通信情報保護學會論文誌
第 10 券 第 1 號, 2000. 3

타원곡선 공개키 암호 알고리즘을 이용한 전자지불 프로토콜*

이 혁**, 이 정 규***

Electronic Payment Protocol using Elliptic Curve Public Key Algorithm

Hyurk Lee,** Jong-kyu Lee***

요 약

전자지불 프로토콜은 하드웨어 구현의 용이성과 안전성의 두 가지 측면을 고려하여 설계되어야 한다. 본 논문에서 제안하는 전자지불 프로토콜은 하드웨어구현을 용이하게 하고 유지비용을 줄이기 위해 초기변수값을 최소로 하는 전자지불 프로토콜에 대칭키 암호알고리즘을 적용하여 주고받는 데이터량을 줄였으며, 안전성을 증가시키기 위해 타원곡선 공개키 암호 알고리즘을 적용하여 타원곡선 공개키 암호 알고리즘이 가지는 특성들을 상속받도록 하였고, 마지막으로 전자지불 프로토콜의 안전성과 효율성을 분석하였다.

ABSTRACT

Electronic Payment Protocol has to consider the security and the cost in hardware implementation. This paper proposes a Electronic Payment Protocol which is shortened the transfer message of Electronic Payment Protocol which has few initial variables, by using symmetric key algorithm for the hardware implementation and cost, and which is applied elliptic curve public key algorithm for this high security property. We analyze efficiency and security of the protocol.

keyword : *Electronic payment, Elliptic curve, Security analysis.*

Ⅰ. 서 론

1988년 Chaum, Fiat와 Naor에 의해 전자화폐 프로토콜이 처음 발표된 이후로 실물화폐를 이용하지 않고 거래를 가능하게 하는 많은 전자지불 프로토콜들이 제안되어왔다.[1] 전자지불 프로토콜은 크게 두 가지 방향으로 연구되어 왔으며, 첫 번째는 프로토콜 자체의 보안성을 증가시키는 것이고 두 번째는 실제로 프로토콜을 하드웨어로 구현하여 사용할 때 소요되는 비용을 줄이는 것이다. 그러나, 보안성을 증가시키기 위해서는 하드웨어의 구현 및 시스템 유지비용이 증가하게 되고 유지비용을 낮추기 위해서는 보안성이 감쇄 되어야 하는 관계를 가지고 있다. 따라서 많은 전자지불 프로토콜이 제안되어 왔음에도 불구하고 활발하게 이용되지 못하였다.

전자지불 프로토콜을 하드웨어로 구현하기 쉽게 하기 위한 요건은 사용자가 지불에 참여하기 위해 가지고 있어야 할 데이타가 적어야 하며 지불과정이 일어

* 이 논문은 1999년 한양대학교 교내연구비에 의하여 연구 되었습니다.
** 한양대학교 전자계산학과 정보통신연구실 (hlee@cse.hanyang.ac.kr)
*** 한양대학교 전자계산학과 (jklee@cse.hanyang.ac.kr)

날 때 전송되는 메시지의 크기가 작아야 한다. 이러한 프로토콜 설계 요건 이외에 스마트 카드의 이용과 타원곡선 공개키 암호 알고리즘의 사용은 하드웨어의 구현을 쉽게 하며 사용자들에게 효율적인 지불환경을 제공할 수 있다. 스마트 카드를 이용하였을 경우 사용자는 지불시스템을 사용하기 위하여 매번 지불이 일어날 때마다 생성되는 시스템 관련 변수값을 알지 않아도 되며 PIN(Personal Identification Number)만을 알고 있으면 지불시스템의 이용이 가능하다. 타원곡선 공개키 암호 알고리즘의 사용은 하드웨어의 구현을 용이하게 할 뿐만 아니라 지불시스템을 사용하는데 필요한 계산 시간을 빠르게 할 수 있으며 앞으로의 컴퓨터 계산 속도의 증가에 따른 키길이의 증가에도 기존의 공개키 암호 알고리즘에 비하여 짧은 키길이로 높은 안전성을 제공할 수 있다.

본 논문에서는 Markus Jakobsson이 제안하였던 전자지불 프로토콜을 효율적인 메시지 전달이 이루어지도록 바꾸고 타원곡선 공개키 암호 알고리즘을 적용하여 높은 안전성을 가지며 하드웨어구현을 용이하게 할 수 있는 지불시스템을 설계한다. Markus Jakobsson의 전자지불 프로토콜은 사용자의 현재금액에 관련한 값과 비밀 랜덤 초기값만을 프로토콜에 이용하므로 적은 데이터를 사용하기는 하지만 실제로 많이 사용되는 지불액이 일정하지 않은 경우에 효율적이지 못하다. 따라서 이러한 경우의 효율적인 지불프로토콜을 제안하며 타원곡선 공개키 암호 알고리즘을 이용하여 프로토콜을 변환하고 시스템의 안전성을 분석한다.[2]

II. 이산대수기반 암호 프로토콜의 타원곡선 변환

프로토콜에 타원곡선 공개키 암호 알고리즘을 적용하기 위하여 이산대수상에서의 프로토콜을 타원곡선상으로 변환시키는 방법을 알아본다.

1. 타원곡선 변환

이산대수의 수식에 대응하는 타원곡선의 수식과 매개 변수들은 표 1과 같다.

2. Diffie-Hellman 키 분배방식

이산대수상에서 정의된 Diffie-Hellman 키 분배 프로토콜은 그림 1과 같고 타원곡선상으로 바꾸어주면 그림 2와 같다.

