Robust Software Tokens: Towards Securing a Digital Identity

Taekyoung Kwon

Taekyoung Kwon is with the Faculty of the Department of Software Engineering, School of Computer Engineering, Sejong University, Seoul, 143-747, Korea. E-mail: tkwon@sejong.ac.kr . Phone: +82-2-3408-3758. Fax: +82-2-3408-3662.

Abstract

This paper presents a new method called the *robust software token* for providing users with a stable and portable container in which a private key is stored and kept from adversaries, by simple software-only techniques. The proposed scheme is comparable with the related noble work such as a cryptographic camouflage scheme[5] and a networked cryptographic device[10], but equipped with several advantages; (1) it uniquely supports both closed and open domains on public key infrastructures, (2) it supports more protocol setup, (3) and it is more efficient than the others. This paper handles the new RSA-based scheme only. The DSA-based scheme sharing the basic idea was done in the previous work[9].

Keywords

Digital identity, software token, private key management, public key infrastructure, authentication, digital signature

I. Introduction

The rapid growth of sensitive services on the Internet makes the digital identity of individuals more important than before. A public key infrastructure (PKI) based on famous standards like X.509[6] enables the digital identity by allowing each user to have an authentic public key and private key pair in a distributed environment. This is stronger and more scalable than the identity based on passwords only. The PKI-based digital identity is beneficial to such applications that need identification or digital signature of individuals in a secure manner. However, it is users' responsibility to carry their private keys securely for convincing others of their digital identities. If a user's private key is compromised, it can be used by an imposter to forge the user's digital identity, i.e., forgery and loss of privacy could result from it. Such a cryptographic key is not favorable to human memory. Accordingly a container must belong to each user for carrying the private key. We call this a *token* in this paper by distinguishing from passcode generator-like devices.

A. Previous Solutions

The token is a mean to provide users with a stable and portable container of the private key and to keep the private key from adversaries. A tamper-resistant hardware token such as a crypto smart card, would be a great and promising solution for this but it could have various shortcomings provided that it is used alone, for example, practical deployment problems and possible channels which may be established surreptitiously[2].

Alternatively, a user-controlled software-only computing device such as a desktop computer, notebook and handheld, may contain a software token for easier deployment. However, such a device is not tamper-resistant and a typical form of the software token is the encrypted private key under an encryption key derived from a memorable password of low entropy. The private key is verifiable by off-line attacks, for example, an adversary who compiled a dictionary of likely passwords and stole the software token, can decrypt a candidate key, sign an arbitrary message, verify it with a corresponding authentic public key, and repeat this procedure until (s)he gets a correct one. Such an attack works in a relatively small space of passwords[7], [12]. This is the reason why the PKCS#5-encrypted private key is insecure[15]. Readers are referred to Appendix I of this paper for the detail.

Several methods were proposed to deposit the password-encrypted private key to the trusted server and download it when necessary[13]. A virtual card solution used in practice[18], prevents off-line attacks and provides mobility in that sense, but needs to store all user credential information in the center server. Therefore such a centralized technique is not our main concern. It will be only combined with the proposed scheme for mobility.

Recently software-only device techniques have been proposed for improving security. Those techniques do not deposit all credentials to the center server. They instead exploit network connectivity in a way to cooperate with the trusted server entity for the private key. They include the cryptographic camouflage[5] and the networked cryptographic device[1][10]. The point to see is, however, the cryptographic camouflage is only useful in the so-called closed PKI environments where the trusted server is the only who can verify the user's digital identity, while the networked cryptographic device does not directly support the closed PKIs, meaning that the server only assists a signature. Our scheme will remove these restrictions. As for the open and closed PKIs, a verifier must utilize a user's public key certificate controlled by a certificate authority (CA) for verifying the user's digital identity. The closed PKI is mainly focused on two-factor authentication[5] in that sense.

[1]Though the scheme proposed in [10] originally did not assume the trusted server, we claim the trusted server is eventually necessary for that scheme because both the user's password and the server's private key are actually the Achilles tendon of the scheme in terms of security. For example, it may be a possible threat that the "untrusted" server obtains a user's password by guessing attacks along with a corresponding user's token, or the server misbehaves by refusing the user's request. Accordingly the trusted server must be assumed for the scheme.

	Our Scheme	Crypto Camouflage[5]	Networked Crypto Device[10]
Server/Token Compromises	Offline Attack	Offline Attack	Offline Attack
Server Compromise	.	.	Offline Attack on Passwords
Token Compromise	.	.	.
Closed PKI Support	O	O	X
Open PKI (Assisted) Support	O	X	O
Open PKI (Entrusted) Support	O	X	X

TABLE I

First Comparisons of Related Schemes

B. Contributions

This paper presents a new method called the *robust software token* for keeping the private key from adversaries by software-only techniques. For the purpose, a user must keep two factors such as a password in his or her mind and a private key in the robust software token while a trusted server just holds its own private key. The user can simultaneously keep the private key in a tamper-resistant hardware token. This paper handles the RSA-based schemes only. The DSA-based schemes are respectively described in [9], [5], and [10].

For clearly summarizing our contributions, we present Table I ahead. Our scheme may be comparable with the related work in some sense[5], [10], but advantageous to them because it will be uniquely designed; (1) to support the closed PKI as well as the open PKI, (2) to support many kinds of protocol setup, and (3) to be more efficient. Note that our scheme will protect not only the user's private key but also the user's password only except when the server and the token are compromised together.

II. Preliminaries

In this section we summarize the notation to be used and describe more detail of the related work[5], [10]. Subsequently we state the security goals for our scheme.

