새로운 소액지불 시스템에 관한 연구

김해만°, 채승철, 이임영
순천향 대학교 공과대학 컴퓨터학부

A Study on New Micro Payment System

Hae-Man Kim°, Seung-Chul Chae, Im-Yeong Lee
Department of Computer Science, College of Engineering,
Soonchungyang University

요 약

컴퓨터와 네트워크 등의 발달로 이를 이용한 전자상거래에 대한 관심이 커져가고 있다. 전자상거래 중에서 고액에 대한 거래보다는 소액 거래가 더욱 활성화 될 것이라고 예상된다. 본 논문에서는 그러한 소액 거래를 위한 기존에 제안된 프로토콜을 분석하고 보다 안전하고 효율적인 소액지불 시스템을 제안하고자 한다.

1. 서론

컴퓨터 보급의 확산과 네트워크를 이용한 컴퓨터 통신의 발달로 인터넷 사용자가 전세계적으로 급증함에 따라 이를 상업적으로 이용하기 위한 시도가 증가하고 있다. 그 중에서 많은 주목을 받고 있는 분야가 바로 전자상거래(Electronic Commerce)이다.

전자상거래에 대한 개념 정의는 일정하지 않으나, 일반적으로 기업, 정부기관과 같은 독립된 조직간 또는 조직과 개인간에 다양한 전자적 매체를 이용하여 상품이나 용역을 교환하는 것을 의미한다.

이러한 전자상거래가 발전할 수 있었던 배경을 살펴보면 다음과 같다[4].

- 인터넷 이용자수의 급증
  : PC의 보급, 통신 환경의 구축, 그리고 통신 프로토콜 등의 발달로 이용자수가 급증하게 되었다.
- 보안기술의 발전
  : 안전한 전자상거래를 위한 보안 기술의 발달은 전자상거래의 발전을 위한 중요한 요소가 된다.
- 쇼핑몰의 구축 용이
  : 상품을 홍보하고 상품에 대한 정보를 얻을 수 있는 다양한 멀티미디어 기술의 발전으로 상품에 대한 효과적인 홍보가 가능하게 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. 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[5]. 고액지불 시스템은 고액의 금액을 안전하게 지불하기 위한 시스템으로 높은 보안성과 안전성이 요구되고 따라서 많은 유지비용이 필요하게 된다. 소액지불 시스템은 1달러 미만의 소액 금액도 거래가 가능하도록 하는 시스템이다. 이를 위해서는 적은 유지비용이 필요한데 따라서 보안성과 안전성의 강도는 다소 낮아질 것이다.

본 논문에서는 소액지불 시스템에 대해서 기존에 제안된 방식을 분석하고 보다 안전하고 효율적인 소액지불 시스템을 제시하고자 한다.

2. 소액지불 시스템

1996년 3월의 분석에 따르면, 1992년에 사용된 10달러 미만의 현금 거래액은 1조 8천억 달러의 엄청난 금액이다.(이 금액은 신용카드 사용액 420억 달러의 4배에 해당하는 금액이다.) 이처럼 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 전체 출판물에 대해 지불을 하는 것이 아니라, 단지 한 페이지 정도나 전체보다 작은 특별히 관심 있는 부분에 대해서만 거래를 할 수 있기 때문에 사용자 측면에서도 매우 유용하다.

이러한 소액 거래를 위해서는 거래를 위한 처리비용이 적어야 하는데, 전자상거래에서 높은 보안성을 유지하기 위해 사용되는 암호(특히 RSA)는 계산량이 많아서 속도가 느리며 시스템의 부하도 크다. 또한 신용카드를 이용한 지불 시스템은 많은 통신량과 높은 수수료를 필요하게 된다. 이것은 거래를 위한 처리비용이 비싸다는 것을 의미한다. 따라서 1달러 미만과 같은 소액 거래에 적합하지 않기 때문에 전자상거래를 위한 최소한의 보안을 유지하고 처리비용을 줄여서 소액 거래가 가능하도록 하는 전자지불 시스템을 만드는 것이 소액지불 시스템이다.

