Self-Certificate: PKI using Self-Certified Key

Byoungcheon Lee and Kwangjo Kim

Information and Communications University,
58-4, Hwaam-dong, Yusong-gu, Taejon, 305-348, Korea
email:{sultan,kkj}@icu.ac.kr

Abstract

Self-certified key [Gir91, PH97] is a public key which contains the certification information of a CA in the public key itself. It provides implicit authentication for the public key while the certificate-based scheme like X.509 provides explicit authentication. Self-certified key has several advantages over certificate-based scheme, for example, it provides simple non-interactive key renewal mechanism. But one drawback is that it cannot provide explicit authentication for the public key. Any dishonest party can produce a good-looking self-certified key with other's identity, which cannot be distinguished from the real one before any successful communication with the owner occurs. So in a distributed environment like the Internet, it is hard to use self-certified key for real application.

In this study, we address how to use the advantages of self-certified key for PKI(public key infrastructure). To provide explicit authentication for self-certified key, we propose to use a concept of *self-certificate*, which is a user-generated certificate for the self-certified public key by signing the public key and relevant information with the private key corresponding to the public key. In this scenario, user can renew his key pairs by himself without any interaction with CA, while keeping the authenticity of CA's certification. CA can also use the same revocation mechanism as that of certificate-based scheme to revoke an issued key.

Key words: self-certified key, self-certificate, PKI, implicit authentication, key renewal, revocation.

1 Introduction

The authenticity of public key in an asymmetric cryptosystem can be given in two different ways: either explicitly or implicitly. In the first case, the authenticity of the public key can be verified explicitly using the certificate issued by a certificate authority(CA), e.g. X.509 [X509] certificate. In the second case, it can be verified implicitly at the time when the key is used for encryption, signature verification, key exchange or any other cryptographic usage. The concept of self-certified key was introduced by [Gir91] and extended in many ways by [PH97]. Here user's identity ID, public key y, and the corresponding private key x satisfy a computationally unforgeable relationship, which is verified implicitly by proper use of x in any cryptographic protocol. [PH97] shows various advantages of self-certified key: efficiency in hierarchical computation of public key and user-controlled key progression.

But one drawback of self-certified key is that it cannot provide explicit authentication for the public key. Any dishonest party can produce a good-looking self-certified key with other's identity which cannot be distinguished from the real one before any successful communication with the owner occurs. If a signature verification with the public key fails, the verifier cannot tell whether the public key is not authentic or the digital signature is wrong. So the owner of the key pair can repudiate his transaction. A transcript of any previous successful communication can be a proof of the authenticity of the public key, but it sometimes cause privacy problem. So in a distributed environment like the Internet, it is hard to use self-certified key for real application. The best way to solve this problem is to provide explicit authentication for the self-certified key.

Our solution is to introduce a concept of *self-certificate* for the self-certified key. It is a user-generated certificate for the authenticity of the self-certified key. Firstly, a user Alice gets a self-certified key pair from CA through an interactive key issuing protocol. Secondly, Alice generates a self-signed certificate by signing the public key and relevant public information using the corresponding private key. Signature using the private key can be considered as a proxy signature delegated by CA, so self-certificate is issued by Alice on behalf of CA. Any other party receiving the self-certificate can verify the authenticity of the public key explicitly by checking CA's blind signature and Alice's signature. In this approach, Alice can renew her key pairs by herself without any interaction with CA, while keeping CA's certification relation. Moreover general key revocation mechanism using certificate revocation list(CRL) can be applied to revoke an issued key pair.

This paper is organized as follows. In section 2, we briefly review the concept of self-certified key [PH97] and its properties. In section 3, we define self-certificate and present a protocol to generate self-certified key and self-certificate together. Moreover we present a user-controlled key renewal mechanism and revocation mechanism. In section 4, we compare its various features with the original schemes. Finally, we conclude in section 5.

