# 해쉬체인을 이용한 새로운 오프라인 전자화폐

## (New Offline Electronic Cash using Hash Chain)

김 상 진 † 오 희 국 ††
(Sangjin Kim) (Heekuck Oh)

**요 약** 해쉬체인은 계산속도가 빠른 해쉬함수를 이용하여 체인을 구성하는 구조이다. 이 구조를 이용하여 화폐를 만들면 해쉬연산만으로 화폐의 유효성을 확인할 수 있어서 지금까지 주로 적은 금액이 빈번하게 교환되는 실명 거래 환경에서 응용되었다. 최근에 익명 거래가 가능한 범용화폐로 확장하려는 노력이 있었으나 추가 비용이 많이 들어 해쉬체인이 가지고 있는 본래의 장점이 퇴색하는 결과를 초래했다. 이 논문에서는 해쉬체인의 장점을 최대한 유지하면서 익명 거래와 분할이 가능한 오프라인 화폐를 제안한다. 이 화폐는 인출할 때 해쉬체인의 길이만큼 서명이 필요한 기존의 시스템과는 달리 한 번의 은닉서명만 수행하며, 다양한 금액을 추가비용 없이 지불할 수 있다. 새 시스템은 사용하고 남은 화폐에 대해서 환불이 가능하도록 하였으며, 환불티켓이라는 새로운 개념의 환불방식을 사용하여 지불과 환불을 연관시킬 수 없도록 하였다. 그밖에 지불액과 환불액 사이의 차액을 통해 관련 여부를 추측할 수 없도록 환불액을 축적하는 방식을 사용하고 있다.

**키워드** : 전자화폐, 범용화폐, 익명성, 해쉬체인

***Abstract*** A hash chain is highly efficient, attractive structure to use in electronic cash. Previous systems using hash chain were, however, either credit-based vendor-specific cash or debit-based general-purpose cash which lacks efficiency due to double spending problem. In this paper, we propose a new divisible cash system using hash chain. This newly proposed cash is general-purpose, debit-based, anonymous, and offline. The efficiency of the system results from its capacity to pay variable amounts with no additional costs. A client always performs a single blind signature in the withdrawal phase, independent of the length of the chain. During payment, a client performs a single challenge-and-response or generates a single signature, independent of the amount paid. This system provides a new refund mechanism, which uses a refund ticket, that allows clients to refund the unspent part of the chain without revealing any connection to the spent part.

**Key words** : electronic cash, general-purpose cash, anonymity, hash chain

## 1. 서 론

해쉬체인(hash chain)은 어떤 값을 해쉬함수에 적용하여 얻은 값을 다시 해쉬함수에 적용하여 그 다음 값을 얻는 방법으로 해쉬연산을 여러 번 반복하여 얻은 값의 리스트를 말한다. 길이가 $l$인 해쉬체인을 $(c_0, c_1, \ldots, c_l)$로 나타내며, $0 \le i \le l-1$에 대해 $c_i = H(c_{i+1})$ 관계가 성립한다. 여기서 $H$는 충돌회피(collision-resistant) 해쉬함수이고, $c_0$는 체인의 루트(root)라 한다. 해쉬함수의 일방향성 때문에 $c_i$를 알아도 체인을 만든 사람을 제외하고는 $c_j (i < j \le l)$를 만들 수 없다. 해쉬함수의 일방향성을 이용한 이 구조는 Lamport[1]가 패스워드 기반 인증 시스템을 만들면서 처음 사용되었다. 전자화폐에서는 1996년 같은 학회에서 해쉬체인을 이용한 세 개의 지불 시스템[2-4]이 소개되면서 많은 관심을 불러일으켰다.

현재까지 발표된 대부분의 전자화폐는 동전방식[5]이다. 이 방식에서는 고객의 익명성과 액면가를 표현하는 비용을 줄이기 위해 몇 종류의 액면가만을 제공한다. 따라서 고객은 다양한 액면가의 많은 동전을 인출하여 가

† 정 회 원 : 한국기술교육대학교 인터넷미디어공학부 교수
sangjin@kut.ac.kr
†† 종신회원 : 한양대학교 전자컴퓨터공학부 교수
hkoh@cse.hanyang.ac.kr
논문접수 : 2002년 1월 10일
심사완료 : 2002년 7월 20일

지고 있어야 하며, 보통 여러 개의 동전을 이용하여 지불한다. 상점은 각 동전마다 은행의 서명을 확인하고 고객과 시도와 응답(challenge-and-response)을 수행해야 하므로 사용되는 동전의 개수에 비례하여 처리비용이 증가한다. 이 때문에 지불하는 동전의 수가 많으면 효율성이 떨어지며, 고객이 동전을 충분히 가지고 있더라도 지불대금을 정확하게 맞추지 못하면 지불할 수 없는 불편함이 있다. 해쉬체인에 기반한 전자화폐는 동전방식의 이러한 단점에 대한 대안으로 등장한 것이다[2-4,6-11]. 해쉬체인의 각 값을 동전으로 사용하는 전자화폐에서는 일반적으로 체인의 루트에 대해서만 서명을 확인하고 나머지 동전에 대해서는 해쉬연산만을 이용하여 유효함을 확인할 수 있다.

해쉬체인을 이용한 초기 지불시스템에는 Rivest와 Shamir의 Payword[2], Pedersen의 Tick 지불[3], Anderson 등의 NetCard[4], Hauser 등의 iKP 기반 소액지불시스템[6] 등이 있다. Tick 지불과 Netcard는 해쉬체인을 이용한 개념에 대해 소개하고 있지만, 지불시스템에 대한 구체적인 서술은 없다. Payword와 iKP 기반 소액지불시스템은 후불방식(credit-based)의 판매자 전용화폐(vendor-specific cash)이다. 이처럼 해쉬체인을 이용한 초기 지불시스템은 고객의 프라이버시나 화폐의 범용성에 대해 고려하지 않고 해쉬체인의 효율성만 강조하여 시스템을 구성하였다. Mu 등[7]은 해쉬체인을 구성할 때 salt를 추가하여 체인의 안전성을 증가시키고자 하였고, Yen과 Zheng[8]은 해쉬체인을 구성하는 각 값에 가중치를 부여하여 보다 작은 길이의 해쉬체인으로 많은 금액을 지불할 수 있도록 개선하였다. 그러나 [7]과 [8]은 고객의 프라이버시에 대한 고려나 범용화폐(general-purpose cash)로의 확장을 추구하지는 않고 체인의 안전성이나 성능을 개선하고자 하였다.

보통의 지불시스템은 선불방식(debit-based)으로 화폐를 인출할 때 금액만큼 먼저 은행에 지불하고 사용하는 방식이다. 이와는 달리 후불방식은 고객이 스스로 화폐를 만들어 사용하고, 은행이 나중에 사용한 만큼을 고객에게 청구하는 방식이다. 따라서 이 방식은 인출과정이 없다는 장점이 있지만 익명으로 거래할 수 없다는 단점이 있다. 또한 고객이 사용한 만큼의 채무를 이행하지 않을 수 있으므로 고객의 신용한도를 제한할 수 있는 방법이 필요하다. 반대로 선불방식은 익명 거래가 가능하지만 선불이므로 고객이 손해를 보지 않도록 사용하지 않은 화폐나 사용하고 남은 금액은 언제든지 은행으로부터 환불받을 수 있어야 한다.

해쉬체인의 장점을 그대로 유지하면서 고객의 프라이버시를 고려하고 범용화폐를 만들려는 노력이 있었다[9-11]. Mao[9]는 해쉬체인을 사용하고 남은 부분을 다른 상점에게 지불할 수 있도록 상점이 거스름을 만들어 주는 범용화폐를 제안하였다. Nyugen 등[10]은 이중잠금(double-locked) 해쉬체인이라는 구조를 이용하여 범용화폐를 만들고자 하였다. 이들은 또한 [11]에서 일반 오프라인 동전방식에서 각 동전을 해쉬체인으로 연결하여 지불의 효율성을 높이고자 하였다. 그러나 [9]는 가명(pseudonym)을 이용하여 익명성을 제공하므로 하나의 체인뿐만 아니라 서로 다른 체인을 이용한 지불도 연관되는 문제점과 남은 금액을 상점이 서명해줘야 사용할 수 있으므로 체인을 여러 상점에게 사용하면 할수록 그 다음 상점에게 전달하여야 하는 정보의 크기가 커지는 문제점이 있다. [10]은 익명성을 제공하지 않아 거래가 노출되고, 체인 구조에 허점이 있어 상점끼리 공모하면 고객의 체인을 생성할 수 있는 문제점이 있다. [11]은 익명성과 범용성을 모두 만족하지만 인출과정에서 체인의 길이만큼 서명이 필요해서 효율성이 떨어진다.

소액 환경이라도 고객은 자신의 프라이버시가 침해되는 것을 원하지 않는다. 또한 화폐의 편리성과 유용성 측면에서 고객은 판매자 전용화폐보다는 여러 상점에게 지불할 수 있는 범용화폐를 선호한다. 이 논문에서는 해쉬체인의 장점을 최대한 유지하면서 익명으로 거래가 가능한 선불방식의 범용 오프라인 화폐를 제안한다. 이렇게 해쉬체인을 이용하여 범용화폐를 만들면 화폐는 분할성(divisibilty)을 가지게 된다[12]. 분할성이란 화폐의 액면가가 $v$일 때 이 화폐를 $v$보다 적은 액면가를 지닌 여러 개의 화폐로 나누어 사용할 수 있는 것을 말한다. 새 시스템은 기존의 분할 가능한 화폐[12-14]처럼 같은 해쉬체인을 이용한 지불이 서로 연결되는 단점이 있으나 서로 다른 해쉬체인을 이용한 지불은 서로 연결되지 않는다. 또한 지불과정에서 화폐의 유효성을 확인하는 시도와 응답은 화폐에 사용된 노드의 수만큼 필요한 기존의 분할화폐와는 달리 한 번만 수행한다. 새 시스템은 선불방식이므로 고객은 언제든지 체인의 사용하지 않은 부분을 은행으로부터 환불받을 수 있도록 하였다. 이 때 환불티켓이라는 개념을 사용하여 기존 지불과 환불을 연관시킬 수 없도록 하였다.

이 화폐는 해쉬체인과 표현 문제(representation problem)[5]를 이용하여 화폐를 구성하며, Solages와 Traore의 제한적 은닉서명(restrictive blind signature)[15]을 사용하여 고객의 익명성을 보장한다. 그러나 익명의 화폐가 범죄에 악용되는 것을 막기 위해 조건부 익명성을 제공한다. 이 화폐는 신뢰기관을 이용한 화폐

추적과 인출자 추적 메커니즘을 제공한다. 해쉬체인은 다양한 형태로 나누어 사용할 수 있으므로 이중사용한 고객을 찾아내는 비용을 줄이기 위해 인출자 추적 메커니즘을 이중사용에 대해서도 활용한다.

