식별 불능의 활용 가능 정보만을 저장하는 서버의 머클 루트를 활용한 블록체인 기반 사용자 정보 검증

(A Blockchain-based Verification of User's Information using Merkle Roots by a Server that Only Stores Unidentifiable but Utilizable Information)

박준철*
(Jun-Cheol Park)

요 약

최근 쿠팡 사태까지 이어진 일련의 인터넷 사이트 서버 해킹 사태에서 보듯이 사이트마다 회원 가입을 유도하고 개인정보를 저장하는 형태가 변하지 않는 한, 서버 보안을 강화하더라도 이런 사태는 언젠가 또 발생할 것이다. 이에 개인정보 보안을 위한 접근 방식의 근원적 변화가 필요하게 되었다. 본 논문에서는 사이트에서 정보의 소유자 특정은 불가능한 수준의 정보만을 서버가 보유하면서도 개인별 서비스(구매 내용 등) 패턴을 분석하여 맞춤 서비스 제안 등은 할 수 있으며, 서비스 제공에 요구되는 사용자 정보 검증을 사이트의 인증 서버가 안전하고 신속하게 실행하는 기법을 제시한다. 이 기법에서는 제안하는 블록체인 트랜잭션의 머클(Merkle) 루트 값만을 활용해 서버 내의 정보 사용 없이 제공된 사용자 정보 요소를 검증한다. 검증 기록은 블록체인에 저장되어, 서버 또는 사용자 누구나 유실 걱정 없이 불변의 저장된 기록을 언제든 확인할 수 있다.

■ 중심어 : 식별 불능 정보 ; 검증 ; 블록체인 ; 머클 루트 ; 보안

Abstract

Recent large-scale server hacking incidents, such as the Coupang case, demonstrate that security breaches are likely to recur as long as websites continue to require user registration and store extensive personal information. Even with strengthened server-side security, these incidents remain inevitable, indicating the need for a fundamental shift in approaches to personal data protection. We propose a novel scheme in which servers retain only unidentifiable information while still enabling personalized services such as the analysis of individual service usage and purchase patterns. User information verification required for service provision is securely and quickly executed by a verifying server. The scheme utilizes only the Merkle root of blockchain transactions to verify user information elements without the use of information in the server. Verification records are immutably stored on the blockchain, allowing both service providers and users to retrieve tamper-resistant verification logs at any time without the risk of data loss.

■ keywords : unidentifiable information ; verification ; blockchain ; Merkle root ; security

Ⅰ. 서 론

최근 쿠팡 개인정보 유출 사례에서 보듯이 전자상거래, 통신사, 웹사이트 서버의 해킹 사건은 꾸준히 이어지고 있다. 해킹 발생 시 업체의 대처는 대개 서버 보안성을 높이고 직원 관리를 철저히 하며 보안에 더욱 투자하겠다는 정도였다. 그런 개선책이 물론 도움이 되지만, 현재처럼 사이트마다 회원 가입을 받고 회원 정보와 서비스 내용을 보유하는 한, 새로운 기술로 더 정교하게 설계된 공격에 피해를 볼 가능성은 늘 존재한다.

전자상거래 등 서비스 제공사가 고객 관련 정보를 수집하고 보유하려는 것은 사용자 인증뿐 아니라 고객별 마케팅이나 제품 또는 서비스의 소비 형태 통계 분석 및 AI 모델 개발 등에 활용하려는 이유가 크다. 또한 추후 고객 문의나 민원에 대비하려는 목적도 있다. 이런 서비스 제공

* 종신회원, 홍익대학교 컴퓨터공학과
이 논문은 2025학년도 홍익대학교 학술연구진흥비에 의하여 지원되었음.
접수일자 : 2026년 01월 13일
수정일자 : 2026년 02월 10일
게재확정일 : 2026년 03월 02일
교신저자 : 박준철 e-mail : jcpark@hongik.ac.kr

사의 합리적 요구를 무시한 채, 유출 시 피해를 막고자 사용자 정보를 전혀 보유하지 못하게 함은 사업자가 수용하기 어려울 것이다.

본 논문의 제안 기법은 서비스 제공사 서버가 개인을 특정할 정보를 저장하지 않으면서도 맞춤형 추천이나 통계 분석 및 AI 모델에 활용될 수 있는 데이터를 보유하도록 한다. 그리고 사용자는 서비스 사이트마다 회원 가입을 하거나 패스워드를 설정하고 기억할 필요 없이, 필요할 때만 서비스나 구매에 요구되는 최소한의 정보만을 제시하고 서버에게 안전하고 신속하게 검증받을 수 있다. 검증 과정에선 제안하는 블록체인(Merkle-Roots Chain: 이하 MRChain) 트랜잭션의 머클 루트 값을 통해 사용자가 제시한 정보 요소(들)값의 유효성을 확인한다. 서버는 또한 검증 기록을 블록체인에 남기며, 내, 외부 유출 시에도 소유자로의 연결 불가능한 형태로 제공한 서비스 내용을 데이터베이스에 기록한다.

본 논문의 구성은 다음과 같다. 2장에서 블록체인의 서버 인증 데이터베이스 역할 관련 연구 결과들을 제시하고, 3장에서 참조 대상인 비트코인 시스템과 비교한 MRChain을 트랜잭션 측면에서 설명한다. 4장에서 개인정보 요소(들) 등록, 검증, 변경, 폐기 프로토콜을 설명하고, 5장에서 제안 방식의 보안성과 검증 기능의 성능 관련 분석을 서술하며, 6장에서 결론을 맺는다.

Ⅱ. 관련 연구

사용자 인증을 위한 정보의 저장소 역할로서의 서버 데이터베이스를 대체하거나 보완하려는 블록체인 활용 관련 기존 연구 결과를 분석한다.

첫째, 블록체인의 스마트 계약 또는 트랜잭션 등의 저장 정보를 인증의 도구로 삼는 연구를 살펴본다. 사용자가 개인정보를 암호화시켜 분산형 파일 시스템인 IPFS(Inter-Planetary File System)에 저장한 후, 인증을 요구하는 웹사이트에는 이더리움(Ethereum) 스마트 계약을 통해 필요한 정보의 검증 권한을 부여하는 방식이 발표되었다[1]. 다만 논문 [1]에서 저장 시 정보를 초기 검증하는 방법이나 절차는 제시되지 않았다. 또한 트랜잭션을 통해 사용자가 어떤 서비스 제공자에게 등록(인증)된 후 스마트 계약 내의 등록된 사용자 크리덴셜을 통해 다른 서비스 제공자에게 별도 등록 절차 없이 인증받는 기법도 제시되었다[2]. 그리고 한 특정 MetaMask 지갑의 계정으로부터 어떤 스마트 계약에 접근하는 트랜잭션을 발생시킴으로써 해당 사용자가 이중(2-factor) 인증의 한 요소로 블록체인을 활용하도록 하는 연구도 등장했다[3]. 제안 기법과 달리, 이런 연구들은 스마트 계약을 사용하므로 생성과 사용에 수수료 부담이 발생하여 기존의 무비용 인증에 익숙한 사용자나 웹사이트에서 수용하기 어렵다. 또 다른 연구로, 한 기관 내에서 사용자에게 여러 자원에 대한 접근권한을 이더리움 스마트 계약으로 제어하는 기법이 제시되었는데[4], 사용자 인증의 범위와 목적이 한 기관 내의 자원 접근 제어라는 점에서 제안 기법과 차이가 있다. 한편 논문 [5]는 사용자가 자격이나 면허를 검증자에게 입증하기 위해 관련 기관(예: 대학 졸업장이라면 출신 대학)이 입증에 필요한 크리덴셜을 사용자에게 제공하여 지갑에 보유하도록 함과 동시에 발급 사실을 블록체인에 기록하는 구조를 제안하였다. 검증자는 사용자의 크리덴셜을 검증할 때 기관의 서명과 블록체인의 발급 기록을 확인한다. 이 연구는 미리 발급된 크리덴셜에 나온 자격이나 면허만을 입증할 수 있어, 최초 대면하는 상대에게 일회성의 입증을 위해 즉시 필요 정보를 제시할 수 있는 제안 기법과는 차이가 있다. 또 다른 연구로 제안 기법과 마찬가지로 사용자의 선택적 제공 정보를 블록체인 트랜잭션을 통해 서버가 검증하는 기법이 발표되었다[6]. 다만 논문 [6]의 기법은 트랜잭션의 입출력 수가 검증에 필요한 요소의 수만큼 필요하여 그 구조가 제안 기법보다 더 복잡하다. 또 서버 데이터베이스를 상정하지 않아 기존 서비스 기록으로부터 유의미한 자료를 추출할

