ECC를 이용한 SSET 프로토콜 설계

조 인 석° 이 병 관
관동대학교 전자계산공학과
cho9088@hanmail.net, bklee@kwandong.ac.kr

SSET(Strengthened and Secure Electronic Transaction) Protocol Design by using ECC

In-Seock, Cho° Byung-Kwan, Lee
Dept. of Computer Science, Kwandong University

요 약

기존의 SET은 전자서명(digital signature), 데이터 암호화(data encryption), 전자 봉투(digital envelope)로 구성되어 있으며, RSA, SHA, DES를 사용하여 프로토콜을 구현하는데, 본 논문은 ECC의 공개키와 개인키를 이용하여 암호 강도가 강화된 대칭키 알고리즘을 제안하고 SET의 전자 봉투를 생략한 SSET를 제안하고 있다.

1. 서론

SET(Secure Electronic Transation)은 기존의 SSL(Secure Soket Layer)보다 인터넷에서의 보안성이 크게 증대되어 메시지 무결성(Integrity), 인증(Authentication), 부인 방지(Non-repudiation)등이 강화되고, 이로 인해 거래자간의 신뢰도를 높여 전자 상거래를 활성화시켰다[1]. SET은 인증과 부인 방지를 위한 전자 서명(digital signature)에는 RSA 공개키 암호화와 해쉬 알고리즘을, 메시지의 보안을 위해 대칭키 암호화와 전자 봉투를 사용하고 있으나, RSA의 파괴 가능성과 대칭키가 분실, 도청될 위험성을 가지고 있다. 이러한 위험을 피하기 위한 보안 강도의 강화가 요구되고 있을 뿐만 아니라, 미래는 보안을 위해 보다 키의 길이가 짧으면서, 짧은 시간에, 더욱 강력한 암호 강도를 요구하고 있다. 이러한 관점에 부응하여 활발히 연구, 구현되고 있는 것이 타원 곡선 암호화(ECC, Elliptic Curve Cryptosystem)이다[2]-[4].

본 논문에서는 ECC를 전자 서명의 공개키 알고리즘에 사용하여 보안 강도를 강화하고, 이 때 발생되는 개인키와 공개키를 변형시켜 본 논문에서 제안하고 있는 새로운 대칭키 암호화 알고리즘의 키로 사용하여 전자 봉투를 생략한 SSET를 제안한다. 이를 위해 2장에서는 필요한 개념의 설명과 3장에서는 제안된 SSET의 설명과 4장에서는 결론과 향후 과제에 대하여 논한다.

2. 기본 개념

2.1 EC(Elliptic Curve)

1985년 N.Kobilz[2]와 V.Miller[3]에 의해 제안된 EC

C는 p가 3보다 큰 홀수 소수이고, a 와 b는 F p의 원소이며, $4a^3+27b^2 \not\equiv 0 \pmod p$일 때, 유한체(Finite Fields) F p 상의 타원 곡선 E는 $y^2 = x^3 + ax + b$ 형태의 방정식으로 정의된다. 집합 E(F p)는 무한점과 이 방정식을 만족하면서 F p 상의 원소인 x, y로 이루어진 모든 점 P(x, y)으로 구성된다. 타원 곡선에서 덧셈인 점과 점의 연산은 기하학적 연산으로 Addition되며 아래와 같이 정의한다.[5][6]

① 항등원인 무한점 0이 정의된다.

P + 0 = 0 + P = P

② (x , y) + (x , $-y$) = 0

P = (x , y), -P = (x , $-y$)

③ Addition

P = (x_1, y_1), Q = (x_2, y_2), P ≠ ±Q일 때

P + Q = (x_3, y_3)가 된다.

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2,$$

$$y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1,$$

④ Doubling

P = (x_1, y_1), P ≠ -P일 때,

P + P = 2P = (x_3, y_3)가 된다.

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1,$$

$$y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1$$

2.2 ECC(Elliptic Curve Cryptosystem)

A와 B 두 사용자 간에 키의 생성과 암호화는 다음과 같이 설명할 수 있다.[6]

① [A] 소수 p 선택

② [A] $4a^3+27b^2 \not\equiv 0 \pmod p$가 되는 임의 a, b를 선택하여 $y^2 = x^3 + ax + b$를 고정

③ [A] 타원 곡선상 임의 초기점 P를 선택

④ [A] 임의 정수 k를 선택하여 A의 개인키로 정한다.