A. Notation

Let us borrow partly the noble notation used in [10]. Let κ be the security parameter such that $\kappa = 160$, while λ another parameter such that $\lambda = 1024$ or 2048. Let rst denote

the robust software token, dvc the user-controlled device, and svr the trusted server. Let π be the user's password, pk_{svr} the trusted server's authentic public key, and sk_{svr} the corresponding private key. Let $< e, N >$ and $< d, N, \phi(N) >$ be the user's authentic public key and private key pair where $ed \equiv 1(\bmod\ \phi(N))$ and N is the product of two large prime integers[16], [14]. Note the Euler totient function $\phi(N)$ is necessary for initializing software tokens[5], [10]. Notation not described here will be declared in each part of this paper.

B. Related Work in Detail

B.1 Cryptographic Camouflage

Hoover and Kausik presented a pioneer study called cryptographic camouflage in 1999[5]. Their basic idea was simple but strong to make the plaintext unidentifiable by encrypting random-looking plaintext only, for example, the RSA private exponent d only (not n). However, they also had to encrypt the user's public key under pk_{svr} for security reasons, meaning that the authentic public key was not public any more. Consequently this property restricted its use to authentication in the closed PKI only, say, a user cannot prove his or her digital identity to most parties who do not know sk_{svr}. In addition they needed a longer private key and public key pair, for example, $d + l\phi(N)$ having $(|l| + 1)\lambda$ bits in length where $|l|$ means the bit length of a small integer l and a public key e having κ bits in length[5]. Readers are referred to Appendix II of this paper for more detail.

B.2 Networked Cryptographic Device

MacKenzie and Reiter presented the networked cryptographic device along with a noble security proof in 2001[10] inspired by work of [4]. Their basic idea was to split the user's private key and share them between dvc and svr by encrypting the server's portion under pk_{svr}. They also suggested the new ability for the user to disable the private key provided that the token is compromised. However, the system was designed for the open PKI only, meaning that, dvc had to show the user's password information rather than an authentic public key to svr for obtaining a complete digital signature. This scheme accordingly does not support the digital identity in the closed PKI environments[5], [9] because svr actually uses the password information rather than the certified public key for authentication. Readers are referred to Appendix III of this paper for more detail.

C. Security Goals

The proposed scheme should allow dvc and svr to communicate securely over a public network in order to run the digital identity operations based on PKIs. Note that dvc contains rst. An adversary is assumed to have a dictionary of likely passwords and control the whole network, meaning that (s)he controls any inputs to dvc and svr, and hears all of their outputs. Then we define the following security goals.

- The proposed scheme must guarantee that the adversary needs sk_{svr} as well as rst for breaking the system. In other words, the adversary should compromise both dvc and svr for getting information on the user's private key.
- The proposed scheme must guarantee that additional off-line attacks are necessary for obtaining the correct private key even if dvc and svr are both compromised.
- The security must hold in the closed PKI as well as the open PKI, but each domain must be distinguished. That means, the digital identity aimed for the closed PKI must not be re-used for the open PKI and vice versa.

Let vrf denote an actual verifier for the user's digital identity. Then vrf is only svr in the closed PKI while it can be any party in the open PKI.

III. How to Make a Robust Software Token

This section states the basic idea of our scheme and a method to make rst in detail.

A. Basic Idea

The idea is conceptually identical with that of our previous work on the digital signature standard (DSS)[9] firstly inspired by the cryptographic camouflage method. Our idea is: *one public key is paired to two kinds of private keys and left unencrypted*. One of those private keys is a pure private key while the other is a protected key. The pure key is used in the RSA signature algorithm as usual and must be securely stored in a tamper-resistant device. The protected key works with the proposed protocols and will be handled by rst.

B. Making a Robust Software Token

A user's key pair, a password, and a server's public key are loaded by the entity kgc who is responsible for key generation. The values are $< e, N >$, $< d, N, \phi(N) >$, π, and

pk_{svr}. Then kgc computes the followings:

$e_1 \leftarrow_R Z_{\phi(N)}$	: choose another RSA exponent at random in length κ.
$d_1 \leftarrow e_1^{-1} \bmod \phi(N)$	: compute a mod $\phi(N)$ inverse of e_1.
$a \leftarrow k(\pi)$	: derive a symmetric encryption key from π by $k()$.
$b \leftarrow_R \{0,1\}^1$	: choose one bit at random.
$c \leftarrow (dd_1 \bmod \phi(N)) - b$	: multiply d and d_1 (mod $\phi(N)$), and subtract b from it.
$x \leftarrow E_a(\mathcal{R}(c))$	: encrypt c under a by random padding, $\mathcal{R}()$.
$y \leftarrow \mathcal{E}_{pk_{\mathsf{svr}}}(R(e_1))$	: encrypt e_1 under pk_{svr} by randomized encoding, $R()$.

$E_i()$ denotes symmetric encryption while $\mathcal{E}_i()$ does asymmetric encryption, under each key i. $\mathcal{R}()$ means random padding. Let the random encoding function $R()$ include a timestamp readable by svr. This may be useful for controlling lifetimes and further resisting replays. $k()$ means a key derivation function as defined in PKCS#5. Above $dd_1 \bmod \phi(N)$ may remind readers of the noble work, digital multisignatures[3], stemming from the very different idea. Note that $-b$ is normal integer subtraction, not in $Z_{\phi(N)}$. It may perturb c to be odd or even with probability $\frac{1}{2}$. An adversary cannot verify correct guesses by decrypting x through likely passwords because c is perturbed and x is random padded, i.e, (s)he cannot find any notable characteristics from c even if x is correctly decrypted.

Several policies are allowed to make rst, e.g., either svr or dvc can act kgc. As for svr acting kgc, y can be made more efficient such that $y = E_\xi(e, e_1)$ where ξ is svr's secret key.

