m^{4\Delta d a e + b e} = m^{4\Delta a + b e} = m \pmod{N}$. That is, y is the digital signature of m by the private key d .

4 Intrusion-tolerant Password-enabled PKI

4.1 Intrusion-tolerant virtual soft token

Just as a virtual soft token is the composition of a PAKE and a secure download of the password-encrypted private key, an intrusion-tolerant password-encrypted private key, an intrusion-tolerant virtual soft token scheme is implemented as the composition of an intrusion-tolerant PAKE and multiple secure downloads of the password-encrypted private key *shares*. In an intrusion-tolerant virtual soft token scheme, for each user, his PVD is shared among the n servers using a (t, n) -Shamir secret sharing over finite field F_q . The user's password-encrypted private key is also shared among the same n servers using a (t, n) -Shamir secret sharing.

When a user needs to use his private key, he first runs the intrusion-tolerant PAKE protocol with Γ , a subset of these multiple servers, $|\Gamma| \geq t$, $\Gamma \subseteq \{1, 2, \dots, n\}$ (see Section 3.1 and the Appendix section for more details). Afterward the user will have one authenticated session key (thus, one secure connection) with each of the servers in Γ . Then, these subset of servers will send their password-encrypted private key shares to the user (via the secure connections). The user reconstructs the password-encrypted private key, decrypts it with the password and uses the private key as in the conventional PKI.

Remark. In the above virtual soft token scheme, neither the shared PVD nor the password-encrypted private key is reconstructed at any single server. The minimal number of servers required for a user login is $(2t - 1)$, $t \leq n$.

4.1.1 The parameter selection

In the above intrusion-tolerant virtual soft token, both the PVD and the password-encrypted private key are shared among the multiple servers. The sharing of PVD is performed in the finite field $F_{\bar{q}}$. The sharing of the password-encrypted private key can be operated in another finite field $F_{\bar{q}}$ where \bar{q} is another prime. Note that the size of d is in the same order as the size of N and, the size of the encryption of d by a password (say, using the PKCS # 5 standard [22]) will *not* increase significantly. (If PKCS #5 is used to encrypt the private key, the salt and iteration count can be simply replicated to each server. And only the encryption of d is shared). Thus, the size of \bar{q} should be no less than the size of N . Another option is to use \hat{q} as \bar{q} . If this is the case, \hat{q} should be no less than N .

4.2 Intrusion-tolerant virtual smart-card PKI

Just as a virtual smartcard [26] is the composition of a PAKE and a password-adapted non-threshold distributed RSA, our intrusion-tolerant virtual smartcard scheme is the composition of an intrusion-tolerant PAKE and a password-adapted threshold RSA. In the intrusion-tolerant virtual smartcard scheme, a user's PVD (related to password \hat{p}) is shared among the n servers using a (t, n) -Shamir secret sharing scheme. The user's RSA private key d is split as a password-derived value d_1 (derived from password \hat{p}) and d_2 . d_2 is further shared as $d_2 \xrightarrow{(t, n)} (d_{21}, d_{22}, \dots, d_{2n})$. In addition to its PVD share, a server, i , $1 \leq i \leq n$, also holds d_{2i} .

When a roaming user wants to digitally sign a message, m , he first runs the intrusion-tolerant PAKE protocol with Γ , a subset of the multiple servers, $|\Gamma| \geq t$, $\Gamma \subseteq \{1, 2, \dots, n\}$, and establishes an authenticated session key (thus, a secure connection) with each of them. Then, the user sends m , via the secure connections, to server j , $j \in \Gamma$. Server j computes $c_{2j} = m^{2d_{2j}} \bmod N$ and sends it back to the user through the secure channel. The user computes ω and y as described in Section 3.2, where y is the digital signature of m by the user's private key d .

Remark. In the above virtual smartcard scheme, none of the shared PVD, d_2 and m^{d_2} is reconstructed at any single server. The minimal number

of servers required for a user login is $(2t - 1)$, $t \leq n$.

5 Some Operational Considerations

Operational quality is a big concern for the intrusion-tolerant password-enabled PKI since introducing multiple servers increases the operational complexity [26]. However, we can automate the management to minimize the manual management overhead.

5.1 The user enrollment

Compared to the virtual soft token [25, 26], at the user enrollment phase, our intrusion-tolerant virtual soft token scheme introduces one additional step: the share generations of the user's PVD and password-encrypted private key and the share distribution to the multiple servers. This additional step can be fully automated by a management server, which performs the Shamir secret sharing on the PVD and the password-encrypted private key and, then *securely* sends, via SSL, these shares to the multiple servers.

