Self-Certified Signatures

Byoungcheon Lee[1] and Kwangjo Kim[2]

[1] Joongbu University,
San 2-25, Majon-Ri, Chuboo-Meon, Kumsan-Gun, Chungnam, 312-702, Korea
sultan@joongbu.ac.kr

[2] International Research center for Information Security (IRIS),
Information and Communications University (ICU),
58-4, Hwaam-dong, Yusong-gu, Daejeon, 305-732, Korea
kkj@icu.ac.kr

Abstract. A digital signature provides the authenticity of a signed message with respect to a public key and a certificate provides the authorization of a signer for a public key. Digital signature and certificate are generated independently by different parties, but they are verified by the same verifier who wants to verify the signature. In the point of a verifier, verifying two independent digital signatures (a digital signature and the corresponding certificate) is a burden.

In this paper we propose a new digital signature scheme called *self-certified signature*. In this scheme a signer computes a temporary signing key with his long-term signing key and its certification information together, and generates a signature on a message and certification information using the temporary signing key in a highly combined and unforgeable manner. Then, a verifier verifies both signer's signature on the message and related certification information together. This approach is very advantageous in efficiency. We extend the proposed self-certified signature scheme to *multi-certification signature* in which multiple certification information need to be verified. We apply it to public key infrastructure (PKI) and privilege management infrastructure (PMI) environments.

Keywords: digital signature, self-certified signature, self-certified key, multi-certification signature, public key infrastructure, privilege management infrastructure

1 Introduction

1.1 Digital Signature and Certification

A digital signature is computed by a signer from a message and his signing key. When the signature is verified to be valid with the corresponding public key, it provides the authenticity of the signed message with respect to the public key. But the signature is only linked to the public key and does not provide the authorization of the signer by itself. To provide the authorization of the signer for the public key, certificate is used, which is signed by a trusted third party. In X.509 [PKI], a certification authority (CA) can provide the signer with a certificate which is a digital signature of CA on the public key and relevant certification information such as serial number, identity of the signer, identity of CA, period of validity, extensions, *etc.* In other words a certificate provides an unforgeable and trusted link between a public key and a specific signer. Whenever a verifier wants to use the public key to verify a signature, he first has to check the validity of the certificate using CA's public key.

Public key infrastructure (PKI) [PKI] is a hierarchical framework to issue and manage certificates. It is also said that PKI is a trust management infrastructure. It is a key infrastructure for the digital signature technology to be adapted in real world. Recently, many countries over the world enact the digital signature act which provides legal support to the validity of digital signature. Nowadays, PKI industry is booming and digital signature technology is being adapted quickly in our real life.

Digital signature and certificate are generated independently by different parties, but they are verified by the same verifier who wants to verify the signature. In the point of a verifier, verifying two independent digital signatures (a digital signature on a message and the corresponding certificate) is a burden. Moreover the verifier has to locate and keep the corresponding certificate by himself. Therefore more elegant and efficient approach for the verification of digital signature and certificate is required.

To solve this problem in more efficient way, we propose a new digital signature scheme called *self-certified signature (SCS)*. In this scheme a signer computes a temporary signing key with his long-term signing key and certification information together, and generates a signature on a message and certification information using the temporary signing key in a highly combined and unforgeable manner. Then, a verifier verifies both signer's signature on the message and related certification information together. This approach has many advantages in efficiency (computation and communication) and in real world usage.

Moreover, in PKI and PMI environment many additional certification information need to be checked together, such as certificate revocation list (CRL), certification path from the root CA to the signer, attribute certificates, *etc.* We extend the SCS scheme to *multi-certification signature (MCS)* in which multiple certification information are verified, and apply it to PKI and PMI environment.

1.2 Related Concepts

The concept of SCS has some similarity with identity-based cryptosystem, self-certified key (SCK), and proxy signature scheme. These concepts commonly deal with the issue that how to certify public keys. The most familiar approach to certify a public key is using explicit certificate such as X.509, but these concepts show other possibilities to manage certification.

In identity-based scheme, first introduced by Shamir [Sha84], the public key is nothing but the identity of the signer and the related secret key is computed from some trapdoor originated by CA. This scheme is very attractive because it needs no certificate and no verification of certificate, hence reduces the amount of storage and computation, but it's disadvantage is that the secret key is known to CA.

The concept of self-certified key (SCK), introduced in [Gir91], is a sophisticated combination of certificate-based and identity-based models. Using an RSA cryptosystem a user chooses his secret key, computes public key, and gives it to the authority. Then the authority computes a certification parameter for the user which satisfies a computationally unforgeable relationship with the public key and the identity. A verifier can compute the public key from the identity and the certification parameter. [PH97] extended [Gir91] to DLP-based cryptosystem in which self-certified key is issued securely using weak blind Schnorr signature protocol. A problem of SCK schemes is that it provides only implicit authentication, *i.e.*, the validity of a SCK is verified only after a successful communication. [LK00] improved [PH97] such that explicit authentication of SCK is provided by using the concept of self-certificate.