이 논문의 구성은 다음과 같다. 2장에서 이 논문의 이해를 위해 필요한 수학적 배경을 간략히 소개하고, 해쉬체인을 이용한 기존 전자화폐의 특성과 문제점을 분석한다. 3장에서는 이 논문에서 제안하는 새 시스템을 상세히 서술한다. 4장에서는 새 시스템의 안전성을 분석하고, 다른 시스템과 비교한다. 끝으로 5장에서 결론과 향후 연구 방향에 대해 서술한다.

## 2. 수학적 배경과 관련 연구

이 논문에 있는 모든 수학 연산은 군(group)의 위수(order)가 매우 큰 소수 $q$(160 비트 이상)인 $G_q$군에서 이루어진다. 이 군은 먼저 매우 큰 소수 $p$(512 비트 이상)를 선택하고 $p-1$의 소인수 중 하나인 $q$를 선택하여 구성된다. 따라서 $G_q$군은 $Z_p^*$의 부분군(subgroup)이며, 군 연산은 법(modulo) $p$에서의 곱셈이다. 이 군에서 1를 제외한 모든 수는 군의 생성자(generator)가 된다. 시스템의 안전성은 이 군에서 이산대수(discrete logarithm)를 구하는 것이 계산적으로 어렵다는 것에 기반한다. 이 논문에서 이루어지는 연산의 법은 문맥에서 유추할 수 있으므로 생략한다.

### 2.1 표현 문제

표현 문제란 이산대수 문제를 응용한 것으로 여러 개의 생성자로 만든 어떤 값이 있을 때, 이 값을 만들기 위해 사용한 생성자의 색인(index)값을 알아내는 것이 계산적으로 어렵다는 것에 바탕을 두고 있다. 다음은 표현 문제와 관련된 정의와 정리이다. 정리 1과 따름정리 1에 대한 증명은 이산대수 가정을 이용하여 쉽게 증명할 수 있다[5].

**정의 1.** 길이가 $l(\geq 2)$인 생성자 튜플은 $(g_1, \ldots, g_l)$을 말하며, 이 때 $g_i \in G_q - \{1\}$는 $G_q$의 생성자이며 $i \neq j$이면 $g_i \neq g_j$이다. 생성자 튜플 $(g_1, \ldots, g_l)$에 대한 $y \in G_q$의 표현은 $\prod_{i=1}^{l} g_i^{x_i} = y$인 색인 튜플 $(x_1, \ldots, x_l)$을 말하며, 이 때 $\forall x_i \in Z_q$이다.

**정리 1.** $G_q$에서 이산대수를 계산하는 것이 어렵다고 가정하면 모든 $y \in G_q - \{1\}$에 대해 임의로 선택한 생성자 튜플 $(g_1, \ldots, g_l)$을 입력하였을 때, 어느 정도 성공할 수 있는 확률을 가지고 $y$의 표현을 찾는 다항시간 알고리즘은 존재하지 않는다.

**따름정리 1.** $G_q$에서 이산대수를 계산하는 것이 어렵다고 가정하면 임의로 선택한 생성자 튜플 $(g_1, \ldots, g_l)$을 입력하였을 때, 어느 정도 성공할 수 있는 확률을 가지고 어떤 수 $y \in G_q$와 $y$의 서로 다른 두 가지 표현을 찾는 다항시간 알고리즘은 존재하지 않는다.

### 2.2 제한적 은닉서명

제한적 은닉서명이란 용어는 Chaum과 Pedersen이 제안한 서명 기법[16]을 Brands[5]가 응용하면서 처음으로 사용되었다. 제한적 은닉서명은 기존의 cut-and-choose 기법을 사용하지 않고 서명 받는 사람이 부정을 못하게 하는 은닉서명이다. 이 논문에서는 그림 1에 기술되어 있는 Solages와 Traore[15]가 제안한 제한적 은닉서명을 사용한다. 이 서명에서 서명자의 서명키는 $x \in G_q$이며, 확인키는 $y = g^x$이다. 이 프로토콜이 정상적으로 종료되면 수신자는 메시지 $m$에 대한 서명 $Sig(m) = (z, c, s)$을 얻는다. 이 때 서명값 $z = m^x, c, s$는 다음 식을 만족한다.

$$c = H_q(M \| m \| z \| g^s y^c \| m^s z^c)$$

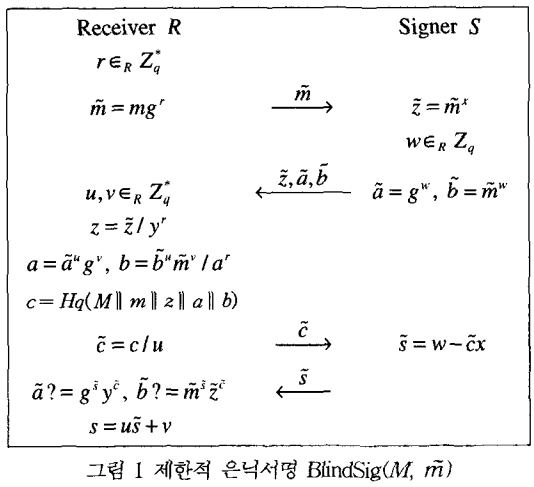


그림 1 제한적 은닉서명 $BlindSig(M, \tilde{m})$

서명 확인은 위 식을 이용한다. 여기서 '||'은 비트 결합을 나타내며, $H_q$는 $\{0,1\}^*$를 입력받아 $Z_q$의 임의의 원소로 매핑해주는 충돌회피 해쉬함수이다. 또한 $M \in \{0,1\}^*$은 서명할 때 수신자가 서명과 연관하고 싶어 서명에 포함하는 값으로서 필요가 없으면 사용하지 않아도 된다. 이 서명에 관한 몇 가지 정리는 다음과 같다. 각 정리에 대한 증명은 이미 [5,15-18]에 있으므로, 이 논문에서는 정리 4에 대해서만 개략적으로 설명한다.

**정리 2.** (**정확성**) 수신자가 증명을 인정하면 $Sig(m)$은 $m$에 대한 올바른 서명이다.

**정리 3.** (**은닉성**) 수신자가 프로토콜을 충실하게 수행하면 서명자는 $m$과 $Sig(m)$에 대한 어떤 정보도 얻을 수 없다.

**가정 1.** (**위조불가능성**) BlindSig를 다항 번(순차적으로 또는 병행으로) 반복 수행하여도 원래 얻을 수 있는 서명 쌍을 제외하고는 추가적인 서명 쌍을 얻을 수 없다. 뿐만 아니라 프로토콜의 수행을 통해 서명자의 서명키 $x$에 대한 어떤 정보도 노출되지 않는다.

이 서명의 위조불가능성에 대해서는 아직 증명되어 있지 않다. 그렇다고 하여 이 서명을 위조할 수 있는 공격방법이 있는 것은 아니다. 이 가정의 정당성에 관한 논의는 이 논문의 범위를 벗어나며, [15-18]을 참조하면 보다 자세한 내용을 얻을 수 있다.

**정리 4.** (**은닉 제한**) 수신자가 $\tilde{m}$을 서명자에게 전달한 후에는 생성자 $g$ 외에 다른 생성자를 이용하여 서명받을 메시지 $m \in G_q$을 결정할 수 없다.

수신자가 $\tilde{m}$을 전달하면 서명자는 자신의 서명키 $x$로 이 값을 거듭제곱하여 돌려준다. 이 값으로부터 고객은 자신이 원하는 메시지 $m$에 $x$가 거듭제곱된 값 $z=m^x$을 만들고, 이것에 대응되는 $b$ 값을 만들 수 있어야 한다. 전자는 시스템에 따라 가능하지만 후자는 은행이 임의로 선택하는 $w$ 값을 알아야 가능하므로 이산대수 가정에 의해 계산적으로 어렵다. 만약 $m$을 구성하는 어떤 생성자 $g_i$에 대해 수신자가 이산대수 $a=\log_g g_i$를 알고 있으면 은행이 확인한 $\tilde{m}$에 포함된 $m$과 다른 $m'$에 대한 서명을 얻을 수 있다. 그러나 이것도 이산대수 가정에 의해 계산적으로 어렵다. 따라서 이 은닉서명 프로토콜은 오직 생성자 $g$를 이용해서만 서명받을 메시지를 결정할 수 있다.

### 2.3 해쉬체인을 이용한 기존 전자화폐

Payword는 오프라인이며, 후불방식의 판매자 전용화폐이다[2]. Payword에서 고객은 인출과정 없이 스스로 해쉬체인을 만들고, 체인의 루트에 자신의 개인키로 서명을 하여 만든 서약(commitment)과 체인의 특정 값을 함께 전달함으로써 지불한다. 체인의 각 값이 100원일 때 길이가 $l$인 체인을 만들어 화폐로 사용하면 $l\times100$원까지 쓸 수 있다. 지불은 $c_1$부터 시작하며, 지불액이 100원 이상일 경우에는 중간 값을 보내지 않고 액수만큼 건너 뛰어 마지막 값만 전달하여 지불한다. $c_i$가 전달되었을 때 이 값의 유효성은 $c_i$에 해쉬함수를 $i$번 적용한 값이 체인의 루트인 $c_0$와 같은지 비교하여 확인한다. 상점은 마지막으로 수신된 체인의 값과 서약을 은행에 전달하여 입금한다. 은행은 해당 고객에게 사용한 액수만큼 청구하게 된다. 같은 체인이 서로 다른 상점으로부터 입금되면 이것을 고객이 이중사용하였는지 아니면 상점이 사용하였는지 알 수 없으므로 하나의 체인을 특정 상점에게만 지불할 수 있도록 제한하였다.

Payword는 인출과정이 없고 해쉬함수를 이용하여 화폐의 유효성을 빠르게 확인할 수 있으므로 효율성 측면에서 매우 우수한 시스템이며, 같은 상점에게 여러 번 지불할 때에는 매우 유리하다는 장점을 지니고 있다. 그러나 실명거래로 인해 고객의 프라이버시가 노출되며, 하나의 체인으로 여러 상점에게 지불할 수 없다는 단점이 있다. 또한 고객에 따라 신용한도를 제한해야 하는 문제가 있고, 이중사용 여부를 확인하기 위해 상점이 수신한 화폐의 데이타베이스를 유지하여야 하는 불편함이 있다.

iKP 기반 소액지불시스템은 Payword와 마찬가지로 후불방식의 판매자 전용 화폐이다[6]. 그러나 후불방식에서 신용한도를 제한하기 위해 체인을 상점에게 처음으로 지불할 때 체인의 액면가만큼 고객이 지불할 수 있는지 상점이 은행으로부터 확인하는 방식을 사용하고 있는 온라인과 오프라인의 중간형태의 지불방식을 사용하고 있다.