⑤ [A] kP를 Addition한 결과를 A의 공개키로 정한다.

⑥ [A] p, a, b, P, kP를 전송

⑦ [B] p, a, b, P, kP를 수신

⑧ [B] 임의 정수 r을 선택하여 B의 개인키로 정한다.

⑨ [B] rP를 Addition한 결과를 B의 공개키로 정한다.

개인키와 공개키가 완성되어지면 암호화를 한다. B는 암호화를 위해 Addition인 r(kP) 결과를 이용하고, A는 암호를 복호화하기 위해 Addition k(rP) 결과를 이용한다. Addition r(kP)와 Addition k(rP)가 일치되는 것을 알 수 있다. ECC의 키는 짧지만 보안 강도가 강하므로 디지털 서명인 공개키 알고리즘에도 충분히 사용될 수 있다. 또한, 이 키에 변형을

가하여 본 논문에서 제안하고 있는 전자봉투를 제거하기 위한 대칭키 알고리즘의 키로도 사용할 수 있다.

3. SSET(Strengthened and Secure Electronic Transaction)

제안된 SSET는 ECC를 전자 서명의 공개키 알고리즘에 사용하여 보안 강도를 강화시키고, 이 때 발생되는 개인키와 공개키의 값을 변형시켜, 본 논문에서 제안하고 있는 대칭키 암호화 알고리즘의 키로 사용하여 그림 1과 같이 전자 봉투를 생략할 수 있다.

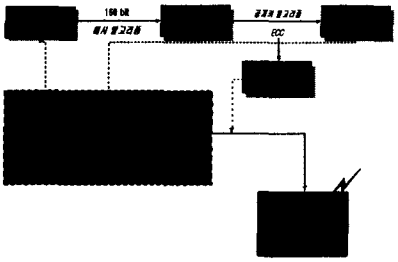

그림 1 SSET의 암호화

3.1 디지털 서명

기존의 SET의 RSA 공개키 암호 알고리즘을 ECC로 대치하여 사용한다. 해쉬 알고리즘으로 생성된 메시지 요약을 ECC를 사용 서명한다.

3.2 대칭키 암·복호화 알고리즘

데이터의 보안과 속도를 위해 사용하는 SET의 대칭키 알고리즘인 DES(Data Encryption Standard)을 대치할 새로운 알고리즘을 제안한다. 제안되는 알고리즘은 ECC의 암·복호화 키를 변형하여 두 개의 키를 생성하고 이 키를 바이트 교환 암호화와 비트 XOR 연산의 키로 사용하여 데이터를 암호화한다.

제안하는 대칭키 알고리즘은 다음과 같다.

전자서명에 사용되는 ECC키, P= (x, y)를 이용하여 x 와 y를 4개의 숫자로 고정하기 위해서 x 나 y 가 4개의 숫자 미만일 경우 오른쪽에 0을 패딩하고, x 나 y 가 4개 숫자 이상일 경우 오른쪽 4개만 남기고 상위 자리는 절단한다. 의사 코드는 표 1과 같다.

표 1 자릿수 고정 의사 코드

```
/*fix digit*/
P(x,y)//a point over ECC
x=4 digit, y=4digit
if digit of x < 4
move x to left of x'
fill 0 to remainder area of x'
move y to left of y'
fill 0 to remainder area of y'
else
move 4 right digit of x to x'
move 4 right digit of y to y'
end-if
move x' to x
move y' to y
```

바이트 교환키(Kex)는 x 와 y를 연접(concatenation)시킨 후 8개 각 숫자를 mod 8 연산을 하여 기본형 키로 한다(N). 비트 XOR 키(Kxor)는 표 1에서 생성된 x 와 y를 연접하여 각각의 숫자를 ASCII 코드로 변환한 후 이 값과 역전(reverse)시킨 값을 OR 연산한 결과를 키로 정한다. 키의 생성은 표 2와 같다.