In the point of cryptographic primitive, SCS is similar to proxy signature schemes [MUO96,PH97,KPW97,LKK01a,LKK01b]. Proxy signature is a signature scheme in which an original signer delegates her signing capability to a proxy signer, and then the proxy signer creates a signature on behalf of the original signer. From a proxy signature a verifier can check both original signer's delegation and proxy signer's digital signature. The basic methodology used in proxy signature scheme is that the original signer creates a signature on delegation information (ID of the designated proxy signer, validity period, specification on message space, or any warrant information) and gives it secretly to the proxy signer, and then the proxy signer uses it to generate a proxy key pair. Because the proxy key pair is generated from original signer's signature on delegation information, any verifier can check original signer's delegation from a proxy signature. In SCS scheme certification information is used in a similar way as the delegation information in proxy signature scheme.

More recently, proxy certificate profile was proposed in [PC]. Proxy certificate can be used for entity A to give the delegation information to entity B in the form of certificate. Then B can authenticate with others as if it were A.

These concepts commonly specify how to make key pair. SCK scheme is a key issuing protocol with no specification on signing and verification, but SCS scheme contains signing and verification algorithms together with key generation algorithm. Proxy signature schemes [PH97,KPW97,LKK01a] specify signing and verification algorithms, but they are signatures only on message. As will be shown in later Section, there are possibilities of forgery if proxy signature scheme is a signature only on message. Proxy signature schemes in [LKK01b] are signatures on message and delegation information together, but detailed analysis was not given. On the other hand, SCS schemes are signatures both on message and certification information together such that a verifier can verify both the digital signature on message and certification information in an efficient way. Since a SCS contains certification information in a digital signature, it provides more concrete non-repudiation evidence than a normal digital signature.

1.3 Our Contributions

To provide the authenticity of a digital signature and the authorization of a public key together in an efficient way, we introduce a new digital signature scheme called self-certified signature. In this approach the signer generates a temporary signing key using his long-term signing key and CA's certification information together and signs a message and certification information using this temporary signing key. In the verification stage both the signature on the message and certification information are checked together.

We extend the proposed SCS scheme to multi-certification signature (MCS) in which multiple certification information need to be verified. We apply MCS scheme to public key infrastructure (PKI) and privilege management infrastructure (PMI) environments in which many additional certification information, such as certificate revocation list (CRL), path validation from the root CA to the signer, attribute certificate (AC), *etc*, have to be verified. A signer can collect all the certification information required to verify the authorization for his public key and compute a MCS. Then it provides more concrete non-repudiation evidence than a normal digital signature. In the point of a verifier, he doesn't need to locate all the required certification information by himself.

The paper is organized as follows: In Section 2 we define SCS scheme and show a general implementation of SCS based on DLP. We also show a distinguished implementation of SCS and discuss its security. In Section 3 we extend the proposed SCS scheme to MCS and apply it to PKI and PMI environments in which many additional certification information have to be verified. We compare the efficiency of MCS scheme with a general multiple signature approach. Finally, we conclude in Section 4.

2 Self-Certified Signature

Assume that a signer $\mathcal{S}$ has a long-term key pair (x_0, y_0) where x_0 is a secret signing key and y_0 is the corresponding public key. The public key y_0 is certified by a certification authority $\mathcal{CA}$ with a certificate $Cert_s$. $\mathcal{CA}$ issues $Cert_s$ as a certificate for the public key y_0 and the signer $\mathcal{S}$ by signing a certification information CI_s prepared by himself. According to X.509, CI_s can include information such as serial number, signer's identity, issuer's identity, public key y_0, period of validity, extensions, *etc.* To sign a message using SCS scheme, the signer $\mathcal{S}$ computes a temporary key pair (x, y) using his long-term key pair (x_0, y_0) and the certificate $Cert_s$.

The basic idea of the proposed SCS scheme is as follows.

1. A signer $\mathcal{S}$ computes a temporary signing key x for SCS using his long-term signing key x_0 and certificate $Cert_s$ such that it can be computed only by himself.
2. $\mathcal{S}$ signs a message and related certification information using the temporary signing key x to generate a self-certified signature σ.

3. A verifier $\mathcal{V}$ computes the temporary public key y from signer's long-term public key y_0 and certification information, and verifies the self-certified signature σ using y.

The resulting SCS is a combination of general signature scheme and certification scheme, therefore it should satisfy the non-repudiation requirement of general signature scheme and certification requirement of certification scheme.

In this paper we use the following notation.