〔표 1〕 DLP와 ECDLP의 매개변수와 수식
〔Table 1〕 Parameters in DLP and ECDLP

구분	이산대수문제	타원곡선문제
방식	$h=g^a \bmod p$ h, g가 주어졌을 때 a를 찾는 문제	$Q=aP$ Q, P가 주어졌을 때 a를 찾는 문제
군	Z_p^*	$E(Z_p)$
연산	법(modulo) p상의 곱셈	점들의 덧셈
원소	$\{1,2,3,\cdots,p-1\}$ 소수 p order q generator α group $\{\alpha^0, \alpha^1, \cdots, \alpha^{q-1}\}$	$(x,y) \cup O$ $E(F_q)$, $q=p$ or 2^m order n generator point P group $\{O, P, 2P, \cdots, (n-1)P\}$
표기법	element g, h multiplication $g \times h$ division g/h inverse g^{-1} exponentiation g^a	element P, Q addition $P+Q$ subtraction $P-Q$ inverse $-P$ scalar multiplication aP
키생성	$[2, q-2]$의 구간에서 임의의 정수 u를 택한다. $x=g^u \bmod p$ private key : u public key : x	$[2, q-2]$의 구간에서 임의의 정수 d를 택한다. $Q=dP$ private key : d public key : Q

사용자 1		사용자 2
비밀키 a를선택		비밀키 b를 선택
$y_A=g^a \bmod p$	$\xrightarrow{y_A}$ $\xleftarrow{y_B}$	$y_B=g^b \bmod p$
$(y_B)^a=g^{ab} \bmod p$		$(y_A)^b=g^{ab} \bmod p$

〔그림 1〕 이산대수상의 Diffie-Hellman 키분배 프로토콜
〔Fig. 1〕 Diffie-Hellman key agreement protocol on DLP

이산대수상에서 사용자 1과 사용자 2는 결과적으로 정수값인 비밀키 a, b를 가지고 $g^{ab} \bmod p$ 인 정수값인 키를 분배하게 된다.

타원곡선상에서 비밀키는 a와 b로 정수값을 가지며 공개키는 타원곡선 E_p상의 좌표값이다. 즉 $A=E_p(x_A, y_A)$, $B=E_p(x_B, y_B)$와 같이 좌표값을 가지며 사용자1과 사용자 2는 $abP=E_p(x_{AB}, y_{AB})$의 좌표를 분배하게 된다.[3,4]

사용자 1		사용자 2
비밀키 a를선택		비밀키 b를 선택
$A=aP$	A ⟶ ⟵ B	$B=bP$
$aB=a(bP)=abP$		$bA=b(aP)=abP$

〔그림 2〕 타원곡선상의 Diffie-Hellman 키분배 프로토콜
〔Fig. 2〕 Diffie-Hellman key agreement protocol on ECDLP

3. 타원곡선 좌표값의 암호프로토콜 적용

이산대수상에서 그림 2에서와 같이 타원곡선상의 Diffie-Hellman 키분배 프로토콜에서 분배되는 키는 x좌표값과 y좌표값 (x,y)의 형태로 생성된다. 따라서 기존의 암호 프로토콜에 이러한 키값이 사용되기 위해서는 생성된 좌표값이 하나의 정수값을 가지도록 해야 한다. 이러한 타원곡선상의 좌표값을 키값으로 사용하기 위한 방법은 다음과 같다.

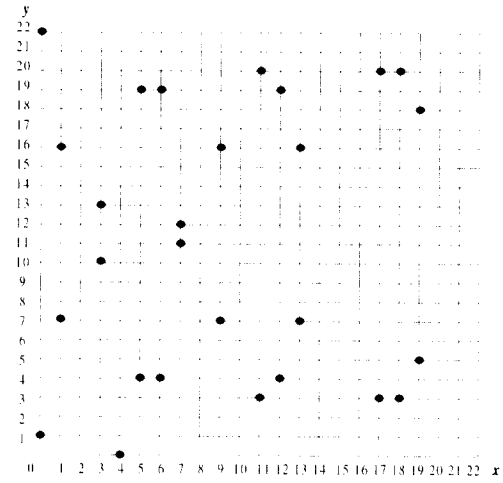
3.1 좌표값을 모두 복원해야 하는 경우

Z_{23}상에서의 $y^2=x^3+x+1$점들은 다음과 같다.

(0,1)	(0,22)	(1,7)	(1,16)	(3,10)	(3,13)
(4,0)	(5,4)	(5,19)	(6,4)	(6,19)	(7,11)
(7,12)	(9,7)	(9,16)	(11,3)	(11,20)	(12,4)
(12,19)	(13,7)	(13,16)	(17,3)	(17,20)	(18,3)
(18,20)	(19,5)	(19,18)			

좌표값을 전송할 때 (11,3)을 2진수를 사용하여 전송할 경우 x좌표 01011과 y좌표 00011을 붙여서 0101100011을 전송한다. 그러나 타원곡선상의 점의 좌표는 그림 3과 같이 대칭적인 구조를 가지고 있으므로 원래의 좌표값을 표기하기 위해서는 x좌표값에 1비트만 추가함으로서 좌표값을 나타낼 수 있다.

그림 3에서와 같이 타원곡선상의 점들은 하나의 x좌표값에 대하여 y좌표값이 대칭적으로 두개가 존재하므로 실제로 (x,y) 좌표값을 전송할 때 $0\le y\le 11$일 경우 $(x,0)$을 전송하고 $12\le y\le 22$일 경우 $(x,1)$을 전송하면 좌표값을 복원해 낼 수 있다. 따라서 (11,3)을 전송할 때 010110 만을 전송하면 원래의 좌표값을 복원해 낼 수 있다.



〔그림 3〕 F_{23}상의 $y^2=x^3+x+1$
〔Fig. 3〕 $y^2=x^3+x+1$over F_{23}

3.2 좌표값의 복원이 필요없는 경우

점의 좌표를 복원할 필요없이 단순히 키값만을 사용할 경우에는 x좌표값만을 취하여 비밀키값으로 사용한다. 본 논문에서는 생성된 좌표값의 x좌표값만을 취하는 방식을 사용한다.[5,6]

Ⅲ. 지불 프로토콜의 설계

지불프로토콜은 하드웨어로 구현되었을 때 효율성과 안전성을 고려하여 설계되어야 한다. 본문에서는 제안하고자하는 지불프로토콜의 기본 모델을 설정한다.

