익명성을 부여한 소액지불 프로토콜에 관한 연구

김해만, 이임영
순천향대학교 공과대학 컴퓨터학부

A Study on Micro Payment System with Anonymity

Hae-Man Kim°, Im-Yeong Lee
Department of Computer Science, College of Engineering,
Soonchunhyang University

요 약

현재 많은 관심을 받고 있는 전자상거래는 앞으로 더욱 활성화되어질 전망이다. 전자상거래는 그 특성상 디지털 데이터와 같은 소액 거래가 쉽게 이루어질 수 있다. 소액 거래는 비록 거래 비용은 적지만 그 응용 분야가 넓기 때문에 중요성을 가진다. 이러한 소액 거래를 위해서는 처리비용이 적어야만 한다. 소액 거래를 위한 많은 프로토콜이 연구되고 있는데, 본 고에서는 지금까지 개발된 대표적인 소액지불 프로토콜을 살펴보고 기존의 소액지불 프로토콜에서 대부분 지원하지 않았던 익명성을 부여한 소액지불 프로토콜을 제안한다.

1. 서론

컴퓨터 보급의 확산과 네트워크를 이용한 컴퓨터 통신의 발달로 인터넷 사용자가 전세계적으로 급증함에 따라 이를 상업적으로 이용한 전자상거래가 많은 주목을 받고 있다.

전자상거래는 원래 미국 국방 예산의 비용을 절감하기 위한 효율적인 운영에 대한 방안으로 구상된 CALS 및 EDI로부터 발전한 개념으로서, 단순한 개인적인 상품 매매의 관점을 벗어나 정부 및 기업의 모든 거래에 필요한 전자적 수단을 통칭한다. 하지만 최근들어 인터넷이 확산되어 각종 상품의 매매가 통신망을 통하여 이루어지면서, 전자상거래의 개념이 통신망을 이용한 거래 행위를 가리키게 되었다.

전자상거래는 지불 방식에 따라 크게 지불브로커 시스템과 전자화폐 시스템으로 나눌 수 있다. 지불브로커 시스템은 독립적인 신용구조를 가지지 않고 신용카드나 은행이 계좌를 이용해 네트워크상에서 지불을 하는 방식이다. 현재 널리 사용되고 있는 신용카드

를 이용할 수 있기 때문에 손쉽게 적용할 수 있다는 장점이 있다. 전자화폐 시스템은 독립적인 신용구조를 가지고 있어서 물품 구입시 은행이나 카드 발행사로부터의 거래 승인이 필요없다. 전자화폐 시스템은 플라스틱 카드 위에 부착된 IC칩을 이용해 오프라인 대금결재에 활용하는 IC카드형 전자화폐와 화폐가치를 디지털 정보의 형태로 발행하여 네트워크를 통한 온라인 대금결재를 가능하도록 한 Network형 전자화폐로 나눌 수 있다.

또한 전자화폐를 지불 금액의 정도에 따라 고액지불 시스템과 소액지불 시스템으로 구분할 수 있다. 고액지불 시스템은 고액의 금액을 안전하게 지불하기 위한 시스템으로 높은 보안성과 안전성이 요구되고 따라서 많은 처리비용이 필요하게 된다. 소액지불 시스템은 1달러 미만의 소액 금액도 거래가 가능하도록 하는 시스템이다. 이를 위해서는 보안성과 안전성의 강도는 다소 낮아지더라도 적은 처리비용이 필요하게 된다.[5]

본 논문에서는 소액지불 시스템에 대해서 기존에 제안된 방식을 분석하고 기존의 소액지불 프로토콜에서 제공되지 않았던 익명성을 보장함으로써 보다 안전하고 효율적인 소액지불 시스템을 제시하고자 한다.

2. 소액지불 시스템

소액 거래는 비록 거래 단위가 작지만 실생활에서 아주 커다란 부분을 차지하고 있다. 또한 소액 거래를 위한 지불 시스템을 구축한다면 많은 새로운 비즈니스를 제공할 수 있게 될 것이다. 인터넷상에서 잡지, 신문, 만화, 음악, 비디오 등의 판매가 가능하게 되고 특별히 관심 있는 부분에 대해서만 거래를 할 수 있기 때문에 사용자 측면에서도 유용하다.