- $\mathcal{S}$: a signer
- $\mathcal{V}$: a verifier
- $\mathcal{CA}$: a certification authority
- (x_0, y_0): signer's long-term key pair (secret signing key, public key)
- (x, y): signer's temporary key pair for SCS
- CI_s: certification information, prepared by $\mathcal{CA}$, for the public key y_0 and the signer $\mathcal{S}$
- $Cert_s$: a certificate, issued by $\mathcal{CA}$, for the public key y_0 and the signer $\mathcal{S}$
- $S_x(m)$: a signing algorithm on message m using a signing key x
- $V_y(s, m)$: a verification algorithm of a signature s using a public key y
- $S_x(m_1, m_2)$: a two-message signing algorithm on messages m_1 and m_2 using a signing key x
- $V_y(s, m_1, m_2)$: a two-message verification algorithm of a signature s on messages m_1 and m_2 using a public key y
- $h(), h_1(), h_2()$: collision resistant hash functions
- m: a message
- σ: a self-certified signature

2.1 Definition of SCS

First, we need to consider how to sign two messages together.

Definition 1 (Two-message signature). *Let m_1 and m_2 be two messages that a signer $\mathcal{S}$ wants to sign together. Let $S_x(m)$ be a signing algorithm which is a signature on message m using a signing key x. Then two-message signature is a signature on two messages m_1 and m_2 together and we denote it as $S_x(m_1, m_2)$, where m_1 and m_2 are called the first and second message.*

The most general approach of two-message signature is to prepare a new message by concatenating two messages as $m = (m_1 || m_2)$ and then sign m using a general signature scheme. But there can be numerous modifications. We will show an example in later Section. Now we define self-certified signature.

Definition 2 (Self-certified signature). *Let (x_0, y_0) be a signer's long-term key pair where x_0 and y_0 are a secret signing key and the corresponding public key, respectively. Let $Cert_s$ be a certificate for the public key y_0 issued by $\mathcal{CA}$. Self-certified signature scheme consists of the following three algorithms.*

1. ***Key generation algorithm*** *takes the long-term key pair* (x_0, y_0) *and certificate* $Cert_s$ *and outputs a temporary key pair* (x, y)

$$x = f(x_0, Cert_s), \ y = g(y_0, Cert_s)$$

   where f, g *are public algorithms.*
2. ***Signing algorithm*** *is a probabilistic algorithm which takes a message* m*, a certificate* $Cert_s$*, and the temporary signing key* x *as input and outputs a self-certified signature* $\sigma = S_x(m, Cert_s)$ *using the two-message signature scheme where the first message is* m *and the second message is* $Cert_s$*.*
3. ***Verification algorithm*** *takes a self-certified signature* σ*, a message* m*, a certificate* $Cert_s$*, a long-term public key* y_0 *as input and outputs a binary value 0 (invalid) or 1 (valid). It is a three-step algorithm.*
   - *It computes the temporary public key* $y = g(y_0, Cert_s)$*,*
   - *It verifies the self-certified signature using* y*,* $V_y(\sigma, m, Cert_s) \stackrel{?}{=} 1$*.*
   - *It checks whether* y_0 *is stated in* $Cert_s$ *correctly (This is just a document test, not a signature verification).*

If all the verifications hold, it represents that the signature of the signer on message m *is valid and certification by CA is also confirmed. If the document test of* $Cert_s$ *is not valid, the self-certified signature is considered to be invalid, although the signature verification was passed.*

Self-certified signature scheme should satisfy the following security requirements.

1. ***Non-repudiation:*** *The self-certified signature should be generated only by the signer* $\mathcal{S}$ *who knows the long-term signing key* x_0*. Therefore the signer should not be able to repudiate his signature creation later.*
2. ***Certification:*** *From the self-certified signature a verifier should be convinced that the signer* $\mathcal{S}$ *is authorized to use the public key* y_0 *by the trusted authority* $\mathcal{CA}$*.*

2.2 Attack Models against SCS

The most powerful attack model on digital signature scheme is the adaptively chosen message attack [PS00] in which an adversary is allowed to access the signing oracle as many times as she wants and get valid signatures for messages of her choice. In SCS scheme the attacker is more advantageous since she has additional knowledge of certification information. There is also possibility for the signer to misuse the temporary signing key. We consider the following additional attack scenarios.

1. **Forgery using partial information (by third party):** In SCS scheme partial information of the temporary signing key x is known to third parties, *i.e.*, the certification information $Cert_s$ is published (but the long-term signing key x_0 is kept secret). Moreover the algebraic relationship between

x and x_0 is publicly known. The long-term signing key x_0 can be used to generate normal signatures, while the temporary signing key x is used to generate SCS. If the SCS scheme is not secure, an active attacker can try to change the certification information, induce a valid normal signature from a valid SCS, or induce a valid SCS from a valid normal signature. For the SCS scheme to be secure, this partial information should be of no help for any malicious third party to forge a valid signature.

2. **Key generation stage attack (by signer):** In SCS scheme the signer computes the temporary signing key x by himself. A malicious signer can try to use it for another malicious purpose. For example, he gets a certificate for (x, y) from another CA without exposing the previous certification information and uses it for malicious purpose. He can show the previous certification information later when it is necessary. For the SCS scheme to be secure, this kind of malicious usage of the temporary key pair should be prevented and detected easily.