방법이 없다는 점에서 제안 기법과 차이가 있다.

둘째로, 블록체인을 보안 데이터베이스로 활용하려는 연구를 살펴본다. 의료기관 간의 의료정보 교환을 위해 블록체인을 기록 공간으로 제안하고 생체인식 기반의 인증으로 접근 제어를 달성하려는 기법이 발표되었다[7]. 또 다른 시도에서는 확장성을 고려해 별도의 사이드체인에 정보를 저장하되 사이드체인에 대한 접근권한 부여를 위해 주 체인의 스마트 계약을 활용하였다[8]. 이들은 블록체인을 보안 데이터베이스로 활용하므로 블록체인을 인증의 도구로 활용하는 제안 기법과 해결하려는 문제가 다르다.

Ⅲ. MRChain 트랜잭션 및 비트코인과의 설계 요소 차이

비트코인[9,10]과 비교하여 MRChain 트랜잭션의 구조와 역할, 설계 근거를 제시하고 블록 생성과 MRChain 참여 노드 간 합의 방식을 설명한다. 참여자들은 사용자들 정보를 보유한 신뢰기관의 마스터 서버(이하 masterS), 다수 사용자와 여러 서비스 제공자의 인증 서버들이다. 신뢰기관이란 대한민국의 행정안전부같이 주요 개인정보를 보유하고 관리하는 기관으로 어느 나라든 운전면허증이나 여권 등의 기재 정보 관리 기구가 존재하기 마련이다. 신뢰 기관 서버의 중요성을 고려할 때 masterS는 통상의 웹사이트나 포털의 서버보다 훨씬 더 견고한 방어 및 가용성 제고를 위해, 철저한 보안 교육과 더불어 망 분리, 관리자 권한 접근 제어, 방화벽 등 여러 보안 장비, 분산 서버 시스템, 하드웨어 이중화, ISP 망 다중 접속 구조 등을 채택하여 운용할 수 있다. 따라서 masterS는 높은 보안성을 보장함과 함께 일시적 장애에도 서비스 제공을 이어갈 것이라 가정함에 무리가 없다.

1. 트랜잭션의 Input, Output과 이어가기

비트코인 트랜잭션의 예로, 그림 1(a)는 input 0에서 BTC가 포함된 이전 트랜잭션의 output을 Alice가 참조해서(가져와서) Bob의 주소로 전송(output 0)하고, 남은 BTC를 Alice의 다른 주소로 전송(output 1)함을 보여준다. Output은 트랜잭션의 ID(트랜잭션에 해시 함수 적용한 결과)와 0부터 시작하는 인덱스 번호를 합쳐 구분한다(예: txid:0 등). 이처럼 트랜잭션은 사용되지 않은 output(Unspent Transaction Output: 이하 UTXO)을 참조하여 사용할 권한이 있음을 입증하는 부분인 input과 사용을 위해 만족시켜야 할 조건이 명시된 BTC 전송의 목적지인 output으로 구성된다. 따라서 BTC의 전송 흐름은 이전 트랜잭션의 output과 현재 트랜잭션의 input이 이어짐(그림 1(b))을 통해 나타낼 수 있다.

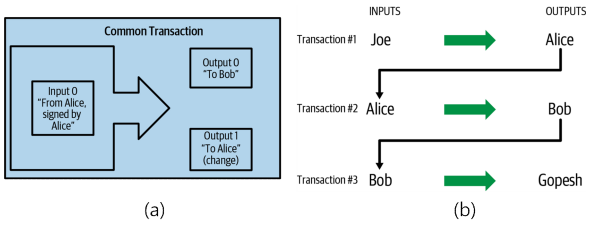

그림 1. 비트코인 트랜잭션 형태 및 트랜잭션 이음[10]

제안하는 MRChain의 트랜잭션은 개인정보 요소(들)값의 검증을 위한 것이기에 검증 시 신뢰기준점(trust anchor) 역할을 하는 머클 루트(2절에서 상세 설명) 값을 포함한다. 트랜잭션은 하나의 input과 하나의 output의 쌍을 포함하는데, 단일 input에선 해당 사용자의 직전 트랜잭션의 output을 참조하며, 단일 output(항상 인덱스 번호 0임)에선 다음 검증 시 이 output을 가져다 쓰기 위한 권한을 확인하기 위해 공개키의 해시값을 보유한다. 다만 가치의 전송이 없으므로 어떤 가상 화폐도 필요치 않다. 제안하는 트랜잭션은 비트코인 트랜잭션에서 이전의 UTXO를 사용할 때 입증을 위한 정보를 제공하는 아이디어를 참조하여, 공개키의 해시로 락(lock)을 걸고 이를 해제(참조하여 쓰기)하기 위해 공개키 및 공개키로 검증할 수 있는 서명을 포함하도록 한

다. 개인정보 검증, 등록, 변경, 폐기의 트랜잭션은 각기 정해진 역할만을 하도록 input 및 output의 형태가 고정되어 있다.

2. 트랜잭션의 머클 루트의 역할 및 계산법

트랜잭션의 핵심 정보는 머클 루트인데, 이는 머클 트리의 루트(root) 노드를 일컫는다. 머클 트리는 비트코인 블록의 모든 트랜잭션(들)을 포함하는 포화 이진 형태의 해시 트리이다. 즉 블록의 각 트랜잭션은 해시된 후 머클 트리의 한 잎(leaf) 노드가 되고, 이어서 마지막 루트 해시를 생성할 때까지 재귀적으로 왼쪽과 오른쪽의 자식 노드를 이은 후의 해시 결과가 부모 노드에 저장된다. 이런 상향식 해시 계산 구조는 특정 트랜잭션의 포함 여부를 효율적으로 결정할 수 있게 한다. 예로, 그림 2에서 트랜잭션 C의 포함 여부는, H_C 계산 후 H_D를 이용해 H_{CD}를 계산하고, 이어 H_{AB}를 이용해 루트인 H_{ABCD}계산이 가능하므로 이 값과 머클 루트 노드의 주어진 값을 비교하는 식으로 트리의 깊이에 비례하는 수의 연산만으로 결정할 수 있다.