Similarly, compared to the virtual smartcard [26], at the user enrollment phase, our intrusion-tolerant virtual smartcard also introduces one additional step: the share generations of the user's PVD and his secret component (d_2) and the share distribution to the multiple servers. We can also automate this step by using a management server, which performs the Shamir secret sharing on the PVD and the secret components and, *securely* sends, via SSL, these shares to the multiple servers.

Thus, our intrusion-tolerant password-enabled PKI schemes do not bring much operational overhead to the user enrollment stage.

5.2 User authentication

In the intrusion-tolerant password-enabled PKI schemes, when a user interacts with the servers to use his private key, he just needs to type in his password and all other steps are automatically performed by programs. Thus, our intrusion-

tolerant password-enabled PKI schemes do not increase user's operational complexity.

5.3 Password change

In a password-enabled PKI, a user may want to change his password while keeping his long-term private key unchanged.

In the intrusion-tolerant virtual soft token PKI, a change in the password requires the update of the corresponding PVD shares and the update of the password-encrypted private key shares (the private key remains unchanged but its encrypted form by the password should be updated accordingly). The intrusion-tolerant PAKE used in this article (see Section 3.1) allows a user to securely change his password and update the corresponding PVD shares stored on each server. This naturally enables the password change in our intrusion-tolerant virtual soft token: a user first runs the intrusion-tolerant PAKE protocol, securely downloads the password-encrypted private key shares, reconstructs the password-encrypted private key, decrypts it with the old password, re-encrypts it with the new password, generates a (t, n) -Shamir secret shares and securely uploads these new shares to the multiple servers respectively; then, he can run the intrusion-tolerant PAKE password change protocol given in [30] to update the PVD shares stored on each server.

In a virtual smartcard PKI, the change of a user's password causes the change of d_1 and thus, requires the update of the secret component, d_2 , since d remains unchanged and $d_1 + d_2 = d \bmod \phi(N)$. As $\phi(N)$ is unknown to the user and the server, the change of d_2 is difficult unless the shared d is reconstructed and $\phi(N)$ is recovered. Technically, after authenticating himself to the server and establishing a session key, a user can securely download d_2 from the server, reconstruct d and recover $\phi(N)$, compute the new d_2 (from the new d_1 and the recovered $\phi(N)$), and update this new d_2 to the server. However, this process is computation-intensive and looks awkward. Indeed, in the only virtual smartcard PKI scheme proposed in [26], password change is not discussed. This difficulty remains in our intrusion-tolerant virtual smartcard PKI and is the topic of our future research.

6 Conclusion

Password-enabled PKI, including the virtual soft tokens and virtual smartcards, facilitates the private key management by integrating easy-to-use passwords into PKI. However, compared to the conventional PKI, password-enabled PKI introduces a security-critical server where large number of password-related credentials are stored. The compromise of this server will render these password-based credentials susceptible to the dictionary attack and, thus, damage the security of numerous private keys.

To address this attack, using multiple servers, we proposed an intrusion-tolerant virtual soft token PKI scheme and an intrusion-tolerant virtual smartcard PKI scheme. In our schemes, compromising up to a threshold number of these servers will not help an attacker mount the dictionary attack and the intrusion-tolerant password-enabled PKI schemes can still function in the presence of some server failures. Compared to previous work, our virtual soft token is threshold and does not require all servers when a user needs his private key. We designed our virtual soft token PKI by compositing an intrusion-tolerant PAKE with a secret sharing. Our intrusion-tolerant virtual smartcard PKI is achieved through the composition of an intrusion-tolerant PAKE with the password-adapted threshold RSA. The multiple servers introduced in our intrusion-tolerant password-enabled PKI can be easily managed and PKI users can roam with human memorable passwords.

Acknowledgement

The author is indebted to Samuel Redwine for the discussions and Hua Lin for reviewing the earlier drafts of this paper. The author also wishes to thank the anonymous reviewers for suggestions to improve this article.

