

*Secure Authenticated Key Exchange protocol based on EC using Signcryption Scheme

Rack-Hyun Kim¹, Heung-Youl Youm²

^{1,2}Department of Information Security Engineering, Soonchunhyang University,
San 53-1, Eupnae-ri, Shinchang-myun, Asan-si
Chungchungnam-do, KOREA

¹rhkim@sch.ac.kr

²hyyoum@sch.ac.kr

Abstract

A Signcryption proposed by Yuliang Zheng in 1997 is a hybrid public key primitive that combines a digital signature and an encryption. It provides more efficient method than a straightforward composition of a signature scheme with an encryption scheme. In a mobile communication environment, the authenticated key agreement protocol should be designed to have lower computational complexity and memory requirements. Therefore, in this paper, our proposed method that efficient authenticated key exchange protocol using Signcryption scheme and based on EC using Signcryption in mobile and wireless network. And we analyze computational cost and security requirements of proposed protocol.

1. Introduction

Two of the most important services offered by cryptography are those of providing private and authenticated communications. Much research has been done into creating encryption schemes to meet highly developed notions of privacy.

In 1997 Zheng proposed a primitive that he called Signcryption[1]. The idea of a Signcryption schemes is to combine the functionality of an encryption scheme with that of a signature scheme. It must provide privacy; Signcryption must be unforgeable; and there must be a method to settle repudiation disputes.

In [2], Zheng proposed two key exchange protocol using Signcryption scheme that he called DKEUN(Direct Key Exchange Using a Nonce) and DKEUTS(Direct Key Exchange Using Time-stamp).

But Zheng's key exchange protocols are heavy in mobile network and wireless network. Therefore, our proposed two key exchange protocols using Signcryption and based on EC using Signcryption scheme.

PAK protocol used to establish a short-term session key between a client and a server is known as a key agreement protocol. The protocol refers to be authenticated if one party is authenticated to the other during the authentication protocol run. The authenticated key exchange protocol is to be authenticated, if simple password is used to authenticate each other.

We describe working environments for PAK protocol. Two entities, who only share the secret information, and who are communicating over an insecure network, want to authenticate each other and agree on a large session key to be used for protecting their subsequent communication. In this case, password is shared among the servers and a client can be authenticated by a group of servers using the shared secret password.

This paper is organized as follows. In section 2, we describe the basic concepts of PAK and ITU-T PAK protocol based on our solutions. In section 3, we review the basic concepts of signcryption and key agreement protocols based on signcryption. We present the SAKE and EC-SAKE protocols in section 4. We show properties and efficiency comparison of proposed secure authenticated-key exchange protocols in section 5. And finally, we conclude in section 6.

* This work was supported in part by the NSRI(National Security Research Institute), Korea.

2. Password-based Authenticated Key Exchange Scheme

2.1. PAK Protocol

In this subsection we describe two basic PAK Protocols in [9] and ITU-PAK in [10]. These schemes do not require any external authentication infrastructure but use only password to authenticate each other and derive a common session key.

Figure 2 describes that a basic building block for our basic PAK protocols and figure 3 describes ITU-PAK. Suppose that user A and User B are sharing password for individual certification beforehand. Following parameters are used for the PAK and ITU-PAK protocol.

PAK Protocol allows two parties to authenticate each other, while maintaining a perfect forward secrecy by performing the Diffie-Hellman key exchange procedure. The authentication relies on a pre-shared secret, which is concealed (i.e., remains unrevealed) from an eavesdropper preventing an off-line dictionary attack.

Follow table 1 describes system parameters for PAK and ITU-PAK protocol.

Table 1. System parameters for PAK protocol

System parameter for PAK protocol
A : user A ID, B : user B ID
π : password of user A, B
$p = rq + 1$, $\gcd(r, q) = 1$
p is a 1024 bit prime
q is a 160 bit prime
g is a generator of the subgroup Z_p^* of order q
h is a generator of the subgroup Z_r , $h^q = (g^r)^q$ and $Z^{qr} = g^{(p-1)} = 1$
H_1 : a collision-free hash function with 1024+160=184 bits output [1].
Generate $\alpha[A, B, \pi] \in_R Z_q$ and store it.
Then generate $h \in_R Z_p^*$ and $\beta \in_R Z_{\lfloor 2^\eta / p \rfloor}$,
and return $(h^q g^{\alpha[A, B, \pi]} \bmod p) + \beta p$.
Note that will be indistinguishable from a random bit string of length η , since $(h^q g^{\alpha[A, B, \pi]} \bmod p)$ is random element from Z_p^{*8} and $(2^\eta \bmod p) / 2^\eta$ is negligible.