Mao는 익명인증서(anonymous certificate)에 포함된 공개키를 화폐에 포함하는 전자화폐를 제안하였다[9]. 고객은 공개키에 대응되는 개인키로 서명을 함으로써 지불을 하게 되며, 이중사용하면 개인키가 노출되도록 하여 이중사용을 방지하고 있다. 그러나 익명인증서를 이용하므로 같은 해쉬체인을 이용한 지불뿐만 아니라 서로 다른 체인을 이용한 지불도 같은 고객의 것이라는 것을 알 수 있다. 뿐만 아니라 입금과정에서 은행이 화폐를 확인하지 않기 때문에 가짜 화폐를 입금할 수 있는 치명적인 허점이 있다. 이 시스템에서는 해쉬체인을 사용하고 남은 부분을 다시 사용할 수 있도록 은행이 아닌 상점이 서명을 하여 만들어 준다. 이렇게 하면 범용성은 충족되지만 화폐가 사용되면 될수록 그 크기가 커지는 문제점이 있다.

Nguyen 등은 이중잠금 해쉬체인이라는 두 개의 해쉬체인을 이용한 구조를 사용하여 Payword를 범용화폐로 바꾸고자 하였다[10]. 이중잠금 해쉬체인은 먼저 두 개의 해쉬체인 $(c_0,\ldots,c_l)$, $(c'_0,\ldots,c'_l)$을 만들고, 체인의 루트를 $(c_0, c'_0, l+1)$로 사용한다. 이 때 각 동전은 $(c_i, c'_{l-i})$쌍이 되며, $(c_0, c'_l)$, $(c_1, c'_{l-1})$, … 순으로 사용한다. 이렇게 하면 단일 체인을 사용할 때와는 달리 $(c_i, c'_{l-i})$을 가지고 있어도 그것의 이전 값 $(c_{i-1}, c'_{l-i+1})$이나 이후 값 $(c_{i+1}, c'_{l-i-1})$을 만들 수

없다. 그러나 만약 $j>i$인 $(c_i, c'_{l-i})$와 $(c_j, c'_{l-j})$를 가지고 있으면 두 값 사이에 있는 모든 값을 만들 수 있다는 문제점이 있다.

Nguyen 등은 또한 같은 학술대회에서 오프라인 동전 방식의 익명화폐에서 여러 개의 동전을 이용하여 지불하여야 할 경우에 지불의 효율성이 떨어지는 문제점을 해쉬체인을 이용하여 극복하고자 하였다[11]. 이를 위해 오프라인 동전들을 해쉬체인을 이용하여 연결하여 사용하고 있다. 그러나 해쉬체인에 있는 동전 각각에 대해 은행으로부터 은닉서명을 받아 사용하여야 하므로 인출과정이 복잡하다. 반면에 각 지불에서 첫 동전에 대한 확인과정은 기존의 오프라인 동전과 비슷한 연산량이 요구되지만 그것을 제외한 나머지 동전은 해쉬체인의 특성을 이용하여 확인하므로 저렴하게 확인할 수 있다. 이 시스템은 선불방식의 화폐이지만 환불에 대한 언급은 없다.

## 3. 해쉬체인을 이용한 새로운 오프라인 범용화폐

이 장에서는 이 논문에서 제안하는 해쉬체인을 이용한 새로운 오프라인 방식의 분할 가능한 범용화폐를 자세히 설명한다. 제안하는 화폐를 자세히 서술하기에 앞서 먼저 해쉬체인을 이용하여 고객의 프라이버시를 보장하는 오프라인 범용화폐를 만들 때 고려해야할 점들을 살펴본다.

### 3.1 해쉬체인을 이용하여 오프라인 범용화폐를 만들 때 문제점

해쉬체인은 체인의 루트를 통해 식별되므로 익명성을 제공하기 위해서는 루트값이 인출과정에서 은닉되어야 한다. 또한 선불방식이므로 고객이 체인의 길이에 대해 부정을 할 수 없어야 한다. 길이에 대한 부정을 방지하는 방법으로는 다음과 같은 몇 가지를 생각할 수 있다.

- **방법1**: 인출할 수 있는 체인의 길이를 고정시킴
- **방법2**: 인출할 수 있는 길이와 은행의 공개키를 연관시킴
- **방법3**: 표현 문제를 사용하여 특정 생성자의 색인값으로 길이를 나타냄

방법 1은 유연성이 떨어지는 문제점이 있다. 고객은 항상 시스템에서 정한 길이의 해쉬체인만 인출할 수 있으므로 불편하다. 방법 2는 기존 화폐에서 동전의 액면가를 나타내기 위해 사용하는 방법이다. 기존에는 은행의 특정 공개키가 특정 액면가를 나타낸 반면, 해쉬체인의 경우에는 특정 길이를 나타내도록 하는 것이다. 이렇게 하면 은행이 유지하여야 하는 공개키가 많아지므로 불편하다. 이를 극복하기 위해 체인의 길이를 서명의 공개부분으로 하는 부분은닉서명[19]을 사용할 수 있다. 방법 3은 표현 문제를 이용하여 화폐를 구성한다. 다음은 생성자 튜플 $(g_U, g_R, g_L, g_T)$를 사용하여 이 논문에서 화폐를 구성한 예이다.

$$C = g_U^{x_U} g_R^{c_0} g_L^{l} g_T^{r}$$

여기서 $x_U$는 고객의 비밀신원정보이고, $c_0$는 체인의 루트이다. $l$은 체인의 길이이며, $r$은 이 화폐를 인출할 때 사용한 은닉요소로서 추적 기능을 제공하고 다른 요소를 숨기기 위해 사용된다. $e=H_a$(만료날짜)일 때 위 $C$ 대신에 $C'=Cg_E^e$을 사용하여 유효기간까지 나타낼 수도 있다. 그러나 이 논문에서 이 부분은 생략하고 시스템을 설명한다. 은행은 인출과정에서 은닉서명을 수행하기 전에 이 값의 구성을 확인한다. 방법 2가 방법 3보다는 효율적이지만 방법 3을 사용하면 Solages와 Traore가 제안한 제한적 은닉서명과 추적 기능을 적용할 수 있어[15], 이 논문에서는 방법 3을 사용한다.

전자화폐는 서명키를 주기적으로 변경해야 하고, 이중사용을 검출할 때 필요한 데이타베이스의 크기를 일정한 수준으로 유지해야 하는 등의 이유 때문에 유효기간이 있다. 따라서 선불방식의 화폐는 은행에 반납하여 환불받을 수 있어야 한다. 일반적인 오프라인 화폐는 유효기간 전에 은행에 입금함으로써 쉽게 환불받을 수 있다. 그러나 해쉬체인이나 분할 가능한 화폐처럼 동전 간에 연결고리가 있는 화폐는 이미 지불한 부분의 익명성을 유지하면서 환불받기가 쉽지 않다. 또한 이런 연결고리 때문에 지불액과 환불액 사이의 차액을 이용하여 환불받을 때 그것이 어떤 화폐에 대한 환불인지 추측이 가능하다. 따라서 환불 프로토콜은 기존 지불의 익명성을 유지해주어야 할 뿐만 아니라 지불액과 환불액 사이의 차액을 통한 연결고리를 제거해주어야 한다. 이것을 해결하기 위한 다음과 같은 방법을 생각해 볼 수 있다.

- **방법1**: 기존 오프라인 수표시스템[15]에서 사용하는 환불방법을 응용함.
- **방법2**: 기존 온라인 수표시스템[20]에서 사용하는 환불방법을 응용함.

오프라인 수표시스템에서는 수표를 지불에 사용할 부분과 환불에 사용할 부분으로 구성하여 사용한다. 방법 1은 이렇게 화폐를 두 부분으로 구성하여 하나는 지불에 사용하고 다른 하나는 환불에 사용하는 것이다. 이 방법을 사용하면 기존 지불의 익명성은 유지할 수 있으나 화폐의 형태가 복잡해지며, 지불과정에서 환불할 때 부정을 못하도록 하기 위한 추가적인 연산과 정보 교환이

필요하다. 또한 이 방법을 이용하면 지불액과 환불액을 이용한 연결고리를 없애는 것이 어렵다. 온라인 수표시스템에서는 지불과정에서 발생한 거스름을 은행으로부터 받아 이것을 나중에 환불받거나 지불에 다시 사용한다. 방법 2는 이처럼 환불받을 부분을 은행에 지불하고 나중에 환불을 받을 수 있는 티켓을 받아 사용하는 것이다. 이 방법은 환불을 받기 위해 은행에 다시 접촉해야 하는 불편함이 있지만 인출이나 지불과정과 무관하게 환불기능을 제공할 수 있으므로 환불이 필요 없는 경우에는 아무런 추가비용이 없다는 장점이 있다. Chaum의 온라인 수표[20]에서 사용한 쿠키통(cookie-jar) 방식을 환불티켓으로 활용하면 하나의 티켓에 환불액을 계속 축적할 수 있다. 이렇게 축적한 것을 이용하여 환불받으면 기존과 달리 지불액과 환불액 사이에 직접적인 연관성이 없어지므로 익명성 유지에 큰 도움이 된다.

해쉬체인을 이용하여 범용화폐를 만들면 분할성을 지니므로 이중사용하는 경우가 다양해진다. 이중사용은 [11]에서 언급된 것처럼 크게 두 가지 경우로 분류할 수 있다.

- **분류1:** 이중사용한 체인 부분의 시작 위치가 같은 경우
- **분류2:** 이중사용한 체인 부분의 시작 위치는 다르지만 중첩되는 부분이 있는 경우

오프라인 화폐에서는 이중사용한 고객을 알아내기 위해 보통 고객의 신원정보를 화폐에 포함하고 이중사용되면 이 신원정보가 드러나도록 한다. 이를 위해 이중사용된 두 지불의 시도와 응답값을 이용하면 화폐에 포함된 고객의 신원정보를 계산할 수 있도록 고안한다. 그러나 해쉬체인의 경우에는 다양한 형태로 나누어 사용할 수 있으므로 체인 중 어떤 부분이 이중사용되더라도 고객의 신원을 밝혀낼 수 있어야 한다. 이것을 해결하기 위해 [11]에서처럼 해쉬체인의 각 동전마다 시도와 응답에 사용할 값을 미리 고정시키는 방법을 사용할 수 있다. 이것은 다음 세 가지 방법으로 제공할 수 있다. 세 방법 모두 인출할 해쉬체인의 길이가 $l$이라고 가정하며, 응답에 사용할 값 $b_i(i=1,\ldots,l)$를 임의로 선택하여 이 값을 화폐 또는 체인에 포함한다. 지불에 사용할 마지막 동전이 $c_i$이고 지불대금이 $v$일때, 첫 동전 $c_{i-v}$에 대해서는 시도값 $c$에 대한 $s=b_{i-v}-cx_U$를 계산하여 전달하고, 나머지 동전에 대해서는 $b_j(i-v+1\le j\le i)$를 전달한다. 따라서 어떤 경우에도 고객의 비밀신원정보 $x_U$를 계산할 수 있다.