소액지불 시스템을 이루기 위해서는 거래를 위한 처리비용을 최대한 낮춰야 한다. 이를 위한 고려 사항을 살펴보면 다음과 같다.

- 계산량의 감소 : RSA와 같은 처리 속도가 느린 암호 알고리즘의 사용을 최대한 줄이고 해쉬 함수와 같이 처리속도가 빠른 암호기술을 사용하여 처리비용을 줄여야 할 것이다.
- 통신량의 감소 : on-line형 시스템의 경우에는 매 거래마다 broker를 통한 인증을 하기 때문에 통신량이 증가하게 되며 따라서 처리비용도 증가하게 된다. 그러므로 off-line형 시스템을 이용하여 통신량을 줄여야 할 것이다.
- 신용 risk의 감소 : 신용카드거래와 같은 후불 방식에서는 고객의 신용 risk가 생기게 되고, 이는 수수료를 높이는 원인이 된다. 따라서 선불 방식과 같은 방법으로 고객의 신용 risk를 없애면 처리비용을 줄일 수 있을 것이다.
- DB의 감소 : 데이터의 인증이나 법률적인 논쟁 등의 경우를 위해 일정한 데이터를 DB로 관리해야 하는데, 관리해야할 데이터를 줄여 처리비용은 줄여야할 것이다.

현재 소액 거래를 위한 많은 프로토콜이 제안되고 있는데, 대표적인 것으로 Millicent[1],

MPTP(Micro Payment Transfer Protocol)[2], Mini-Pay[3] 등이 있다.

2.1 Millicent

1997년 3월 11일, Digital Equipment Corporation에서는 인터넷상에서 진정한 의미의 소액 상거래를 이루기 위해 Millicent 계획을 발표하였다.

가. 참여 주체

- Customer : Broker로부터 scrip을 구입한 후 vendor와 거래한다.
- Vendor : Customer로부터 scrip을 받고 서비스와 거스름 scrip을 제공한다.
- Broker : Customer와 broker의 계정을 관리하고 scrip과 실제 돈의 거래를 다룬다.

나. Scrip

- Millicent에서는 물건을 구입하기 위해 scrip이라고 전자 현금을 사용한다. scrip은 broker와 vendor가 발행을 할 수 있으며, 특정한 vendor에게만 사용이 가능하다.
- Scrip의 구조

Vendor	Value	ID#	Cust_ID#	Expires	Props	→ Scrip Body
H(Scrip Body ll master_scrip_secret)						→ Certificate

  · Vendor : scrip을 발행한 상거래 서버의 ID.
  · Value : scrip의 화폐 가치.
  · ID# : scrip의 이중사용을 막기 위한 scrip의 유일한 번호.
  · Cust_ID# : scrip을 사용하는 사용자의 ID.
  · Expires : scrip의 유효기간.
  · Pros : 기타 데이터.
  · Certificate : scrip에 대한 변조 여부의 확인을 위한 인증서

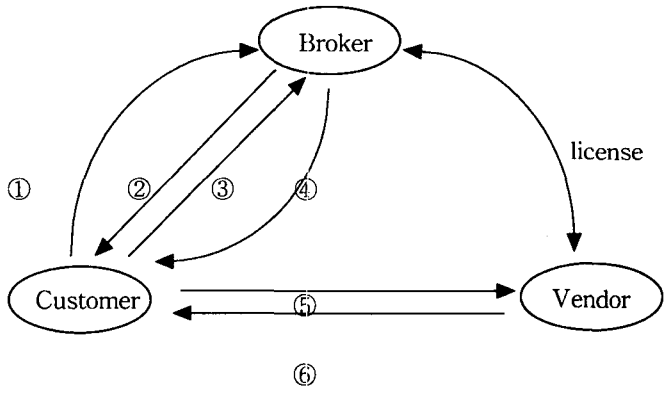
다. 보안 메커니즘

- master_scrip_secret 사용
  · scrip의 유효성을 확인하기 위한 인증서인 certificate를 생성하기 위해 사용되어진다.
  · certificate 생성방법은 DB에 있는 master_scrip_secret들 중 하나를 scrip_body부분의 ID#를 이용하여 선택한 후 scrip_body부분과 함께 해쉬함으로써 생성된다.
- master_customer_secret 사용
  · customer_secret를 생성하는데 사용되어진다.
  · customer_secret의 생성방법은 scrip_body부분의 Cust_ID#를 이용하여 DB에 있는