Note that $(H_1(A, B, \pi))^r = (h^q g^{\alpha[A, B, \pi]})^r = h^{qr} g^{r \cdot \alpha[A, B, \pi]} = g^{r \cdot \alpha[A, B, \pi]}$.

$H, H_1, H_{2a}, H_{2b}, H_3$: a collision-free hash function with 160 bits output

$a|b$: concatenate a and b

Key is the resulting session key

Alice(A) : Client	Bob(B) : Server
$x \in_R Z_q$ $m = g^x \cdot H_1(A, B, \pi)^r \bmod p$	\xrightarrow{m}
	$m \stackrel{?}{=} 0 \bmod p$ $y \in_R Z_q$, $\mu = g^y \bmod p$
$\xleftarrow{\mu, k}$	$\sigma = \left(\frac{m}{H_1(A, B, \pi)^r}\right)^y = g^{xy} \bmod p$ $k = H_{2a}(A, B, m, \mu, \sigma, \pi)$
$\sigma = \mu^x = g^{xy} \bmod p$ $k \stackrel{?}{=} H_{2a}(A, B, m, \mu, \sigma, \pi)$ $k' = H_{2b}(A, B, m, \mu, \sigma, \pi)$	$\xrightarrow{k'}$
	$k' \stackrel{?}{=} H_{2b}(A, B, m, \mu, \sigma, \pi)$
	$Key = H_3(A, B, m, \mu, \sigma, \pi)$

Fig 1. PAK Protocol

ITU-PAK is specified in ITU-T Recommendation X.805. And it is based on Diffie-Hellman key exchange. Diffie-Hellman key exchange, although it provides the Perfect forward secrecy, is vulnerable to the man-in-the-middle attack, as is well known.

Alice(A) : Client	Bob(B) : Server
Select $R_A \in_R Z_q$ $m = H(P) \cdot (g^{R_A} \bmod p)$	\xrightarrow{m}
	$m \stackrel{?}{=} 0 \bmod p$ $\frac{H(P) \cdot (g^{R_A} \bmod p)}{H(P)} = g^{R_A}$
$\xleftarrow{m', S_1}$	Select $R_B \in_R Z_q$ $S_1 = H("1" P g^{R_A} \bmod p g^{R_B} \bmod p g^{R_A R_B} \bmod p)$ $m' = H(P) \cdot (g^{R_B} \bmod p)$
$S_1' = H("1" P g^{R_A} \bmod p g^{R_B} \bmod p g^{R_A R_B} \bmod p)$ $S_1' \stackrel{?}{=} S_1$	$\xrightarrow{S_2}$
	$S_2 = H("2" P g^{R_A} \bmod p g^{R_B} \bmod p g^{R_A R_B} \bmod p)$

$S_2' = H("2" P g^{R_A} \bmod p g^{R_B} \bmod p g^{R_A R_B} \bmod p)$ $S_2 \stackrel{?}{=} S_2'$
$K = H("3" P g^{R_A} \bmod p g^{R_B} \bmod p g^{R_A R_B} \bmod p)$

Fig 2. ITU-PAK Protocol discussed in ITU-T

Initially, A selects a (secret) exponent R_A and computes $g^{R_A} \bmod p$; B selects (a secret) exponent R_B and computes $g^{R_B} \bmod p$. For efficiency purposes, the values of the exponents should be smaller than their range allows. Neither computed quantity may be equal to zero. Then