2 Self-Certified Key

2.1 Schnorr signature

A certification authority(CA) chooses large primes p, q with $q|p-1$, a generator g of a multiplicative subgroup of Z_p^* with order q. $h(\ )$ denotes a collision resistant hash function. He publishes p, q, g, and h. Assume that a signer Alice has a private key x_A and the corresponding public key $y_A = g^{x_A}$. To sign a message m, Alice chooses a random number $k \in_R Z_q^*$ and computes $r = g^k$, $s = x_A h(m, r) + k$. Then the triple (m, r, s) becomes the signed message. The verification of signature is checked by $g^s = y_A^{h(m,r)} r$.

This signature scheme has been proven to be secure under the random oracle model [PS96]. They have shown that existential forgery under an adaptive chosen message attack is equivalent to the discrete logarithm problem.

2.2 Self-certified keys

We review the secure key issuing protocol of [PH97]. CA has a private key x_{CA} and the corresponding public key $y_{CA} = g^{x_{CA}}$. CA signs Alice's identity ID_A using a weak blind Schnorr signature [HMP95]. Using the following interactive protocol, CA issues a self-certified key pair (x_A, y_A) to Alice.

Secure key issuing protocol

1. CA chooses $\tilde{k}_A \in_R Z_q^*$, computes $\tilde{r}_A = g^{\tilde{k}_A}$, and transmits $\tilde{r}_A$ to Alice.
2. Alice chooses $a \in_R Z_q^*$, computes $r_A = \tilde{r}_A g^a$, and sends (ID_A, r_A) to CA.
3. CA computes the signature parameter $\tilde{s}_A = x_{CA} h(ID_A, r_A) + \tilde{k}_A$ and sends $\tilde{s}_A$ to Alice.
4. Alice obtains her private key $x_A = \tilde{s}_A + a$. The tuple (r_A, x_A) is CA's signature on Alice's identity ID_A. She verifies the validity of CA's

signature by

$$y_A \equiv g^{x_A} = y_{CA}^{h(ID_A, r_A)} r_A. \tag{1}$$

Then her public key is y_A.

Alice publishes the public parameter r_A and her identity ID_A and keeps her private key x_A secretly. Anyone can compute Alice's public key y_A using the published values r_A, ID_A and CA's public key y_{CA}.

The private key x_A is hidden to CA because it is blinded by the random value a. The public key y_A is self-certified because it contains CA's signature on ID_A in the key itself.

The problem is that y_A is not authentic because the public parameter r_A could have been modified by an attacker. Any illegal attacker can generate a good-looking public key y'_A using a modified public parameter r'_A and Alice's identity ID_A by $y'_A = y_{CA}^{h(ID_A, r'_A)} r'_A$, which cannot be distinguished from Alice's certified public key y_A at first glance. Its authenticity has to be verified implicitly by the proper use of the corresponding secret key x_A. If CA runs a trusted public directory to distribute the public parameter r_A, an extra online communication is required, which can be a drawback compared with the offline verifiability of the certificate-based schemes.

2.3 Proxy signature

The self-certified key can be used for proxy signature scheme [PH97, Kim98]. In the above secure key issuing protocol, let CA be an original signer and Alice be a proxy signer. Then the key pair (x_A, y_A) is considered as a proxy key pair. If the proxy signer signs a message m with x_A, then any verifier checks the validity of the signature using y_A and checks the authenticity of the public key by equation (1). If both checks are verified, then the signature is considered as a valid proxy signature, i.e., the signature was generated by the proxy signer on behalf of the original signer.

3 Self-Certificates

3.1 Definition

As mentioned in the previous section, the self-certified key y_A is not authentic by itself. A simple solution to provide explicit authenticity to y_A is that Alice signs her public parameter r_A and relevant information with her private key x_A. Her signature can be used as a certificate for her public key y_A. Because self-certified key can be used for proxy signature scheme, her signature is considered as a proxy signature of CA on the public parameter r_A. In this paper we define self-certificate as follows.

Definition 1 (Self-Certificate) *Let the key pair* (x_A, y_A) *be a self-certified key pair issued by CA,* r_A *be Alice's public parameter, and* ID_A *be Alice's identity. Alice signs on* ID_A, y_A, r_A *with her private key* x_A *to generate*

$$SelfCert_A = Sig_A(ID_A, y_A, r_A).$$

Then $SelfCert_A$ *is called self-certificate for the public key* y_A.