- **방법1:** $B_i=g_U^{b_i}$를 계산하고 인출과정에서 은닉서명을 다음과 같이 받는다.

$$\mathrm{BlindSig}(B_1\|\ldots\|B_l,\tilde{C})$$

- **방법2:** 생성자 튜플 $(g_1,\ldots,g_N)$를 두고, $B=\prod_{i=1}^{l}g_i^{b_i}$를 계산하고 인출과정에서 은닉서명을 다음과 같이 받는다. 여기서 $N(\ge l)$은 인출할 수 있는 체인의 최대 길이이다.

$$\mathrm{BlindSig}(B,\tilde{C})$$

- **방법3:** $B_i=g_U^{b_i}(i=1,\ldots,l)$를 계산하고 해쉬체인을 생성할 때 다음과 같이 한다.

$$c_i = H(c_{i+1}\|B_{i+1})$$

방법 1은 새로운 상점에게 지불할 때마다 $B_1$부터 $B_l$를 항상 전달해야 하므로 지불마다 고객이 전달하는 정보가 많다는 문제점이 있다. 그러나 전달된 후에는 추가비용이 소요되지 않는 장점이 있다. 방법 2는 지불과정에서 $B_{i-v}=g_U^{b_{i-v}}$를 전달하고 $g_U$에 대한 $B_{i-v}$의 이산대수와 생성자 튜플 $(g_1,\ldots,g_l)$에 대한 $B$의 표현 중 $g_{i-v}$ 해당하는 값이 같음을 항상 증명하여야 한다. 이 증명 때문에 지불과정에서 추가적인 군 연산이 요구된다. 방법 3은 새로운 상점에게 지불할 때마다 최종 동전의 유효성을 확인하기 위해 $B_1$부터 $B_i$까지 전달하여야 하므로 체인의 후반부를 어떤 상점에게 처음 지불할 경우에는 많은 정보를 전달하여야 하는 문제점을 가지고 있다. 결국 이 세 방법 모두 데이타 연산량과 교환량이 많아서 해쉬체인을 사용하여 효율성을 높이겠다는 본래의 의도가 퇴색되는 문제점이 있다. 이 논문에서는 익명의 화폐가 범죄에 악용되는 것을 막기 위한 추적기능을 이중사용한 경우에도 이용한다. 이렇게 하면 미리 시도와 응답할 값을 인출과정에서 고정시킬 필요가 없고 지불과정에서 이중사용 때문에 별도로 교환하여야 할 정보가 없어 해쉬체인의 장점을 계속 유지할 수 있다.

해쉬체인을 이용하여 범용화폐를 만들면 이중사용하는 형태가 다양해질 뿐만 아니라 상점끼리 공모하면 스스로 계산할 수 없는 체인의 값을 얻을 수 있다. 고객이 단순하게 동전만 전달하여 지불하는 경우에 이중사용이 발견되면 두 상점이 공모한 것인지 고객이 이중사용한 것인지 알 수 없다. 따라서 지불 받은 동전은 해당 상점만이 입금할 수 있도록 하여야 하며, 상점끼리 공모하여도 자신이 지불 받지 않은 동전은 입금할 수 없어야 한다. 이 논문에서는 이것을 제공하기 위해 지불 동전과 상점 식별자를 고객만이 알고 있는 값으로 서명하여 지불하도록 하였다. 여러 개의 동전을 지불하더라도 마지막 동전, 지불 대금, 상점 식별자를 서명하여 주도록 하

여 가변적 금액도 동일한 비용으로 지불할 수 있다. 그러나 동전을 하나 씩 지불하게 되면 동전마다 서명을 하여야 하므로 그 비용이 부담될 수 있다.

### 3.2 시스템 설정 및 계정 개설

은행은 계정을 관리하며, 고객과 상점은 다음 절에 설명된 방법으로 은행에 계정을 개설하여야 지불에 참여할 수 있다. 신뢰기관은 익명의 화폐가 범죄에 악용되는 것을 방지하기 위한 화폐 추적과 인출자 추적 기능을 제공하는 기관이다. 은행과 신뢰기관은 시스템을 설정하기 위해 우선 $q \mid p-1$인 두 개의 매우 큰 소수 $p$와 $q$를 선택하여 $Z_p^*$의 부분군이며 군의 위수가 $q$인 $G_q$군을 설정한다. 은행은 표 1에 있는 다섯 개의 $G_q$군 생성자를 임의로 선택한다. 은행은 화폐를 발행할 때 사용할 서명키 $x_B \in Z_q$를 선택하고 대응되는 확인키 $y_B = g_B^{x_B}$를 계산한다. 은행은 환불티켓 인출을 위한 RSA 법 $n$을 하나 선택하고 환불액을 나타낼 공개지수들에 대한 비밀지수를 생성한다. 은행은 $p$, $q$, $g_B$, $g_U$, $g_S$, $g_R$, $g_L$, $y_B$, $n$을 공개한다. 신뢰기관은 $G_q$군 생성자 $g_T$를 임의로 선택하고, 화폐 추적을 위한 개인키 $x_{CT} \in Z_q$와 인출자 추적을 위한 개인키 $x_{OT} \in Z_q$를 임의로 선택한다. 그 다음에 대응되는 공개키 $y_{CT} = g_B^{x_{CT}^{-1}}$와 $y_{OT} = g_T^{x_{OT}^{-1}}$를 계산한다. 신뢰기관은 $g_T$, $y_{CT}$, $y_{OT}$를 공개한다.

고객은 자신의 비밀신원정보 $x_U \in Z_q$를 임의로 선택하여 자신의 공개신원정보 $y_U = g_U^{x_U}$를 계산하고 $y_U$를 은행에 전달한다. 고객은 $\log_{g_U} y_U$를 알고 있음을 영지식(zero knowledge)으로 은행에 증명한다[21]. 은행은 $y_U$를 고객 식별자로 기록해 놓는다. 상점도 고객과 유사하게 $x_S \in Z_q$를 임의로 선택하여 $y_S = g_S^{x_S}$를 계산하고 $y_S$를 은행에 전달하여 계정을 개설한다. 상점도 고객과 같은 방법으로 $\log_{g_S} y_S$를 알고 있음을 증명한다.

표 1 새 화폐에서 사용하는 생성자

| | 생성자 | 용도 |
|---|---|---|
| 은행 | $g_B$ | 은행의 서명키와 공개키를 위한 생성자 |
| | $g_U$ | 고객의 신원정보를 표현하기 위한 생성자 |
| | $g_S$ | 상점의 신원정보를 표현하기 위한 생성자 |
| | $g_R$ | 해쉬체인의 루트를 표현하기 위한 생성자 |
| | $g_L$ | 해쉬체인의 길이를 표현하기 위한 생성자 |
| 신뢰기관 | $g_T$ | 인출자 추적을 위한 생성자 |

### 3.3 인출 프로토콜

새 시스템의 인출 프로토콜은 그림 2와 같으며 다음과 같이 진행된다.

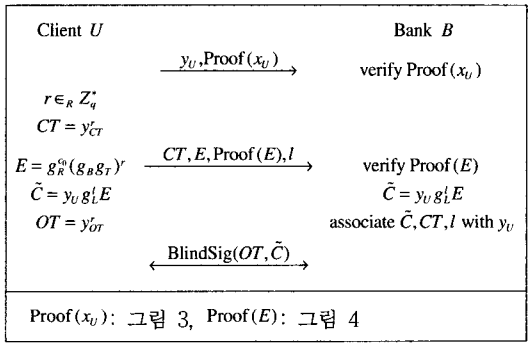


그림 2 인출 프로토콜

- **단계 1:** 고객은 길이가 $l$인 해쉬체인 $(c_0, c_1, \ldots, c_l)$을 생성한다.
  실제 구현에서는 인출할 수 있는 체인의 길이 $l$을 몇 가지로 고정하여 이 정보를 이용하여 화폐를 인출한 고객을 유추할 수 없도록 하여야 한다.
- **단계 2:** 고객은 자신의 신원정보 $y_U$와 Proof($x_U$)를 은행에 전달한다.
  그림 3에 기술된 Proof($x_U$)는 기저 $g_U$에 대한 $y_U$의 이산대수를 알고 있음을 증명하고 있으며, 공격자가 이 증명을 나중에 사용할 수 없도록 시간정보 $T_U$를 증명에 포함한다.
- **단계 3:** 고객은 은닉요소 $r$을 임의로 선택하여 $CT = y_{CT}^r$과 $E = g_R^{c_0}(g_B g_T)^r$을 계산하고, $CT$, $E$, Proof($E$), $l$을 은행에 전달한다.
  그림 4에 기술된 Proof($E$)는 생성자 튜플 $(g_R, g_B g_T)$에 대한 $E$의 표현 $(c_0, r)$을 알고 있음을 영지식으로 증명함과 동시에 이 표현의 $g_B g_T$의 값과 기저 $y_{CT}$에 대한 $CT$의 이산대수가 같음을 영지식으로 증명하고 있다. 은행은 수신한

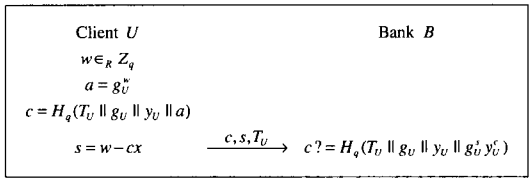


그림 3 Proof ($x_U$)

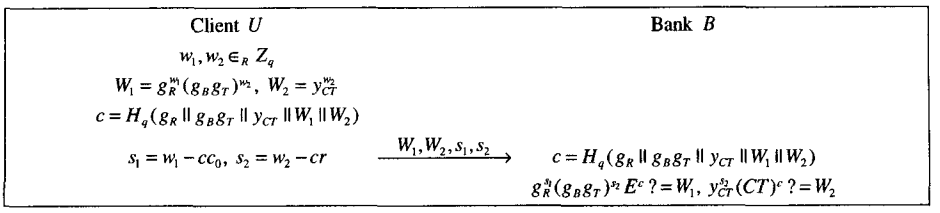


그림 4 Proof ($E$)

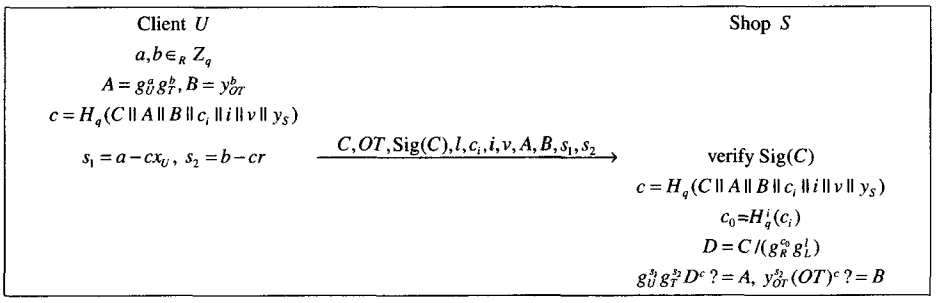


그림 5 각 상점에서 첫 지불에 사용하는 지불 프로토콜

증명을 확인하여 $E$의 구성이 올바르고, $CT$를 통해 화폐에 대한 추적을 할 수 있음을 확인한다. 확인이 끝나면 $C$, $CT$, 체인의 길이 $l$을 인출 데이타베이스에 고객과 연관시켜 놓는다.