Assume the user's public key is certified by CA and $\mathcal{C}$ denotes the certificate. Then the values $\mathcal{C}$ and $< x, y, N >$ are packed in rst while the values $\mathcal{C}$ and $< d, N, \phi(N) >$ are stored in a tamper-resistant hardware token. The values $e, d, N, \phi(N), \pi, e_1, d_1, a, b, c, x, y$, and pk_{svr} are permanently erased from kgc. Note that pk_{svr} may be further necessary for the open PKI protocols. So we add it to rst if necessary. Finally kgc can issue the robust software token as well as the tamper-resistant hardware token to the corresponding user. Hence rst may be saved in stable storage on the user-controlled device.

The tamper-resistant hardware token is used in the open PKI only if the reading device is ready. In most cases, the robust software token rst can be used in the closed PKI and in the open PKI as well. Next two sections will describe this.

dvc		svr
		$\mathcal{G} \leftarrow_R \{0,1\}^\kappa$
	$\xleftarrow{\mathcal{G}}$	
$a \leftarrow k(\pi)$		
$c \leftarrow \mathcal{R}^{-1}(D_a(x))$		
$c \leftarrow c+1$ if even		
$m \leftarrow h(\mathsf{svr}, h(\mathcal{G}))$		
$M \leftarrow R(m)$		
$s \leftarrow M^c \bmod N$		
	$\xrightarrow{\mathcal{C}, y, s}$	verify $\mathcal{C}$
		$e_1 \leftarrow R^{-1}(\mathcal{D}_{sk_{\mathsf{svr}}}(y))$
		$M_1 \leftarrow (s^e)^{e_1} \bmod N$
		$m_1 \leftarrow R^{-1}(M_1)$
		Abort if $m_1 \neq h(\mathsf{svr}, h(\mathcal{G}))$
		Otherwise, accept

Fig. 1. Two-factor Authentication Protocol

IV. Robust Software Token in the Closed PKI

As we mentioned already, the only entity who knows sk_{svr} should verify the user's digital identity in the closed PKI. Two kinds of services such as *two-factor authentication* and *closed digital signature* can be implemented between dvc and svr who has sk_{svr}.

A. Two-factor Authentication

A server may authenticate a user by verifying something (s)he knows, something (s)he has, or something (s)he is. The most widely used methods are based on something (s)he knows, say memorable passwords[1], [7], [8], [12], but they are relatively deficient in security and scalability[5]. Two-factor authentication based on something (s)he knows and something (s)he has may improve the security and scalability.

In the protocol of Figure 1, the user who knows π and has rst can prove his or her digital identity to svr who has sk_{svr}. The user types π in dvc while loading rst and connecting

dvc		svr
$m \leftarrow \{0,1\}^*$		
$a \leftarrow k(\pi)$		
$c \leftarrow \mathcal{R}^{-1}(D_a(x))$		
$c \leftarrow c+1$ if even		
$M \leftarrow R(h(\mathsf{svr}, h(m))$		
$s \leftarrow M^c \bmod N$		
	$\xrightarrow{\mathcal{C}, y, s, m}$	verify $\mathcal{C}$
		$e_1 \leftarrow R^{-1}(\mathcal{D}_{sk_{\mathsf{svr}}}(y))$
		$M_1 \leftarrow (s^e)^{e_1} \bmod N$
		$m_1 \leftarrow R^{-1}(M_1)$
		Abort if $m_1 \neq h(\mathsf{svr}, h(m))$
		Otherwise, accept

Fig. 2. Closed Signature Protocol

to svr. An initiating signal to svr is omitted in the Figure. Upon receiving the random challenge $\mathcal{G}$ from svr, dvc derives a from π, recovers c from x, and adds 1 to c if it is even. Subsequently dvc computes $m = h(\mathsf{svr}, h(\mathcal{G}))$, signs m by random encoding, and sends out $< \mathcal{C}, y, s >$ to svr where $h()$ is a typical one-way hash function.

The reasons for $h(\mathsf{svr}, h(\mathcal{G}))$ are distinct; (1) to prevent an adversary who may masquerade svr from signing an arbitrarily chosen message instead of $\mathcal{G}$, and (2) to keep s from being re-used outside the svr's domain, even if svr is compromised or misbehaving.

Upon receiving the values from dvc, svr verifies the signature by using e from $\mathcal{C}$ (so that the system works in the PKI) and e_1 from y. Since the signature s is random encoded, an adversary's attempt to re-sign $h(\mathsf{svr}, h(\mathcal{G}))$ under guessed decryption of x is ineffective, meaning that $s \neq s^{'}$ even with a correct guess.

B. Closed Signature

A user's signature in the closed PKI is simply derived from the former protocol. The user signs an arbitrary message m instead of $\mathcal{G}$ as depicted in Figure 2. A timestamp in $R()$ may work against replays. The reasons for $h(\mathsf{svr}, h(m))$ are as same as above.

V. Robust Software Token in the Open PKI

In this section, we state two methods for using rst in the open PKI. One allows an entrusted signature protocol setup while the other does an assisted signature protocol setup. Both setup assume the same security but may arouse various applications respectively. The basic idea of our scheme in open PKIs is very simple that svr is fully trusted and able to complete the dvc's open signature from the information sent by dvc. It is svr's given authority to decide whether (s)he sends the dvc's signature to the out world or sends it back to dvc as requested by dvc. This may not deteriorate security and privacy compared to an ordinary signature scheme because the signature is already purported open. Note that we omit hashing message m before signing for simplicity in this section.