master_customer_secret들 중 하나를 선택해서 Cust_ID#와 함께 해쉬함으로써 생성

라. Millicent 프로토콜

Millicent 프로토콜의 과정은 다음과 같다.



[그림 1] Millicent 프로토콜 흐름도

(1) ①, ② 단계(customer_secret, broker_scrip구입)

안전한 채널상에서 broker_scrip과 사용자의 비밀키인 customer_secret을 broker로부터 받는다. Customer_secret은 거래가 종료될 때까지 계속하여 재사용되기 때문에 사용자는 이 비밀키의 관리에 신중을 기해야 한다.

(2) ③, ④ 단계 (vendor_scrip 구입)

사용자가 customer_secret과 broker_scrip을 이용하여 broker에게 거래하고자 하는 vendor의 scrip을 구입 요청한다. Broker는 request문을 확인하고 해당 vendor_scrip을 발행한다. 그리고 사용자가 제시한 broker_scrip과 broker가 발행한 vendor_scrip의 차는 다시 새로운 거스름 broker_scrip을 발행함으로써 해결한다. Broker는 certificate를 이용해서 사용자로부터 받은 broker_scrip의 유효성을 확인하고, ID#필드의 일련 번호를 통해서 이중 사용(Double Spending)을 확인한다.

(3) ⑤, ⑥ 단계 (상품 구입)

Customer는 vendor에게 request문을 보낸다. Request문은 (request || scrip || H(request || scrip || customer_secret))로 구성이 되어 있다. 제 3자는 customer_secret를 알 수 없으므로 부정한 request문을 만드는 것이 불가능하다. Request문에 대한 유효성 검사를 한 후에 vendor는 사용자가 선택한 상품의 배송과 함께 거스름 scrip을 만들어 사용자에게 보낸다.

마. 프로토콜 분석

해쉬를 통하여 메시지를 인증함으로써 scrip이나 request문을 제 3자가 위조하거나 변경할 수 없다. 또한 vendor에서 ID#를 점검함으로써 이중사용을 방지할 수 있다. 그리고 scrip의 구입이 선불 방식이므로 customer의 신용이 필요 없다는 장점이 있다.

반면에 Millicent는 익명성이 보장되지 않고, 거래 시 vendor의 거스름돈이 필요하게 된다. 또한 vendor가 부정한 동전을 생성할 수 있는 문제가 있다

2.2 MPTP (Micro Payment Transfer Protocol)

MPTP는 1995년에 나온 W3C Working Draft로써 현재 새로운 버전은 나오지 않고 있지만 새로운 token 생성 방식의 사용과 프로토콜에 대하여 많은 관심을 받고 있다.

가. 참여 주체

MPTP도 Millicent와 마찬가지로 참여하는 주체는 customer, vendor, broker로 구성되어 있다.

나. 지불 메커니즘

지불 명령은 크게 지불 authority와 지불 token으로 나누어지는데 지불 authority는 지불에 필요한 paychain_root 값, 식별자 등 지불에 필요한 정보를 디지털 서명한다. 지불 token은 연쇄 해쉬 함수를 이용하여 금액을 결정한다.

여기에서 연쇄 해쉬 함수란 token을 인증하기 위해 사용되는데, 우선 customer는 랜덤한 w_n 값을 선택한다. 그리고나서 w_i = h(w_i+1)를 계산함으로써 일련의 지불 토큰(paychain) w_0, w_1, · · ·, w_n을 계산한다. 이 때, 금액은 해쉬 횟수에 의해서 결정된다. 여기서 paychain을 생성하는 최초의 root값(w_0)을 paychain_root라고 한다. h는 암호학적으로 안전한 MD5와 같은 one_way 해쉬 함수를 나타낸다.