References

- [1] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances*

- in *Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155, Bruges, Belgium, May 2000.
- [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] S. M. Bellare and M. Merritt. Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 244–250, 1993.
- [4] C. Boyd. Digital multisignatures. In H. Beker and F. Piper, editors, *Cryptography and coding*, pages 241–246. Clarendon Press, Royal Agricultural College, Cirencester, December 15–17 1989.
- [5] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In B. Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 156–171, Bruges, Belgium, May 2000.
- [6] CERT. CERT advisory CA-1999-13 multiple vulnerabilities in WU-FTPD. Available from <http://www.cert.org/advisories/CA-1999-13.html>, October 19 1999.
- [7] CERT. CERT advisory CA-2001-33 multiple vulnerabilities in WU-FTPD. Available from <http://www.cert.org/advisories/CA-2001-33.html>, November 29 2001.
- [8] CERT. CERT advisory CA-2002-09 multiple vulnerabilities in Microsoft IIS. Available at <http://www.cert.org/advisories/CA-2002-09.html>, April 11 2002.
- [9] CERT. CERT advisory CA-2002-29 buffer overflow in Kerberos administration daemon. Available at <http://www.cert.org/advisories/CA-2002-29.html>, October 25 2002.
- [10] CERT. Vulnerability note VU#124003 apache HTTP server on Win32 systems does not securely handle input passed to CGI programs. Available at <http://www.kb.cert.org/vuls/id/124003>, April 11 2002.
- [11] 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.
- [12] Y. Desmedt. Threshold cryptography. *European Trans. on Telecommunications*, 5(4):449–457, July-August 1994. (Invited paper).
- [13] Y. Deswarte, L. Blain, and J.-C. Fabre. Intrusion tolerance in distributed computer systems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 110–122, May 1991.
- [14] Y. Deswarte, K. Kanoun, and J.-C. Laprie. Diversity against accidental and deliberate faults. In *Proceedings of the Computer Security, Dependability and Assurance: From Needs to Solutions*, pages 171–181, 1998.
- [15] T. Dierks and C. Allen. The TLS protocol version 1.0. Internet RFC 2246, January 1999.
- [16] W. Ford and B. Kaliski, Jr. Server-assisted generation of a strong secret from a password. In *Proceedings of the 9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2000)*, pages 176–180, Gaithersburg, MD, USA, June 14–16 2000.
- [17] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. *Information and Computation*, 164(1):54–84, 2001.
- [18] D. P. Jablon. Strong password-only authenticated key exchange. *ACM SIGCOMM Computer Communication Review*, 26(5):5–26, October 1996.
- [19] D. P. Jablon. Password authentication using multiple servers. In D. Naccache, editor, *Progress in Cryptology - CT-RSA 2001 Proceedings of the Cryptographers' Track at RSA Conference*, volume 2020 of *Lecture Notes in Computer Science*, pages 344–360, San Francisco, CA, USA, April 8–12 2001. Springer-Verlag.
- [20] D. Klein. Foiling the cracker: A survey of, and improvements to, password security. In *Proceedings of the UNIX Security Workshop II*, August 1990.

- [21] 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.
- [22] R. Laboratories. PKCS #5 v2.0 password-based cryptography standard. Available from <http://www.rsasecurity.com/rsalabs/pkcs/pkcs-5/>, March 1999.
- [23] P. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold password-authenticated key exchange (extended abstract). In M. Yung, editor, *Advanced in Crypto: - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 385–400. Springer-Verlag, August 2002.
- [24] R. Morris and K. Thompson. Password security: a case history. *Communications of the ACM*, 22(11):594–597, November 1979.
- [25] 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.
- [26] 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.
- [27] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.
- [28] V. Shoup. Practical threshold signatures. In *Advance in Cryptology - EUROCRYPT 2000*, pages 207–220, May 2000.
- [29] D. R. Stinson. *Cryptography: Theory and Practice*. CRC, Boca Raton, 1st edition, 1995.
- [30] X. Wang. *A Distributed Password-Authenticated Key Exchange Protocol Secure Against Server Compromise-Based Dictionary Attacks*. Department of Computer Science, James Madison University, Harrisonburg, VA, USA, January 2003.
- [31] T. Wu. The secure remote password protocol. In *Proceedings of the 1998 Network and Distributed System Security Symposium*, pages 97–111, 1998.
- [32] T. Wu. A real-world analysis of Kerberos password security. In *Proceedings of the 1999 Network and Distributed System Security Symposium*, 1999.

APPENDIX

In this article, the intrusion-tolerant PAKE protocol given in [30] is used as a building block. Let \hat{N} be a safe prime, $\hat{N} = 2\hat{q} + 1$ where \hat{q} is also a prime. \hat{g} is an element of finite field $F_{\hat{N}}$ with order \hat{q} . $(\hat{N}, \hat{q}, \hat{g})$ are the system parameters. H denotes a secure hash function such as SHA-1. We use $(\tau_1, \tau_2, \dots, \tau_n) \xrightarrow{(t, \hat{N})} \tau$ to denote the fact that a value, τ , is reconstructed from t or more shares τ_i , $1 \leq i \leq n$.