- **단계 4:** 고객과 은행은 각자 고객의 공개신원정보 $y_U$, 해쉬체인의 길이 정보 $g_L^l$, 추적 정보와 해쉬체인의 루트가 포함된 $E$를 이용하여 인출할 화폐의 은닉된 형태 $C = y_U g_L^l E$를 계산하고 BlindSig($OT$, $C$)을 수행한다.

  여기서 $OT = y_{OT}^r$는 인출자 추적을 위해 사용되는 값이며, 은행은 이 값을 보지 못한다. 은행은 이렇게 스스로 은닉된 형태를 만들어 제한적 은닉서명을 수행하므로 나중에 고객이 얻게 되는 화폐의 형태가 올바르다는 것을 확신할 수 있다.

이렇게 하여 프로토콜이 종료되면 고객은 $C = y_U g_R^{c_0} g_L^l g_T^r$에 대한 은행의 서명 $\mathrm{Sig}(C) = (z, c, s)$를 얻게 되며, 서명값 $z = C^{x_B}$, $c, s$는 다음을 만족한다.

$$c = H_q(OT \| C \| z \| g_B^s y_B^c \| C^s z^c)$$

### 3.4 지불 프로토콜

고객은 인출한 화폐 $C$를 어떤 상점에게 처음 지불할 때에는 그림 5에 기술된 프로토콜을 사용한다. 그림 5에서 고객은 $c_{i-v}$부터 $c_i$까지를 이용하여 지불하고 있으며, $v$는 지불대금을 나타낸다. 이 프로토콜은 다음과 같이 진행된다.

- **단계 1:** 고객은 $A$와 $B$를 그림 5에서 처럼 계산하고, $C$, $A$, $B$, 사용할 최종 동전 $c_i$, $i$, $v$, 상점 식별자 $y_S$를 이용하여 시도값 $c$를 계산한다. 그 다음 시도값에 대한 응답값 $s_1$과 $s_2$를 계산하고, $C$, $OT$, $\mathrm{Sig}(C)$, $l$, $c_i$, $i$, $v$, $A$, $B$, $s_1$, $s_2$를 상점에게 전달한다.

  $A$는 $C$의 표현을 알고 있음을 증명하기 위해 사용되며, $B$는 인출자 추적이 가능함을 증명하기 위해 사용된다. 고객이 $C$의 실제 표현을 이용하지 않고 다른 표현을 이용하여 증명하지 못하도록 하기 위해 $A$와 $B$를 시도값을 계산할 때 포함한다. 또한 상점이 자신이 받은 금액 이상으로 청구하지 못하도록 하기 $c_i$와 $v$를 포함하며, 지불을 입금하여 돈을 청구할 수 있는 상점을 제한하기 위해 $y_s$를 시도값을 계산할 때 포함한다.
- **단계 2:** 상점은 $\mathrm{Sig}(C)$을 확인하고 시도값 $c$를 계산한다. 그 다음 $c_i$를 이용하여 직접 $c_0$를 생성한다. 그 다음 $c_0$와 $l$을 이용하여 $D$를 만들고 $g_U^{s_1} g_T^{s_2} D^c$와 $A$가 같은지 확인하여 고객이 $C$의 표현을 알고 있음과 $c_i$의 유효성을 확인한다. 또한 $y_{OT}^{s_2}(OT)^c$와 $B$가 같은지 확인하여 인출자 추적이 가능함을 확인한다.

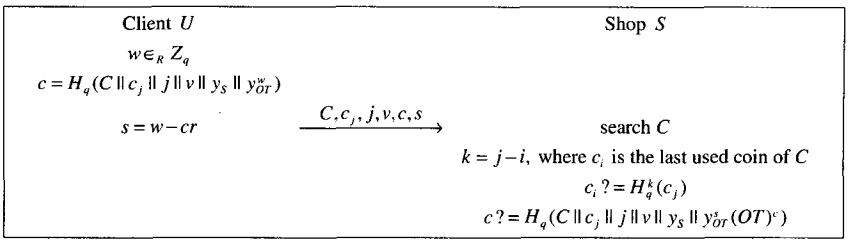


그림 6 각 상점에서 첫 지불 이후에 사용하는 지불 프로토콜

상점이 스스로 시도값을 계산하는 것은 자신이 청구할 수 있는 지불인지 확인하기 위함이다. 여기서 $H_q^k(c_j)$란 $c_j$에 해쉬함수 $H_q$를 $k$번 연속해서 적용한 결과를 나타낸다.

이런 확인이 끝나면 상점은 입금을 위해 수신한 트랜스크립트를 보관하며, 입금한 후에는 $C$를 이용한 고객과의 다음 거래를 위해 $C$, $OT$, $c_i$, $i$를 보관한다.

만약 고객이 어떤 상점에게 $C$를 이미 지불한 적이 있으면 그림 6에 기술된 프로토콜을 이용하여 지불한다. 그림 6에서 고객은 $c_{j-v}$부터 $c_j$까지를 이용하여 지불하고 있으며, 다음과 같이 진행된다.

- **단계 1**: 고객은 기저 $y_{OT}$에 대한 $OT$의 이산대수 $r$을 서명키로 $C$, 최종 동전 $c_j$, 동전의 색인 $j$, 지불대금 $v$, 상점 식별자 $y_s$를 Schnorr 서명[21]한 다음, 그 결과값 $c$와 $s$을 $C$, $c_j$, $j$, $v$와 함께 상점에게 전달한다.
  만약 이 화폐가 판매자 전용이면 $C$, $c_j$, $j$, $v$만 전달하고, 상점은 $c_j$의 유효성만 확인하면 된다. 그러나 범용화폐이므로 두 상점이 공모하여 부당한 이득을 얻을 수 없도록 하여야 한다. 이 때문에 인출자만이 알고 있는 값을 서명키로 $c_j$를 서명하여 전달하는 것이다.
- **단계 2**: 상점은 데이타베이스에서 $C$를 찾아 지난 지불에서 받은 $c_i$를 이용하여 $c_j$를 확인한다. 그 다음 $OT$를 확인키로 수신한 서명을 확인한다.

### 3.5 입금 프로토콜

상점은 첫 지불의 경우에는 $C$, $OT$, $\mathrm{Sig}(C)$, $A$, $B$, $c_i$, $i$, , $v$, $l$, $s_1$, $s_2$, $y_s$를 은행에 전달하여 지불대금을 청구한다. 은행은 먼저 스스로 시도값을 만들고 상점과 같은 방법으로 수신한 트랜스크립트의 이상여부를 확인한다. 스스로 시도값을 만드는 것은 이 트랜스크립트를 이용하여 돈을 청구할 수 있는 상점을 확인하기 위함이다. 정당한 지불임이 확인되면 입금 데이타베이스를 이용하여 현재 입금된 해쉬체인의 동전들이 기존 지불과 중첩되는지 검사한다. 중첩되지 않으면 해당 금액을 상점 계좌에 입금하여 주지만 반대의 경우에는 입금된 트랜스크립트와 저장되어 있는 트랜스크립트를 비교한다. 만약 중첩되는 두 트랜스크립트가 정확하게 일치하면 상점이 이중청구하는 것이며, 그렇지 않으면 고객이 이중사용한 것이다. 전자의 경우에는 입금을 거부하고, 후자의 경우에는 두 트랜스크립트를 증거로 신뢰기관에게 인출자 추적을 요청하여 고객의 신원을 알아낸다.

첫 번째 이후 지불에 대해서는 $C$, $c_j$, $j$, $v$, $c$, $s$를 전달하여 입금한다. 은행은 상점이 했던대로 서명을 확인하고, 첫 지불에 대한 입금과 같은 방법으로 이중사용 또는 이중청구 여부를 검사하여 처리한다.

### 3.6 익명성 제어

익명의 화폐는 실물화폐처럼 돈세탁, 협박, 불법구매와 같은 범죄에 악용될 수 있다. 이것을 막기위해 이중으로 사용되지 않은 화폐도 필요하면 익명성을 철회할 수 있어야 한다. 보통 익명성 철회는 신뢰기관을 통해 이루어지며, 화폐 추적과 인출자 추적 두 가지 형태로 제공된다. 화폐 추적은 협박과 같은 범죄에 대항하기 위해 제공되며 인출과정에서 얻은 정보로부터 인출된 화폐를 구성할 수 있도록 하여 그 화폐가 나중에 입금될 때 식별할 수 있게 해준다. 인출자 추적은 돈세탁과 같은 범죄에 대항하기 위해 제공되며 입금된 화폐에서 고객의 신원 정보를 계산할 수 있게 해준다.