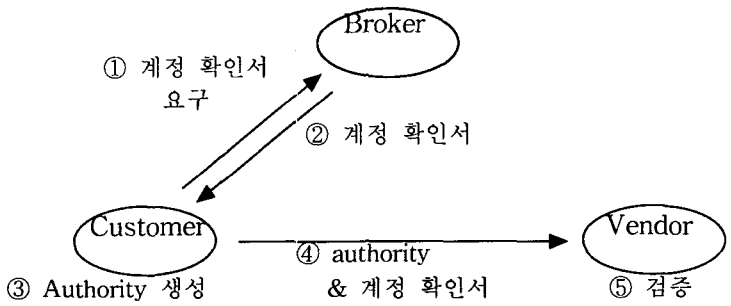
다. MPTP 프로토콜

지불 처리는 Session Establishment 단계와 Payment Transfer 단계로 나눌 수 있다.

- Session Establishment 단계

Session Establishment 단계는 지불 처리를 하기 위해 미리 지불 처리에 필요한 정보(authority, 계정 확인서)를 broker, customer, vendor가 서로 주고받는 단계이다.

이 단계는 거래를 하기 전에 한번만 수행을 하고 그 후에는 이 정보를 기반으로 계속적인 거래를 할 수 있다.



[그림 2] MPTP 프로토콜 흐름도 : Session Establishment 단계

(1) ①, ② 단계

Customer는 broker에게 계정 확인서를 요구하고 계정 확인서를 받는 단계이다. 만약 broker가 공개키 서명을 사용하여 계정 확인서를 생성하면, vendor는 broker의 서명을 확인함으로써 계정 확인서를 인증한다. 만약 broker가 대칭키 서명을 사용할 경우 vendor는 broker에게 계정 조회를 의뢰한다.

(2) ③ 단계

Customer가 authority를 생성하는 단계이다. Authority는 authority를 인증할 수 있는 정보와 paychain을 생성할 수 있는 정보를 포함한다.

(3) ④ 단계

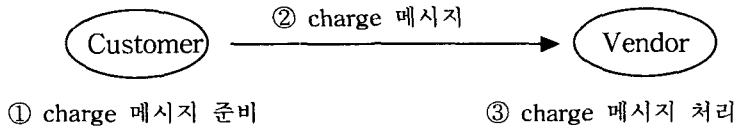
Customer는 authority에 서명을 해서 계정 확인서와 함께 vendor에게 전송한다.

(4) ⑤ 단계

Vendor는 전송 받은 authority와 계정 확인서를 서명을 통하여 검증하는 단계이다.

- Payment Transfer

지불 전송하는 단계로써 세션 설정이 이루어진 후 계속해서 지불 처리를 할 수 있다.



[그림 3] MPTP 프로토콜 흐름도 : Payment Transfer 단계

(1) ① 단계

Customer가 charge 메시지를 준비하는 단계로써, charge 메시지는 금액을 결정할 수 있는 정보인 authority_id와 payword 등의 정보를 포함한다.

Vendor_id와 일치하는 authority 정보를 검색해서 해당 vendor에 맞는 paychain_root값을 적절한 횟수만큼 연쇄 해쉬함으로써 요구된 금액에 맞는 payword를 결정한다.

(2) ② 단계

Charge 메시지를 전송한다.

(3) ③ 단계

Charge 메시지를 점검하는 단계이다. Authority_id를 이용해서 해당 paychain_root를 선택한 후, 이것을 사용하여 연쇄 해쉬 함수를 수행함으로써 payword를 확인한다.

라. 프로토콜 분석

동전을 생성할 수 있는 정보(paychain_root)는 안전한 서명 방식으로 설정되어지기 때문에, 그 정보를 모르는 제 3자는 부정한 동전을 생성할 수 없고 유일한 authority 식별자의 점검으로 이중사용 방지할 수 있다. 또한 customer가 거래 금액에 맞는 동전을 생성하므로 broker의 부하가 감소하고, vendor의 거스름돈이 필요 없다는 장점이 있다.