- 1) A initiates the exchange by sending $H(P) \cdot (g^{R_A} \bmod p)$ to B ;
- 2) B , upon receiving that quantity, verifies that it is not a zero and then divides it by $H(P)$ (thus recovering what has to be $g^{R_A} \bmod p$). Then B computes $S_1 = H("1" | P | g^{R_A} \bmod p | g^{R_B} \bmod p | g^{R_A R_B} \bmod p)$, and sends to A a message that contains both quantities, $H(P) \cdot (g^{R_B} \bmod p)$ and S_1 .
- 3) Upon receiving that message, A can authenticate B by recovering what should be $g^{R_B} \bmod p$, ensuring that it is not zero, and computing S_1 itself. If the result is equal to the received value, A computes the key $K = H("3" | P | g^{R_A} \bmod p | g^{R_B} \bmod p | g^{R_A R_B} \bmod p)$. To authenticate itself and complete the exchange, A also computes the quantity $S_2 = H("2" | P | g^{R_B} \bmod p | g^{R_A} \bmod p | g^{R_A R_B} \bmod p)$ and sends it to B .
- 4) B authenticates A by computing S_2 itself and checking it against the value received from A . If both are the same, it also computes the key $K = H("3" | P | g^{R_A} \bmod p | g^{R_B} \bmod p | g^{R_A R_B} \bmod p)$.

If any of the above verifications fails, the protocol halts; otherwise, both parties have authenticated each other and established the common session key.

3. Signcryption

3.1. Signcryption

Signcryption is a novel public key primitive first proposed by Zheng in 1997 [1]. A signcryption scheme combines the functionality of a digital signature scheme with that of an encryption scheme. It therefore offers the three services: privacy, authenticity and non-repudiation. Since these services are frequently required simultaneously, Zheng proposed signcryption as a means to offer them in a more efficient manner than a straightforward composition of digital signature scheme and encryption scheme. An ingenious scheme was proposed to meet such a goal.

The scheme may be used to signcrypt messages from $\{0,1\}^n$, where $k = n + k_0 + k_1$ for integers k_0 and k_1 . Before f is applied to a message some random padding is applied. We describe how the scheme works below.

Security Parameters

The scheme requires two hash functions

$$H : \{0,1\}^{n+k_0} \longrightarrow \{0,1\}^{k_1} \text{ and } G : \{0,1\}^{k_1} \longrightarrow \{0,1\}^{n+k_1}$$

Signcryption	Unsigncryption
For Alice to signcrypt a message $m \in \{0,1\}^n$ for Bob:- 1. $r \leftarrow \{0,1\}^{k_0}$ 2. $\omega \leftarrow H(m r)$ 3. $s \leftarrow G(\omega) \oplus (m r)$ 4. $c \leftarrow f(s \omega)$ 5. Send c to Bob	For Bob to unsigncrypt a cryptogram c from Alice:- 1. $s \omega \leftarrow f^{-1}(c)$ 2. $m r \leftarrow G(\omega) \oplus s$ 3. If $H(m r) = \omega$ accept m Else reject

Fig 3. Signcryption scheme

3.2. Key exchange protocol using signcryption

In [2], Zheng proposed two key exchange protocol using Signcryption scheme that he called DKEUN(Direct Key Exchange Using a Nonce) and DKEUTS(Direct Key Exchange Using Time-stamp). But Zheng's key exchange protocols are heavy in mobile network and wireless network. Therefore, our proposed two key exchange protocol using Signcryption and based on EC using Signcryption scheme.

Alice(A) : Client	Bob(B) : Server	
$\leftarrow NC_b$	$NC_b \in_R \{0, 1\}^{\ln}$	
$key \in_R \{0, 1\}^k$ $x \in_R [1, \dots, q-1]$ $(k_1, k_2) = hash(y_b^x \text{ mod } p)$ $c = E_{k_1}(key)$ $r = KH_{k_2}(key, NC_b, \text{etc})$ $s = (x/(x + x_a)) \text{ mod } q$	$\xrightarrow{c, r, s}$	
	$(k_1, k_2) = hash((y_a \cdot g^r)^{s \cdot x_a} \text{ mod } p)$ $key = D_{K_1}(c)$ Accept key only if $KH_{K_2}(key, NC_b, \text{etc}) = r$	
	$key^* \in_R \{0, 1\}^k$ $x^* \in_R [1, \dots, q-1]$ $(k_1^*, k_2^*) = hash(y_b^{x^*} \text{ mod } p)$ $c^* = E_{k_1^*}(key^*)$ $r^* = KH_{k_2^*}(key^*, key, \text{etc})$ $s^* = (x^*/(r^* + x_b)) \text{ mod } q$	$\leftarrow c^*, r^*, s^*$
$(k_1^*, k_2^*) = hash((y_a \cdot g^{r^*})^{s^* \cdot x_a} \text{ mod } p)$ $key^* = D_{K_1^*}(c^*)$ Accept key^* only if $KH_{K_2^*}(key^*, key, \text{etc}) = r^*$		
	Verify $tag = MAC_{key \oplus key^*}(NC_b)$	