A. Entrusted Open Signature (Figure 3)

Firstly dvc generates a closed signature under c and entrusts svr with it by attaching an identifier of vrf. The reason for encrypting s under pk_{svr} is to prevent an adversary who obtained dvc and s_1 from verifying the correct private key through likely passwords. It also enables to use a common deterministic encoding $\delta()$ for M. After decrypting s_0, svr can check a timestamp against replays. Then svr computes s_1 by raising s to e_1. Assuming this an open signature of dvc (actually the owner of rst), svr verifies its validity with $< e, N >$ (of $\mathcal{C}$). If it is correct, then svr on behalf of dvc sends s_1 to vrf. It is optional for svr to log the record and report it to dvc. Such a record might be beneficial to non-repudiation requirements. Accordingly vrf can verify the dvc's signature with $\mathcal{C}$ as usual. Various applications can be considered in this setup, for example, svr acts a secure mail server while dvc runs a mail client. The signed email can be delivered to any recipients, vrf.

B. Assisted Open Signature (Figure 4)

In the assisted setup, dvc generates a signature under c and asks svr for completing the open signature. Similarly s must be encrypted under pk_{svr} and such decrypted. After checking a timestamp, svr computes s_1 by raising s to e_1. Assuming this an open signature of dvc (actually the owner of rst), svr verifies its validity with $< e, N >$ (of $\mathcal{C}$). Provided that it is correct, svr sends back s_1 to dvc. It is also optional for svr to log the record that might be beneficial to non-repudiation requirements. Upon receiving s_1, dvc verifies its

dvc		svr		vrf
$m \leftarrow \{0,1\}^*$				
$a \leftarrow k(\pi)$				
$c \leftarrow \mathcal{R}^{-1}(D_a(x))$				
$c \leftarrow c+1$ if even				
$M \leftarrow \delta(m)$				
$s \leftarrow M^c \bmod N$				
$s_0 \leftarrow \mathcal{E}_{pk_{\mathsf{svr}}}(R(s))$				
	$\xrightarrow{\mathcal{C}, y, s_0, m, \mathsf{vrf}}$	verify $\mathcal{C}$		
		$e_1 \leftarrow R^{-1}(\mathcal{D}_{sk_{\mathsf{svr}}}(y))$	
		$s \leftarrow R^{-1}(\mathcal{D}_{sk_{\mathsf{svr}}}(s_0))$		
		$s_1 \leftarrow s^{e_1} \bmod N$		
		$M_1 \leftarrow (s_1)^e \bmod N$		
		$m_1 \leftarrow \delta^{-1}(M_1)$		
		Abort if $m_1 \neq m$		
		Record(optional)		
	$\xleftarrow{\mathsf{report(ack)}}$		$\xrightarrow{\mathcal{C}, s_1, m}$	verify $\mathcal{C}$
Record(optional)				$M_1 \leftarrow s_1^e \bmod N$
				$m_1 \leftarrow \delta^{-1}(M_1)$
				Abort if $m_1 \neq m$
				Otherwise, accept

Fig. 3. Entrusted Signature Protocol

validity with $< e, N >$. If it is correct, dvc sends s_1 to vrf. Accordingly vrf can verify the dvc's signature with $\mathcal{C}$ as usual. Every application needing the dvc's open signature can be considered in this setup because dvc is the final sender of the signature.

C. Towards Mobility

Mobility is another requirement for digital identities. If rst is stored in a handheld computer or a portable storage device such as a memory card, flash drive, USB token[17],

vrf	dvc		svr
	$m \leftarrow \{0,1\}^*$		
	$a \leftarrow k(\pi)$		
	$c \leftarrow \mathcal{R}^{-1}(D_a(x))$		
	$c \leftarrow c+1$ if even		
	$M \leftarrow \delta(m)$		
	$s \leftarrow M^c \bmod N$		
	$s_0 \leftarrow \mathcal{E}_{pk_{\mathsf{svr}}}(R(s))$		
		$\mathcal{C}, y, s_0, m \longrightarrow$	verify $\mathcal{C}$
			$e_1 \leftarrow R^{-1}(\mathcal{D}_{sk_{\mathsf{svr}}}(y))$
			$s \leftarrow R^{-1}(\mathcal{D}_{sk_{\mathsf{svr}}}(s_0))$
			$s_1 \leftarrow s^{e_1} \bmod N$
			$M_1 \leftarrow (s_1)^e \bmod N$
			$m_1 \leftarrow \delta^{-1}(M_1)$
			Abort if $m_1 \neq m$
			Record(optional)
		$\longleftarrow s_1$	
	$M_1 \leftarrow (s_1)^e \bmod N$		
	$m_1 \leftarrow \delta^{-1}(M_1)$		
	Abort if $m_1 \neq m$		
verify $\mathcal{C}$	$\longleftarrow \mathcal{C}, s_1, m$		
$M_1 \leftarrow s_1^e \bmod N$			
$m_1 \leftarrow \delta^{-1}(M_1)$			
Abort if $m_1 \neq m$			
Otherwise, accept			

Fig. 4. Assisted Signature Protocol

etc., the mobility may not be an issue in the software token. However, dvc includes localized systems such as desktop computers so that a supplementary method is necessary. Remind that the virtual card solution[18] needed to store all user credential information in the center server and it was a weak spot because such information was not entirely protected. We can apply this model but with our enhanced security, meaning that a user deposits rst to the mobility server msv and requests it in an authentic manner such as using a password only[13], [16], before cooperating with svr. The security of svr must be assumed. The point is that svr and msv are clearly separated in our mobility scheme. That means, svr and msv are different and each password for rst and msv also must be different.

VI. Analysis

A. Security

Upon trusting svr, it is not problematic to allow svr even to complete the open signature in both open models. On line attacks are also frustrated in that sense. If svr is untrusted or compromised, however, the security assumed in the software token methods is deteriorated because a further compromise of dvc is to be a critical weak point as well as a denial of service is possible. This is the shared property of the related schemes[5], [9], [10] as we noted earlier. However, our scheme does not allow the compromised server to obtain password information by dictionary attacks before explicitly obtaining dvc compared to [10] (see Appendix III). Our security goals mentioned in Section II-C are achieved.