반면에 MPTP는 익명성이 보장되지 않고, customer의 신용 리스크가 생기고, vendor의 부정한 동전 생성이 가능하다는 단점이 있다.

2.3 Mini-Pay

Mini-Pay는 IBM에서 소액지불을 위해 개발한 프로토콜이다.

가. 참여 주체

- Buyer : 상품을 구입하는 사람으로 MiniPay Wallet이 제공되어진다.
- Seller : 서비스 제공자이다.
- IAP(Internet Access Provider) : buyer의 billing system이다.
- ISP(Internet Service Provider) 또는 bank : seller의 billing system이다.

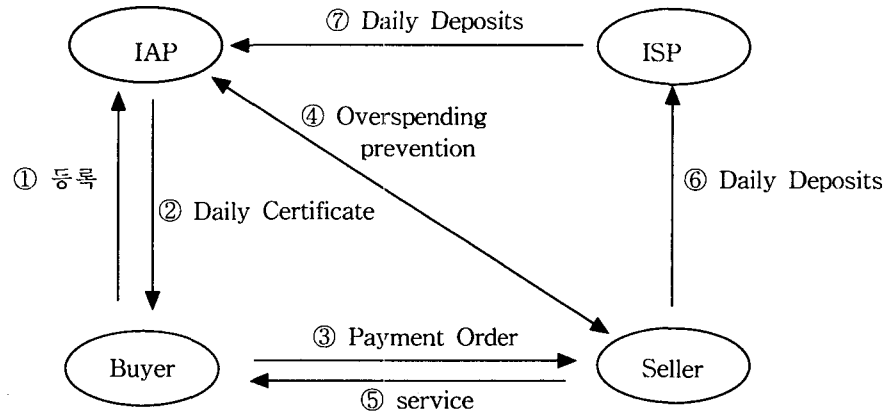
나. 메커니즘

- 하나의 기관이 ISP와 IAP의 기능을 모두 수행할 수도 있다.
- IAP와 ISP의 연결은 중간에 하나 또는 두개의 은행을 통해서 할 수도 있고 또는 IAP와 ISP를 직접 연결할 수도 있다.

다. *Mini-Pay* 프로토콜

Mini-Pay의 프로토콜의 과정은 다음과 같다.

Mini-Pay의 프로토콜의 과정은 다음과 같다.



[그림 4] Mini-Pay 프로토콜 흐름도

(1) ① 단계 (등록)

계정을 설정하고 이 계정에 사용될 공개키를 확인하는 단계이다. buyer는 IAP로부터 비밀값 code_B를 받고 공개키(PUB_B)와 개인키(PRIV_B)를 생성한 후, IAP에게 PUB_B, S_B(Reg_req, H(code_B, salt1, PUB_B, acct_B, time)), salt1을 전송하여 등록을 요구하고, IAP는 buyer에게 S_IAP(Reg_res, OK/fail_code, acct_B, PUB_B, time, fees)를 전송함으로써 등록 요구에 대한 응답을 한다.

(2) ② 단계 (Daily Certificate)

이 단계는 매일 시작할 때 한번 실행을 하는 단계로써 전날의 전체 잔액을 바꾸고, IAP는 buyer의 wallet에 offline_limit를 포함하는 daily certificate인 Daily_B를 제공한다. offline_limit는 IAP에게 on-line 확인을 하지 않고 날마다 buyer가 off-line으로 거래할 수 있는 최대 구매 금액을 나타낸다. Daily_B의 형식은 다음과 같다.

Daily_B=S_IAP(Daily_res, OK/fail_code, acct_B, PUB_b, time, reco_offline_lim, H(total_lim, salt, real_bal))

(3) ③ 단계 (Payment Order)

Buyer는 seller에게 payment order인 Pay_Order 생성하여 Daily_B와 함께 전송하는 단계이다. Pay_Order의 형식은 다음과 같다.