표 2 N과 Kxor의 생성 의사 코드

```
/*create N and Kxor*/
//create N
P(x, y)
concatenate x and b
move remainder of 8 digit with 8 to N
//create Kxor
P(x, y)
xx = convert x to ASCII code
yy = convert y to ASCII code
p = concatenate xx and yy
move reverse of p to p'
or of p and p'
```

데이터 암호화는 8byte×8byte 크기인 64byte를 기본으로 하며, 이 64byte를 1개 블록으로 정의한다. 블록의 구조는 데이터가 56byte, 데이터 순서 번호가 4byte, 블록 번호가 4byte인 그림 2와 같은 형태이다.

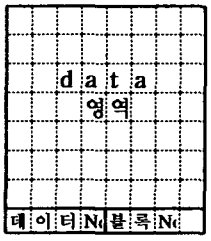

그림 2 블록 구조

암호화를 위해 입력되는 데이터를 56byte 크기로 절단한다. 56byte 미만의 데이터일 경우 난수를 이용 56byt 가 되게 임의 문자로 패딩한다. 데이터 블록의 수는 2의 배수로 완성되는데 56byte 블록 단위가 2배수가 되지 않을 때는 1개 블록을 난수를 이용, 임의 문자로 패딩하여 생성시킨다. 절단된 56byte 순서대로 64byte 블록의 데이터 영역에 저장하고 데이터 번호를 데이터 번호 영역에 첨부한다. 블록 번호는 블록의 순서가 아니라 바이트 교환을 할 한 쌍의 블록을 선택하기 위한 임의 수로 데이터 순서와 전혀 관련성이 없다. 난수를 이용하여 발생된 블록 번호를 블록의 블록 번호 영역에 첨부하는데 동일한 블록 번호는 2회 이하 발생되어야 하며 3회 이상 발생되면 다시 난수를 발생시켜 첨부시킨다. 또한 전체 블록 수/2이하의 블록 번호가 발생 되도록 난수 발생을 조정하여야 한다.

표 3 블록의 각 영역 부착 의사 코드-1

```
/*input data, data sequential number and block random
number to each area*/
clear datasqnum, randomnum
loop: for end of totalblock //input data to data area
      move next 56 bytes of data to data area
if data < 56 byte
fill with random character in empty of data area
end-if
move datasqnum to data sequential number area
add 1 to data datasqnum
//input randomnum to block random number area
 move randomnum to block number area
 set 'NOT_OK' to block-num-ok
 loop : until block-num-ok is 'OK'
 if ( random number >= 1 ) and
                (random number < totalblock/2)
 set 'OK' to block-num-ok
 else
 set 'NOT_OK' to block-num-ok
 end-if
//count randomnum
```

표 4 블록의 각 영역 부착 의사 코드-2

```
clear i , dup-ran-num
loop: for end of block number table
add 1 to i
if (randomnum=block number table(i))
add 1 to dup-ran-num
if dup-ran-num=3
set 'NOT_OK' to block-num-ok
end-if
end-if
end-loop
end-loop
end-loop
```

데이터의 암호화는 블록간 byte 교환을 수행한 후 비트 XOR 연산을 하여 암호화한다. Byte 교환은 Byte 교환 횟수에 따라 byte 교환 위치를 변화시켜 블록 번호가 같은 블록 한 쌍을 호출하여 교환시킨다.

- Byte 교환 위치 = (row, col)
  row = N * 교환 횟수 mod 8
  col = ((N * 교환 횟수 mod 8) + 3) mod 8
- 교환 과정
  난수 발생(RN), 1 ≤ RN ≤ 총 블록 수 / 2
  발생된 난수의 블록 쌍을 호출하여 byte 교환

비트 XOR 연산은 표 2에서 생성시킨 키 Kxor과 64bit 단위의 데이터로 이루어진다. 의사 코드는 표 5와 같다.