- The adversary needs sk_{svr} as well as rst for breaking the proposed system simply because (1) c is perturbed and x is random padded, (2) s is random encoded or encrypted under pk_{svr}, and (3) e_1 has κ bits in length. As for (1), an adversary cannot expect any kinds of known characteristics from c since $D_a(x)$ and $\mathcal{R}^{-1}(D_a(x))$ will both produce random data even with a correct guess. As for (2), (i) signing the same message will not produce the same signature s even with a correct private key, and (ii) the adversary, who listened s_0 and s_1 in the open protocol and obtained dvc, cannot find any relation between them because s is explicitly encrypted to s_0 under pk_{svr} in the open protocol. A timestamp included in s_0 by $R()$ will resist replays sent to svr. As for (3), e_1 is securely encrypted and not guessable at all so that the adversary cannot utilize it for his or her attack.

Number of Multiplications	*Client* dvc	*Server* svr	*Communication Costs in Bytes*	
Cryptographic Camouflage 1[5]	1.5λ	$2.25\lambda + (\sigma + 1)$	$m1 : \lceil \frac{\kappa}{2^3} \rceil$	$m2 : \lceil \frac{3\lambda}{2^3} \rceil$
Cryptographic Camouflage 2[5]	$1.5(\sigma + 1)\lambda$	$1.5\lambda + (1.5\kappa + \sigma + 1)$	$m1 : \lceil \frac{\kappa}{2^3} \rceil$	$m2 : \lceil \frac{3\lambda}{2^3} \rceil$
Robust Software Token	1.5λ	$1.5\lambda + (1.5\kappa + 2\sigma + 2)$	$m1 : \lceil \frac{\kappa}{2^3} \rceil$	$m2 : \lceil \frac{3\lambda}{2^3} \rceil$

TABLE II

Performance Evaluation of Software Tokens in Closed PKIs

Number of Multiplications	*Client* dvc	*Server* svr	*Communication Costs in Bytes*	
Networked Crypto Device[10]	$1.5\lambda + (3\sigma + 4)$	9λ	$m1 : \lceil \frac{5\lambda+\kappa}{2^3} \rceil$	$m2 : \lceil \frac{\lambda}{2^3} \rceil$
Robust Software Token 1	$1.5\lambda + (\sigma + 1)$	$3\lambda + (1.5\kappa + 2\sigma + 2)$	$m1 : \lceil \frac{3\lambda+\kappa}{2^3} \rceil$	$m2 : \lceil \frac{\kappa}{2^3} \rceil$
Robust Software Token 2	$1.5\lambda + (2\sigma + 2)$	$3\lambda + (1.5\kappa + 2\sigma + 2)$	$m1 : \lceil \frac{3\lambda+\kappa}{2^3} \rceil$	$m2 : \lceil \frac{\lambda}{2^3} \rceil$

TABLE III

Performance Evaluation of Software Tokens in Open PKIs

- The adversary still needs off-line attacks for obtaining the correct private key even if (s)he compromised dvc and svr, because c is encrypted under the password-derived key a and dvc does not show the password information to any party ahead in the protocol run.
- The digital identity aimed for the closed PKI cannot be re-used for the open PKI and vice versa. For the purposes, the closed signature explicitly included the recipient's identifier and perturbed the pre-image like $h(\mathsf{svr}, h(\mathcal{G}))$ while the open signature did not.

As we examined above, the security may hold[2] in the closed PKI as well as the open PKI, and each domain can be distinguished even with the same public key and private key pair.

B. Efficiency

If the RSA exponent is chosen at random, RSA encryption using the repeated square-and-multiply algorithm will take λ modular squarings and expected $\frac{\lambda}{2}$ modular multiplications[11]. From this point of view, we can evaluate the computational performance of

[2]Though the security can be hopefully proved by informal descriptions above, a formal security proof would like to be discussed in the next version.

software token methods in a simple manner. Let us compare the proposed scheme with the cryptographic camouflage[5] in closed PKIs (Table II) and the networked cryptographic device[10] in open PKIs (Table III) separately since both object schemes are such restricted. We assume m's length κ in open schemes, and cryptographic camouflage schemes and our schemes include the certificate verification process. Let $\sigma = 16$ while $e = 2^{16} + 1$ and $pk_{\mathsf{svr}} = 2^{16} + 1$. The key size and the communication costs are such approximated.

In Table II, two versions of the camouflaged tokens for resisting a Wiener's attack[20], [19] as well as a brute force attack[5] are considered. Note that camouflage scheme 1 needs e of length $\frac{\lambda}{2}$ while camouflage scheme 2 does e of length κ but with d such as $d + l\phi(N)$ of length $(\sigma + 1)\lambda$[5]. In camouflage scheme 1, dvc needs $\frac{3\lambda}{2}$ multiplications while svr does $\frac{9\lambda}{4} + (\sigma + 1)$. In camouflage scheme 2, dvc needs $\frac{3\lambda(\sigma+1)}{2}$ multiplications while svr does $\frac{3(\lambda+\kappa)}{2} + (\sigma + 1)$. Appendix II describes the protocol. In our scheme, dvc needs $\frac{3\lambda}{2}$ multiplications while svr does $\frac{3(\lambda+\kappa)}{2} + 2(\sigma + 1)$. Camouflage scheme 1 explicitly needs much more computation for svr and camouflage scheme 2 does for dvc than our scheme.