Pay_Order=S_B(Order, amount, day_total, acct_B, time, URL, acct_S)

데이터를 받은 Seller는 Pay_Order의 IAP 서명 확인하여 buyer를 인증하고 공개키를

않은 경우에는 곧바로 (5)단계를 수행하고 초과한 경우에는 (4)단계를 수행한다.

(4) ④ 단계 (Overspending prevention)

Seller가 IAP에게 Pay_Order를 보내면, IAP는 seller에게 추가적인 금액의 사용을 승인하거나 거절하는 메시지를 전송한다. 이 단계는 상대적으로 드물 것이다.

Buyer는 overspending을 예상하고 직접 IAP에 payment order를 보낼 수 있다.

(5) ⑤ 단계 (Service)

Seller는 요구되어진 정보나 서비스를 buyer에게 제공

(6) ⑥ 단계 (Daily Deposits)

Seller는 payment order를 모은 후 time과 함께 서명한 후 ISP에게 전송을 하고 결제를 요구하면 ISP는 seller에게 결제한 결과를 전송한다.

(7) ⑦ 단계 (Daily Deposits)

(6)단계와 유사한 동작으로써, ISP는 payment order를 모은 후 time과 함께 서명한 후 해당 IAP에게 전송하고 결제를 한다.

라. 프로토콜 분석

서명을 이용함으로써 안전한 메시지 인증은 가능하지만 계산량이 다소 많아질 것이다. 오버헤드를 줄이기 위해서 여러 데이터를 한꺼번에 처리하도록 하였고, Daily Certificate의 offline_limit를 통하여 off-line 거래가 가능하도록 함으로써 통신량을 줄일 수 있도록 하였다. 하지만 buyer가 offlin-limit 금액으로 여러 seller와 거래한다면 buyer는 자신이 사용할 수 있는 금액의 범위를 넘어서 사용하는 overspending의 문제가 있다. 또한 후불 방식으로 IAP가 신용 리스크를 부담하게 된다.

3. 새로운 소액 지불 시스템

기존의 방식에서는 대부분 익명성을 제공하지 않고 있다. 익명성은 개인의 프라이버시를 위해 요구되는 기본적인 조건이라 할 수 있다. 본 제안 방식에서는 신뢰할 수 있는 기관인 CA를 통하여 간단한 방식으로 익명성을 제공하도록 하였다. 또한 불법적인 사용자에 대해서는 CA와 broker의 협조를 통해서 추적이 가능한 방식이다.

가. 참여 주체

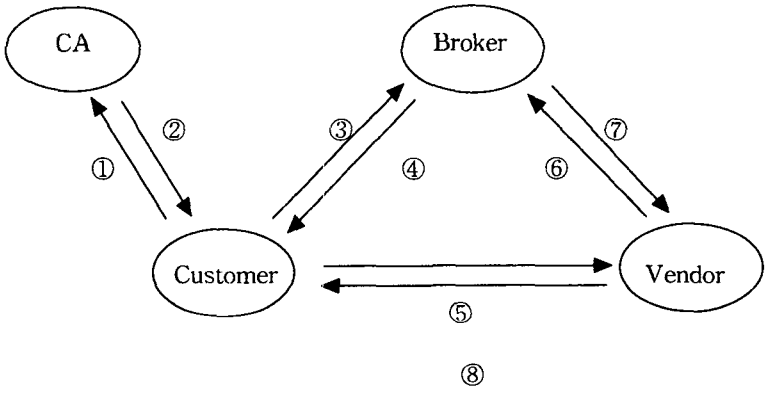
- Customer : broker에게 입금을 한 후 vendor와 거래한다.
- Vendor : customer에게 서비스를 제공한다.
- Broker : customer와 vendor 사이의 실제 돈의 흐름을 관리하고 거래를 인증한다.
- CA : 신뢰된 기관으로써 customer의 신원을 확인하고 PID(pseudo_ID)를 제공한다.