Fig 4. DKEUN(Direct Key Transport Using a Nonce) protocol

Alice(A) : Client	Bob(B) : Server
$key \in_R \{0, 1\}^k$ $x \in_R [1, \dots, q-1]$ $(k_1, k_2) = hash(y_b^x \text{ mod } p)$ Get a current time-stamp TS $c = E_{k_1}(key, TS)$ $r = KH_{k_2}(key, TS, \text{etc})$ $s = (x/(x + x_a)) \text{ mod } q$	$\xrightarrow{c, r, s}$
	$(k_1, k_2) = hash((y_a \cdot g^r)^{s \cdot x_a} \text{ mod } p)$ $(key, TS) = D_{K_1}(c)$ Accept key only if TS is fresh and $KH_{K_2}(key, TS) = r$

$key^* \in_R \{0, 1\}^k$ $x^* \in_R [1, \dots, q-1]$ $(k_1^*, k_2^*) = hash(y_b^{x^*} \text{ mod } p)$ Get a current time-stamp TS $c^* = E_{k_1^*}(key^*, TS^*)$ $r^* = KH_{k_2^*}(key^*, TS^*, key, \text{etc})$ $s^* = (x^*/(r^* + x_b)) \text{ mod } q$	$\leftarrow c^*, r^*, s^*$
$(k_1^*, k_2^*) = hash((y_b \cdot g^{r^*})^{s^* \cdot x_b} \text{ mod } p)$ $(key^*, TS) = D_{K_1^*}(c^*)$ Accept key^* only if TS^* is fresh and $KH_{K_2^*}(key^*, TS, key, \text{etc}) = r^*$	
	Verify $tag = MAC_{key \oplus key^*}(TS)$

Fig 5. DKEUTS(Direct Key Transport Using a Time-stamp) protocol

Alice(A) : Client	Bob(B) : Server
1. Random x $x \in [1, \dots, q-1]$ $(K_1, K_2) = hash(x \cdot y_b)$ 2. for m , generate (c, r, s) $c = E_{K_1}(m)$ $r = KH_{K_2}(m)$ $s = (x/(r + x_a)) \text{ mod } q$	$\xrightarrow{c, r, s}$
	$u \leftarrow r, s, g, y, y_a, x_b$ $u = s \cdot x_b \text{ mod } q$ computes K_1, K_2 $(K_1, K_2) = hash(u \cdot y_a + u \cdot r \cdot G)$ $m = D_{K_1}(c)$ if $(KH_{K_2}(m) \stackrel{?}{=} r)$ Accept m

Fig 6. EC-Signcryption

4. SAKE and EC-SAKE protocol

Parameters using SAKE same defined parameters in Table 1. In remainder of this paper, following parameters are used in EC-SAKE protocol.

Table 2. Parameters for EC-SAKE protocol

Parameters public to all :

C : an elliptic curve over $GF(p^m)$, either with

$(p \geq 2^{150}, m = 1 \text{ or } p = 2 \text{ and } m \geq 150)$
 q : a large prime whose size is approximately of $|p^m|$
 G : a point with order q , chosen randomly from the points on C
 H_1, H_2 : one way hash function
 KH : one way keyed hash function
 (E, D) : the encryption and decryption algorithm of private key cipher

Alice's Keys :
 x_a : Alice's private key ($\in [1, \dots, q-1]$)
 y_a : Alice's public key ($y_a = x_a \cdot G$)

Bob's Keys :
 x_b : Bob's private key ($\in [1, \dots, q-1]$)
 y_b : Bob's public key ($y_b = x_b \cdot G$)