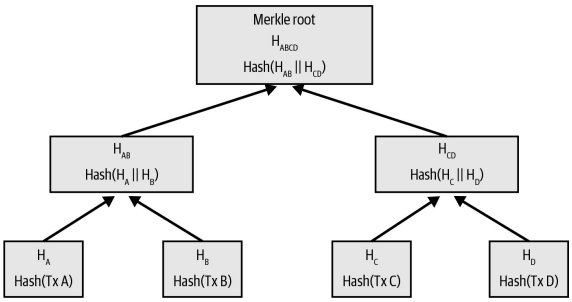

그림 2. 비트코인 머클 트리[10]

제안 기법은 사용자가 요소값 해시를 각각 하나의 잎 노드로 삼아 머클 트리를 만들고, 구한 머클 루트 값을 제시된 개인정보 요소(들)값 유효성 확인에 사용한다. 즉, 사용자가 개인정보 요소(들)값 일부(또는 전부) 및 검증에 사용될 해시값(들)을 제시하면, 검증자는 이들로부터 계산한(그림 3 점선 화살표는 한 요소값이 머클 루트 값에 반영되는 흐름) 머클 루트가 저장된 머클 루트 값과 같을 때 상대방을 제시한 정보의 소유자라 판정한다(그림 3의 점 패턴의 노드는 '계산'하는 노드, 색칠 노드는 값이 '제공'되는 노드).

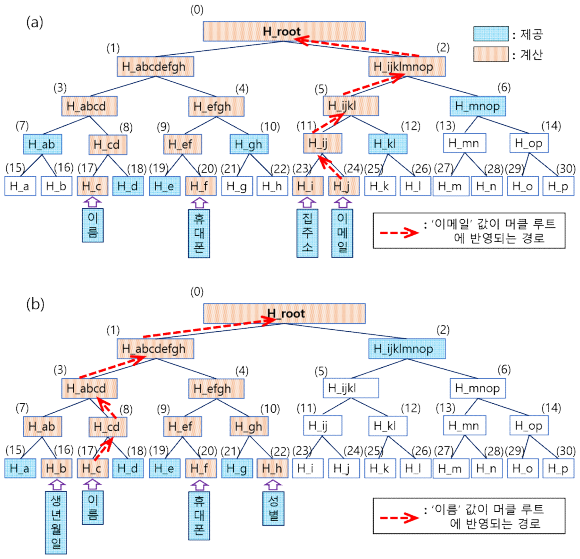

그림 3. 사용자 개인정보 요소(들)값 포함 여부 확인 예

3. MRChain 블록 생성 및 체인 합의 방식

MRChain은 비트코인 시스템과 달리 사전에 승인된 노드들만 트랜잭션을 검증하고 블록을 생성하도록 한다. 따라서 MRChain에는 경쟁이나 수익 추구의 개념이 없으며, 단지 랜덤하게 선택된 노드가 블록을 생성, 제안하고 다른 노드들이 이를 검증 후 수용하는 식으로 MRChain이 확장될 뿐이다. 시스템 내에 오직 승인된 노드들만 존재하므로 블록을 생성하고 체인을 이어갈 때 노드 간의 의견 불일치가 없음이 보장된다.

MRChain의 트랜잭션을 이용해 사용자 정보 요소(들)값을 검증함은 인증 서버 내의 절차만으로 매우 신속히 완료된다. MRChain의 블록 생성 주기는 12초로 한다. 검증을 요청하려면 직전 검증 결과 트랜잭션의 output을 참조해야 하므로, 한 사용자가 최대한 기다린다고 하더라도 12초 후면 다른(이전 서버와 같아도 무방) 서버에게 다음 검증을 연이어 요청할 수 있다. 서비스 동안(예: (연령대와 성별 검증받고) 온라인 설문조사 응답함)의 시간을 고려하면 물리적으로 이

보다 더 짧은 시간 내에 사람이 다음 검증을 요청하기는 어려워 이 크기는 문제가 되지 않는다. 또한 이는 신뢰할 수 없는 검증자들을 가정한 이더리움의 생성 주기와도 같아 무리 없이 실현 가능하며, 필요하면 더 단축할 여지도 충분하다.

Ⅳ. 개인정보 등록, 검증, 변경, 탈퇴 프로토콜 및 인증 서버의 데이터베이스

MRChain을 활용하여 최초로 사용자 개인정보 요소(들)값을 masterS 확인 후 등록하며, 사용자가 개인정보 요소(들)값의 유효성을 서비스 제공자 서버에게 검증받고, 요소(들)값의 변경 적용 및 폐기하기를 위한 각 프로토콜을 제시한다. 이때 필요한 머클 트리 구성, 서버에 제공할 정보를 생성, 제공된 정보로부터 머클 루트를 계산하는 알고리즘들을 우선 서술한다. 모든 프로토콜의 전송 메시지는 TLS 1.3으로 보호됨을 가정한다. 표기법으로, 연접 연산자를 ‖로 쓰고, 크립토 해시 함수 중 강력한 보안성의 SHA3-512[11]와 BLAKE2-512[12]를 각각 h_1과 h_2로 쓴다.

1. 머클 트리 생성

사용자 개인정보 요소(들)값으로부터 구성되는 잎 노드 n개 머클 트리의 형태는 그림 4와 같다 (그림 4의 $n=16$). 실제 요소의 수는 최대 $n-1$으로, 최소 하나 이상의 랜덤 더미(dummy) 값이 트리의 잎 노드에 포함된다. 개인의 “이름”, “생년월일”, “성별”, “휴대폰번호”, “이메일주소”, “SNS주소”, “집주소”, “직장주소”, “직장전화번호”, “은행계좌번호”, “신용카드정보”, “운전면허정보”, “병역사항”과 같이 한 사람에 대한 매우 자세한 수준의 설명도 13개의 요소로 표현될 수 있다. 따라서 15개는 요소의 수로 충분하여 제안 기법은 n을 16으로 택한다. n은 필요하면 32, 64 등 2^h 꼴을 취하는 값으로 쉽게 확장될 수 있다.

트리의 노드는 번호(그림 4의 괄호 안 값)로 특정하는데 루트 노드의 번호는 0이며, 부모 노드 번호가 j일 때 왼쪽 자식 노드 번호는 $2j+1$, 오른쪽 자식 노드 번호는 $2j+2$이다. 한편 각 요소 $elmt_i (i=0, \ldots, n-1)$는 $elmt_i$ 값의 해시 적용 결과가 트리의 번호 $(n-1+i)$인 잎 노드가 된다. 예로, 그림 4의 요소 7은 해시 적용 후 번호 22(=16-1+7)의 잎 노드가 된다.

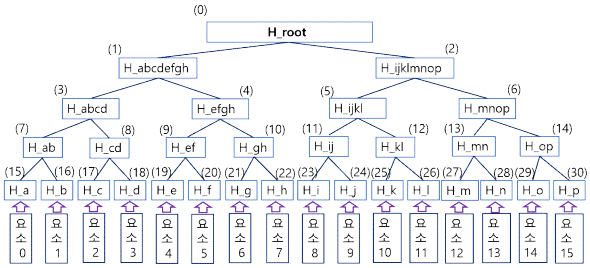

그림 4. 사용자 개인정보 요소(들)값의 머클 트리 형태

사용자가 자신의 개인정보 요소(들)값 및 랜덤 더미(들) 값을 이용해 저장할 머클 트리를 생성하는 방법은 알고리즘 1과 같다.