현재 소액 거래를 위한 많은 프로토콜이 제안되고 있는데, 대표적인 것으로 Millicent[1], MPTP[2](Micro Payment Transfer Protocol), MiMi[3](Micro Mint) 등이 있다.

2.1 기존의 소액지불 시스템

2.1.1 Millicent

Millicent에서 참여하는 주체는 broker, vendor, customer로 구성되어 있다. Broker는 scrip 생성과 계정을 관리하고, vendor는 scrip을 받고 정보나 서비스를 제공하고, customer는 scrip을 구입해서 상품 구입을 한다.

여기서 scrip은 거래를 위한 화폐의 가치를 나타낸다. Scrip은 broker나 vendor에서 생성하게 되는데, 특정 vendor에게만 사용할 수 있는 화폐가치를 나타낸다. Scrip의 안전성은 해쉬 함수를 통해서 이루어진다. 수행 속도가 빠른 해쉬 함수만을 사용하기 때문에 효율적인 처리가 가능하다.

가. Scrip의 구조

Scrip의 구조는 다음과 같다.

Vendor	Value	ID#	Cust_ID#	Expires	Props
H(scrip_body ll master_customer_secret)					

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

나. 보안 메커니즘

보안을 위한 메커니즘으로는 master_scrip_secret과 master_customer_secret이 사용된다.

Master_scrip_secret은 scrip의 certificate를 생성하기 위해 사용되어지고, master_customer_secret은 customer_secret를 생성하는데 사용되어진다.

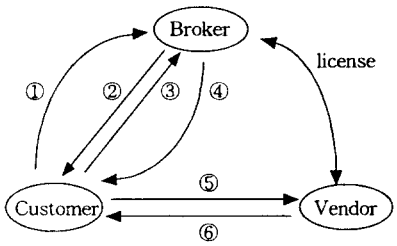
Certificate는 scrip의 유효성을 확인하기 위한 인증서이다. 생성방법은 scrip_body부분의 ID#를 이용하여 DB에 있는 master_scrip_secret들 중 하나를 선택해서 scrip_body부분과 함께 해쉬를 함으로써 생성된다.

Customer_secret은 broker가 customer에게 주는 비밀 정보로써 customer는 request문에 비밀키를 사용하므로써 소유권을 증명하는데 사용한다. 따라서 이 비밀키를 모르는 제 3자는 request문을 변경하거나 생성할 수 없다. Customer_secret의 생성방법은 scrip_body부분의 Cust_ID#를 이용하여 DB에 있는 master_customer_secret들 중 하나를 선택해서 Cust_ID#와 함께 해쉬를 함으로써 생성된다.

Vendor와 broker에서 certificate를 확인 절차는 동일한 방법으로 certificate를 생성해서 customer가 전송한 certificate와 비교함으로써 이루어진다.

다. Millicent 프로토콜

Millicent 프로토콜의 과정은 다음과 같다.



[그림 1] Millicent 프로토콜

1) ①, ② 단계(customr_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.

Request문의 확인 과정을 살펴보면 vendor는 먼저 scrip 안에 포함되어 있는 expires 필드를 이용하여 유효기간을 확인한다. 그 다음에 vendor는 request문의 certificate를 이용하여 request문의 유효성을 검사한다.

Request문에 대한 유효성 검사가 끝나면 vendor는 사용자가 선택한 상품의 배송과 함께 거스름 scrip을 만들어 사용자에게 보낸다.

2.1.2. MPTP (Micro Payment Transfer Protocol)

MPTP도 Millicent와 마찬가지로 참여하는 주체는 broker, vendor, Custmer로 구성되어 있고 거래를 위한 알고리즘으로 해쉬 함수를 사용한다.

그러나 동전을 생성하는 방법 및 처리 과정에서 많은 차이점이 있다.