Alice(A) : Client	Bob(B) : Server
Random $x \in [1, \dots, q-1]$	
$(K_1, K_2) = H_1(y_b^x)$	
Generate (m, r, s)	$\xrightarrow{m, r, s}$
$m = H_2(A, B, K_1) \cdot g^x$	
$r = KH_{K_2}(m)$	
$s = (x/(r + x_a)) \bmod q$	
	$u = s \cdot x_b \bmod q$
	Generate (K_1, K_2)
	$(K_1, K_2) = H_1(y_a \cdot g^u) \bmod p$
	Random $y \in [1, \dots, q-1]$
	if $(KH_{K_2}(m) \stackrel{?}{=} r)$
$\xleftarrow{\mu, \delta}$	$\mu = y \cdot G, \frac{m}{H_2(A, B, K_1)} = x \cdot G$
	$\sigma = (x \cdot y \cdot G), \delta = KH_{K_1}(\sigma)$
	$\sigma = x \cdot \mu$
	$\delta \stackrel{?}{=} KH_{K_1}(\sigma)$
	$K = H_3(A, B, K_1, K_2, \sigma, \mu)$
$\sigma = \mu^x$	
$\delta \stackrel{?}{=} KH_{K_1}(\sigma)$	
$K = H_3(A, B, K_1, K_2, \sigma, \mu)$	

Fig 7. SAKE Protocol

In step 1, Alice selects random number $x \in [1, \dots, q-1]$, and computes K_1 and K_2 . Alice computes m using H_2 , r and S . And, Alice sends the computed value m , r and S to Bob.

In step 2, Bob selects a random number $y \in_R Z_q$ and computes μ , $\sigma = g^{xy}$. Bob sends the

computed values μ and δ using KH_{K_1} hash function to Alice.

In step 3, Alice computes δ' using KH_{K_2} and compare received δ from Bob. If satisfy, computes session key K using H_3 hash function. By the results, distribute same session key to Bob and Alice.

In step 4, Bob and Alice compute K each other. By the results, distribute same session key ($K = H_3(A, B, K_1, K_2, \sigma, \mu)$) to Bob and Alice.

Alice(A) : Client	Bob(B) : Server
Random $x \in [1, \dots, q-1]$	
$(K_1, K_2) = H_1(x \cdot y_b)$	
Generate (m, r, s)	$\xrightarrow{m, r, s}$
$m = H_2(A, B, K_1) \cdot x \cdot G$	
$r = KH_{K_2}(m)$	
$s = (x/(r + x_a)) \bmod q$	
	$u = s \cdot x_b \bmod q$
	Generate (K_1, K_2)
	$(K_1, K_2) = H_1(u \cdot y_a + u \cdot r \cdot G) \bmod p$
	Random $y \in [1, \dots, q-1]$
	if $(KH_{K_2}(m) \stackrel{?}{=} r)$
$\xleftarrow{\mu, \delta}$	$\mu = y \cdot G, \frac{m}{H_2(A, B, K_1)} = x \cdot G$
	$\sigma = (x \cdot y \cdot G), \delta = KH_{K_1}(\sigma)$
	$\sigma = x \cdot \mu$
	$\delta \stackrel{?}{=} KH_{K_1}(\sigma)$
	$K = H_3(A, B, K_1, K_2, \sigma, \mu)$

Fig 8. EC-SAKE Protocol

In step 1, Alice selects random number $x \in [1, \dots, q-1]$, and computes K_1 and K_2 . Alice computes m using H_2 , r and S . And, Alice sends the computed values m , r and S to Bob.

In step 2, Bob selects a random number $y \in_R Z_q$ and computes μ , $\sigma = x \cdot \mu$. Bob sends the computed values μ and δ using KH_{K_1} hash function to Alice.

In step 3, Alice computes δ' using KH_{K_2} and compare received δ from Bob. If satisfy, computes session key K using H_3 hash function. By the results, distribute same session key to Bob and Alice.

In step 4, Bob and Alice compute K each other. By the results, distribute same session key ($K = H_3(A, B, K_1, K_2, \sigma, \mu)$) to Bob and Alice.

5. Efficiency and Characteristics

5.1. Key exchange protocol using signcryption

We would like to attain the following security properties:

- An eavesdropper on the conversation between client and server can't learn client's secret information, client's configuration information, or be able to verify a guess of client's secret information from the eavesdropped information
- Server and client use two other hash functions, so that message and secret information are protected for replay attack.
- Proposed protocols are designed based underlying problems are (computational) Diffie-Hellman Problem and Discrete Logarithm Problem in a finite field.
- Previous session key can't know even if two entity's secret information is exhibited because compute in step using two entity's secret information and random numbers.