1. 지불프로토콜의 구성요소

지불프로토콜의 구성요소는 다음과 같다.

- 지불자(Payer) : 지불을 하는 객체(entitiy).
- 상인(Merchant) : 지불을 받는 객체.
- 은행(Bank) : 지불자와 상인들의 계좌에 관한 데이터와 자금이체등의 데이터를 가지고 있음.
- 지불장치발행기관(Issuer) : 지불장치를 생산하고 사용자들에게 보급.

지불자와 상인은 모두 은행에 속해있는 사용자들이며 지불과정이 이루어질 때 지불을 하는 쪽이 지불자가 되고 지불을 받는쪽은 상인이 된다.

2. 지불프로토콜의 매개변수

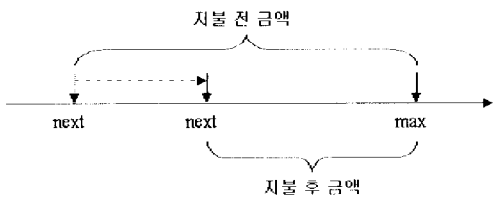
본 논문에서 사용하게 될 매개변수는 표 2와 같다.

〔표 2〕 프로토콜에 사용되는 매개변수
〔Table 2〕 Parameters for Protocol

σ	사용자와 은행만 가지고 있는 비밀 랜덤 초기값 (secret random seed)
max	지불받기 과정에서 사용되는 카운터 (counter) 변수
$next$	지불하기 과정에서 사용되는 카운터 변수
g	이산대수문제의 생성자(generator)
G	타원곡선문제의 생성자
$H(), f()$	일방향 해쉬함수(one way hash function)
$E(K,M)$	대칭키 K로 메시지 M을 암호화
$D(K,M)$	대칭키 K로 메시지 M을 복호화
x	지불과정에서 생성하는 비밀키
y	지불과정에서 생성하는 공개키
n	지불되는 금액
M, Mi $(1 \le i \le n)$	상인(Merchant) : 지불을 받는 객체 ex) max_M : 상인의 max변수 max_{Mi} : 상인의 i번째 max변수
P, Pi $(1 \le i \le n)$	지불자(Payer) : 지불을 하는 객체 ex) max_P : 지불자의 max변수 max_{Pi} : 지불자의 i번째 max변수

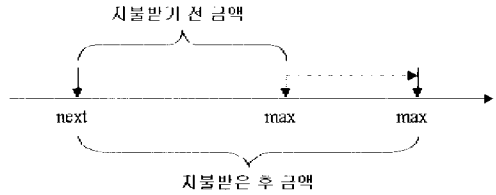
3. 지불과정의 원리

지불과정은 max와 $next$ 변수를 가지고 이루어진다. 사용자들은 지불을 위해 max와 $next$ 변수를 가지고 있으며, 이때 자신이 현재 가지고 있는 금액은 $max-next$값으로 확인한다. 지불을 받게 되는 상인은 지불과정이 끝난 후 자신의 max변수를 증가시고 지불을 하는 지불자는 자신의 $next$변수를 증가시켜서 자신의 현재 금액을 변화한다. 이러한 지불하기 과정과 지불받기 과정은 다음과 같다.



〔그림 4〕 지불하기 과정
〔Fig. 4〕 Process of making a payment

지불을 하는 지불자는 그림 4와 같이 자신의 $next$ 변수를 증가시킴으로서 지불을 하게된다. 이때 지불자는 $max - next \ge 0$ 일 경우에만 지불이 가능하다. 즉 자신이 가지고 있는 잔고 금액보다 적은 액수의 금액일 경우만 지불이 가능하다.



〔그림 5〕 지불받기 과정
〔Fig. 5〕 Process of receiving a payment

지불 받기 과정은 그림 5와 같이 max 변수를 증가시킴으로서 지불을 받게 된다. 이러한 지불과정은 지불자, 상인과 은행이 모두 참여한 온-라인(on-line)상태에서 이루어진다.

4. 지불장치의 초기화

지불과정이 이루어지기 위하여 필요한 매개 변수들을 설정하는 초기화가 필요하다. 필요한 매개변수는 σ, max, $next$가 있으며 다음과 같이 초기화가 이루어진다.

지불프로토콜이 수행되기 위해서는 초기에 은행과 사용자는 은행과 사용자만이 알고 있는 비밀 랜덤 초기값 σ를 저장하고 있어야 한다. 이것은 그림 1과 같이 Diffie-Hellman 키 분배 방식을 이용하여 $\sigma = g^{ab} \bmod p$를 분배한다. 분배된 σ값은 사용자의 저장장치와 은행의 데이터 베이스에 저장이 되며 사용자의 저장장치와 해당 사용자에 대한 은행의 데이터 베이스에 있는 max변수와 $next$변수는 모두 0으로 초기화 된다.

5. 카운터를 하나씩 증가하는 지불 프로토콜

지불프로토콜에서 매번 지불금액이 같은 경우에는 지불금액에 대해 모두 처리해주지 않고 매번 지불이 일어날 때 카운터를 하나씩만 증가시킴으로서 프로토콜을 구성할 수 있다. 이러한 방식은 버스카드와 같이 일정금액을 지불할 때 사용할 수 있다. 이번 장에서는 지불금액이 매번 일정한 경우의 지불프로

토콜을 설계하며 이것은 지불이 일어날 때 카운터값을 하나씩 증가시키는 방식으로 구성한다..

1) 지불을 받게 되는 사용자인 상인은 자신의 비밀 랜덤 초기값 σ_M과 자신의 max_M변수로 f 함수를 이용하여 비밀키 x_M을 생성해 낸다. x_M을 이용하여 y_M을 생성해 낸다.

$$x_M = f(\sigma_M, max_M + 1, 1)$$
$$y_M = g^{x_M} \bmod p \quad (1)$$

생성해낸 y_M값을 지불자에게 전송한다.