**화폐 추적.** 은행은 인출과정에서 확인한 $CT$를 신뢰기관에게 전달하여 화폐 추적을 요청한다. 신뢰기관은 법원의 승인이 있으면 화폐 추적을 위한 개인키 $x_{CT}$를 이용하여 $(CT)^{x_{CT}} = g_B^r$을 계산하고, 이 값을 은행에게 돌려준다. 은행은 받은 값을 이용하여 다음과 같이 인출된 화폐를 구성할 수 있다.

$$C = \tilde{C} / g_B^r$$

**인출자 추적.** 은행은 입금된 수표의 $OT$를 신뢰기관에게 전달하여 인출자 추적을 요청한다. 신뢰기관은 법

원의 승인이 있으면 인출자 추적을 위한 개인키 $x_{OT}$를 이용하여 $(OT)^{x_{OT}} = g_T^r$를 계산하고, 이 값을 은행에게 돌려준다. 은행은 $g_T^r$, $c_0$, $l$를 이용하여 다음과 같이 고객의 공개신원정보를 계산한다.

$$y_U = C / (g_R^{c_0} g_L^l g_T^r)$$

### 3.7 환불 프로토콜

고객은 해쉬체인을 사용하고 남은 부분을 은행에 전달하여 환불받을 수 있다. 이 때 고객은 먼저 그림 7에 기술되어 있는 프로토콜을 사용하여 환불티켓을 받은 후에 이 티켓을 나중에 은행에 전달하여 환불받게 된다. 이 프로토콜에서는 Chaum의 온라인 수표시스템[20]에서 사용한 쿠키통을 환불티켓으로 사용한다. 쿠키통은 환불받을 것을 축적하는 값으로서 환불액을 나타내기 위해 여러 개의 RSA공개지수를 이용한다. 이 논문에서는 RSA 공개지수 3, 5, 7, 11, 13을 이용하며, 이들은 각각 100원, 200원, 400원, 800원, 1600원을 나타낸다. 이렇게 소수를 공개지수로 사용하는 이유는 하나의 지수를 다른 지수의 조합으로 표현할 수 없도록 하기 위함이다. 만약 이것이 가능하면 환불액을 조작할 수 있기 때문이다. 티켓이 공개지수에 대응하는 비밀지수로 암호화되어 있으면 해당 금액이 축적되어 있는 것을 나타낸다. 따라서 $H_n(y_U \| N_U)^{\frac{1}{3\cdot5\cdot7\cdot11}}$은 1500원이 축적되어 있는 것이고, $H_n(y_U \| N_U)^{\frac{1}{3\cdot7}}$은 500원이 축적되어 있는 것을 나타낸다. 여기서 $N_U$는 난스(nonce)로서 이중청구를 못하도록 하기 위해 사용되며, $H_n$은 $\{0,1\}^*$를 입력받아 $Z_n$의 임의의 원소로 매핑해주는 충돌회피 해쉬함수이다.

그림 7에 기술된 환불티켓 인출 프로토콜에서 고객은 이미 300원을 환불받을 수 있는 티켓을 가지고 있고, 이 프로토콜을 수행하여 500원을 티켓에 추가하게 된다. 자세한 진행과정은 다음과 같다.

- **단계 1**: 고객은 임의로 선택한 은닉요소 $r'$을 요청하는 환불액에 해당하는 공개지수로 거듭제곱하고, 그 수를 기존 티켓에 곱하여 티켓을 은닉한다. 그 결과를 $\tilde{T}$라 한다.
- **단계 2**: 고객은 지불 프로토콜과 유사한 방법으로 해쉬체인의 남은 부분을 은행에 지불함으로써 환불티켓을 요청한다.
  이 때 지불 프로토콜과 다른 점은 시도값을 만들 때 $\tilde{T}$를 포함한다는 것이다. 이것은 전달되는 $\tilde{T}$를 공격자가 바꾸어 고객 대신에 환불 티켓을 인출받는 것을 막기 위한 조치이다.
- **단계 3**: 은행은 $C$의 서명, 해쉬체인의 최종 동전 $c_l$의 유효성, $C$의 표현을 고객이 알고 있는지 확인한다. 또한 입금 데이타베이스에서 $C$를 검색하여 $C$를 이용한 기존 지불과 환불 요청한 부분이 중첩되는지 검사한다.
- **단계 4**: 은행은 환불액에 해당하는 비밀지수로 $\tilde{T}$에 서명하여 고객에게 돌려준다. 고객은 은닉요소를 제거하고 환불티켓을 확인한다.

고객은 나중에 $T$, $y_U$, $N_U$, 환불액, 이 금액을 나타내기 위해 각 공개지수가 몇 번 사용되었는지를 은행에 전달하여 실제로 환불을 받게 된다.

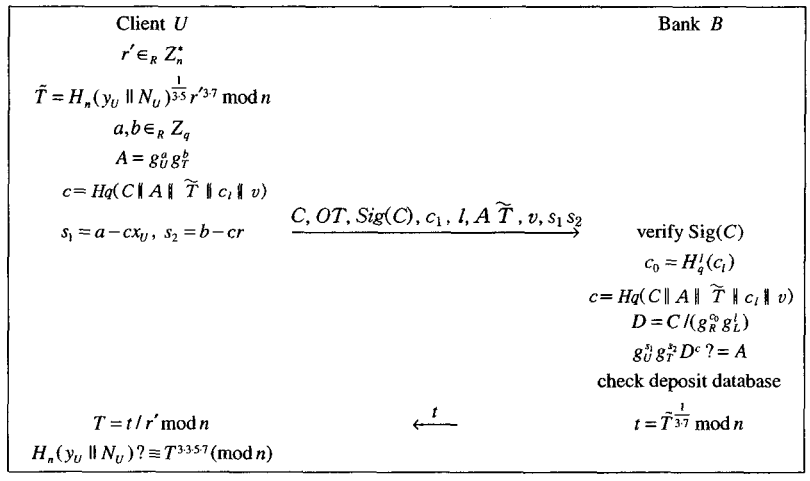


그림 7 환불티켓 인출 프로토콜

## 4. 시스템 분석

이 장에서는 제안된 시스템의 안전성을 분석하고, 기존 해쉬체인 기반 시스템과 분할 가능한 화폐 시스템과 각각 비교하여 장단점을 논의한다.

### 4.1 안전성

**가정 2.** 이떤 기저에 대한 어떤 값의 이산대수를 모를 때 그 값의 이산대수를 알고 있음을 비상호작용으로 영지식 증명을 하는 것은 계산적으로 어렵다[21].

**가정 3.** 어떤 생성자 튜플에 대한 어떤 값의 표현을 모를 때 그 값의 표현을 알고 있음을 비상호작용으로 영지식 증명하는 것은 계산적으로 어렵다[22].

**가정 4.** RSA 은닉서명 프로토콜을 다항 번(병행으로 또는 순차적으로) 수행하여도 RSA 서명을 위조하는 것은 계산적으로 어렵다[23].

위 세 가지 가정의 정당성에 관한 논의는 이 논문의 범위를 벗어나며, 각 가정의 참고문헌을 참조하면 보다 자세한 내용을 얻을 수 있다.

**정리 5.** 이 시스템에서 다른 고객 행세를 하여 화폐를 인출하는 것은 계산적으로 어렵다.

*증명.* 이 시스템에서 화폐를 인출하기 위해서는 먼저 자신의 신원을 증명할 수 있어야 한다. 이 증명은 매번 새로운 타임스템프를 시도값에 포함하여야 하는 이산대수에 관한 비상호작용 영지식 증명이다. 따라서 이산대수 가정과 가정 2에 의해 다른 고객으로 신원을 증명하는 것은 계산적으로 어렵다. 또한 증명에 타임스템프가 포함되므로 공격자는 기존 증명을 다시 사용할 수 없다. □

**정리 6.** 인출 프로토콜을 다항 번(순차적으로 또는 병행으로) 반복 수행하여도 이 시스템에서 화폐를 위조하는 것은 계산적으로 어렵다.

*증명.* 이 시스템에서 화폐는 Solages와 Traore의 제한적 은닉서명 프로토콜을 통해 발행된다. 만약 서명키를 모르는 공격자가 서명을 직접 위조하였다면 이는 기저 $g_B$에 대한 은행의 공개키 $y_B$의 이산대수를 계산하였다는 것을 의미한다. 그러나 이것은 이산대수 가정에 위배된다. 만약 서명키를 모르는 공격자가 인출 프로토콜을 순차적으로 또는 병행으로 다항 번 반복 수행하여 위조하였다면 이것은 가정 1에 위배된다. 따라서 이 시스템에서 화폐를 위조하는 것은 계산적으로 어렵다. □

**정리 7.** 이 시스템에서 인출과정을 공격하여 화폐를 조작하는 것은 계산적으로 어렵다.

*증명.* 이 시스템의 화폐는 고객 정보, 체인 정보, 추적 정보로 구성되어 있다. 체인 정보는 체인의 루트와 길이로 구성되어 있다. 이들 정보는 같은 형태로 화폐에 표시되어 있으므로 이 중 한 가지에 대한 조작 공격이 가능하면 나머지도 같은 방법으로 조작할 수 있다. 따라서 이 증명에서는 고객 정보에 대해서만 조작할 수 없음을 보인다. 은행은 은닉서명의 입력을 스스로 만들어 사용하므로 이런 공격을 하기 위해서는 Proof($E$)나 제한적 은닉서명 자체를 공격하여야 한다. 각 경우에 대해 설명하면 다음과 같다.

- Proof($E$)에 대한 공격: 기존 $E$ 대신에 $E' = Eg_U^r$을 전달하여 증명에 성공할 수 있으면 최종으로 얻게 되는 화폐는 $C = g_U^{x_U} g_R^{c_0} g_L^l g_T^r$이 아니라 $C' = g_U^{x_U + r} g_R^{c_0} g_L^l g_T^r$을 얻게 된다. 공격자는 $E$에 대한 표현을 증명해야 하므로 기저 $g_R$에 대한 $g_U$의 이산대수 $\log_{g_R} g_U$를 알아야 은행을 속일 수 있다. 그러나 이것은 이산대수 가정에 의해 계산적으로 어렵다.
- BlindSig에 대한 공격: 정리 4에 의해 서명의 입력값 $\tilde{c}$가 결정된 후에는 생성자 $g_B$만을 이용하여 $C$의 서명을 얻을 수 있다. 또한 기저 $g_B$에 대한 $g_U$의 이산대수 $\log_{g_B} g_U$를 계산하는 것이 어려우므로 $g_U$로 표현되는 고객의 신원정보를 은닉서명 과정에서 변경할 수 없다.

따라서 이 시스템에서 인출과정을 공격하여 화폐를 조작하는 것은 계산적으로 어렵다. □

**정리 8.** 이 시스템에서 은행이 발행된 화폐를 추적하는 것은 계산적으로 어려우며, 이것은 상점과 공모하여도 마찬가지이다. 같은 해쉬체인을 이용한 고객의 지불은 서로 연결되지만 다른 체인을 이용한 고객의 지불은 연결할 수 없다.

*증명.* 은행은 인출과정에서 $\tilde{c}$, $CT$, $E$, Proof($E$), $l$을 얻으며, 고객은 $C$와 Sig($C$)를 얻게 된다. 정리 3에 의해 은행은 서명과정에서 얻은 정보로부터 메시지나 서명 결과에 대한 어떤 정보도 얻을 수 없다. 뿐만 아니라 Proof($E$)로부터 $C$에 대한 어떤 정보도 얻을 수 없다. 이것은 이산대수 $\log_{y_{CT}} CT$를 계산할 수 없기 때문이다. 나중에 화폐가 입금되어 $C$와 $C$를 구성하는 $c_0$와 $l$을 알게 되어도 정리 1에 의해 나머지 $x_U$와 $r$값을 알 수 없다. 또한 은닉서명을 받을 때 포함한 $OT$를 보지 못하므로 이 값을 이용하여 수표를 추적할 수도 없다. 뿐만 아니라 익명으로 지불이 이루어지면 상점은 지불자에 대한 어떤 정보도 얻을 수 없다. 따라서 은행은 상점과 공모를 하여도 지불에 사용된 수표의 인출자를 알 수 없다.