In Table III, the networked cryptographic device is compared to our two open schemes. Note that τ needs at least three λ-blocks and γ does two λ-blocks depending on m in the networked cryptographic device[10]. Appendix III describes the scheme in detail. In the networked cryptographic device, dvc needs $\frac{3\lambda}{2} + 3(\sigma + 1) + 1$ multiplications while svr does at best $\frac{18\lambda}{2}$ $(= \frac{3\lambda}{2} * 3 + \frac{3\lambda}{2} * 2 + \frac{3\lambda}{2})$ because of τ and γ. In our scheme 1 (entrusted), dvc needs $\frac{3\lambda}{2} + (\sigma + 1)$ multiplications while svr does $\frac{(6\lambda+3\kappa)}{2} + 2(\sigma + 1)$ including the certificate verification. In our scheme 2 (assisted), dvc needs $\frac{3\lambda}{2} + 2(\sigma + 1)$ multiplications while svr does $\frac{(6\lambda+3\kappa)}{2} + 2(\sigma + 1)$ including the certificate verification. The networked cryptographic device explicitly needs much more computation for svr and communications costs than our schemes. As for the computation of dvc, our schemes are still slightly more efficient.

VII. Conclusion

The private key management is very important for security and privacy related to digital identities. This paper presented the new robust software token for securing private keys by software-only techniques. We would like to say that our schemes improved the related noble work[5], [10] in many features. Compared to the related work, the proposed schemes were more efficient and uniquely allowed more protocol setup in closed as well as open

PKIs without sacrificing security. Also our schemes allowed the conventional encoding rule for open signatures. We hope for the proposed schemes to be accepted in practice because one public key and two private keys (one for *the robust software token* and the other for *the tamper-resistant hardware token*) must be useful for securing digital identities.

References

[1] S. Bellovin and M. Merrit, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," In Proceedings of the *IEEE Symposium on Security and Privacy*, pp.72-84, 1992.

[2] S. Brands, *Rethinking Public Key Infrastructures and Digital Certificates*, The MIT Press, pp.219-224, 2000.

[3] C. Boyd, "Digital multisignatures," *Cryptography and Coding*, Oxford University Press, pp.241-246, 1989.

[4] R. Ganesan, "Yaksha: Augmenting Kerberos with public key cryptography," In Proceedings of the *ISOC Network and Distributed System Security Symposium*, February 1995.

[5] D. Hoover, B. Kausik, "Software smart cards via cryptographic camouflage," In Proceedings of the *IEEE Symposium on Security and Privacy*, 1999, `http://www.arcot.com` .

[6] ISO/IEC 9594-8, "Information technology - Open Systems Interconnection - The Directory: Authentication framework," *International Organization for Standardization*, 1995 (equivalent to ITU-T Recommendation X.509, 1993).

[7] D. Jablon, "Strong password-only authenticated key exchange," *ACM Computer Communications Review*, vol.26, no.5, pp.5-26, 1996

[8] T. Kwon, "Authentication and key agreement via memorable password," In Proceedings of the *ISOC Network and Distributed System Security Symposium*, February 2001.

[9] T. Kwon, "Digital signature algorithm for securing digital identities," To appear in *Information Processing Letters*, 2001.

[10] P. MacKenzie and M. Reiter, "Networked cryptographic devices resilient to capture," In Proceedings of the *IEEE Symposium on Security and Privacy*, 2001, a full and updated version is DIMACS Technical Report 2001-19, May 2001.

[11] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, pp.290-291, 1997.

[12] R. Morris and K. Thompson, "Password security: a case history," *Communications of the ACM*, vol.22, no.11, pp.584-597, 1979.

[13] R. Perlman and C. Kaufman, "Secure Password-Based Protocol for Downloading a Private Key," In Proceedings of the *ISOC Network and Distributed System Security Symposium*, February 1999.

[14] PKCS #1, "RSA cryptography standard," *RSA Laboratories Technical Note*, Version 2.0, 1998.

[15] PKCS #5, "Password-based encryption standard," *RSA Laboratories Technical Note*, Version 2.0, 1999.

[16] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol.21, pp.120-126, 1978.

[17] Rainbow Technologies, *http://www.rainbow.com/* .

[18] RSA Security Laboratories, *http://www.rsasecurity.com/* .

[19] G. Simmons, "A "weak" privacy protocol using the RSA crypto algorithm," *Cryptologia*, vol.7, pp.180-182, 1983.

[20] M. Wiener, "Cryptanalysis of short RSA secret exponents," *IEEE Transactions on Information Theory*, vol.36, no.3, May 1990.

Appendix

I. PKCS#5-Encrypted Private Key

Appendix I is described for a beginner. Advanced readers may skip this. PKCS#5 is the password-based cryptography standard intended for protecting sensitive information[15]. PKCS#5 offers useful tools such as key derivation functions, encryption schemes, and message-authentication schemes based on a human memorable password having the relatively low entropy. This is accordingly the most widely used form of protecting the private key in software. Actually such user credential information can be stored and transported as defined in PKCS#8 and #12 owing to this. However, this scheme may disclose the plain key in polynomial time to an adversary who compiled a dictionary of likely passwords.

Let t be salt, o a counter, $dkLen$ the required key length, and dk the derived key. Then $dk \leftarrow \mathcal{F}(\pi, t, o, dkLen)$ denotes the key derivation function $\mathcal{F}()$, namely PBKDF, outputs dk[15]. We usually define $t \leftarrow \{0,1\}^{\kappa}$ and $o \leftarrow 1000$. Also we say private key sk is PKCS#5 encrypted by describing $\varepsilon \leftarrow E_{dk}(sk)$. That is, sk is BER encoded and encrypted under dk[15]. We may decrypt this with a function $D()$, say $sk \leftarrow D_{dk}(\varepsilon)$. Note that salt is traditionally used for the benefits such as disallowing to precompute all the keys corresponding to a dictionary of passwords, and minimizing the possibility of selecting the same key, while counters for increasing the cost of producing keys from passwords.