가. 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.

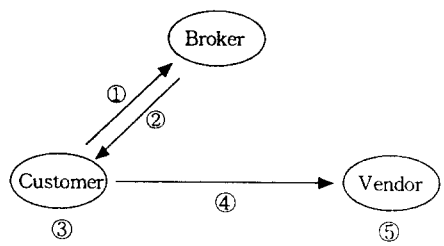
나. Payment Flow

지불 처리는 크게 Session Establishment(세션 설정) 단계와 Payment Transfer(지불 전송) 단계로 나눌 수 있다.

1) Session Establishment

세션 설정 단계는 지불 처리를 하기 위해 미리 지불 처리에 필요한 정보(authority, 계정 확인서)를 broker, customer, vendor가 서로 주고 받는 단계이다.

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



[그림2] MPTP 세션 설정 과정

(1) ①, ② 단계

Customer는 broker에게 계정 확인서를 요구하고 계정 확인서를 받는 단계이다.

만약 broker가 공개키를 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 포함.

(3) ④ 단계

Customer는 authority에 서명을 해서 계정 확인서와 함께 안전하게 vendor에게 전송한다.

(4) ⑤ 단계

Vendor는 전송받은 authority와 계정 확인서를 점검하는 단계이다.

점검 단계는 다음과 같다.

i) authority 유효기간 점검
ii) 이중사용여부 점검 (authority_id 점검)
iii) authority_id 추가
iv) 계정 확인서의 서명 확인
v) authority의 서명 확인
vi) authority를 online 파일에 추가

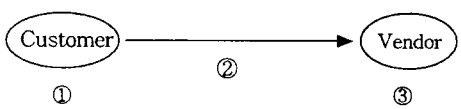
2) Payment Transfer

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

Customer는 vendor에 접속해서 원하는 상품을 선택한 후에 그 상품 금액에 해당하는 token을 생성하여 charge 메시지를 만든 후에 이것을 vendor에게 전송하고 상품을 전송 받는다.

Vendor는 이 메시지를 broker에게 전송하고 자신의 계정에 입금을 요구하면 broker는 유효성을 확인한 후 customer의 계정에서 vendor의 계정으로 입금을 한다.

이 과정을 단계별로 살펴보면 다음과 같다.



[그림3] 지불 전송 과정

(1) ① 단계

Customer는 charge 메시지를 준비하는 단계로써 금액을 결정할 수 있는 authority_id와 payword 등의 정보를 포함한다.

Vendor_id와 일치하는 authority 정보를 검색해서 해당 vendor에 맞는 paychain_root값을 적절한 횟수만큼 연쇄 해쉬함으로써 요구된 금액과 일치하는 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 확인한 후 세션 레코드에 payword정보와 증가량을 갱신한다.

2.1.3 기존 방식 분석

가. Millicent

Millicent 프로토콜은 해쉬 함수를 이용해서 메시지를 인증하므로 scrip이나 request문을 제 3자가 위조하거나 변경할 수 없다. 또한 vendor에서 ID#를 점검함으로써 이중사용을 방지할 수 있다. 그리고 scrip의 구입이 선불 방식이므로 customer의 신용이 필요없다는 장점이 있다.

반면에 Millicent는 다음과 같은 문제점이 있다.

· 양도성 및 익명성이 보장되지 않는다.
· 거스름돈이 필요하다.
· Vendor가 부정한 동전을 생성할 수 있다.

나. MPTP

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 다음과 같은 문제점이 있다.

· 익명성이 보장되지 않는다.
· Customer의 신용이 필요하다. (후불 방식)
· Vendor의 부정한 동전 생성 가능하다.

3. 새로운 소액 지불 시스템

위에서 살펴본 Millicent와 MPTP는 장·단점이 서로 보완적이다. 따라서 각각의 단점을 보완하고 장점을 살린다면 이상적인 프로토콜이 될 것이다.