같은 고객이 인출한 $C$와 $C'$이 입금되어도 이 둘을 연결할 정보가 없다. 물론 이 두 값에는 같은 고객 신원정보가 들어있지만 정리 1에 의해 고객을 제외하고는 계산할 수 없다. 따라서 해쉬체인 때문에 같은 체인을 이용한 두 지불은 연결할 수 있지만 서로 다른 체인을 이용한 지불은 연결할 수 없다. □

**정리 9.** 그림 5의 프로토콜을 이용하여 $C$를 지불하기 위해서는 생성자 튜플 $(g_U, g_R, g_L, g_T)$에 대한 $C$ 의 표현과 $C$에 포함된 체인의 구성을 알아야 한다. 그림 6의 프로토콜을 이용하여 $C$를 지불하기 위해서는 기저 $y_{OT}$에 대한 $OT$의 이산대수와 $C$에 포함된 체인의 구성을 알아야 한다.

*증명.* 그림 5의 지불 프로토콜에서는 고객은 $C$의 표현을 알고 있음을 영지식으로 증명한다. 가정 3에 의해 표현을 알지 못하면 이 증명에 성공할 수 없다. 그림 6의 지불 프로토콜에서는 $\log_{y_{OT}} OT$를 키로 여러 값에 Schnorr 서명을 한다. Schnorr 서명이 안전하다고 가정하면 $\log_{y_{OT}} OT$를 알고 있는 사람만이 지불할 수 있다. 그런데 $C$의 표현과 $\log_{y_{OT}} OT$는 정리 1과 이산대수 가정에 의해 오직 인출자만이 알 수 있다. 따라서 원래 인출자를 제외하고는 $C$를 이용하여 지불할 수 없다. □

**따름정리 2.** 다른 상점과 공모를 하여도 상점은 자신에게 지불된 동전만 입금할 수 있다.

*증명.* 상점은 $C$의 표현을 알 수 없고, $\log_{y_{OT}} OT$를 알 수 없으므로 정리 9에 의해 고객과 실제 수행하여 얻은 지불 트랜스크립트 외에 유효한 또 다른 트랜스크립트를 만들 수 없다. 그림 5의 지불 프로토콜에서는 상점 식별자를 시도값 계산에 포함하며, 그림 6의 지불 프로토콜에서는 상점 식별자를 서명에 포함한다. 따라서 다른 상점으로부터 받은 동전을 입금할 수도 없다. □

**따름정리 3.** 만약 고객이 지불 프로토콜을 정직하게 수행하고 이중사용을 하지 않으면 은행은 이중사용에 대한 누명을 고객에게 씌울 수 없다.

*증명.* 이 따름정리는 따름정리 2와 유사하게 증명할 수 있다. □

**정리 10.** 환불티켓 인출 프로토콜을 다항 번 반복 수행하여 환불티겟을 위조하는 것은 계산적으로 어렵다.

*증명.* 환불티켓은 RSA 은닉서명을 통해 발행되므로 만약 환불티켓 인출 프로토콜을 다항 번 수행하여 환불티켓을 위조할 수 있다면 이것은 가정 4에 위배된다. □

**정리 11.** 생성자 튜플 $(g_U, g_R, g_L, g_T)$에 대한 $C$의 표현과 $C$에 포함된 체인의 구성을 알고 있는 사람만이 $C$에 사용되지 않은 동전에 대한 환불티켓을 인출할 수 있다.

*증명.* 이 증명은 정리 9와 유사하게 증명할 수 있다. □

**따름정리 4.** 환불을 먼저 한 다음에 지불하거나 지불한 다음에 환불하여 초과환불받을 수 없으며, 은행은 고객에게 초과환불하였다고 누명을 씌울 수 없다. 뿐만 아니라 같은 티켓으로 이중환불을 받을 수 없다.

*증명.* 은행은 이미 처리한 환불티켓을 보관하므로 이중환불과 초과환불을 쉽게 발견할 수 있다. 더욱이 환불은 실명으로 이루어지므로 이렇게 부정하는 고객을 식별할 수 있다. 이 시스템에서 초과환불은 이중사용과 같으므로 따름정리 3과 같은 이유에서 은행은 고객에게 초과환불하였다고 누명을 씌울 수 없다. □

### 4.2 기존 해쉬체인 기반 시스템과의 비교

기존 해쉬체인 기반 시스템과의 비교는 표 2에 요약되어 있다. 이 절에서는 Nguyen 등의 시스템[11], 일반 동전방식의 오프라인 익명화폐[5]와의 비교에 대해서만 설명한다. 새 시스템과 Nguyen 등의 시스템은 길이가 $l$인 해쉬체인을 인출한다고 가정하며, 일반 동전방식의 화폐는 $l$개의 동전을 인출한다고 가정[1]하고 비교한다. 먼저 인출과정을 비교하면 새 시스템은 길이가 $l$인 해쉬체인을 인출할 때 단지 한 번의 은닉서명을 수행한다. 반면에 Nguyen 등의 시스템이나 일반 오프라인 동전에서는 $l$번의 은닉서명을 수행하여야 한다. 지불과정에서 $v$개의 동전에 해당하는 금액을 한번에 지불하는 경우(최상의 지불형태)와 $v$개의 동전을 하나씩 지불하는 경우(최악의 지불형태)를 비교하면 다음과 같다. 전자의 경우, 새 시스템은 $C$에 대한 서명확인과 한 번의 시도와 응답을 수행한다. Nguyen 등의 시스템에서는 $v$개의 동전 모두에 대해 은행의 서명을 확인해야 한다. 다만 시도와 응답은 첫 번째 동전에 대해서만 수행한다. 후자의 경우, 새 시스템은 첫 동전 이후 각 동전마다 Schnorr 서명을 하여 상점에게 전달하여야 한다. Nguyen 등의 시스템은 전자 때와 같은 비용이 소요된다. 일반 동전방식의 경우에는 전자나 후자나 모두 모든 동전에 대해 은행의 서명을 확인해야 하고 각 동전에 대해 시도와 응답을 수행하여야 한다. 따라서 최악의 경우 새 시스템에서 고객은 서명 생성 때문에 지불비용이 부담될 수 있지만 상점 측면에서는 Nguyen 등의 시스템과 차이가 없으며, 고객 측면에서도 인출과 지불 비용을 합한 총 비용은 더 저렴하다. 새 시스템이나 Nguyen 등의 시스템은 해쉬체인을 사

1) 실제로 일반 동전방식에서는 다양한 액면가의 동전을 인출하여 사용할 수 있지만 여기서는 동전의 액면가가 같다고 가정한다. 따라서 비교가 공평하지 않을 수도 있다.

표 2 기존 해쉬체인 기반 지불시스템과의 비교

| | 오프라인: ○ 온라인: □ | 선불: ○ 후불: □ | 범용: ○ 판매자 전용: □ | 익명성 | 가변금액 지불 | 인출비용 | 지불비용 |
|---|---|---|---|---|---|---|---|
| Payword | ○ | □ 신용한도: No | □ | × | ○ | × | - 체인루트에 대한 서약(서명) 확인<br>- 나머지: 해쉬연산 |
| iKP 기반 소액지불 시스템 | △ | □ 신용한도: Yes | □ | × | ○ | × | - 체인루트에 대한 서약(서명) 확인<br>- 신용한도 확인<br>- 나머지: 해쉬연산 |
| Mao의 시스템 | ○ | ○ 환불: No | ○ 상점이 거스름 | △ 가명 사용 | ○ | 한번 은닉서명 | - 여러 상점에게 지불할수록 그 다음 상점이 확인하여야 하는 정보가 많음 |
| Nyugen의 이중잠금 해쉬체인 | ○ | ○ 환불: No | ○ 이중잠금 해쉬체인 | × | ○ | 한번 일반서명 | - 체인루트에 대한 서약(서명) 확인<br>- 나머지: 해쉬연산<br>- 영수증 발급 |
| Nyugen의 시스템 | ○ | ○ 환불: No | ○ 각 동전에 서명 | ○ 각 동전에 은닉서명 | × | $l$ 번 은닉서명 | - 첫 동전: 서명과 시도와 응답<br>- 나머지: 서명 |
| **제안된 시스템** | ○ | ○ **환불: Yes** | ○ **지불대금에 서명** | ○ **루트에 은닉서명** | ○ | **한번 은닉서명** | - **첫 지불: 서명과 시도와 응답**<br>- **나머지: 서명** |
| 일반 오프라인 동전방식 | ○ | ○ 환불: Yes | ○ 각 동전에 서명 | ○ 완전한 연결불가능성 | × | $l$ 번 은닉서명 | - 각 동전마다 서명과 시도와 응답 |

용하므로 같은 체인을 이용한 여러 지불이 서로 연결된다는 점이 일반 동전방식의 오프라인 익명화폐와 다른 점이다. 익명성에 대한 약간의 희생이 있지만 그 결과 지불의 효율성은 높아졌다.

Nguyen 등의 시스템과 새 시스템은 둘 다 선불방식이지만 Nguyen 등은 사용하고 남은 동전을 환불하는 방법을 제공하지 않고 있다. 그러나 선불방식이므로 이런 서비스는 필요하다. 기존 오프라인 동전처럼 단순하게 동전을 반납함으로써 환불받을 수 있다고 생각할 수 있지만 해쉬체인으로 동전들이 연결되어 있으므로 이렇게 환불받으면 기존 지불의 익명성이 깨진다. 새 시스템은 사용하고 남은 부분을 환불받을 수 있도록 하였으며, 이 때 기존 지불의 익명성은 계속 유지된다. 특히 지불액과 환불액 간에 차액 때문에 고객을 추측할 수 있다는 문제점을 극복하기 위해 환불액을 축적할 수 있는 방법을 사용하고 있다. 그러나 환불받기 위해 은행에 다시 접촉하여야 하는 단점이 있다.