The PKCS#5-encrypted private key is vulnerable to off-line attacks because t and o are saved in plain form, and the deterministic encoding method is used. For example, the following attack works in a password space. An adversary who captured the PKCS#5 based token, $< t, o, dkLen, \varepsilon, \mathcal{C} >$, tries to decrypt ε using his or her dictionary of π'.

$$dk' \leftarrow \mathcal{F}(\pi', t, o, dkLen)$$
$$sk' \leftarrow D_{dk'}(\varepsilon)$$

If sk' is not BER decoded, (s)he can discard the wrong guesses. Otherwise, (s)he tries to sign an arbitrary message m with the decrypted and verify the signed message with $\mathcal{C}$.

Provided that the signature is not valid, (s)he can discard the wrong guesses. Otherwise (s)he can expect it correct. (S)he repeats this procedure until (s)he gets a correct one.

II. Cryptographic Camouflage

Hoover and Kausik presented a pioneer study called cryptographic camouflage in 1999[5]. The key idea was to make the plaintext unidentifiable by encrypting random-looking plaintext only and hiding the user's public key from others except the trusted server[5]. Their scheme inspired many related work including this paper[9], [10], but has several distinct disadvantages against the related schemes.

For initializing the camouflaged token, a user must choose carefully the public key.

$e \leftarrow_R \{Z_{\phi(N)}\}^{\kappa}$

The user's key pair, a certificate, a password, and a server's public key are loaded by kgc. They are $< e, N >$, $< d, N, \phi(N) >$, $\mathcal{C}$, π, and pk_{svr}. Then kgc computes the followings:

$$
\begin{aligned}
&a \leftarrow k(\pi)\\
&l \leftarrow_R \{0,1\}^{\sigma}\\
&d_1 \leftarrow d + l\phi(N)\\
&b \leftarrow Q(d_1)\\
&c \leftarrow E_a(b)\\
&\rho \leftarrow \mathcal{E}_{pk_{\mathsf{svr}}}(R(\mathcal{C}))
\end{aligned}
$$

The values $< c, N, \rho >$ are stored in the software token and the other values are all erased. Note that $d + l\phi(N)$ is a normal integer addition and $Q()$ truncates MSB and LSB. The user having the software token can run the two-factor authentication protocol depicted in Figure 5, with the trusted server only. This protocol is valid in closed PKIs only and comparable to the protocol depicted in Figure 1. As we mentioned already, this work inspired many related work including this paper[9], [10], but has several distinct disadvantages against the related schemes.

Firstly the authentic public key is not public any more. Consequently its application is restricted to authentication in the closed PKI only. Its security does not hold in open PKIs.

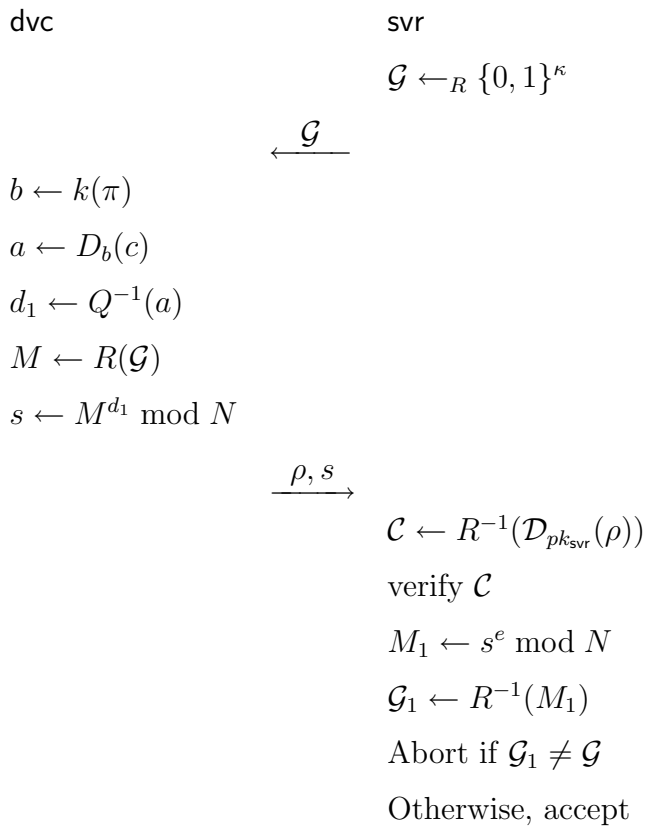

Fig. 5. Cryptographic Camouflage's Two-Factor Authentication Protocol

Secondly it is less efficient than the closed scheme proposed in this paper (see Table II of this paper). Two versions were made for resisting a Wiener's attack[20], [19] as well as a brute force attack[5]. They needed a longer private key and public key pair, for example, $d + l\phi(N)$ having $(\sigma + 1)\lambda$ bits in length where we let σ the bit length of l, and a public key e having κ bits in length. Otherwise, e of length $\frac{\lambda}{2}$ is necessary for having d of length λ[5]. For the reasons, performance is worse than the protocol proposed in this paper, as we discussed in Table II. In camouflage scheme 1 that needs e of length $\frac{\lambda}{2}$, dvc needs $\frac{3\lambda}{2}$ multiplications while svr does $\frac{9\lambda}{4} + (\sigma + 1)$. In camouflage scheme 2 that needs d of length $(\sigma + 1)\lambda$, dvc needs $\frac{3\lambda(\sigma+1)}{2}$ multiplications while svr does $\frac{3(\lambda+\kappa)}{2} + (\sigma + 1)$.

Finally it cannot accommodate the deployed user credentials, meaning that a user has to be issued another new key pair for using the camouflage scheme. The reason is clear that the public key must be encrypted and each key must be chosen by carefully considering its length as we mentioned above.