나. 지불 메커니즘

- 기존의 지불브로커 시스템의 지불 방식은 대부분 직불이나 후불 방식인데, 이러한 방식에서는 익명성을 제공할 수 없다. 본 방식에서는 익명성을 제공하기 위해 선불 방식을 사용하였다.
- CA와 broker는 비밀값 BCA_key를 공유한다. 이 비밀값은 broker가 customer의 PID에 대해 인증할 수 있도록 한다.
- Broker와 customer는 비밀값 BCus_key를 공유한다. 이 비밀값은 customer가 전송하는 데이터에 대한 인증을 제공한다.

다. 지불 프로토콜

제안 방식의 프로토콜의 과정은 다음과 같다.



[그림 5] 제안 방식 프로토콜 흐름도

(1) ① 단계 (PID 요구)
익명성을 유지하기 위한 PID를 요구하는 단계이다. Customer는 CA에게 자신의 실제 신원정보를 전송하고 PID를 요구한다.

(2) ② 단계 (PID 전송)
Customer에 신원을 확인하고 PID와 PID 인증서를 제공한다.
PID 인증서의 형식은 다음과 같다.

$$h(PID \parallel BCA\_key)$$

여기서 BCA_key는 broker와 CA가 공유한 비밀값이므로 제 3자가 조작할 수 없다.

(3) ③ 단계 (사용할 금액을 선불 입금)
Customer는 PID와 PID 인증서를 이용하여 자신이 사용할 적당한 금액을 선불로 입금을 한다.

(4) ④ 단계 (지불 확인과 비밀값 발급)

Broker는 customer로부터 받은 PID와 BCA_key로 해쉬함으로써 PID 인증서를 검증하고 지불된 금액과 customer와 broker의 비밀 값 BCus_key 전송한다. BCus_key는 안전성을 위한 중요한 데이터이므로 공개키 암호와 같은 안전한 방식으로 전송한다.

전송하는 데이터의 형식은 다음과 같다.

Ec(입금 금액 || BCus_key)

(5) ⑤ 단계 (상품 구입)

Customer는 vendor에 접속하여 원하는 상품을 선택하고 주문서 C_order 작성한 후 전송한다. 주문서의 형식은 다음과 같다.

C_order = 금액, PID, Vend_ID, S_N, h(금액 || PID || Vend_ID || S_N || BCA_key)

여기서 S_N은 순차적으로 증가하는 serial number로서 주문서에 대한 식별자 역할을 함으로써 vendor의 부정한 이중 사용을 막기 위해 사용된다.

(6) ⑥ 단계 (Broker에게 검증 및 결제 요구)

Vendor는 customer로부터 받은 C_order와 자신의 계정을 broker에게 전송하여 C_order에 대한 검증과 상품 금액에 대한 이체를 요구한다.

(7) ⑦ 단계 (검증)

Broker는 BCA_key를 이용한 해쉬를 통하여 C_order를 검증하고 검증 결과 B_respond를 vendor에게 전송한다. 검증이 올바르게 이루어졌다면, vendor의 계정에 해당 금액만큼 이체를 하게 된다. B_respond의 형식은 다음과 같다.

B_respond = accept/reject, h(accept/reject || S_N || BCus_key)

(8) ⑧ 단계 (인증 결과 및 상품 전송)

broker로부터 받은 B_respond와 결과가 accept일 경우 상품을 전송한다.

라. 프로토콜 분석

(1), (2) 단계는 거래를 시작하기 전에 한번만 수행하면 되고 (3), (4) 단계는 입금한 금액을 다 사용했을 경우에만 수행하면 된다. 사용된 암호 알고리즘을 살펴보면 대부분 속도가 빠른 해쉬 알고리즘을 사용함으로써 처리 비용을 줄일 수 있도록 하였다. (공개키 암호 알고리즘은 비밀값 BCus_key를 공유할 때 한번만 사용됨)

본 방식에서는 선불 방식을 채택하여 CA를 통한 customer의 익명성을 제공할 수 있고 불법적인 사용과 같은 추적이 필요한 경우에는 CA와 broker의 협조를 통하여 신원

을 조회할 수 있는 부분적인 추적성을 가진다. Customer는 PID에 대한 인증서를 broker에게 전송해야 하기 때문에 자신의 신원을 속일 수 없다.