표 5 데이터 암호화 의사 코드

```
/*encrypt data I*/
loop : for end of totalblock/2
add 1 to exchange-times
//create random number
set 'NOT-OK' to block-num-ok
loop : until block-num-ok is 'OK'
move random number to block-number-1
if ( random number >= 1 ) and
        ( random number  < totalblock/2 )
set 'OK' to block-num-ok
else
set 'NOT_OK' to block-num-ok
end-if
end-loop
//byte exchange between two blocks(Kex)
call two blocks having the same number
add 1 to i
loop : until 8
row = ( N(i) * exchange-times ) mod 8
col = ( ( N(i) * exchange-times  mod 8 ) + 3 ) mod 8
exchange block1( row, col )byte for
        block2( row, col )byte
end-loop
end-loop
/*encrypt data II*/
loop : end of totalblock
xor of data by 64bit and Kxor
end-loop
```

복호화 과정은 먼저 Byte 교환키 Kex와 bit 교환키 Kxor을 연산한다. 이어서 수신된 암호문을 수신된 순서대로 64bit 단위로 절단한 후 암호문 데이터를 키 Kxor을 사용하여 Bit XOR 연산을 행한다. Bit XOR 연산이 끝난 후, 수신된 순서대로 64byte 단위 2개씩 짝을 지어 Byte 교환을 수행한다. 이상의 두 가지 처리를 끝내면 데이터(56byte), 데이터 번호(4byte), 블록 번호(4byte)가 원위치에 원래의 값으로 나타난다. 이어서, 데이터 번호에 따라 56byte 단위를 연속 연결하면 완전한 평문이 생성된다. 의사 코드는 표 6과 같다.

표 6 복호화 의사 코드

```
creation Kex and Kxor
loop : for end of receipt data
xor next 64 bit of receipt data and Kxor
end-loop
loop : for end of block
byte exchange of next 64byte receipt data by using Kex
end-loop
data arrange along datasqnum
```

4. 결론

제안된 SSET 프로토콜은 기존의 SET 프로토콜에 비해 전자 서명에 ECC 키를 이용하여 암호화 강도와 속도가 증가되고 이 키를 변형하여 대칭키 알고리즘에 사용함으로서 대칭키의 전송이 필요치 않아 전자 봉투를 생략할 수가 있다. 이런 결과로 SET이나 ECSET보다 보안성이 높고, 이중 서명에서의 전자 봉투도 생략이 가능하므로 프로토콜이 보다 간략해지고 전자 봉투 암·복호화 과정 감소와 전송 트래픽을 줄일 수 있어 능률적이다. 또한, 제안된 대칭키 알고리즘은 바이트 교환과 비트 연산과정을 통해 암호화되어, 일부의 암호 데이터가 도청되어도 해독하기가 극히 어렵고, 데이터의 정렬도 쉽지 않아 전체 평문의 내용을 파악하기 어렵다. 이로서 암호화 강도도 강화되어 효율적인 암호화가 가능하다. 향후 제안된 대칭키 알고리즘의 속도 증강과 처리 블록 크기를 적정 크기로 확대하면 처리 면과 암호화 강도를 한층 더 증가시킬 수 있다고 기대된다.

표 7 SET, ECSET, SSET의 비교

SET	RSA	DES	필수
ECSET	ECC	DES	필수
SSET	ECC	제안 대칭키	불필요

참고 문헌

[1] 이병관, 전자상거래 보완, 서울, 남두도서, 2002.

[2] N.Koblitz, "Elliptic Curve Cryptosystems", Mathematics of Computation, 48, pp.203-209, 1987.

[3] V.S. Miller, "Use of elliptic curve in cryptography", Advances in Cryptology-Proceedings of Crypto'85, Lecture Notes in Computer Science, vol.218,pp.417-426, Springer-Verlag, 1986.

[4] G. Haper, A. menezes, and S.vanstone, "Public-key Cryptosystem with very small key lengths", Advances in Cryptology-proceedings of Eurocrypt'92, Lecture Notes in Computer Science, vol.658, pp.163-173, Springer-Verlag, 1993.

[5] A. Menezes, Elliptic Curve Public Key Cryptosystem, Kluwer Academic publishers, Boston, 1993

[6] S. H. Yang and B. K. Lee, ECC for ECSET Design, KIPS, vol.8, no.2. Nov. 2001.

[7] I. S. Cho and B. K. Lee, ASEP Protocol Design, ICIS, Aug. 2002.