2) 지불을 할 지불자는 자신의 비밀 랜덤 초기값 σ_P와 $next_P$를 이용하여 두 개의 비밀키 x_P와 k를 생성해 낸다.

$$x_P = f(\sigma_P, next_P, 1)$$
$$k = f(\sigma_P, next_P, 2) \quad (2)$$

생성한 x_P와 k를 가지고 공개키 y_P와 r을 생성해 낸다.

$$(y_P, r) = (g^{x_P}, g^k) \quad (3)$$

k, x_P와 r을 가지고 상인으로부터 전송받은 y_M에 대한 전자서명 값을 생성한다.

$$s = k - x_P H(y_M, r) \quad (4)$$

3) 생성된 값들 (y_P, y_M, r, s)을 은행으로 보낸다. 이때 은행에서는 전자서명값을 확인한다.

$$\begin{aligned} r &= g^s y_P^{H(y_M, r)} \\ &= g^s g^{x_P H(y_M, r)} \\ &= g^{k - x_P H(y_M, r)} g^{x_P H(y_M, r)} \\ &= g^k (\bmod p) \end{aligned} \quad (5)$$

4) 은행은 자신이 저장하고 있던 상인에 대한 σ_M, max_M과 지불자에 대한 σ_P, $next_P$변수를 이용하여 y_P와 y_M을 생성한 후 수신받은 y_P, y_M과 비교하여 현재 지불자의 $next_P$값과 상인의 max_M값이 맞는지를 확인한다.

5) 위와 같은 확인과정이 끝난 후 은행은 해당 지불자와 상인에게 지불이 성공적으로 이루어 졌음을 알려주고 지불자는 $next_P$값을 하나 증가시키고 상인의 경우 max_M값을 하나 증가시킨다. 또한 은행의 데이터베이스에 있는 해당 상인의 max_M과 지불자의 $next_P$변수값을 하나씩 증가시킨다.

위의 과정은 그림 6과 같다.

지불자 (σ_P, max_P, $next_P$)	상인 (σ_M, max_M, $next_M$)
$x_P = f(\sigma_P, next_P, 1)$ $k = f(\sigma_P, next_P, 2)$ $(y_P, r) = (g^{x_P}, g^k)$	$x_M = f(\sigma_M, \max_M + 1, 1)$ $y_M = g^{x_M} \bmod p$ y_M을 지불자에게 전송
$s = k - x_P H(y_M, r)$	
(y_P, y_M, r, s)값을 은행으로 전송	
은행확인 후 $next_P$증가	은행확인 후 max_M 증가

〔그림 6〕 카운터를 하나씩 증가하는 지불프로토콜
〔Fig. 6〕 One count payment protocol

6. 카운터를 하나이상 증가하는 지불 프로토콜

매번 지불금액이 일정하지 않고 상품구매와 같이 일정하지 않은 지불금액을 계산하는 경우의 프로토콜은 카운터를 하나 증가하는 지불 시스템으로부터 금액에 따라 여러번 카운터 할 수 있도록 변환하여 구성한다. 금액 n을 지불하는 프로토콜은 다음과 같다.

1) 상인측에서는 금액 n에 대하여 아래와 같이 i 값을 증가시켜서 금액 n만큼의 y_{Mi}값을 생성하여 μ_M값을 지불자에게 전송한다($1 \le i \le n$).

$$x_{Mi} = f(\sigma_M, max_M + i, 1)$$
$$y_{Mi} = g^{x_{Mi}} \bmod p$$
$$\mu_M = (y_{M1}, \cdots, y_{Mn}) \quad (6)$$

2) 지불을 할 지불자는 i 값을 증가시켜서 금액만큼의 y_{Pi}값을 다음과 같이 생성하여 μ_P값을 만든다.

$$x_{Pi}=f(\sigma_P, next_P+i, 1)$$
$$y_{Pi}=g^{x_{Pi}} \bmod p$$
$$\mu_P=(y_{P1}, \cdots, y_{Pn}) \quad (7)$$

서명에 필요한 k값과 r값을 다음과 같이 생성한다.

$$k=f(\sigma_P, next_P+n, 2)$$
$$r=g^k \bmod p \quad (8)$$

μ_M과 μ_P을 붙여서 μ값을 생성한다.

$$\mu=(\mu_M, \mu_P)$$
$$=((y_{M1}, \cdots, y_{Mn}), (y_{P1}, \cdots, y_{Pn})) \quad (9)$$

weight 값 $w_i=hash(\mu_P, i)$과 x_{Pi}값을 이용하여 X값을 생성하고 이를 이용하여 Y값을 생성한다.

$$X=\sum_{i=1}^{n} w_i x_{Pi}$$
$$Y=g^X=g^{\sum_{i=1}^{n} x_{Pi} \cdot w_i} \quad (10)$$

μ에 대한 전자서명 값을 생성한다.

지불자 (σ_P, max_P, $next_P$)	상인 (σ_M, max_M, $next_M$)
$x_{Pi}=f(\sigma_P, next_P+i, 1)$ $k=f(\sigma_P, next_P+n, 2)$ $(y_{Pi}, r)=(g^{x_{Pi}}, g^k)$	$x_{Mi}=f(\sigma_M, \max_M+i, 1)$ $y_{Mi}=g^{x_M} \bmod p$ $\mu_M=y_{M1}, \cdots, y_{Mn}$ μ_M을 지불자에게 전송
$\mu_P=y_{P1}, \cdots, y_{Pn}$ $\mu=(\mu_M, \mu_P)$ $w_i=hash(\mu_P, i)$ $Y=g^{\sum_{i=1}^{n} x_{Pi} \cdot w_i}$ $=\prod_{i=1}^{n}\{g^{x_{Pi}}\}^{w_i}$ $=\prod_{i=1}^{n} y_{Pi}^{w_i}$ $X=\sum_{i=1}^{n} w_i x_{Pi}$ $s=k-XH(\mu, r)$	
(μ, Y, r, s)값을 은행으로 전송	
은행확인 후 $next_P$를 n만큼 증가	은행확인 후 max_M을 n만큼 증가

〔그림 7〕 카운터를 하나이상 증가하는 지불프로토콜
〔Fig. 7〕 Batch payment protocol

$$s=k-XH(\mu, r) \quad (11)$$

3) 생성된 값들 (μ, Y, r, s)을 은행으로 보낸다. 이때 은행에서는 전자서명값을 확인한다. 이때 은행에서는 $r=g^s Y^{H(\mu, r)}$로 확인하며 이때 사용하는 $Y=\prod_{i=1}^{n} y_{Pi}^{w_i} \bmod p$으로 계산해낸다.

$$r=g^s Y^{H(\mu, r)}$$
$$=g^{k-XH(\mu, r)} g^{\left(\sum_{i} x_{Pi} \cdot w_i\right) H(\mu, r)}$$
$$=g^{k-XH(\mu, r)} g^{XH(\mu, r)}=g^k \quad (12)$$

위의 과정은 그림 7과 같다.