제안 방식에서는 MPTP의 token 생성 방식에 선불 처리가 가능하도록 함으로써 거스름돈도 필요 없고, 신용 문제도 없는 프로토콜을 제안하고자 한다. 또한 Millicent와 MPTP 모두에게 문제가 되고 있는 vendor의 부정한 동전 생성을 방지하도록 한다.

3.1 지불 메커니즘

가. 구성요소

참여하는 주체는 앞에서 설명한 방식과 마찬가지로 broker, vendor, customer로 구성되어져 있다.

나. Paychain의 생성 방법

여러 금액 단위로 paychain_root를 구분하여 paychain을 생성한다.

ex) 1회 거래시 사용한도금액이 10,000이하일 경우

- 1 단위 : $w_i_1 = h(w_i_1+1)$를 계산
- 10 단위 : $w_i_{10} = h(w_i_{10}+1)$를 계산
- 100 단위 : $w_i_{100} = h(w_i_{100}+1)$를 계산
- 1000 단위 : $w_i_{1000} = h(w_i_{1000}+1)$를 계산

다. Token의 구조

Vendor _ID#	Cust _ID#	payword	paychain _new_root	hash _count	props
h(token_body || secret_key)					

- Token_body
  - Vendor_ID# : 거래할 상거래 서버의 ID
  - Cust_ID# : token을 생성하는 사용자의 ID
  - payword : paychain_new_root를 해당금액만큼 해쉬한 값
  - paychain_new_root : payword를 생성하는 root 값으로 해쉬할 때마다 해쉬값이 새로운 root 값이 된다. 최초의 사용시 paychain_new_root 값은 paychain_root값이 된다.
  - hash_count : 최초의 paychain_root값으로부터 paychain_new_root까지의 해쉬 횟수이다.
  - props : 기타 데이터
- Certificate : token_body와 broker로부터 받은 비밀키인 secret_key를 해쉬하여 생성한다.

라. 선불 방식 메커니즘

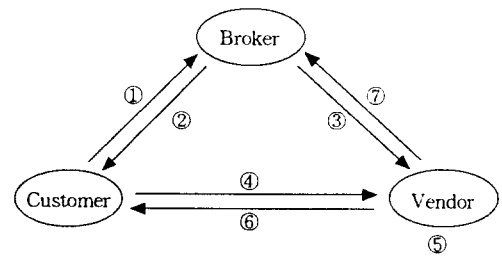
Broker가 금액 생성 범위(각 금액의 해쉬 횟수)을 인증함으로써 선불로 처리 가능하도록 한다.

ex) 선불로 20,000원을 받았을 경우 다음의 사용 금액(해쉬 횟수)을 인정한다.

- $w\_l_1 \sim w\_5000_1$: 일단위는 5000회(5,000원)까지
- $w\_l_{10} \sim w\_5000_{10}$: 십단위는 500회(5,000원)까지
- $w\_l_{100} \sim w\_5000_{100}$: 백단위는 50회(5,000원)까지
- $w\_l_{1000} \sim w\_5000_{1000}$: 천단위는 5회(5,000원)까지

Customer가 payword를 생성하게 되면 해쉬 횟수만큼 paychain_new_root값이 변하게 된다. 따라서 vendor와 broker는 각 금액의 해쉬 횟수를 점검할 수 있고 정해진 범위 내에서 사용했는지를 점검함으로써 token을 인증한다.

3.2 지불 프로토콜



[그림4] 제안 방식 프로토콜

가. ① 단계 (authority와 secret_key 요구)

안전한 지불 수단을 이용하여 금액을 지불하고 각각의 금액 단위에 대하여 원하는 금액을 결정하여 authority와 secret_key를 요구한다.

나. ② 단계 (authority와 secret_key 전송)

Broker는 customer로부터 지불 받은 후 금액 생성 범위 맞는 각각의 paychain_root를 생성하고 secret_key와 authority를 전송한다.