Compared with this definition, [LK99] has introduced another concept of self-certificate in PKI environment. In [LK99], user generates a key pair (x_u, y_u) and also generates a self-certificate by signing the public key y_u and relevant information with the private key x_u. After that, user presents his self-certificate to CA and gets CA's certification. User can get multiple certifications from multiple CAs for the same key pair. So its advantage is user's convenience by sharing the same key material for multiple purposes.

The definition of self-certificate in this paper is different from [LK99] in the sense that the key pair is firstly certified from a single CA and then the user generates self-certificate by himself. The same key pair can not be shared for multiple purposes.

3.2 Generation of self-certificate

To generate self-certificate for a self-certified key, we modify the secure key issuing protocol as follows.

Generation of self-certificate:

1. CA chooses $\tilde{k}_A \in_R Z_q^*$, computes $\tilde{r}_A = g^{\tilde{k}_A}$, and transmits $\tilde{r}_A$ to Alice.
2. Alice chooses $a \in_R Z_q^*$, computes $r_A = \tilde{r}_A g^a$, sends (ID_A, r_A) to CA.
3. CA prepares certification information CI_A depending on her policy, for example, she uses Alice's identity, CA's identity, Alice's public parameter r_A, certificate serial number, validity period, CA's public key and any other relevant extension information.

$$CI_A = [ID_A||ID_{CA}||r_A||CertNo||Period||y_{CA}||Ext].$$

She computes the signature parameter $\tilde{s}_A = x_{CA}h(CI_A) + \tilde{k}_A$. She sends CI_A and $\tilde{s}_A$ to Alice.

4. Alice obtains her private key $x_A = \tilde{s}_A + a$. The tuple (r_A, x_A) is CA's signature on certification information CI_A. She verifies the validity of CA's signature by
$$y_A \equiv g^{x_A} = y_{CA}^{h(CI_A)} r_A. \tag{2}$$
Then her public key is y_A.

5. Alice signs (CI_A, y_A) with her private key x_A to generate the self-certificate of y_A.
$$SelfCert_A = Sig_A(CI_A, y_A). \tag{3}$$

Verification of self-certificate:

When the $SelfCert_A$ is presented, anyone can explicitly verify the validity of y_A by

1. Checks the validity of Alice's signature in $SelfCert_A$ using y_A.

2. Checks the validity of CA's certification by checking equation (2).

3.3 User-controlled key renewal

A user might want to change his keys from time to time. If he has any suspicion that his key is compromised, he will need to change his key pair. In other words, if a user changes his key pair frequently, the risk that he uses compromised key will be reduced a lot. In certificate-based schemes like X.509, changing key pair requires complicated interaction between user and CA. But in our scheme, user can renew his key pair by himself without any interaction with CA.

Let (x_A, y_A) be the basic self-certified key pair issued by CA and let $SelfCert_A$ be the self-certificate issued by Alice. She can generate a renewed key pair (x'_A, y'_A) and a renewed certificate $SelfCert'_A$ by the following procedure.

Key renewal procedure:

1. Chooses $k'_A \in_R Z_q^*$ and computes $r'_A = g^{k'_A}$.

2. Computes new key pair as
$$x'_A = x_A h(CI_A, r'_A) + k'_A, \quad y'_A = y_A^{h(CI_A, r'_A)} r'_A. \tag{4}$$

3. Signs (CI_A, y_A, r'_A, y'_A) with her new private key x'_A to generate

$$SelfCert'_A = Sig'_A(CI_A, y_A, r'_A, y'_A). \tag{5}$$

Through the above key renewal procedure, Alice signs CI_A with the basic private key x_A to generate a signature parameter x'_A, which is used as a renewed private key and $y'_A = g^{x'_A}$ becomes the corresponding public key.