Ⅳ. 대칭키 암호알고리즘을 이용한 지불프로토콜

1. 지불프로토콜의 단점

카운터를 하나이상 증가시키는 지불프로토콜은 지불되는 금액에 따라서 계산시간과 전송하는 메시지의 길이가 다르게 나타나게 된다. 그림 7의 $\mu=(\mu_M, \mu_P)$ 값을 계산하기 위해서는 지불되는 금액만큼 계산을 반복하게 된다. 즉 n만큼의 금액을 보낼 경우 상인측에서 x_{Mi}, y_{Mi} 지불자측에서 x_{Pi}, y_{Pi}의 계산을 n번씩 반복해야 한다. 또한 이렇게 생성되어진 메시지들이 전송되어 지려면 카운터를 하나씩 증가시키는 프로토콜보다 n배의 데이터를 전송하여야 한다. 전송되는 메세지는 $\mu_M=\{y_{M1}, \cdots, y_{Mn}\}$, $\mu_P=\{y_{P1}, \cdots, y_{Pn}\}$이며 $\mu=\{\mu_M, \mu_P\}$이므로 지불되는 금액이 많아지면 전송되는 데이터의 양이 많아진다. 따라서 본문에서는 카운터가 하나이상 증가하는 지불프로토콜의 효율적인 방안을 제안한다.

2. 대칭키 암호 알고리즘을 이용한 지불프로토콜

위와 같은 단점을 보완하기 위하여 지불금액만큼의 공개키값들을 전송해주는 것이 아니라 지불금액 n을 대칭키 암호 알고리즘을 이용하여 암호화하여 전송한 후 은행에서 이를 복호화하여 지불의 확인과정에 이용한다.

대칭키 암호 알고리즘을 이용한 카운터를 하나이상 증가하는 지불프로토콜을 설계하면 다음과 같다.

1) 상인은 다음과 같이 지불금액 n을 이용하여 비밀키 값 x_M을 생성하고 이를 이용하여 공개키값 y_M을 생성하여 지불자에게 전송한다.

$$x_M = f(\sigma_M, \max_M + n, 1)$$
$$y_M = g^{x_M} \bmod p \quad (13)$$

2) 지불자는 다음과 같이 비밀키 x_P와 k값을 생성한다.

$$x_P = f(\sigma_P, next_P, 1)$$
$$k = f(\sigma_P, next_P, 2) \quad (14)$$

생성한 x_P와 k를 가지고 공개키 y_P와 r을 생성해 낸다.

$$(y_P, r) = (g^{x_P} \bmod p, g^k \bmod p) \quad (15)$$

x_P와 r을 가지고 상인으로부터 전송받은 y_M에 대한 전자서명 값을 생성한다.

$$s = k - x_P H(y_M, r) \quad (16)$$

x_P를 비밀키로 하여 지불금액 n을 대칭키 암호 알고리즘을 이용하여 암호화한다.

$$E(x_P, n) \quad (17)$$

3) 생성된 값들 $(y_P, y_M, r, s, E(x_P, n))$을 은행으로 보낸다. 이때 은행에서는 전자서명값을 확인한다.

$$r = g^s y_P^{H(y_M, r)}$$
$$= g^s g^{x_P H(y_M, r)}$$
$$= g^{k - x_P H(y_M, r)} g^{x_P H(y_M, r)}$$
$$= g^k (\bmod p) \quad (18)$$

4) 은행은 자신이 저장하고 있던 상인에 대한 σ_M, max_M과 지불자에 대한 σ_P, $next_P$변수를 이용하여 x_P를 생성해내고 이를 이용하여 n값을 얻는다.

$$n = D(x_P, E(x_P, n)) \quad (19)$$

생성해낸 x_P값을 이용하여 y_P를 생성한 후 전송받은 값과 비교하여 현재 지불자의 $next_P$값이 맞는지 확인한다.

5) 얻어낸 n값을 이용하여 상인의 비밀키인 x_M값을 생성해낸다.

$$x_M = f(\sigma_M, \max_M + n, 1) \quad (20)$$

x_M으로 y_M을 생성하여 현재 상인의 max_M값이 맞는지 확인한다.

6) 위와 같은 확인과정이 끝난후 은행은 해당 지불자와 상인에게 지불이 성공적으로 이루어 졌음을 알려주고 지불자는 $next_P$변수값을 n만큼 증가시키고 상인의 경우 max_M값을 n만큼 증가시킨다. 또한 은행의 데이터베이스에 있는 해당 상인의 max_M과 지불자의 $next_P$변수값을 n만큼 증가시킨다.

이러한 과정은 그림 8과 같다.