5.2. Characteristic and performance

We would like to attain the following characteristic analysis (Table 3) and performance of computations (Table 4).

Table 3. Characteristic analysis

Scheme	Mutual authentication	Communication	Key exchange	Alice's Key element	Bob's Key element
DKEUN DKEUTS	O	3-way (+1opt)	O	O	X
EC-DKEUN EC-DKEUTS	O	3-way (+1opt)	O	O	X
SAKE EC-SAKE	O	2-way	O	O	O

Table 4. Performance of computation

scheme		Computational cost							Time-stamp
		Exp	hash	Enc	Dec	Multi	Div	nonce	
DKEUN	A	3	5	1	1	2	3	2	0
	B	3	5	1	1	2	3	1	0
DKEUTS	A	3	5	1	0	2	3	1	1
	B	3	5	0	1	2	3	1	1
SAKE	A	3	5	0	0	1	1	1	0
	B	3	5	0	0	1	1	1	0
EC-DKEUN	A	0	5	1	1	5	1	2	0
	B	0	5	1	1	5	1	3	0
EC-DKEUTS	A	0	5	1	1	5	1	2	1
	B	0	5	1	1	5	1	2	1
EC-SAKE	A	0	5	0	0	4	1	1	0
	B	0	5	0	0	5	1	1	0

6. Conclusions

In this paper, we propose the Secure Authenticated Key Exchange protocol using Signcryption scheme and

based on EC using Signcryption. It was proven that our schemes are efficient than the other Signcryption key exchange protocols in terms of computation complexity and communication performance.

7. References

- [1] Y. Zhen g , "Digital signcryption or how to achieve cost (signature and encryption)< < cost (signature) + cost (encryption) ", Advances in Cryptology , Proceeding s of CRYPT O'97, LNCS Vol. 1294, Springer - Verlag , pp . 165-179, 1997.
- [2] Y. Zheng, "Shortened Digital Signature, Signcryption and Compact and Unforgeable Key Agreement Schemes" IEEE P1363a:Standard Specifications for Public-key Cryptography : Additional Techiques, 1998
- [3] Y. Zhen g an d H. Imai, How to Construct Efficient Signcryption Schemes on Elliptic Curves , Information Processing Letters , Vol.68, pp .227- 233, 1998.
- [4] Y. Zhen g , "Signcryption an d it s application in efficient public key solutions ", Proc. Of Information Security Work shop (ISW '97), LNCS Vol. 1396, Springer - Verlag , pp. 291- 312, 1998.
- [5] H. Y. Jun g , D. H. Lee, J . I. Lim an d K. S . Chan g , "Signcryption Schemes with Forward Secrecy ", Proceeding of Workshop on Information Security Applications (WISA '2001), Vol.2, pp .463- 475, 2001.
- [6] F. Bao and H. Deng , "A signcryption scheme with signature directly verifiable by public key ", Proceeding of Public Key Cryptography (PKC'98), LNCS Vol.1431, pp.55-59, 1998.
- [7] "Proposed Federal Information Proceeding Standard for Digit al Signature Standard (DSS)", Federal Register , Vol. 56, No.169 30, 1991.
- [8] A. J. Menezes P. C. van Oorschot and S. A. Van stone "Handbook of Applied Cryptography" 1997.
- [9] Victor Boyko, Philip MacKenzie and Sarvar Patel, "Provably Secure Password Authentication and key Exchange Using Diffie-Hellman(extended abstract)", *EuroCrypt 2000*, pp.156-171, 2000.
- [10] Alec Brusilovsky, Igor Faynberg, Sarvar Patel, Zachary Zeltsan, "Password-Authenticated Key Exchange(PAK) Protocol", STUDY GROUP 17, COM 17-D121-E, Jan, 16, 2006.
- [11] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures - How to sign with RSA and Rabin. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of Lecture Notes in Computer Science, pages 399-416. Springer-Verlag, 1996.
- [12] J.-S. Coron and M. Joye and D. Naccache and P. Paillier. Universal Padding Schemes for RSA. In *Advances in Cryptology - CRYPTO 2002*, volume 2442 of Lecture Notes in Computer Science, pages 226-241. Springer-Verlag, 2002.