Verification of the renewed self-certificate:

When the $SelfCert'_A$ is presented, anyone can explicitly verify the validity of y'_A by

1. Checks the validity of Alice's signature in $SelfCert'_A$ using y'_A.
2. Checks the validity of CA's certification by

$$y'_A = y_{CA}^{h(CI_A)h(CI_A,r'_A)} r_A^{h(CI_A,r'_A)} r'_A. \tag{6}$$

The renewed key pair (x'_A, y'_A) uses the same CI_A and serial number. Note that if Alice uses only the renewed key pair for real communication, the original certificate $SelfCert_A$ needs not be generated or verified at all.

Security considerations:

- Anyone who does not have the original secret key x_A cannot forge a valid key pair (x'_A, y'_A) which satisfies equation (5) and (6) together.
- If the basic key pair (x_A, y_A) is not used for any real communication and used only for key renewal, it is not vulnerable to any attack such as adaptive chosen message attack.
- Although an attacker manages to get the renewed private key x'_A, he cannot benefit from the knowledge of it to compute the basic private key x_A or any future renewed private key x''_A of the user because x'_A in equation (4) has two unknowns x_A and k'_A.

3.4 Revocation mechanism

Although [PH97] have not described anything about revocation mechanism which is one of the basic issues in PKI, it can be easily extended to provide one. In the proposed scheme, CA can revoke a certificate although we use a user-generated certificate rather than CA-generated certificate for PKI.

When a certified key pair is found to be compromised, CA can revoke it using the general certificate revocation list(CRL) mechanism. CA publishes CRL which contains the certificate serial number $CertNo$ of the revoked key. Anyone who wants to use a self-certificate should check whether it was revoked or not using the recent version of CRL.

4 Analysis

In this section, we compares the proposed PKI scheme with the original schemes of certificate-based scheme and self-certified key. Table 1 summarizes the result briefly. The proposed PKI scheme provides explicit authentication, non-repudiation, user-controlled key renewal, key revocation and proof of possession of private key.

To verify the authenticity of public key, certificate-based scheme requires one verification of CA's signature. In the self-certified key scheme, it requires one exponentiation and one online communication if a trusted public directory is used for the distribution of public parameter r_A. In the proposed self-certificate scheme, it requires one verification of user's signature and one extra exponentiation for the verification of CA's certification(If renewed key is used, one verification and two extra exponentiations are required as in (6)). Compared with certificate-based scheme, our proposed PKI scheme provides us with various advantages with only one extra exponentiation.

Table 1: Comparison of PKI schemes

Features	Certificate-based scheme	Self-certified key	Self-certificate scheme
Authentication	Explicit	Implicit	Explicit
Non-repudiation	O	X	O
User-controlled key renewal(security)	X	O	O
Key revocation	O	X	O
Proof of possession of private key	X	X	O
Computation/ communication	1 V	1 E + 1 C	1 V + 1 E (1 V + 2 E)

* V : signature verification, E : exponentiation, C : online communication.

5 Conclusion

We introduced a concept of self-certificate which is a user-generated certificate for the self-certified public key. When the self-certificate is presented, anyone can explicitly verify the validity of public key by checking CA's certification and the owner's signature. In this scenario, user can renew his key pairs by himself without any interaction with CA, while keeping the authenticity of CA's certification. To revoke an issued key, CA can also use the same revocation mechanism as that of certificate-based scheme.

The proposed self-certificate scheme is different from the traditional certificate-based scheme like X.509 in the key issuing protocol, but it can be used for PKI in the same way and enjoys many advantages. Our work is a preliminary proposal of a new PKI scheme. We believe that more extensive studies should be followed for real application.

References

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

[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.

[HMP95] P. Horster, M. Michels, H. Petersen "Hidden signature schemes based on the discrete logarithm problem and related concepts", In *Proc. Communications and Multimedia Security'95*, pages 162 - 177, Chapman & Hall, 1995.

[Kim98] S. Kim, "Improved privacy and authenticity in digital signature/key management", Ph.D. Thesis, SungKyunKwan University, 1998.

[LK99] B. Lee and K. Kim, "A proposal for user-oriented public key infrastructure", In *CISC'99*, Vol.9, No.1, pp.47-59, 1999.

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

[X509] ITU-T Recommendation X.509, "The Directory: Authentication Framework", 1993.