III. Networked Cryptographic Device

MacKenzie and Reiter presented the networked cryptographic device along with a noble security proof in 2001[10] inspired by work of [4]. The key idea was to split the user's private key and share them between dvc and svr by encrypting the server's portion under pk_{svr}. This scheme is advantageous to the related schemes in their formal security proof. In spite of the proof, their assumption on untrusted server was wrong. As we noted earlier, the trusted server is eventually necessary for security reasons. They also suggested the new ability for the user to disable the private key provided that the token is compromised.

For initializing the software token, the user's key pair, a password, and a server's public key are loaded by kgc. They are $< e, N >$, $< d, N, \phi(N) >$, π, and pk_{svr}. Then kgc computes the followings:

$t \leftarrow_R \{0,1\}^\kappa$
$u \leftarrow h_{\mathsf{dsbl}}(t)$
$v \leftarrow_R \{0,1\}^\kappa$
$a \leftarrow_R \{0,1\}^\kappa$
$b \leftarrow h(\pi)$
$d_1 \leftarrow f(v, \pi)$
$d_2 \leftarrow d - d_1 \bmod \phi(N)$
$\tau \leftarrow \mathcal{E}_{pk_{\mathsf{svr}}}(< a, b, u, d_2, N >)$

The values $< t, v, a, \tau, N, pk_{\mathsf{svr}} >$ are considered a private key portion and stored in the software token along with $\mathcal{C}$ while the other values are all erased. Here note again that the bit length of τ is bounded to at least 3λ and affects the efficiency. The values $< t, \tau >$ are backed up off line for key disabling features. The user having the software token can run the assisted signature protocol depicted in Figure 6, with svr. The networked cryptographic device scheme can provides the assisted signature only. Also note again that the bit length of γ is bounded to at least 2λ and affects heavily the efficiency depending on the size of m. Table III only assumed the size of m at around 160 bits but larger m will severely deteriorate the efficiency.

This protocol is valid only for the assisted setup in open PKIs and comparable to the protocol depicted in Figure 4. In Figure 6, random encoding is explicitly described through r and enc, an arbitrary encoding function.

Though their scheme is elegant, a comparison should be made more clearly with our schemes. At this time we cautiously urge that our schemes are advantageous to it.

Firstly their scheme did not allow svr to verify the digital identity of the counterpart, based on the certificate issued in the PKI, while our scheme did it. In other words, their system was designed only for assisting signatures in the open PKI. Authentication to the trusted server was simply by showing the user's password information, exactly a hash value of the password, rather than an authentic public key to svr. Note that, however, the password information was not verified by svr at initialization time, meaning that svr actually cannot verify the real identity of dvc at protocol run time. The thing to be done by svr at protocol run time is only assisting dvc to sign an arbitrary message without exactly identifying who (s)he is. Accordingly this scheme does not support the digital identity in closed PKIs[5], [9]. Even worse, such password information can be verified by adversaries when dvc communicates with the compromised or misbehaving server. Then, afterward, those adversaries can forge the user's digital identity as soon as compromising dvc. These may be disadvantageous to the related work[5], [9]. If one says these property is rather beneficial to anonymous transactions with svr, our scheme also supports this idea in a simple way. That is, dvc does not give $\mathcal{C}$ but rather e only to svr (see Figure 3 and 4). For this function, e can be included in y at initialization time.

Secondly our scheme even allows to use the commonly used encoding functions such as BER/DER encoding for open signatures. However, their scheme needs to use a specific encoding rule for security reasons. As for the open signaure rather than the closed one, our encoding-free feature to follow the conventional encoding rule must be advantageous.

Thridly our scheme also supports the entrusted setup while their scheme does not. Note that the entrusted setup as well as the assisted setup must be useful for these schemes.

Finally their scheme is less efficient than our scheme as described in Table III. In the networked cryptographic device, dvc needs $\frac{3\lambda}{2} + 3(\sigma + 1) + 1$ multiplications while svr does at best $\frac{18\lambda}{2}$ $(= \frac{3\lambda}{2} * 3 + \frac{3\lambda}{2} * 2 + \frac{3\lambda}{2})$ depending on m, because of τ and γ.

vrf	dvc		svr
	$\beta \leftarrow h(\pi)$		
	$\rho \leftarrow_R \{0,1\}^{\lambda}$		
	$r \leftarrow_R \{0,1\}^{\kappa_{sig}}$		
	$\gamma \leftarrow \mathcal{E}_{pk_{\mathsf{svr}}}(m, r, \beta, \rho)$		
	$\delta \leftarrow \mathsf{mac}_a(\gamma, \tau)$		
		$\xrightarrow{\gamma, \delta, \tau}$	
			$< a, b, u, d_2, N > \leftarrow \mathcal{D}_{sk_{\mathsf{svr}}(\tau)}$
			abort if $\mathsf{mac}_a(\gamma, \tau) \neq \delta$
			$< m, r, \beta, \rho > \leftarrow \mathcal{D}_{sk_{\mathsf{svr}}(\gamma)}$
			abort if $(\beta \neq b)$
			$\nu \leftarrow (\mathsf{enc}(m, r))^{d_2} \bmod N$
			$\eta \leftarrow \rho \oplus \nu$
		$\xleftarrow{\eta}$	
	$\nu \leftarrow \rho \oplus \eta$		
	$d_1 \leftarrow f(v, \pi)$		
	$s \leftarrow \nu(\mathsf{enc}(m, r))^{d_1} \bmod N$		
	$m_1 \leftarrow s^e \bmod N$		
	$m_2 \leftarrow \mathsf{enc}(m, r)$		
	abort if $m_1 \neq m_2$		
verify $\mathcal{C}$	$\xleftarrow{\mathcal{C}, s, m, r}$		
$m_1 \leftarrow s^e \bmod N$			
$m_2 \leftarrow \mathsf{enc}(m, r)$			
Abort if $m_1 \neq m_2$			
Otherwise, accept			

Fig. 6. Networked Cryptographic Device's Open Signature Protocol