Secret_key는 broker와 customer가 가지는 비밀정보로서 안전하게 보관한다.

다. ③ 단계 (authority 전송)

Paychain_root와 식별번호 등 거래를 위해 필요한 정보를 포함한 authority를 전송한다.

라. ④ 단계 (request문 전송)

Customer는 vendor에게 request문을 보낸다. Request문은 (request || token || certificate) 로 구성되어 있다. 여기서 request는 상품 정보(상품명, 금액)를 나타내고 certificate는 H(request || token || paychain_root)을 나타낸다.

제 3자는 paychain_root값을 알지 못함으로 부정한 request문의 생성이 불가능하다.

마. ⑤, ⑥ 단계 (request문, token 확인 및 서비스)

Vendor는 request문과 token을 검증을 하고 이상이 없으면 서비스를 제공한다.

검증의 과정은 다음과 같다.

- vendor는 paychain_root값을 알고 있기 때문에 request문의 certificate를 검증할 수 있다.
- hash_count로 지불 받은 token의 지불 생성 범위를 검사하여 유효성을 판단한다.
- paychain_new_root의 해쉬를 통해서 payword의 금액을 확인한다.
- paychain_new_root를 검사한다.

  Vendor는 최초의 paychain_root를 알고 있으므로 hash_count를 이용해서 검사할 수 있다.

바. ⑦ 단계 (token 반환)

Vendor는 수집한 token을 broker에게 반환하고 자신의 계정에 입금을 요구한다. 입금 요구를 받은 broker는 token의 유효성을 검증하고 이상이 없으면 vendor의 계정에 입금을 한다.

Broker가 token의 유효성을 확인하는 과정은 다음과 같다.

- certificate를 확인
  : DB에 있는 해당 Cust_ID#의 secret_key와 token_body를 해쉬한 값을 certificate와 비교한다.
- paychain_new_root를 검사한다.
- payword를 검사한다.
- token의 지불 생성 범위를 검사한다.

3.3 제안 방식 분석

제안 방식은 customer가 지불 금액에 맞게 token을 생성하므로 vendor의 거스름돈이 필요 없고, 선불 방식을 따르므로 customer의 신용에 대한 문제가 없다. 중복 사용의 경우에는 paychain_new_root 값의 점검으로 발견할 수 있다.

또한 vendor나 제 3자는 secret_key를 모르기 때문에 부정한 token을 만들 수 없다.

4. 결 론

앞으로 전자상거래에서 소액지불은 중요한 위치를 차지하게 될 것이다. 이번 제안서에서 기존의 소액지불 시스템의 장점을 살리고 단점을 보완해서 보다 효율적이고 안전한 프로토콜을 설계해보았다.

기존의 소액지불 시스템과 제안 방식을 비교해보면 다음과 같다.

	Millicent	MPTP	제안 방식
지불 방식	선불 방식	후불 방식	선불 방식
Customer 신용	필요 없음	필요함	필요 없음
거스름돈	필요함	필요 없음	필요 없음
Vendor의 부정	가능	가능	불가능
익명성	없음	없음	없음

앞으로 익명성 등 미비한 부분을 좀 더 보완해서 인터넷상에서 안전한 소액지불 거래가 이루어진다면, 실생활에서 편리함과 다양한 비즈니스 제공 등 많은 이익을 제공할 것이다.

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

<참고 문헌>

[1] "Millicent", http://www.millicent.digital.com/

[2] "MPTP", http://www.w3.mag.keio.ac.jp/TR/WD-mptp -951122

[3] "MiMi", http://www.infosys.tuwien.ac.at/Staff/vh/mimi.html

[4] 이임영외 8명, "전자 상거래 환경을 위한 기술조사 연구", 한국 전산원 연구 보고서, 1996. 10

[5] 송상헌, 박정수, 강신각, 류재철, "전자상거래를 위한 소액 지불 시스템 구현", 정보보호학술발표대회 논문집 제1권, 1호, pp.121-137, 1997.