알고리즘 1: U_i(폰 M_i) 개인정보 요소(들) 머클 트리 구성

입력: $M[0:n-1]$: $M[j]=elmt_j$인 배열, $0 \le j < n=2^h$인 각 j에 대해 $elmt_j$는 U_i의 개인정보 요소값 또는 랜덤 더미 값(들)

출력: $T[0:2n-2]$: 노드 해시값 포함하는 배열(트리)

1. for $i=0$ to $n-1$ do: // 잎 노드들 전부 생성
2. $\quad T[n-1+i] \leftarrow h_1(h_2(M[i]))$
3. for $i=h$ downto 1 do: // $h=\log_2 n$
4. $\quad$ for $j=0$ to $2^{i-1}-1$ do:
5. $\quad\quad T[\lfloor \frac{2^i-1+2j}{2} \rfloor] \leftarrow h_1(h_2(T[2^i-1+2j] \| T[2^i+2j]))$
6. Root $\leftarrow T[0]$

2. 사용자 개인정보 요소(들)의 등록

개인정보 요소(들)값을 최초 MRChain 트랜잭션으로 등록하려면 프로토콜 1의 절차를 따른다.

프로토콜 1 : 사용자 U_i(폰 M_i) 개인정보 요소(들)값 masterS에 인증 및 MRChain 트랜잭션으로 등록

1. $U_i\,(M_i) \leftrightarrow masterS$: 상호 인증
2. $U_i\,(M_i) \rightarrow masterS$: 트랜잭션 tx_{reg} 및 등록할 개인정보 요소(들)값
   // tx_{reg}: 한 input-output의 쌍
   // $MRoot_{U_i}$: U_i의 머클 루트 값(알고리즘 1로 계산)
   // $hPubKey^0$: tx_{reg} 참조 시 제시할 공개 키의 해시
   // 결과가 이 값과 같은지 확인(공개 키 소유 확인)
   ($void : void$, // input: 이전 output 없음을 표현
   $MRoot_{U_i} : hPubKey^0 : \langle masterS\_URI : \mathbf{0} \rangle$)
   // s개($1 \le s < n$) 개인정보 요소(들)값 배열:
   $M[0 : n-1]$, 단, $j = 0,1, \ldots, s-1$에 대해
   $M[j] = elmt_j$ 이며, $j = s, \ldots, n-1$에 대해
   $M[j] = rand_j$ (랜덤 값)
3. $masterS$: 개인정보 요소(들)값 검증 진행;
   3.1 $\forall j (j = 0,1, \ldots, s-1)$에 대하여,
   $elmt_j$가 U_i의 해당 개인정보 요소값임을 확인;
   (예: 행정안전부 데이터베이스 기록과 대조);
   3.2 $M[0 : n-1]$을 입력으로 알고리즘 1을 실행한 결과의 출력 $T[0]$와 $MRoot_{U_i}$가 같은지 확인;
4. $masterS$: tx_{reg} 및 현재 시각 $ctime$ 이용해 블록에 포함될 확장 트랜잭션 $tx_{reg}^{cnfrmd,masterS}$를 다음과 같이 생성 $tx_{reg} \| sig_{reg}^{masterS}$;
   ($sig_{reg}^{masterS}$ 는 등록 tx_{reg} 검증했음을 블록에 기록하기 위해 $masterS$가 tx_{reg} 해시에 한 서명)
5. $masterS$: U_i와 $tx_{reg}^{cnfrmd,masterS}$ 연결해 자체 저장

$tx_{reg}^{cnfrmd,masterS}$의 머클 루트 값(masterS가 확인, 인증함(프로토콜 1의 3.2와 4))은 추후 U_i의 정보 요소(들)값 검증 시 최초 신뢰 기준점의 역할을 한다. 또한 프로토콜 1의 5는 폰 저장 데이터 삭제 등 상황에서 U_i의 요청에 따라 $materS$가 U_i의 검증 트랜잭션들을 추적을 위한 초기 시작 정보를 제공하려는 목적을 가진다.

3. 서버의 사용자 개인정보 요소(들) 검증

우선 사용자 U_i가 서비스 제공자 서버 $authS_m$에 검증 요청 시 제시할 정보를 생성하는 절차를 알고리즘 2에서 서술한다. 그 동작 원리를 그림 3(a)를 예로 설명한다. '이름', '휴대폰', '집주소', '이메일'에 해당하는 실제의 값을 해시 적용해 각각 H_c, H_f, H_i, H_j를 만든 후, 이들의 형제(sibling) 노드 중 이미 계산된 값(예: H_i와 H_j의 관계)이 아닌 것을 제공할 노드에 포함한다. 이제 한 단계 위 수준부터 다시 계산된 노드(들)로부터 형제 노드 중 제공할 노드를 추출하는 과정을 루트 계산 전까지 반복한다. 그러면 그림 3(a)의 색칠 노드인 H_d, H_e, H_{ab}, H_{gh}, H_{kl}, H_{mnop}가 이 순서대로 제공 노드 번호 집합에 포함되고, 이들 번호의 해당 값을 배열 T에서 가져와 배열 H에 복사하고 H를 $authS_m$에 제공한다.

알고리즘 2: U_i(폰 M_i)가 $authS_m$에 개인정보 요소(들) 값 유효성을 검증받기 위해 제공할 정보 생성

입력: P: $(j, elmt_j)$를 원소로 하는 리스트(포함된 $elmt_j$ (들)($0 \le j < n-1$)을 제출), T: 머클 트리(배열)
출력: $H[0 : 2n-2]$: 필수 노드 해시값 포함하는 배열
1. $H[0 : 2n-2] \leftarrow NULL$ // 모두 NULL로 초기화
2. $C_0 \leftarrow [\,]$ // C_0는 제공 요소의 트리 번호(들) 포함
3. for each $(j, elmt_j)$ in P do:
4. $\quad C_0.append(n-1+j)$ // $j \Rightarrow n-1+j$ 매핑
5. for $i = 0$ to $h-1$ do: // $h = \log_2 n$
6. $\quad C_{i+1} \leftarrow [\,]$, $S_i \leftarrow [\,]$
7. $\quad$ for each k in C_i do:
8. $\quad\quad$ if $\lfloor \frac{k-1}{2} \rfloor$ not in C_{i+1} then
$\quad\quad\quad C_{i+1}.append(\lfloor \frac{k-1}{2} \rfloor)$
   // $sb(k)$: 번호 k 노드의 형제(sibling) 노드의 번호
9. $\quad\quad$ if $sb(k)$ not in C_i then $S_i.append(sb(k))$
10. for $i = h-1$ downto 0 do: // i : 레벨
11. $\quad$ for each j in S_i do: // S_i: 제공 값(들) 번호
12. $\quad\quad H[j] \leftarrow T[j]$

다음으로 서버 $authS_m$가 사용자 U_i가 제시한 여러 정보로부터 머클 루트를 계산하여 트랜잭션 output의 머클 루트 값과 비교함으로써 정보의 유효성을 판정하는 알고리즘 3을 제시한다.