These attacks can work in proxy signature schemes [PH97,KPW97,LKK01a] if it is a signature only on message.

2.3 General Implementation of SCS based on DLP

The SCS scheme can be implemented easily using the DLP-based cryptosystem if system parameters are shared among signer's key pair and CA's key pair. We consider the Schnorr signature scheme as an underlying signature scheme.

Firstly, we review Schnorr signature scheme briefly. Let p and q be large primes with $q|p-1$. Let g be a generator of a multiplicative subgroup of Z_p^* with order q. $h()$ denotes a collision resistant cryptographic hash function. Assume that a signer has a secret signing key x and the corresponding public key $y = g^x \bmod p$. To sign a message m, the signer chooses a random number $k \in_R Z_q^*$ and computes $r = g^k$, $s = x \cdot h(m, r) + k$. Then the tuple (r, s) becomes a valid signature on message m. The validity of signature is verified by $g^s \stackrel{?}{=} y^{h(m,r)}r$. Note that the signing process requires one offline modular exponentiation and the verification of signature requires two online modular exponentiations. This signature scheme has been proven to be secure under the random oracle model [PS96,PS00]. They have shown that existential forgery under the adaptively chosen message attack is equivalent to the solution of discrete logarithm problem.

We assume that a signer $\mathcal{S}$ has a long-term key pair (x_0, y_0) where $y_0 = g^{x_0} \bmod p$. He also has a certificate $Cert_s$ on the public key y_0 issued by $\mathcal{CA}$. We also assume that the same system parameters p and q are shared among signer's key pair and $\mathcal{CA}$'s key pair. Let (x_{CA}, y_{CA}) be $\mathcal{CA}$'s key pair where $y_{CA} = g^{x_{CA}}$. The certificate $Cert_s = (r_c, s_c)$ on public key y_0 is $\mathcal{CA}$'s Schnorr signature on some certification information CI_s prepared by $\mathcal{CA}$, which includes serial number, long-term public key y_0, signer's identity, issuer's identity, period of validity, extensions, *etc.* To issue $Cert_s$ $\mathcal{CA}$ chooses $k_c \in_R Z_q^*$ and computes

$$Cert_s = (r_c, s_c) = (g^{k_c}, x_{CA} \cdot h(CI_s, r_c) + k_c).$$

It's validity is verified by $g^{s_c} \stackrel{?}{=} y_{CA}^{h(CI_s,r_c)} r_c$. $\mathcal{CA}$ has issued $Cert_s = (r_c, s_c)$ to $\mathcal{S}$ as a certificate for the public key y_0.

Now the self-certified signature scheme on a message m and a certificate $Cert_s$ is given by the following three algorithms.

General Implementation of SCS:

1. **Key generation:** A signer $\mathcal{S}$ computes a temporary key pair (x, y) by using the long-term key pair (x_0, y_0) and the certificate $Cert_s$ as
$$x = x_0 + s_c,\ \ y = y_0 y_{CA}^{h(CI_s,r_c)} r_c.$$
2. **Signing:** Using the temporary signing key x, $\mathcal{S}$ computes a self-certified signature $\sigma = (r, s)$ on message m and certificate $Cert_s$ as follows.
   – Prepare a concatenated message $m||CI_s||r_c$.
   – Chooses a random number $k \in_R Z_q^*$ and computes a signature as
$$r = g^k,\ \ s = x \cdot h(m||CI_s||r_c, r) + k.$$
   – Gives $\{(r, s), CI_s, r_c\}$ as a SCS on message m.
3. **Verification:** A verifier $\mathcal{V}$ checks the validity of $\{(r, s), CI_s, r_c\}$ as follows.
   – Computes a temporary public key $y = y_0 y_{CA}^{h(CI_s,r_c)} r_c$.
   – Verifies the signature (r, s) using y by $g^s \stackrel{?}{=} y^{h(m||CI_s||r_c,r)} r$.
   – Checks whether y_0 is stated in CI_s correctly (This is just a document test, not a signature verification).

If the verification holds, it means that the signature of the signer on message m is valid and certification by $\mathcal{CA}$ is also confirmed. If the document test of $Cert_s$ is not valid, the self-certified signature is considered to be invalid, although the signature verification was passed. Therefore the signer should construct a valid temporary key pair using correct certification information.

Because the underlying Schnorr signature scheme is secure, the proposed SCS scheme satisfies the security requirements listed above.

1. Non-repudiation: Since the Schnorr signature scheme is a secure signature scheme, any other party who does not know the temporary signing key x cannot forge a valid Schnorr signature on two messages m and $Cert_s$. Therefore the signer cannot repudiate his signature creation later.
2. Certification: Since the Schnorr signature (r, s) is verified by using the temporary public key $y = y_0 y_{CA}^{h(CI_s,r_c)} r_c$, a verifier is convinced that the signer was authorized to use the public key y_0 by $\mathcal{CA}$.