For this protocol, Table 1 summarizes the password-authenticated key exchange data flow between a user and a particular server, i , $1 \leq i \leq n$, where server i stores the user's PVD share x_i . At the end of this protocol, the user and server i will share a cryptographically strong session key K . This protocol can be repeated between the user and any other server so that the user can share one session with each of the participating servers.

A user shows his ID, I , in the first step, to which server i responds with the user's salt, s . Server i also passes this ID I to a threshold number or more servers (so that they can locate the PVD share for this user). In step 2, the user computes x and l . In step 3, with the assistance of a threshold number or more servers, server i computes a value B (see the following paragraph for details) and passes it back to the user. In this step, the user picks $a \in_R [1, \hat{q} - 1]$, computes A and sends it to server i , which in turns passes A to the participating servers. In step 4, the user checks if the received B satisfies $B = 0 \pmod{\hat{N}}$ and will quit if this is the case. Otherwise, the user computes a value S_c using the given formula and server i uses the subset of the servers to compute value S_s (see the following paragraph for details). Note that $S_c = S_s$ and this value is denoted as S . Step 5 and 6 are used to verify that the user and server i share a common value S . The authenticated session key, K , is derived from S in step 7.

The server-side computation of step 3 Let Γ be the set of participating servers, $\Gamma \subseteq \{1, 2, \dots, n\}$,

Client	Server i	Server 1	Server 2	...	Server n
1	\xrightarrow{I}	(lookup s)	(lookup x_1)	(lookup x_2)	(lookup x_n)
2	\xleftarrow{s}				
	$x = H(s, I, \hat{p})$ $l = x^{-1} \bmod \hat{q}$				
3	$A = \hat{g}^{al \bmod \hat{q}} \bmod \hat{N}$	\xrightarrow{A}			
		\xleftarrow{B}			
4	$S = B^{al \bmod \hat{q}} \bmod \hat{N}$				
5	$M_1 = H(A, B, S)$	$\xrightarrow{M_1}$			
6	(verify M_2)	$\xleftarrow{M_2}$			
7	$K = H(S)$				

Table 1: Intrusion-tolerant PAKE

$|\Gamma| \geq t$. B is computed by the participating servers as follows:

- Using a Joint-Shamir-RSS [17], the participating servers generate $(b_1, b_2, \dots, b_n) \xleftrightarrow{(2t-1, n)}$ $b \bmod \hat{q}$, where b is a random value **unknown** to any individual server and each server j , $j \in \Gamma$, has share b_j .
- Each participating server j , $j \in \Gamma$, computes $d_j = b_j \times x_j \bmod \hat{q}$, $B_j = \hat{g}^{d_j} \bmod \hat{N}$, and sends B_j to server i . Note that $(d_1, d_2, \dots, d_n) \xleftrightarrow{(2t-1, n)}$ $d = b \times x \bmod \hat{q}$ and $(B_1, B_2, \dots, B_n) \xleftrightarrow{(2t-1, n)}$ $\hat{g}^{bx \bmod \hat{q}} \bmod \hat{N}$.
- Using the Lagrange interpolation formula in the exponent [17], server i first computes B as follows: $(B_1, B_2, \dots, B_n) \xrightarrow{(2t-1, n)}$ $B \bmod \hat{N}$. (Note that, $B = \hat{g}^{bx \bmod \hat{q}} \bmod \hat{N}$.) Then, server i sends B back to the user in step 3.

$(2t - 1)$ servers are needed to perform the above steps.

The server-side computation of step 4 The computation of step 4 proceeds as follows:

- Each participating server checks if $A = 0 \bmod \hat{N}$. They will abort if it is. Otherwise, they check if $A^{\hat{q}} = 1 \bmod \hat{N}$. They will abort the computation if it does not hold. Otherwise, they will continue.
- Each participating server j , $j \in \Gamma$, computes $S_j = A^{d_j} \bmod \hat{N}$, and sends S_j to

server i . Note that $(S_1, S_2, \dots, S_n) \xleftrightarrow{(2t-1, n)}$ $A^{bx \bmod \hat{q}} \bmod \hat{N}$.

- Using the Lagrange interpolation formula in the exponent [17], server i computes S as follows: $(S_1, S_2, \dots, S_n) \xrightarrow{(2t-1, n)}$ $S \bmod \hat{N}$. Note that $S = A^{bx \bmod \hat{q}} \bmod \hat{N}$.