알고리즘 3: $authS_m$의 수신 U_i 정보 요소(들)값 검증

입력: P:$(j,elmt_j)$를 원소로 하는 리스트(제출된 U_i의 요소(들)값), $H[0:2n-2]$: 필수 노드 해시값 포함 배열, $Root$: 참조 트랜잭션 output에 기록된 머클 루트 값
출력: $TRUE$(검증 성공) 또는 $FALSE$

1. for each $(j,elmt_j)$ in P do:
2. $H[n-1+j] \leftarrow h_1(h_2(elmt_j))$
3. for $i=0$ to $h-1$ do: // $h=\log_2 n$
4. for $j=2^{h-i}-1$ to $2^{h-i+1}-2$ step 2 do:
5. if $H[j]$ not $NULL$ then
6. $H[\lfloor \frac{j-1}{2} \rfloor] \leftarrow h_1(h_2(H[2\lfloor \frac{j-1}{2} \rfloor + 1] \| H[2\lfloor \frac{j-1}{2} \rfloor + 2]))$
7. if $H[0] == Root$ then return $TRUE$ else return $FALSE$

이제 서버 $authS_m$이 사용자 U_i 제공 정보의 유효성을 검증하고 MRChain에 기록하는 절차를 프로토콜 2에서 설명한다. 사용자 U_i는 폰 저장 비밀값 S_{U_i}와 사이트 URI를 연접한 후 해시하여 사이트 서버에 제시할 핸들을 필요할 때마다 즉시 생성한다(예: https://eshopbiz.com 의 서버에 제시할 핸들은 $h_1(S_{U_i}\|$ "eshopbiz.com")임). 핸들에는 소유자를 유추할 수 있는 어떤 정보도 포함되어 있지 않으며, 512비트의 핸들 길이를 고려할 때 한 인증 서버에서 두 사용자의 핸들이 일치할 경우는 확률적으로 무시할 수준이다.

프로토콜 2 : 서비스 요청을 위한 U_i(폰 M_i)의 최소 개인정보 요소(들)값 $authS_m$에 검증 및 MRChain에 기록

1. $U_i(M_i) \leftarrow authS_m$: 상대방이 $authS_m$임 인증;
   $U_i(M_i) \rightarrow authS_m$: $handle_{U_i}^{authS_m}$, 서비스 요청
   // $handle_{U_i}^{authS_m}$: U_i의 $authS_m$ 구동 사이트 핸들
2. $U_i(M_i) \leftarrow authS_m$: $nonce\|ctime$, U_i에 제공을 요구하는 최소한의 개인정보 요소(들)값 항목
   // $nonce$: $authS_m$이 $ctime$과 합하여 이 입증 // 시도를 다른 시도와 구분함; $ctime$: 현재 시각
3. $U_i(M_i) \rightarrow authS_m$: 트랜잭션 tx_{verify}, 리스트 P, 배열 H
   // tx_{verify}: 1개의 input-output 쌍 및
   // 서명 검증용 공개 키와 서명으로 구성
   ($txid_l:0$, // 마지막 검증 tx의 output(UTXO)
   $MRoot_{U_i}:hPubKey^{c(U_i)+1}:\langle authS_m URI:nonce\|ctime\rangle$)
   $pubKey^{c(U_i)}$; sig_{verify}
   // (sig_{verify}는 $pubKey^{c(U_i)}$의 쌍이 되는 개인 키로
   // tx_{verify}의 해시에 서명한 것. 이에 $txid_l:0$
   // ouput의 뒷부분 $hPubKey^{c(U_i)}$을 해시값으로
   // 가지는 $pubKey^{c(U_i)}$ 사용해 검증할 수 있음)
   // $c(U_i)$는 U_i 정보 요소(들)값이 (재)등록된 후
   // 현재까지 검증된 횟수; $nonce$와 $ctime$은 2에서
   // $authS_m$이 제시한 값($authS_m$의 응답 확인용))
   // (요소 인덱스, 실제 요소값) 튜플(들)의 리스트
   $P=[(a,elmt_a),(b,elmt_b),\ldots,(c,elmt_c)]$
   // P와 U_i의 머클 트리 배열 T를 입력으로
   // 알고리즘 2 실행한 결과 배열 $H[0:2n-2]$
   $H=[H_0,H_1,\ \ldots\ ,H_{2n-2}]$
4. $authS_m$: 다음의 제출된 값들 검증 절차 진행;
   4.1 tx_{verify}의 output에 포함된 $nonce\|ctime$이 2에서 제시한 $nonce\|ctime$과 같은지 확인;
   4.2 $txid_l:0$에 명시된 $MRoot_{U_i}$가 tx_{verify}의 output의 $MRoot_{U_i}$와 같은지 확인;
   4.3 $h_1(pubKey^{c(U_i)}) == hPubKey^{c(U_i)}$인지 확인;
   4.4 sig_{verify}가 유효한지를 서명 검증용 공개키 $pubKey^{c(U_i)}$ 및 tx_{verify} 이용하여 확인;
   4.5 P 및 H를 입력으로 알고리즘 3을 실행한 결과의 반환 값이 $TRUE$인지 확인;
5. $authS_m$: tx_{verify} 이용해 블록에 포함될 확장

트랜잭션 $tx_{verify}^{cnfrmd,authS_m}$: $tx_{verify} \| sig_{verify}^{authS_m}$ 생성;
($sig_{verify}^{authS_m}$는 tx_{verify} 검증했음을 블록에 기록하기 위해 $authS_m$가 tx_{verify} 해시에 한 서명)

6. $authS_m$: $handle_{U_i}^{authS_m}$을 기본 키로 접근해 DB에 U_i가 제시한 개인정보 요소(들)값의 항목 이름 및 서비스 제공 내용을 저장;

"같은 값"이 이어지는(프로토콜 2의 4.2) 머클 루트 값을 활용해 사용자 U_i는 여러 서비스 제공자 서버에 필요한 최소한을 제시한 요소(들)값의 유효성을 검증받는다. 이런 연속된 검증의 과정이 MRChain에 트랜잭션 이어감의 형태(그림 5)로 기록된다.