지불자 (σ_P, max_P, $next_P$)	상인 (σ_M, max_M, $next_M$)
$x_P = f(\sigma_P, next_P, 1)$ $k = f(\sigma_P, next_P, 2)$ $(y_P, r) = (g^{x_P}, g^k)$	$x_M = f(\sigma_M, \max_M + n, 1)$ $y_M = g^{x_M} \bmod p$ y_M을 지불자에게 전송
$s = k - x_P H(y_M, r)$ $E(x_P, n)$	
$(y_P, y_M, r, s, E(x_P, n))$ 값을 은행으로 전송	
은행확인 후 $next_P$를 n만큼 증가	은행확인 후 max_M을 n만큼 증가

〔그림 8〕 대칭키 암호 알고리즘을 사용한 지불프로토콜
〔Fig. 8〕 Payment protocol using symmetric key algorithm

V. 타원곡선을 이용한 지불프로토콜

본문에서는 Ⅳ장에서 설계된 지불시스템을 타원곡선상에서 설계한다.

1. 지불장치의 초기화

지불장치의 초기화를 위해서 초기에 은행과 사용자는 비밀 랜덤초기값 σ를 분배받아야 하므로 타원곡선 상에서의 Diffie-Hellman 키 분배방식을 이용

한다. 타원곡선상에서의 키 분배 방식은 그림 2와 같으며 이때 공유된 좌표값 $abP=E_p(x_{AB}, y_{AB})$중에서 x좌표값인 x_{AB}값을 비밀 랜덤 초기값 σ로 사용한다.

2. 타원곡선상의 지불프로토콜

대칭키 암호 알고리즘을 이용한 지불프로토콜을 타원곡선상에서 설계하면 다음과 같다.

1) 지불을 받게 되는 상인은 자신의 비밀 랜덤 초기값 σ_M, 자신의 max_M변수와 지불금액 n을 가지고 f 함수를 이용하여 비밀키 x_M을 생성해 낸다. x_M을 이용하여 공개키값을 생성해 낸다.

$$x_M=f(\sigma_M, max_M+n, 1)$$
$$(y_{Mx}, y_{My})=x_M \cdot G \quad (21)$$

이때의 공개키값은 x_M 으로부터 생성해낸 좌표값의 x좌표값인 y_{Mx}값을 지불자에게 전송한다.

2) 지불을 할 지불자는 자신의 비밀 랜덤 초기값 σ_P와 $next_P$ 를 이용하여 두 개의 비밀키 x_P와 k를 생성해 낸다.

$$x_P=f(\sigma_P, next_P, 1)$$
$$k=f(\sigma_P, next_P, 2) \quad (22)$$

생성한 x_P와 k를 가지고 공개키값을 생성한다.

$$(y_{Px}, y_{Py})=x_P \cdot G$$
$$(r_x, r_y)=k \cdot G \quad (23)$$

이때의 공개키 값은 x좌표값인 y_{Px}와 r_x만을 취하며, x_P와 r_x값을 가지고 상인으로부터 전송받은 y_{Mx}에 대한 전자서명 값을 생성한다.

$$s=k-x_P H(y_{Mx}, r_x) \quad (24)$$

x_P를 비밀키로 하여 지불금액 n을 대칭키 암호 알고리즘을 이용하여 암호화 한다.

$$E(x_P, n) \quad (25)$$

3) 생성된 값들 $(y_{Px}, y_{Mx}, r_x, s, E(x_P, n))$을 은행으로 보낸다. 이때 은행에서는 다음과 같이 계산하고 r_x 값만을 취하여 지불자로부터 전송받은 r_x값과 맞는지를 비교하여 전자서명값을 확인한다.

$$\begin{aligned}(r_x, r_y) &= s \cdot G + H(y_{Mx}, r_x) \cdot y_{Px} \\ &= s \cdot G + H(y_{Mx}, r_x) \cdot x_P \cdot G \\ &= (k - x_P H(y_{Mx}, r_x)) \cdot G \\ &\quad + H(y_{Mx}, r_x) \cdot x_P \cdot G \\ &= k \cdot G \end{aligned} \quad (26)$$

4) 은행은 자신이 저장하고 있던 σ, max , $next$ 변수를 이용하여 x_P를 생성해내고 이를 이용하여 n값을 얻는다.

$$n=D(x_P, E(x_P, n)) \quad (27)$$

생성해낸 x_P값을 이용하여 y_{Px}를 생성한 후 전송받은 값과 비교하여 지불자의 $next_P$값이 맞는지 확인한다.

5) 얻어낸 n값을 이용하여 상인의 비밀키인 x_M값을 생성해낸다.

$$x_M=f(\sigma_M, \max_M+n, 1) \quad (28)$$

x_M으로 y_{Mx}를 생성하여 상인의 현재 max_M값이 맞는지 확인한다.

지불자 (σ_P, max_P, $next_P$)	상인 (σ_M, max_M, $next_M$)
$x_P=f(\sigma_P, next_P, 1)$ $k=f(\sigma_P, next_P, 2)$ $(y_{Px}, y_{Py})=x_P \cdot G$ $(r_x, r_y)=k \cdot G$	$x_M=f(\sigma_M, \max_M+n, 1)$ $(y_{Mx}, y_{My})=x_M \cdot G$ y_{Mx}을 지불자에게 전송
$s=k-x_P H(y_{Mx}, r_x)$ $E(x_P, n)$	
$(y_{Px}, y_{Mx}, r_x, s,$ $E(x_P, n))$ 값을 은행으로 전송	
은행확인 후 $next_P$를 n만큼 증가	은행확인 후 max_M을 n만큼 증가

〔그림 9〕 대칭키 암호 알고리즘을 이용한 타원곡선 상의 지불프로토콜

〔Fig. 9〕 Elliptic curve payment protocol using symmetric key algorithm

6) 위와 같은 확인과정이 끝난후 은행은 해당 지불자와 상인에게 지불이 성공적으로 이루어 졌음을 알려주고 지불자는 $next_P$변수값을 n만큼 증가시키고 상인의 경우 max_M값을 n만큼 증가시킨다. 또한 은행의 데이터베이스에 있는 해당 상인의 max_M과 지불자의 $next_P$변수값을 n만큼 증가시킨다.

위의 과정은 그림 9와 같다.