Note that $\{(r, s), CI_s, r_c\}$ is a signature on a combined message $m||CI_s||r_c$ (instead of just m) with a temporary signing key $x = x_0 + s_c$. This prevents additional attacks proposed above.

1. Forgery using partial information (by third party): Third party cannot manipulate a valid SCS to generate a new valid SCS (modifying certification information) or a normal signature (deleting certification information), although he knows additional information s_c.
2. Key generation stage attack (by signer): A malicious signer cannot try to hide the certification information on purpose and expose it later. He can get a new certificate for $x = x_0 + s_c$ from other CA and use it as a new certified key. But when he exposes the hidden certification, it is not accepted since certification information should be included explicitly in message.

Since a SCS includes both a signature on message and certification, it provides more concrete non-repudiation evidence than a normal signature. For example, assume that a normal signature is verified to be valid, but the corresponding certificate is turned out to be invalid, then the signature is not qualified. But a valid SCS is qualified by itself. A complete non-repudiation evidence is provided in SCS scheme, while only partial non-repudiation evidence is provided in normal signature schemes.

In the point of communication, a verifier does not need to locate and keep the certification information by himself because it is included in a SCS. In the point of computation, SCS is more efficient than the general approach of independent signature verification. Detailed efficiency analysis will be given in Section 3.

2.4 Comparison with Self-Certified Key

The proposed SCS scheme can be compared with the self-certified key (SCK) scheme in many ways.

First, SCK scheme is a key issuing protocol and it does not specify how to sign a message using the self-certified key. On the other hand, SCS scheme does not specify how to certify a public key, but specifies how to sign a message and verify a signature using a long-term key pair and the corresponding certificate. Therefore in SCS scheme already wide-spread PKI environment can be used without any change. SCS can be considered as signer's additional service to provide more concrete non-repudiation evidence and more efficient verification of signature. As will be shown later, efficiency is provided mainly in verification stage, not in signing stage.

Second, SCK scheme provides only implicit authentication. Since any kind of certificate is not used explicitly, the authenticity of a public key is guaranteed only when it is used successfully in application. On the other hand, SCS scheme provides explicit authentication since certificate in PKI environment is used. Only difference is that the certificate is used in highly combined manner with the signature in the signing and verification algorithms.

SCS scheme is is different from proxy signature scheme.

2.5 Distinguished Implementation of SCS

If the same digital signature scheme (for example, Schnorr signature) is used both as a normal signature scheme and a SCS scheme, then some argument can happen. The SCS (r, s) generated above can also be considered as a normal signature on message $m||CI_s||r_c$ using a new signing key x. Anyone can launch the following attacks using the public information s_c.

– If the signer signs a message something like $m||CI_s||r_c$ using his long-term signing key x_0, anyone can compute a valid SCS by adding the certification component.
– If the signer generates a SCS on $m||CI_s||r_c$ as above, anyone can compute a normal signature on $m||CI_s||r_c$ by deleting the certification component.

The resulting forgery can be considered not so risky in real world, but the signer should be careful not to sign any maliciously formed message. Although the SCS scheme is secure in cryptographic sense, this kind of argument needs to be removed. Therefore normal signature scheme and SCS scheme need to be implemented in distinguished ways.

The first natural approach in designing SCS scheme is to use the message and certification information in distinguished way in the signing algorithm. First, we introduce a distinguished two-message signature scheme in which two messages are used in different hash functions. It is a slight modification of Schnorr signature scheme.

Distinguished two-message signature scheme: Let m_1 and m_2 be two messages to be signed. Let (x, y) be signer's key pair.

1. **Signing algorithm:** A signer chooses a random number $k \in_R Z_q^*$ and computes a signature as
$$r = g^k, \ s = x \cdot h_1(m_1, r) + k \cdot h_2(m_2, r)$$
where $h_1()$ and $h_2()$ are cryptographic hash functions. Note that the first and the second messages are used in $h_1()$ and $h_2()$, respectively.
2. **Verification algorithm:** A verifier verifies the signature (r, s) as
$$g^s \stackrel{?}{=} y^{h_1(m_1,r)} r^{h_2(m_2,r)}.$$

We consider the security of the distinguished two-message signature scheme. It can be proven that the distinguished two-message signature scheme is secure under an adaptively chosen-message attack.