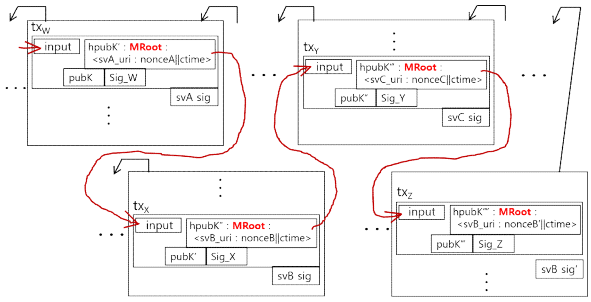

그림 5. 인증 서버에서의 사용자 개인정보 요소(들)값 검증을 통한 MRChain의 트랜잭션들 이어감

서버의 데이터베이스는 사용자의 핸들을 기본 키로 사용자의 서비스 요청 내용(예: 온라인 물품 판매 기록)을 기록한다. 단, 개인정보는 식별 불가한 수준으로만 저장한다. 즉 프로토콜 2의 6의 결과로, 데이터베이스는 각 사용자의 '성별', '거주지역', '연령범위'만을 서비스 내용($nonce \| ctime$와 함께)에 연결하여 저장한다(그림 6). 예로, 한 전자상거래 업체 데이터베이스의 어떤 핸들을 키로 저장된 내용이 <남자, 대전시 유성구, 40-49> 및 물품 구매 내용이라면, 이로부터 유의미한 공격 시나리오를 짜는 것은 불가능하다. 이 업체는 거래 내용만 남기고 배송 후에 배송지 주소 등의 필요 정보를 모두 삭제함으로써 개인 특정에 도움이 될 정보의 보유 기간을 최소화한다. 결과적으로, 데이터베이스 정보로부터 사용자별 맞춤형(제공된 서비스 내용으로부터 도출) 추천을 할 수 있으며, 유형 집단별 관심사(예: 광역시에 사는 30대 여자의 주말 구매 품목 순위)를 통계 분석해 트렌드를 추출하거나 AI 모델 학습에 적용할 수 있다. 또한 트랜잭션의 $nonce \| ctime$이 데이터베이스에도 그대로 기록되므로 서비스 제공자는 사용자별로 제공한 서비스 내용 각각을 변경이나 유실 우려 없는 MRChain의 트랜잭션과 연결할 수 있다. 사용자 또한 트랜잭션의 output 참조를 역방향으로 따라가면 자신의 기존 서비스 요청 내용을 언제든 검색할 수 있다. 만약 핸들 분실 등의 사유로 사용자 U_i가 자신의 서비스 이용 내역 추적 도구를 잃으면, U_i는 대면 등 별도 방식을 통해 $masterS$에 인증받은 후 저장된 등록 트랜잭션을 얻고(프로토콜 1의 5), 이로부터 MRChain의 트랜잭션 output 참조를 순방향으로 따라가면서 특정 사이트 또는 전체 사이트의 서비스 기록을 찾을 수 있다.

handle	성별	거주 지역	연령 범위	서비스(거래, 구매) 내용		
				nonce, ctime	검증 요소(들) 이름	서비스 내용
xyz...	남	대전시 유성구	40-49	123fd... 09:12:49/10/10/2025	이름,배송주소, 휴대폰,결제정보	16"노트북(...), 거치대(...) 구매
				456ab... 11:24:05/09/17/2025	이름,배송주소, 휴대폰,결제정보	24" 모니터(...) 교환(사유:불량 회소)
				⋮		
def...	여	서울시 종로구	30-39	789cd... 19:40:22/11/25/2025	이름,배송주소, 휴대폰,결제정보	전자동 에스프레소 커피머신(...) 구매
				⋮		
⋮	⋮	⋮	⋮	⋮		

그림 6. 서비스 제공자 서버의 데이터베이스 형태

4. 사용자 개인정보 일부 요소(들)값 갱신

프로토콜 3은 갱신된 요소(들)값을 반영해 새 머클 루트 값을 MRChain 트랜잭션에 기록한다.

프로토콜 3 : U_i(폰 M_i)의 일부 개인정보 요소(들)의 갱신 값 masterS에 인증 및 MRChain에 머클 루트 값 등록

1. U_i (M_i) ← $masterS$: 상대방이 $masterS$임 인증;
   U_i (M_i) → $masterS$: 트랜잭션 tx_{update} 및 갱신할 개인정보 요소(들)값 입증 정보
   // tx_{update}: 1개의 input-output 쌍 및
   // 서명 검증용 공개 키와 서명으로 구성

($txid_l:0$, // (기존 값의)마지막 검증 tx의 output
$MRoot*_{U_i}:hPubKey*^0:\langle masterS\_URI:\mathbf{0}\rangle$)
$pubKey^{c(U_i)}$; sig_{update}
// (sig_{update}는 $pubKey^{c(U_i)}$의 쌍이 되는 개인 키로
// tx_{update}의 해시에 서명한 것. 따라서 $txid_l:0$
// ouput의 뒷부분 $hPubKey^{c(U_i)}$을 해시값으로
// 가지는 $pubKey^{c(U_i)}$ 사용해 검증할 수 있음)
// s개($1 \le s < n$) 개인정보 요소(들)값 배열:
$M[0:n-1]$, 단, $j=0,1,\ldots,s-1$에 대해
$M[j]=elmt_j$ 이며, $j=s,\ldots,n-1$에 대해
$M[j]=rand_j$ (랜덤 값)
// s개($1 \le s < n$) 새로운 개인정보 요소(들)값 배열:
$M*[0:n-1]$, 단, $j=0,1,\ldots,s-1$에 대해
$M*[j]=elmt_j*$ 이며, $j=s,\ldots,n-1$에 대해
$M*[j]=rand_j*$ (랜덤 값)
// 각 j에 대해 $elmt_j*$는 $elmt_j$이거나 또는 갱신
// 된($\neq elmt_j$) 값, $rand_j*$는 새로운 랜덤 값
2. $masterS$: 이전/갱신 개인정보 요소(들)값 검증 진행;
2.1 $txid_l:0$이 U_i의 UTXO임을 확인;
2.2 $h_1(pubKey^{c(U_i)}) == hPubKey^{c(U_i)}$ 인지 확인;
2.3 서명 sig_{update}이 유효한지 서명 검증 공개키 $pubKey^{c(U_i)}$ 및 tx_{update} 이용하여 확인;
2.4 $M[0:n-1]$을 입력으로 알고리즘 1을 실행한 결과의 출력 $T[0]$와 $MRoot_{U_i}$가 같은지 확인;
// $MRoot_{U_i}$는 $txid_l:0$에 명시된 머클 루트 값
2.5 $\forall j(j=0,1,\ldots,s-1)$에 대하여, $elmt_j*$가 U_i의 해당 개인정보 요소값임을 확인 (예: 행정안전부 데이터베이스 기록과 대조);
2.6 $M*[0:n-1]$을 입력으로 알고리즘 1을 실행한 결과의 출력 $T[0]$와 $MRoot*_{U_i}$가 같은지 확인;
3. $masterS$: tx_{update} 이용해 블록에 포함될 확장 트랜잭션 $tx_{uptate}^{cnfrmd,masterS}:tx_{update}||sig_{update}^{masterS}$ 생성;
($sig_{update}^{masterS}$ 는 tx_{update} 를 검증했음을 블록에 기록하기 위해 $masterS$가 tx_{update} 해시에 한 서명)

==========

참조된 UTXO가 사용자 U_i의 것임(프로토콜 3의 2.1), 참조 권한 있음을 공개키와 서명으로 확인(프로토콜 3의 2.2와 2.3), 제시된 기존 요소(들)값으로 UTXO의 머클 루트 값 도출됨을 확인해 소유자임을 검증(프로토콜 3의 2.4), 갱신 요소(들)값의 유효성 확인(프로토콜 3의 2.5) 및 새로운 머클 루트 값이 올바름을 확인(프로토콜 3의 2.6)한다.

5. 사용자 개인정보 요소(들)값 폐기(탈퇴)

사용자 개인정보 요소(들) 전체 폐기(MRChain 기반의 검증 탈퇴) 절차는 프로토콜 4와 같다.