Ⅵ. 지불 프로토콜의 안전성 분석

본문에서는 지불프로토콜이 공격자의 공격에 취약해 질 수 있는 각각의 경우에 대한 안전성을 분석한다.[7,8]

1) 상인이 자신의 금액을 임의로 증가시키려는 경우

상인은 지불자가 없는 상태에서 지불자의 비밀데이터를 알게 되면 은행에 허위로 지불 사실을 통보하여 자신의 금액을 증가시키려고 할 수 있다.

그림 9에서 지불자는 지불과정에 참여하기 위하여 자신과 은행만이 알고 있는 비밀 랜덤 초기값 σ와 자신의 $next$ 변수를 키값에 의존하는 일방향 함수 f를 이용하여 x_P와 k를 생성해 낸다. 따라서 x_P와 k값을 공격자가 알아낸다고 하더라도 원래의 입력값으로 사용되었던 σ와 $next$값을 유추해 낼 수 없다. 따라서 공격자인 상인은 정당한 지불자의 참여 없이는 자신의 금액을 임의로 증가할 수 없다.[9]

2) 전송되는 데이터로 반복공격(replay attack)을 하려는 경우

전송되는 $(y_{Px}, y_{Mx}, r_x, s, E(x_P, n))$값을 저장하였다가 다시 전송하는 반복 공격을 행할 수 없다. 지불자는 지불이 이루어 질때마다 $next$변수가 바뀌므로 $x_P = f(\sigma_P, next_P, 1)$, $k = f(\sigma_P, next_P, 2)$와 같이 지불과정에서 생성되는 값들이 매번 바뀌게 된다. 이것은 일회용 서명과 같은 역할을 하게 됨으로서 공격자가 전송하는 데이터를 가로챈다고 하더라도 다시 같은 데이터를 전송하는 방법으로 공격할 수 없다.[10]

3) 전송되는 데이터를 변경하려는 경우

상인과 지불자 사이에 지불이 일어났음을 알리기 위해서 지불자는 $s = k - x_P H(y_{Mx}, r_x)$와 같이 상인에게서 전송받은 y_{Mx}값에 전자서명을 하여 $(y_{Px}, y_{Mx}, r_x, s, E(x_P, n))$를 은행으로 전송하게 된다. 따라서 은행에서는 상인과 지불자의 지불금액을 확인하기 이전에 전자서명값을 확인함으로서 은행으로 전송되는 데이터에 대한 무결성을 확인할 수 있다.[11]

4) 전송되는 $E(x_P, n)$를 변경하려는 경우

은행이 저장하고 있는 있는 지불자의 비밀랜덤 초기값 σ_P와 $next_P$값을 이용하여 $x_P = f(\sigma_P, next_P, 1)$를 구한후 y_{Px}값을 구하여 지불자의 변수값이 맞는지 확인하고 또한 x_P를 사용하여 $D(x_P, E(x_P, n))$을 하여 n값을 구해낸다. n값으로 $x_M = f(\sigma_M, \max_M + n, 1)$을 구해내고 이것을 이용하여 y_{Mx}값을 확인한다. 따라서 $E(x_P, n)$값이 중간에서 변경되었다면 구해내는 n값이 달라지므로 상인의 비밀키 값인 x_M값을 생성해 낼 수 없고 결과적으로 전자서명을 통하여 확인된 y_{Mx}값을 생성해 낼 수 없다. 따라서 $E(x_P, n)$값만을 변경시킬 수 없다.

Ⅶ. 지불프로토콜의 효율성

1) 연산과 메시지 비교

각각의 프로토콜들의 연산과 메시지를 비교하면 표 3과 같다.

〔표 3〕 프로토콜의 연산과 메시지 비교
〔Table 3〕 Group operation and Message length

	카운터를 하나씩 증가	카운터를 하나이상 증가	대칭키 암호 알고리즘 사용
①	1회	n회	1회
②	2회	$n+1$회	2회
③	y_M	μ_M	y_M
④	y_P, y_M, r, s	μ, Y, r, s	$y_P, y_M, r, s, E(x_P, n)$

① 상인의 군연산 회수
② 지불자의 군연산 회수
③ 상인으로부터 지불자로 전송되는 메시지
④ 지불자로부터 은행으로 전송되는 메시지

군연산 회수는 카운터를 하나이상 증가시키는 프로토콜에서 금액 n만큼의 연산이 필요하게 되므로 금액이 많아질 경우 연산에 필요한 시간이 많이 걸리게 된다. 그러나 대칭키 암호 알고리즘을 사용하면 금액 n을 지불하지만 카운터를 하나씩 증가할때와 같이 상인측에서 1회, 지불자측에서 2회만 계산하면 된다. ③,④에서 $\mu_M = \{y_{M1}, y_{M2}, \cdots, y_{Mn}\}$, μ는

$\{y_{M1}, y_{M2}, \cdots, y_{Mn}\}$, $\{y_{P1}, y_{P2}, \cdots, y_{Pn}\}$이므로 카운터가 하나이상 증가하는 프로토콜은 카운터가 하나씩 증가하는 프로토콜에 비해 n배의 메시지를 전송하여야 한다. 그러나 대칭키 암호 알고리즘을 사용하면 지불자로부터 은행으로 전송되는 메시지에 $E(x_P, n)$만을 추가하여 전송하면 금액 n의 지불이 가능하다.

2) 메시지전송

제안하는 지불프로토콜은 지불금액에 상관없이 전송되는 메시지의 양이 일정하다. 카운터를 하나씩 증가하는 지불프로토콜과 같은 계산량과 데이터의 크기를 가지고 있으며 다른점은 대칭키 암호알고리즘에서 $E(x_P, n)$의 계산이 추가되고 전송되는 데이터도 $E(x_P, n)$만큼의 크기가 늘어난다는 것이다. 그러나 모든 금액에 대하여 위와같이 일정한 계산과 일정한 데이터의 전송이 이루어지므로 이전의 방식에서처럼 가변적인 계산량과 데이터량에 비하여 효율적이다. 따라서 사용자의 지불장치에 필요한 버퍼의 크기를 줄일 수 있다. 또한 카운터를 하나씩 증가하는 경우 $n=1$로 하여서 $E(x_P, 1)$을 전송하여 구현할 수 있으나 $E(x_P, 1)$을 제외한 전자서명 부분 (y_{Px}, y_{Mx}, r_x, s)만을 전송하도록 하여 전송되는 메시지의 양을 적게 할 수 있다.