Theorem 1. *Consider an adaptively chosen message attack in the random oracle model against the distinguished two-message signature scheme. Probabilities are taken over random tapes, random oracles and public keys. If an existential forgery of this scheme has non-negligible probability of success, then the discrete logarithm problem can be solved in polynomial time.*

Proof. The signer can be simulated by a simulator (who does not know the secret key) with an indistinguishable distribution. We denote the signature scheme as
$$r = g^k, \ s = x \cdot e_1 + k \cdot e_2$$

where $e_1 = h_1(m_1, r)$ and $e_2 = h_2(m_2, r)$. A simulator who does not know secret key x can choose $s, e_1, e_2 \in_R Z_q$ and compute $r = g^{s/e_2} y^{-e_1/e_2}$. Then, (r, s) computed by the simulator is indistinguishable from signer's signature. Then the attacker and the simulator can collude in order to break the signature scheme, and we can solve the discrete logarithm. Assume that an existential forgery of this scheme has non-negligible probability of success. Using the Forking lemma [PS96,PS00], we get two valid signatures (r, s, e_1, e_2) and (r, s', e'_1, e'_2) such that $g^s = y^{e_1} r^{e_2}$ and $g^{s'} = y^{e'_1} r^{e'_2}$. Then, from

$$g^{s/e_2} y^{-e_1/e_2} = g^{s'/e'_2} y^{-e'_1/e'_2}$$

the signing key x can be computed as

$$x = (s/e_2 - s'/e'_2)/(e_1/e_2 - e'_1/e'_2).$$

□

Now, we implement a SCS scheme using the distinguished two-message signature scheme and call it a distinguished implementation. In this scheme the first message is the message to be signed and the second message is certification information for the public key. In this implementation the key generation algorithm is the same, but signing and verification algorithms are modified as follows.

Distinguished Implementation of SCS:

1. **Key generation:** same as the general implementation of SCS.
2. **Signing:** Chooses a random number $k \in_R Z_q^*$ and computes a signature as

$$r = g^k, \; s = x \cdot h_1(m, r) + k \cdot h_2(CI_s||r_c, r).$$

3. **Verification:** Verifies the signature (r, s) using y by

$$g^s \stackrel{?}{=} y^{h_1(m,r)} r^{h_2(CI_s||r_c,r)}.$$

Note that message m is used in the first hash function and certification information $CI_s||r_c$ is used in the second hash function. Compared with the general implementation, this modification requires one more online exponentiation in verification.

Since the distinguished two-message signature scheme is a secure signature scheme, distinguished implementation of SCS also satisfies non-repudiation and certification requirements.

3 Multi-Certification Signature and PKI

3.1 PKI and PMI Environments

A digital signature provides the authenticity of a signed message with respect to a public key and a certificate issued by a trusted authority provides the authorization of a signer for a public key. Whenever a verifier wants to use the public key to verify a signature, he first has to check the validity of the certificate using CA's public key. The next question is whether the verifier trusts signer's CA or not, or how to manage the trust relationship between a signer and a verifier. Public key infrastructure (PKI) [PKI] is a hierarchical framework to issue and manage certificates. It is also said that PKI is a trust management infrastructure.

As the trust relationship between a signer and a verifier becomes complex in PKI environment, the verifier should check not only the certificate of the signer, but also various extra certification information related with the certificate.

- He has to check CRL [CRL] to check whether the signer's certificate was revoked or not.
- He has to check certification path from signer's CA to the root CA who is trusted by himself (Check certificates and CRLs of CAs located in the middle of the certification path).

Recently, attribute certificate (AC) and PMI [PMI] are becoming an issue. Since the the public key certificate (PKC) provide authentication only for the key pair and is used for relatively long period of time, it is not suitable to authenticate short-term attributes of signer (such as access control, role, authorization, *etc.*) which are used for short period of time. For these applications attribute authority (AA) issues AC to a signer to certify signer's specific attribute. PMI is an infrastructure to manage ACs while PKI is an infrastructure to manage PKCs.

AC does not use an independent key pair, but has a link to a PKC, therefore same key pair is shared among PKC and AC. When a signer signs a message with the key pair and asserts both certifications of PKC and AC, a verifier has to verify both certifications.

- He has to verify certifications related with AC, if it is asserted by the signer.

Therefore, in the point of a verifier the verification process of a digital signature is a burden and he should be very careful to check every required certifications.

The proposed SCS scheme can be extended to multi-certification situation easily in which multiple certification information should be verified. In this Section we introduce multi-certification signature (MCS) and apply it to PKI and PMI environments.

3.2 Multi-Certification Signature

Multi-certification signature (MCS) scheme is a generalization of the self-certified signature scheme in which multiple certification information are verified together.

Definition 3 (Multi-Certification Signatures). *Multi-certification signature is a self-certified signature in which multiple certification information are used.*

Let (x_0, y_0) be signer's long-term key pair. Let $(c_1, \ldots, c_n)$ be n certification information related with y_0, which can be PKCs, CRLs, ACs, etc. Multi-certification signature scheme consists of the following three algorithms.