프로토콜 4: U_i의 모든 개인정보 요소(들)값 폐기 요청의 masterS 인증 및 사용 불가토록 MRChain에 기록

==========

1. $U_i(PC$ 또는 대면) $\leftrightarrow masterS$: 상호 인증 및 U_i의 등록된 개인정보 요소(들)값 폐기 요청
2. $masterS$:
2.1 $UTXO_{U_i}$(마지막 참조된 U_i의 검증 기록 트랜잭션의 output) 검색; 결과 $txid_l:0$ 라 함
2.2 트랜잭션 tx_{revoke} 생성;
// tx_{revoke}: output에 void:void의 형태 포함됨
($txid_l:0$, $void:void:\langle masterS\_URI:-1\rangle$)
2.3 tx_{revoke} 이용해 블록에 포함될 확장 트랜잭션 $tx_{revoke}^{cnfrmd,masterS}:tx_{revoke}||sig_{revoke}^{masterS}$ 생성;
($sig_{revoke}^{masterS}$는 $masterS$가 tx_{revoke} 해시에 한 서명)

==========

$tx_{revoke}^{cnfrmd,masterS}$의 void:void 형태의 output은 사용(권한 입증)될 수 없으므로, 이후 어떤 U_i의 개인정보 요소(들)값 검증 요청도 원천 거부된다.

V. 보안성 및 검증 시간 분석

위험 요소별로 제안 기법의 보안성을 분석하고, 핵심 기능인 인증 서버에 의한 사용자 개인정보 요소(들)값 검증 시간의 성능을 분석한다.

1. 사용자 개인정보 불법 탈취 시도

개인정보 탈취를 위한 공격 경로로 피해자와

서비스 제공자 서버 또는 masterS와의 프로토콜 통신 메시지를 해독함은 교환되는 메시지가 암호화를 사용하는 TLS 1.3으로 보호되므로 극히 어려우며, 인터넷 메시지 탈취와 해독은 제안 기법만의 취약 지점은 아니다. 다음 경로로, 탈취를 위해 MRChain 트랜잭션의 머클 루트 값을 분석하려 할 수 있다. 머클 루트 $MRoot_{U_i}$은 s개의 요소(들)값과 $(n-s)$개의 더미 랜덤 값으로부터 생성된다. 극단적 경우로, 공격자가 피해자의 정보 중 딱 하나만을 몰라서 이 요소값 위치에 무작위 대입을 시도한다고 하자. 설사 요소값의 가능 범위가 제한되더라도(예로, 11자리 전화번호의 총 가짓수는 10^{11}임), 대입한 번호가 올바른지 확인하는 것조차 더미 랜덤 값을 정확히 아는 것을 요구한다. 이는 512비트 더미 값의 전체 가짓수 2^{512}를 고려할 때 불가능에 가깝다. 따라서 ChaCha20[13]와 같은 강력한 무작위성의 소프트웨어 스트림(stream) 암호의 출력을 더미(들)값으로 사용한다면 이런 공격의 성공 가능성은 무시할 수 있는 수준이다. 또한 만약 머클 루트 값으로부터 해시의 역상(preimage)을 추출하여 요소(들)값을 획득하려 한다면, 머클 루트 생성을 위한 총 $2n+2(n-1)=4n-2$ 번의 해시 계산(잎 노드 수는 n개, 내부 노드 수는 $(n-1)$개이고 각 노드 해시값 계산에 h_1과 h_2가 적용되므로)이 필요하다. 그러므로 이는 강력한 역상 저항성의 h_1과 h_2을 고려할 때 불가능하다. 따라서 트랜잭션의 머클 루트 값을 통한 공격은 해당 값을 생성할 때 사용된 개인정보 요소(들)값의 기밀성 방어를 뚫을 수 없다.

다음으로 프로토콜 대신 서버 자체를 통한 경로 공격이 가능하다. 우선 masterS는 사용자 정보의 마스터 데이터베이스를 보유하는 셈이라 뚫리게 되면 큰 문제를 일으킨다. 다만 신뢰 기관의 서버로서 masterS는 자료의 안전한 보관 및 가용성을 위해 최신의 보안 장비 및 보안 기법들이 적용됨을 간주할 수 있다. 한편 서비스 제공자 서버 $authS_m$의 데이터베이스는 탈취되더라도 직접 개인을 특정하거나 연락처를 알아낼 수는 없다. 그럼에도 오랜 기간 다수의 서비스를 받은 내역이 있는 사용자의 데이터베이스 항목에 대해, 공격자가 충분한 시간과 노력을 들여 AI 기반의 추론 공격 등을 가한다면 해당 사용자가 실제로 누구인지를 특정할 가능성은 존재한다. 하지만 이에 대비하여 차분(differential) 프라이버시 등의 제공을 통해, 공격자의 분석에도 개인을 가려내는 것을 억제하려는 목적의 추가 대비책 제안은 본 논문의 범위를 넘어선다.

2. 일부 거짓 개인정보 포함하여 검증 시도

처음 등록 이후 사용자 U_i의 머클 루트 값은 불변함이 보장된다(프로토콜 2의 4.2). 그러므로 U_i가 요소값 중 하나라도 거짓 제시해(예로, 온라인 공연예매 시 경로우대 할인을 위해 생년을 속임) 검증을 통과하려면, 거짓 요소(들)값으로부터 '등록된' 머클 루트 값 $MRoot_{U_i}$를 생성해야 한다. U_i는 이를 위해 거짓 요소값 $elmt_j$*의 해시를 해당 위치의 잎 노드에 배치하고, 이 노드로부터 루트 노드로 가는 경로상의 노드 중 적어도 하나를 '적절한' 값으로 바꾸는 데 성공해야 한다. 하지만 이는 $h_1(h_2(H_{fake}\|X))$ 또는 $h_1(h_2(X\|H_{fake}))$가 어떤 특정한 값이 되도록 X를 구하는 문제 이상의 난도를 가진다. 그러므로 h_1과 h_2의 강한 제2 역상 저항성으로 그 성공 가능성은 극히 낮다. 둘 이상의 거짓 요소값을 제시해야 한다면, 하나의 거짓 요소 제시에 비해 머클 트리에서 임의의 값 부여가 가능한 노드의 선택지가 줄어들거나 유지되므로 공격 난도는 그 이상이 된다.

3. 공격자가 위장 입증 시도

공격자가 인증 서버에 제시할 피해자 U_i의 요소값 중 하나라도 모른다면 1의 분석에 따라 U_i로의 위장 공격은 성공하기 극히 어렵다. 설사,

공격자가 서버가 요구하는 U_i의 요소(들)값을 전부 알더라도, 추가로 [(i) 서버에 제시할 U_i의 핸들, (ii) h_1의 역상으로부터 $UTXO_{U_i}$를 참조할 때 필요한 공개키, (iii) ECDSA 알고리즘 깨기를 의미하는 개인 키 또는 서명]을 전부 구해야 한다. 이는 불가능에 가까운 과업이다. 물론 공격자가 U_i에 대해 잘 안다면 제안 기법이 아닌 다른 인증 수단을 통한 U_i로의 위장은 가능할 수 있다.