3) 데이터저장

사용자의 저장장치에 가지고 있어야 하는 데이터가 적다. 사용자는 저장장치에 σ, *next*, *max*의 세 가지의 변수값만을 저장하고 있으면 된다. 그럼에도 불구하고 사용자는 *next*와 *max*변수를 이용하여 자신의 금액을 확인해 볼 수 있으며 은행의 데이터베이스에 저장해야 할 데이터를 적게 한다. 따라서 저장에 필요한 하드웨어 비용을 줄일 수 있으며 간단한 프로토콜의 구조는 안전성의 증명 및 복잡한 프로토콜로의 확장이 용이하다.

4) 타원곡선 공개키 암호알고리즘의 적용

지불프로토콜은 전자서명과 서명확인 과정을 응용하여 설계되었다. 타원곡선 공개키 암호 알고리즘의 적용은 서명과 확인과정을 빠르게 할 수 있으며 이는 사용자들에게 지불과정의 신속성을 제공할 수 있다. 타원곡선 공개키 암호 알고리즘을 적용은 스마트 카드와 같은 하드웨어의 구현을 쉽게 할 수 있으며, 또한 지불시스템의 안전성 증가를 위한 키길이의 증가가 선형증가율을 가지게 된다.[12,13]

Ⅵ. 결 론

전자지불 프로토콜은 현재 널리 쓰이고 있는 현금, 신용카드와 같이 하드웨어로 구현되었을 때 사용자들에게 편리하고 안전한 지불환경을 제공해야 한다. 사용자의 지불장치는 기억용량과 계산속도에 제한이 있으며 지불시스템을 설계할때는 간단한 프로토콜의 사용, 효율적인 알고리즘의 사용, 효율적인 부가장비의 사용등이 요구된다. 본 논문에서는 간단한 구조를 가지는 지불 프로토콜에 타원곡선 공개키 암호 알고리즘을 적용함으로서 하드웨어 구현시의 비용을 줄일 수 있고 보다 높은 안전성을 가지는 지불시스템을 설계하였다.

향후 컴퓨터 기술의 발전은 이동 컴퓨팅 환경이 보편화될 것으로 예상되고 있으며 이것은 이동단말의 저장량 및 계산량을 최대한 줄임으로서 효율성을 가지게 할 수 있다.[14] 따라서 적은 데이터와 타원곡선을 이용한 지불 프로토콜은 이동 컴퓨팅 환경에서 효과적으로 구현되어 이용될 수 있을 것이다.

참 고 문 헌

[1] D. Chaum, A. Fiat and M. Naor, "Untraceable Electronic Cash," Advances in Cryptology - Proceedings of Crypto '88, pp. 319-327, 1988.

[2] Markus Jakobsson "Mini-Cash : A Minimalistic Approach to E-Commerce," PKC '99, pp. 122-136, 1999.

[3] IEEE P1363, Standard Specifications For Public Key Cryptography, Feb. 13, 1998.

[4] Alfred Menezes, Elliptic Curve Public Key Cryptosystems, Kluwer Academic Publishers, 1993.

[5] Neal Koblitz, A Course in Number Theory and Cryptography, SpringerVerlag, New York, 1987.

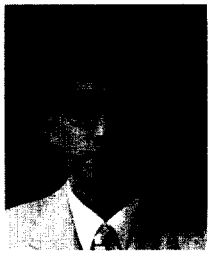
[6] William Stallings, Cryptography and Network Security : Principles and Practice, Prentice Hall, pp. 193-199, 1999.

[7] Alfred Menezes et el., Handbook of applied cryptography, CRC Press, 1997.

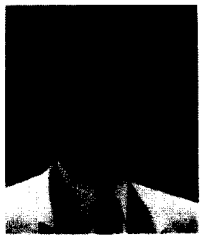
[8] Charles P. Pfleeger Security in Computing, Prentice Hall, 1997.

[9] J.K. Gibson, "Discrete logarithm hash function that is collision free and one way," IEE Proceedings-E.138, pp. 407-410, 1991.
[10] S. Goldwasser, S. Micalli, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," SIAM Journal on Computing, 17, pp. 281-308, 1988.
[11] ANSI X9.62, The Elliptic Curve Digital Signaure Algorithm (ECDSA), working draft, October 1997.
[12] A. Menezes and S.A. Vanstone, "Elliptic curve cryptosystems and their implementations," Journal of Cryptology, 1993.
[13] Certicom, ECC Whitepapers, http: // www.certicom.com, 1999.
[14] G. Horn, B. Preneel, "Authentication and payment in future mobile systems," Proceedings ESORICS '98, LNCS 1485, J.-J. Quisquater, Y. Deswarte, C. Meadows, D. Gollmann, Eds., Springer-Verlag, pp. 277-293, 1998

〈著者紹介〉



이 혁 (Hyurk Lee)
1998년 2월 : 한양대학교 전자계산학과 학사
2000년 2월 : 한양대학교 전자계산학과 석사
〈관심분야〉 전자상거래, 인증프로토콜, 타원곡선 공개키 암호 알고리즘



이 정 규 (Jong-kyu Lee)
1979년 2월 : 한양대학교 전자공학과 학사
1986년 : UCLA 전자공학과 석사
1989년 : UCLA 전자공학과 박사 (컴퓨터 네트워크 전공)
1979년 3월~1984년 5월 : 국방과학연구소 연구원
1989년 3월~1990년 2월 : 삼성전자 종합기술원 정보통신부문 수석연구원
1990년 3월~현재 : 한양대학교 전자계산학과 부교수
1997년 3월~현재 : 한양대학교 전자계산학과 공학기술 연구소 부소장
〈관심분야〉 무선데이타통신, 이동통신, 보안 프로토콜