1. ***Key generation algorithm*** *takes the long-term key pair (x_0, y_0) and n certification information $(c_1, \ldots, c_n)$ and outputs a temporary key pair (x, y)*

$$x = f(x_0, c_1, \ldots, c_n), \;\; y = g(y_0, c_1, \ldots, c_n)$$

   where f, g are public algorithms.
2. ***Signing algorithm*** *is a probabilistic algorithm which takes a message m, n certification information $(c_1, \ldots, c_n)$, and the temporary signing key x as input and outputs a multi-certification signature $\sigma = S_x(m, c_1, \ldots, c_n)$ using the two-message signature scheme where the first message is m and the second message is $(c_1, \ldots, c_n)$.*
3. ***Verification algorithm*** *takes a multi-certification signature σ, a message m, n certification information $(c_1, \ldots, c_n)$, the long-term public key y_0 as input and outputs a binary value 0 (invalid) or 1 (valid). It is a three-step algorithm.*
   - *It computes the temporary public key $y = g(y_0, c_1, \ldots, c_n)$,*
   - *It verifies the multi-certification signature using y, $V_y(\sigma, m, c_1, \ldots, c_n) \stackrel{?}{=} 1$.*
   - *It checks whether $(c_1, \ldots, c_n)$ are valid (This is just a document test, not a signature verification).*

Now consider a general implementation of MCS based on DLP cryptosystem. We assume that a signer $\mathcal{S}$ has a certified key pair (x_0, y_0) where $y_0 = g^{x_0}$ and n certification information $(c_1, c_2, \ldots, c_n)$ related with it. c_i can be PKCs, CRLs, ACs, *etc*, which are all represented by digital signatures. Here we assume that the same system parameters p and q are shared among signer's key pair and n certification information.

The certification information c_i are digital signatures on some certification messages CI_i related with the key pair (x_0, y_0) in any form and are provided by authorities $\mathcal{A}_i$. Let (x_i, y_i) be $\mathcal{A}_i$'s key pair where $y_i = g^{x_i}$. Then c_i is a Schnorr signature of the authority $\mathcal{A}_i$ on certification message CI_i. To generate c_i, $\mathcal{A}_i$ chooses $k_i \in_R Z_q^*$ and computes

$$c_i = (r_i, s_i) = (g^{k_i}, x_i \cdot h(CI_i, r_i) + k_i).$$

It's validity can be verified by $g^{s_i} \stackrel{?}{=} y_i^{h(CI_i, r_i)} r_i$. $\mathcal{A}_i$ has issued (r_i, s_i) as a certification information.

The MCS scheme is given by the following three algorithms.

General Implementation of MCS:

1. **Key generation:** A signer $\mathcal{S}$ computes a temporary signing key pair (x, y) by using the long-term key pair (x_0, y_0) and n certification information $(c_1, \ldots, c_n)$ as

$$x = x_0 + s_1 + s_2 + \cdots + s_n,$$

$$y = y_0 y_1^{h(CI_1, r_1)} r_1 \cdots y_n^{h(CI_n, r_n)} r_n.$$

2. **Signing:** Using the temporary signing key x the signer $\mathcal{S}$ computes a multi-certification signature $\sigma = (r, s)$ on message m and certification information $(CI_1, r_1, \ldots, CI_n, r_n)$ as follows.
   - Prepare a concatenated message $m||CI_1||r_1|| \cdots ||CI_n||r_n$.
   - Chooses a random number $k \in_R Z_q^*$ and computes a signature as

   $$r = g^k, \; s = x \cdot h(m||CI_1||r_1|| \cdots ||CI_n||r_n, r) + k.$$

   - Gives $\{(r, s), CI_1||r_1|| \cdots ||CI_n||r_n\}$ as a MCS on message m.
3. **Verification:** A verifier $\mathcal{V}$ checks the validity of $\{(r, s), CI_1||r_1|| \cdots ||CI_n||r_n\}$ as follows.
   - Computes a temporary public key $y = y_0 y_1^{h(CI_1, r_1)} r_1 \cdots y_n^{h(CI_n, r_n)} r_n$.
   - Verifies the signature (r, s) using y by $g^s \stackrel{?}{=} y^{h(m||CI_1||r_1|| \cdots ||CI_n||r_n, r)} r$.
   - Checks the certification information stated in $(CI_1, \ldots, CI_n)$ (This is just a document test, not a signature verification).

If the verification holds, it means that the signature of the signer is valid and n certification information are also confirmed. We can consider the distinguished implementation of MCS in the same way.

3.3 Efficiency

To compare the efficiency of the proposed MCS scheme, we consider a general approach that the signer just generates a signature on message m with his signing key x_0, and then the verifier has to verify $n + 1$ signatures (a signature of the signer and n certification information) independently. We show the comparison result in Table 1.

In the point of computation the general approach requires 1 signature generation (1 offline exponentiation) and $n+1$ signature verifications ($2(n+1)$ online exponentiations), while the general implementation of MCS scheme requires 1 signature generation (1 offline exponentiation) and 1 signature verification together with n exponentiations ($n + 2$ online exponentiations). In distinguished implementation of MCS scheme $n + 3$ online exponentiations are required in verification. On average the proposed MCS schemes are about 50% more efficient than the general approach. Note that computational efficiency is provided mainly in verification stage, not in signing stage.