### 4.3 기존 분할 가능한 화폐와 비교

대부분의 분할 가능한 화폐[12-14,24]는 이진트리 구조를 이용하고 있다. 이 이진트리 구조는 인출하기 전에 미리 만드는 구조가 아니며 지불할 때 필요한 노드만 만들어 사용한다. $N$이 분할정밀성(총금액/최소분할단위금액)이면 [13,14,24]에서 모든 절차의 복잡성은 $O(\lg N)$이 된다. 만약 10원이 최소분할단위이고 화폐의 액면가가 만원이면 $N=1000$이 되며, 이것을 이진트리 구조로 나타내고자 하면 11개의 단계가 필요하다. 즉, 10개의 노드만 사용하면 만원 이하의 어떤 지불대금도 나타낼 수 있다. 반면에 새 시스템은 해쉬체인을 이용하므로 같은 형태의 화폐를 만들고자 하면 체인의 길이가 1000이 되어야 한다. 그러나 새 시스템에서는 어떤 임의의 대금을 지불하기 위해 그만큼의 노드를 전달할 필요가 없으며, 그 만큼의 시도와 응답을 할 필요도 없다. 항상 최종 값 하나만 전달하며, 시도와 응답을 한번 수행하거나 서명을 하나 생성해야 한다. 기존의 분할 가능한 화폐는 사용한 노드의 개수만큼 시도와 응답을 해야 하므로 요구되는 평균 연산의 양 측면에서는 새 시스템이 더 효율적이다. 새 시스템이나 대부분의 분할 가능한 화폐는 분할된 화폐의 연결불가능성을 제공하지 못한다. [24]는 분할된 화폐를 상호 연관시킬 수 없지만, 여기서 사용하는 이중이산대수(double discrete logarithm) 증명은 cut-and- choose 기법에 기반한 증명 프로토콜을 사용한다. 따라서 효율성이 떨어지는 시스템이다.

## 5. 결 론

해쉬체인은 저렴한 해쉬함수를 이용한 구조이므로 전자화폐에 사용하기에 매력적인 구조이다. 그러나 해쉬체인을 이용한 초창기 시스템[4,6]은 고객의 프라이버시를 보호하지 못하며, 한 판매자에게만 지불할 수 있는 판매자 전용화폐이었다. 해쉬체인의 장점을 유지하면서 선불방식의 범용화폐로 전환하기 위한 노력이 있었지만 이 시스템들은 이중사용 문제를 효율적으로 해결하지 못하였기 때문에 해쉬체인의 장점을 제대로 반영하지 못하고 있다[9-11]. 이 논문에서는 이런 문제를 해결한 범용, 선불, 익명, 오프라인 방식의 화폐를 제안하였다.

새 시스템은 해쉬체인과 표현 문제를 이용하여 화폐를 구성하며, 고객은 인출하는 해쉬체인의 길이와 무관하게 항상 Solages와 Traore의 제한적 은닉서명 프로토콜을 한번 수행하여 화폐를 인출한다. 이 시스템은 화폐의 추적불가능성은 제공하지만 해쉬체인 때문에 같은 체인을 이용한 지불의 연결불가능성은 제공하지 못한다. 그러나 다른 체인을 이용한 지불은 서로 연결할 수 없다. 지불과정에서 고객은 지불대금과 상관없이 상점과의 첫 거래에서는 한 번의 시도와 응답을 하고, 그 이후 거래에서는 한 번의 서명을 생성하여 지불한다. 상점은 고객과 첫 거래에서는 한 번의 서명확인과 시도와 응답을 수행하고, 그 이후 거래에서는 서명 확인만으로 지불을 확인한다. 따라서 기존 동전방식의 화폐나 해쉬체인을 이용한 범용화폐보다 효율적으로 지불할 수 있다. 해쉬체인의 사용 때문에 복잡해지는 이중사용 문제는 익명의 화폐가 범죄에 악용되는 것을 막기 위한 추적 기능을 이용하여 해결함으로써 인출과 지불 비용을 최소화하였다.

기존의 분할 가능한 화폐에서는 환불에 대한 언급이 없다. 이들은 범용화폐이므로 환불이 필요 없다고 생각할 수 있지만 일반적으로 화폐는 유효기간이 있기 때문에 선불방식에서는 환불기능을 제공해야 한다. 그런데 분할된 각 화폐가 서로 연관되는 화폐에서는 지불된 화폐의 익명성을 유지하면서 환불을 제공하기가 쉽지 않다. 이 논문에서는 환불티켓 방식이라는 새로운 개념의 환불방법을 사용하여 환불 문제를 해결하였다. 이 방법은 환불과정을 인출이나 지불과정과 독립적으로 수행한다. 또한 지불액과 환불액 사이의 차액을 통한 연결고리를 없애기 위해 환불을 축적하는 방법을 사용하고 있다. 이 시스템에서는 Chaum이 제안한 쿠키통 방식을 사용하여 환불티켓을 제공하지만 이 방식은 많은 공개키 연산이 소요되며, 축적된 환불액을 확인하기 위해 각 공개지수가 몇 번 사용되었는지 밝혀야 하는 번거로움이 있다. 따라서 이 방식보다 효율적이고 이상적인 축적 방법에 대한 연구가 필요하다.

## 참 고 문 헌


[1] Lamport, L., "Password Authentication with Insecure Communication," *Comm. ACM*, Vol. 24, No. 11, pp. 770-772, 1981.

[2] Rivest, R.L. and Shamir, A., "PayWord and MicroMint - Two Simple Micropayment Schemes," *Proc. of the 1996 Int. Workshop on Security Protocols*, LNCS 1189, pp. 69-87, Springer, 1997.

[3] Pedersen, T.P., "Electronic Payments of Small Amounts," *Proc. of the 1996 Int. Workshop on Security Protocols*, LNCS 1189, pp. 59-68, Springer, 1997.

[4] Anderson, R.J., Manifavas, C., and Sutherland, C., "NetCard - A Practical Electronic-Cash System," *Proc. of the 1996 Int. Workshop on Security Protocols*, LNCS 1189, pp. 49-57, Springer, 1996.

[5] Brands, S., "Untraceable Off-Line Cash in Wallets with Observers," *Advances in Cryptology, Crypto 1993*, LNCS 773, pp. 302-318, Springer, 1994.

[6] Hauser, R., Steiner, M., and Waidner, M., "Micro-payments based on *i*KP," *Proc. of the 14th Worldwide Congress on Computer and Communications Security and Protection, SECURICOM 1996*, pp. 67-82, 1996.

[7] Mu, Y., Varadharajan, V., and Lin, Y., "New Micropayment Schemes Based on PayWords," *Proc. of the 2nd Australasian Conf. on Information Security and Privacy, ACISP 1997*, LNCS 1270, pp. 283-293, Springer, 1997.

[8] Yen, S. and Zheng, Y., "Weighted One-Way Hash Chain and Its Applications," *Proc. of the 3rd Int. Workshop on Information Security, ISW 2000*, LNCS 1975, pp. 135-148, Springer, 2000.

[9] Mao, W., "Lightweight Micro-Cash for the Internet," *Proc. of the 1996 European Symp. on Research in Computer Security, ESORICS 1996*, LNCS 1146, pp. 15-32, Springer, 1996.

[10] Nguyen, K.Q., Mu, Y., and Varadharajan, V., "Micro-Digital Money for Electronic Com- merce," *Proc. of the 13th IEEE Annual Computer Security Applications Conf.*, pp. 2-8, IEEE Computer Society Press, 1997.

[11] Nguyen, K.Q., Mu, Y., and Varadharajan, V., "Secure and Efficient Digital Coins," *Proc. of the 13th IEEE Annual Computer Security Applications Conf.*, pp. 9-15, IEEE Computer Society Press, 1997.

[12] Okamoto, T. and Ohta, K., "Universal Electronic Cash," *Advances in Cryptology, Crypto 1991*, LNCS 576, pp. 324-337, Springer, 1992.
[13] Okamoto, T., "An Efficient Divisible Electronic Cash Scheme," *Advances in Cryptology, Crypto 1995*, LNCS 963, pp. 438-451, Springer, 1995.
[14] Chan, A., Frankel, Y., and Tsiounis, Y., "Easy Come - Easy Go Divisible Cash," *Advances in Cryptology, Eurocrypt 1998*, LNCS 1403, pp. 561-575, Springer, 1998.
[15] de Solages, A. and Traore, J., "An Efficient Fair Off-line Electronic Cash System with Extensions to Checks and Wallets with Observers," *Proc. of the 2nd Int. Conf. on Financial Cryptography, FC 1998*, LNCS 1465, pp. 275-295, Springer, 1998.
[16] Chaum, D. and Pedersen, T.P., "Wallet Databases with Observers," *Advances in Cryptology, Crypto 1992*, LNCS 740, pp. 89-105, Springer, 1993.
[17] Schnorr, C.P., "Security of Blind Discrete Log Signatures against Interactive Attacks," *Proc. of the 3rd Int. Conf. on Information and Communications Security, ICICS 2001*, LNCS 2229, pp. 1-13, Springer, 2001.
[18] Pointcheval, D. and Stern, J., "Security Arguments for Digital Signatures and Blind Signatures," *J. of Cryptology*, Vol. 13, No. 3, pp. 361-396, 2000.
[19] Abe, M. and Camenisch, J., "Partially Blind Signature Schemes," *Proc. of the 1997 Symp. on Cryptography and Information Security Workshop, SCIS 1997*, SCIS97-33D, 1997.
[20] Chaum, D., "Online Cash Checks," *Advances in Cryptology, Eurocrypt 1989*, LNCS 434, pp. 288-293, Springer, 1990.
[21] Schnorr, C.P., "Efficient Signature Generation by Smart Cards" *J. of Cryptology*, Vol. 4, No. 3, pp. 161-174, 1991.
[22] Brands, S., "An Efficient Off-line Electronic Cash System Based on the Representation Problem," CWI(Centrum voor Wiskunde en Informatica) Technical Report, CS-R9323, 1993.
[23] Bellare, M., Namprempre, C., Pointcheval, D., and Semanko, M., "The Power of RSA Inversion Oracles and the Security of Chaum's RSA-based Blind Signature Scheme," *Proc. of the 5th Int. Conf. on Financial Cryptography, FC 2001*, 2001.
[24] Nakanishi, T. and Sugiyama, Y., "Unlinkable Divisible Electronic Cash," *Proc. of the 3rd Int. Workshop on Information Security, ISW 2000*, LNCS 1975, pp. 121-134, Springer, 2000.

김 상 진
1995년 2월 한양대학교 전자계산학과 졸업. 1997년 2월 한양대학교 전자계산학과 석사. 2002년 8월 한양대학교 컴퓨터공학과 박사. 2003년 3월~현재 한국기술교육대학교 인터넷미디어공학부 전임강사. 관심분야는 정보보호, 전자화폐, 암호프로토콜

오 희 국
1983년 한양대학교 전자공학과 졸업 1989년 아이오아주립대학 전자계산학과 석사. 1992년 아이오아주립대학 전자계산학과 박사. 1993년~1994년 한국전자통신연구원 선임연구원. 1995년~현재 한양대학교 전자컴퓨터공학부 부교수. 관심분야는 정보보호, 암호프로토콜응용, 분산컴퓨팅