4. 개인정보 요소(들) 검증 시간 분석

제안 기법의 핵심 기능은 인증 서버가 사용자가 제시한 값들을 검증하는 것이다. 이 절차를 실행하는 프로토콜 2의 단계 4에서, 값 비교, 저장을 제외한 암호화 연산의 횟수는 표 1과 같다. 즉, $s = n-1$일 때 최대로 $2n$번의 h_1 계산 및 $2(n-1)$번의 h_2 계산과 고정 1번의 ECDSA 서명 검증 연산이 필요하다. 검증 트랜잭션의 크기(단위: B(바이트))는 input은 64(txid 길이), output은 64(MRoot 길이)+64(pubKey의 해시 길이)+최대32(사이트 URI 길이)+32(nonce 길이)+8(ctime(UNIX time으로 구현) 길이)로 전체 최대 264B이다. 단계 4.4.의 해시 h_1은 이 트랜잭션에 적용하며, 단계 4.5의 알고리즘 2, 라인 1, 2에서 실제 요소값의 h_2 적용 대상 크기는 각각 다르다(예: "성별"은 1B, "이름"은 32B로 충분). 그런데 매우 복잡한 주소 등 64B보다 긴 값이 있을 수 있지만, 전체 s개 요소값 길이의 평균은 64B 이하라고 가정함에 무리가 없다. 그 외의 모든 해시는 512비트(64B)에 적용한다. 따라서 $n = 16$일 때 h_1의 해시 대상은 최대 $264 + 31 \times 64 = 2{,}248$B, h_2의 해시 대상은 최대 $30 \times 64 = 1{,}920$B 크기를 가진다. 한편 Intel Core i5-6600(Skylake, 3.31GHz)를 사용한 벤치마크 실험 결과 h_1(SHA3-512)은 198 MiB/s, h_2 (BLAKE2)는 947 MiB/s의 성능을 각각 보였다[12]. 이 수치를 적용하면 2,248B에 대한 h_1 해시는 초당 $(198 \times 10^6)/2248 \simeq 88{,}078$번, 1,920B에 대한 h_2 해시는 초당 $(947 \times 10^6)/1920 \simeq 493{,}229$번 실행 가능하다. 다음, Intel Core i7-7820HQ (2.3GHz)에서 libsecp256k1(secp256k1:비트코인 시스템에서 쓰이는 타원 곡선)의 암호화 라이브러리를 써서 ecdsa_verify를 돌려 실측한 결과 초당 18,797회의 ECDSA 서명 검증이 가능했다[14]. 참조한 벤치마크는 모두 단일 core이며 clock 속력도 3.31~2.3GHz인 CPU 상의 측정치로서, 멀티코어 고사양의 서버 컴퓨터는 서버 팜 대신 단일 기계인 경우라도 초당 수만 건의 제안 기법의 검증 처리가 충분히 가능할 것이다.

표 1. 프로토콜 2, 단계 4의 세부 단계의 암호화 연산 횟수

단계	해시 함수		서명 검증	비고
	h_1	h_2		
4.3	1	0	0	
4.4	1	0	1	tx를 해시 후 ECDSA 서명한 것 검증함
4.5	s	s	0	알고리즘 2 라인 1, 2
	각각 $\leq n-1$			
	$\leq n-1$	$\leq n-1$	0	알고리즘 2 라인 3-6
	각각 $\leq [2^0 + \cdots + 2^{h-1}] = 2^h - 1 = n-1$			
총계	$\leq 2n$	$\leq 2(n-1)$	1	

VI. 결 론

서비스 제공자는 사용자 정보 보유로 얻어지는 이익과 보유한 정보 유출 시 겪게 되는 막대한 손실 위험 사이에서 적절한 균형을 잡아야 하는 난제를 떠안고 있다. 제안 기법은 업체 보유 정보의 개인 특정 가능성을 배제하여 유출 시의 피해를 최소화하면서도, 맞춤형 마케팅이나 통계 분석 및 AI 모델 학습 등에 정보가 활용될 수 있게 한다. 인증 서버는 이때 상대방이 제시한 요소(들)값이 올바른지를 자체 데이터베이스 참조 없이 블록체인만을 통해 안전하고 신속하게 판정한다. 따라서 사전 정보 없이도 상대방이 제시한 값(들)을 검증하려는 다양한 상황(예: SMS 인증코드 전달 위한 이름+휴대폰번호, 본인 명의 계좌접근 확인 위한 이름+휴대폰번호+은행계좌번호 등)에서 제안 기법은 매우 유용할 것이다.

REFERENCES

[1] L. Zhang, H. Li, L. Sun, Z. Shi and Y. He, "Poster: Towards Fully Distributed User Authentication with Blockchain," *Proc. of 2017 IEEE Symposium on Privacy-Aware Computing (PAC)*, pp. 202-203, 2017.

[2] L. Yu, M. He, H. Liang, L. Xiong, and Y. Liu, "A Blockchain-Based Authentication and Authorization Scheme for Distributed Mobile Cloud Computing Services," *Sensors*, vol. 23, no. 3, 2023.

[3] C. McCabe, A. Mohideen, and R. Singh, "A Blockchain-Based Authentication Mechanism for Enhanced Security," *Sensors*, vol. 24, no. 17, 2024.

[4] P. Kamboj, S. Khare and S. Pal, "User authentication using Blockchain based smart contract in role-based access control," *Peer-to-Peer Netw. Appl.,* vol. 14, pp. 2961-2976, 2021.

[5] M. Aydar, S. Ayvaz, S. C. Cetin, "Towards a Blockchain based digital identity verification, record attestation and record sharing system," arXiv:1906.09791v2, *TR*, *Cornell University*, 2020.

[6] 박준철, "블록체인의 비트코인 유사 트랜잭션을 활용한 선택된 개인정보의 사용자 소유권 입증," *스마트미디어저널*, 제14권, 제12호, 103-112쪽, 2025년 12월

[7] P. Soni, S. Islam, A. K. Pal, N. Mishra, and D. Samanta, "Blockchain-based User Authentication and Data-Sharing Framework for Healthcare Industries," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 4, pp. 3623-3638, July-Aug. 2024.

[8] L. Yang, R. Jiang, X Pu, C. Wang, Y. Yang, M. Wang, L. Zhang, and F. Tian, "An Access Control Model based on Blockchain Master-Sidechain Collaboration," *Cluster Computing*, vol. 27, pp. 477-497, 2024.

[9] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," https://bitcoin.org/bitcoin.pdf (accessed Jan., 09, 2026.)

[10] A.M. Antonopoulos and D.A. Harding, *Mastering Bitcoin: Programming the Open Blockchain*, 3rd ed., O'Reilly Media, 2023.

[11] NIST Computer Security Resource Center, Hash Functions, SHA-3 Project, https://csrc.nist.gov/Projects/hash-functions/sha-3-project (accessed Jan., 10, 2026.)

[12] BLAKE2 - fast secure hashing, https://blake2.net (accessed Jan., 10, 2026.)

[13] J.P. Degabriele, J. Govinden, F. Günther, and K.G. Paterson, "The security of ChaCha20-Poly1305 in the multi-user setting," *Proc. of the ACM Conf. on Computer and Communications Security*, pp. 1981-2003, Nov. 2021.

[14] libsecp256k1 GitHub mirror: Consider removing x86_64 field assembly #726, https://mirror.b10c.me/bitcoin-core-secp256k1//726 (accessed Jan., 10, 2026.)

저 자 소 개

박준철(종신회원)

1986년 서울대 계산통계학과 학사
1988년 KAIST 전산학과 석사
1998년 U. of Maryland, College Park 전산학 박사
2001년 9월 ~ 현재 홍익대학교 컴퓨터공학과 교수

<주관심분야 : 네트워크 보안, 블록체인 신원 인증>