If we consider a special case that n certification information are somewhat fixed and the verifier can verify them all in advance, then the verifier in MCS scheme also can compute the temporary public key y in advance and can use it repeatedly. Then the amount of computation in MCS scheme and in general approach are the same.

In signature size general approach uses $n+1$ independent signatures ($(n+1)(|p|+|q|)$) while the proposed MCS schemes require a single signature and $(r_1, \ldots, r_n)$ ($(n+1)|p|+|q|$). (Note that if the signer sends certificates themselves as certification information to the verifier, communication size will not be changed.) Therefore MCS scheme is more efficient than the general approach in computation and communication.

Table 1. Comparison of the efficiency of MCS schemes in computation and communication.

Process	General approach	General implementation of MCS	Distinguished implementation of MCS
No. of Exp. in signing	1 (offline)	1 (offline)	1 (offline)
No. of Exp. in verification	$2(n+1)$ (online)	$n+2$ (online)	$n+3$ (online)
Signature size	$(n+1)(\|p\|+\|q\|)$	$(n+1)\|p\|+\|q\|$	$(n+1)\|p\|+\|q\|$

We can consider another efficiency point. In MCS scheme a signer collects all the relevant certification information and provides a verifier with a highly combined digital signature with which both the digital signature on message and all certification information are verified together. If the signature cannot pass the verification process because of wrong certification information, it will not be considered as a valid signature. Therefore, a signer has to provide all the correct certification information and a verifier does not need to locate and keep them by himself. MCS can be considered as signer's additional service to provide more concrete non-repudiation evidence and more efficient verification of signature.

4 Conclusion

In this paper we have considered the real situation of using digital signatures in PKI and PMI environments and derived the necessity of new digital signature schemes called self-certified signature and multi-certification signature. First, we have shown the necessity of signing a message with a long-term signing key and certification information related with the public key together, and proposed an efficient self-certified signature scheme. Then we have considered the PKI and PMI environment and extended SCS to multi-certification signature scheme in which multiple certification information have to be verified together. The proposed schemes turned out to be very efficient in real usage.

In this paper we have implemented SCS and MCS schemes in DLP-based cryptosystems. However, RSA signature schemes are also widely used in practice.

Therefore designing RSA-based SCS scheme is an attractive further work. It is also required to provide more concrete security notions and proofs on SCS schemes.

Acknowledgements

We would like to thank anonymous reviewers for their valuable comments, which help to make this paper more readable one.

References

[CRL] RFC 2459, Internet X.509 Public Key Infrastructure Certificate and CRL Profile, IETF, 1999, http://www.ietf.org/html.charters/pkix-charter.html

[Gir91] M. Girault, "Self-certified public keys", *Advances in Cryptology: Eurocrypt'91*, LNCS 547, Springer-Verlag, 1991, pages 490 - 497.

[KPW97] S. Kim, S. Park, and D. Won, "Proxy signatures, revisited", In *Proc. of ICICS'97, International Conference on Information and Communications Security*, Springer, Lecture Notes in Computer Science, LNCS 1334, pages 223-232, 1997.

[MUO96] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: Delegation of the power to sign messages", In *IEICE Trans. Fundamentals*, Vol. E79-A, No. 9, Sep., pages 1338–1353, 1996.

[LK00] B. Lee and K. Kim, "Self-Certificate: PKI using Self-Certified Key", *Proc. of Conference on Information Security and Cryptology 2000*, Vol. 10, No. 1, pages 65–73, 2000.

[LKK01a] B. Lee, H. Kim and K. Kim, "Strong Proxy Signature and its Applications", *Proc. of SCIS2001*, pages 603–608, 2001.

[LKK01b] B. Lee, H. Kim and K. Kim, "Secure Mobile Agent using Strong Non-designated Proxy Signature", *Proc. of ACISP2001*, LNCS Vol.2119, Springer-Verlag, pages 474–486, 2001.

[PC] S. Tuecke, *et. al.*, "Internet X.509 Public Key Infrastructure Proxy Certificate Profile", IETF, 2002.

[PH97] H. Petersen and P. Horster, "Self-certified keys – Concepts and Applications", In *Proc. Communications and Multimedia Security'97*, pages 102 - 116, Chapman & Hall, 1997.

[PKI] Public-Key Infrastructure (X.509) (pkix), http://www.ietf.org/html.charters/pkix-charter.html

[PMI] Request for Comments, An Internet Attribute Certificate Profile for Authorization (RFC 3281), IETF, 2002.

[PS96] D. Pointcheval and J. Stern, "Security Proofs for Signatures", *Advances in Cryptology: Eurocrypt'96*, pages 387 - 398, Springer, 1996.

[PS00] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures", *Journal of Cryptology*, Vol. 13, No. 3, pages 361 - 396, Springer-Verlag, 2000.

[Sha84] A. Shamir, "Identity-based cryptosystems and signature schemes", *Advances in Cryptology: Crypto'84*, LNCS 196, Springer-Verlag, pages 47 - 53, 1985.