안전성 측면에서 살펴보면, broker와 customer의 비밀키 BCus_key를 사용한 해쉬를 통하여 메시지를 인증함으로써 이 값을 모르는 제 3자나 vendor는 부정한 메시지를 만들 수 없다. 또한 vendor가 부정을 저질을 경우를 생각해보면, vendor가 customer로부터 받은 주문서 C_order를 나중에 다시 사용하는 이중 사용의 경우와 broker의 B_respond에 대해 accept된 것을 reject된 것으로 속여 부당하게 상품 전송을 않하는 경우를 생각할 수 있다. 첫번째 경우에는 C_order에 순차적으로 변하는 S_N을 사용함으로써 이중 사용을 방지하고, 두번째 경우에는 B_respond에 BCus_key를 사용하여 메시지를 인증함으로써 vendor가 인증 결과인 B_respond를 속일 수 없도록 한다.

4. 결론

앞으로 전자상거래는 더욱 활성화될 전망이고 그 중에서 소액 거래는 전자상거래의 특성상 그 응용범위가 넓기 때문에 더욱 중요하게 부각될 것이다. 소액 거래를 위해서는 처리 비용을 최대한 줄여야 하는데 가장 중요한 요소가 통신량과 계산량이다. 통신량을 줄이기 위해서는 off-line형 시스템이 적합한 반면 vendor측에서 전자화폐에 대한 검증을 하는 과정이 필요하게 된다. 계산량을 줄이기 위해서는 속도가 느린 RSA와 같은 공개키 암호 알고리즘 대신에 속도가 빠른 해쉬함수를 이용하는 것이 좋다. 본 고에서는 소액지불을 위해 기존에 제시된 방식을 분석하고 기존에 제공되지 않았던 익명성을 간단한 방법으로 제공할 수 있는 방식을 제안하였다.

기존의 방식과 제안 방식을 비교해보면 다음과 같다.

[표 1] 각 소액지불 프로토콜의 비교

	Millicent	MPTP	Mini-Pay	제안 방식
지불 방식에 의한 분류	전자화폐 시스템 (off-line)	전자화폐 시스템 (off-line)	지불브로커 시스템 (off-line, on-line)	지불브로커 시스템 (on-line)
지불 시점	선불 방식	후불 방식	후불 방식	선불 방식
신용 리스크	×	○	○	×
안전성	○	○	○	○
이중사용방지 (overspending)	○	○	(×)	○
익명성	×	×	×	○
vendor의 부정방지	×	×	○	○
거스름돈	×	○	×	×

통신량과 계산량을 비교해보면, 본 제안 방식은 기존의 방식과 비교하여 통신량은 많은

반면 계산량이 적은 특성을 가지고 있다.

소액지불 시스템에서는 처리 비용을 줄이기 위해 보안 강도가 다소 낮아지게 되는데, 처리 비용을 줄이면서 보안 강도를 높일 수 있다면 유용한 소액지불 시스템이 될 것이다.

[본 연구는 한국과학재단 특정기초연구과제 (과제번호 : 97-01-00-06-01-3) 연구비 지원에 의해 수행되었음]

참 고 문 헌

[1] Steve Glassman, Mark Manasse, Mart Abadi, Paul Gauthier, Patrick Sobalvarro, "The Millicent Protocol for Inexpensive Electronic Commerce", http://HTTP.CS.Berkeley.EDU/~gauthier/millicent/millicent.html

[2] Phillip M. Hallam-Baker, "Micro Payment Transfer Protocol(MPTP) Version 0.1", W3C Working Draft, http://www.w3.mag.keio.ac.jp/TR/WD-mptp-951122, 1995

[3] Amir Herzberg, Hilik Yochai, "Mini-Pay : Charging per Click on the Web" http://www6.nttlabs.com/HyperNews/get/PAPER99.html

[4] Petri Aukia, Jean-Baptiste Lehmann, "Mechanisms in electronic commerce using micropayments", http://studwww.eurecom.fr/~lehmann/study/

[5] 김해만, 채승철, 이임영, "새로운 소액지불에 관한 연구", 한국멀티미디어학회 춘계학술발표논문집, pp181-186, 1998