

블록체인에 대한 해시 기반 서명 기법 적용 방안

Bae Bongjin

June 2017

목차

1 개요	1
1.1 연구 동기	1
1.2 논문 기여	2
1.3 논문 구성	2
2 관련 연구	4
2.1 블록체인	4
2.1.1 블록체인 구조	5
2.1.2 트랜잭션 구조	8
2.1.3 블록체인 관련 연구 및 활용 사례	10
2.2 해시 기반 서명	13
2.2.1 포스트 양자 암호 알고리즘 등장 배경	13
2.2.2 포스트 양자 암호 알고리즘	16
2.2.3 해시 함수	18
2.2.4 해시 기반 OTS(One-Time Signature)	19
2.2.5 해시 기반 MSS(MerkleTree Signature Scheme)	25
2.2.6 해시 기반 서명 기법 관련 연구 및 활용 사례	29
3 제안 기법	30
3.1 트랜잭션 서명 기법	30
3.2 머클 트리 생성 및 머클 서명 생성 기법	31
3.3 트랜잭션 검증 기법	32
4 실험 결과	35

4.1	실험 환경	35
4.2	실험 결과	36
4.2.1	트랜잭션 서명 생성	36
4.2.2	머클 트리 생성 및 머클 서명 생성	37
4.2.3	트랜잭션 서명 검증	37
5	결론	40
A	약어	46
B	초록	48

표 목차

1	SHA-2 계열 해시 함수 특성	19
2	LD-OTS와 W-OTS의 서명키/검증키 크기, 서명 크기, 그리고 키 생성 시간(b : security level, w : Winternitz parameter, m : message length) .	24
3	실험 및 구현/개발 환경	35
4	ECDSA와 W-OTS간 서명 알고리즘 성능 비교(단위 : ms)	36
5	머클 트리 생성 및 머클 서명 생성(단위 : ms)	37
6	트랜잭션 서명 검증(단위 : ms)	38

그림 목차

1	블록 구조	6
2	머클 트리를 이용한 머클 루트 계산	7
3	트랜잭션 구조	8
4	전자 화폐 거래 예시	10
5	IBM Bluemix Blockchain	13
6	IBM Quantum Experience	16
7	양자 컴퓨팅 환경에서의 현대 암호 안전성	17
8	SHA-2 해시 압축 함수 블록 다이어그램	20
9	W-OTS 키 쌍 생성 과정	23
10	Merkle tree Signature Scheme 구조(트리 높이 3)	26
11	Merkle tree Signature Scheme 서명 생성 과정(트리 높이 3)	28
12	제안 기법의 머클 트리 생성 과정	31
13	제안 기법의 머클 서명 생성 과정	32
14	제안 기법의 머클 서명 검증 과정	33

1 개요

1.1 연구 동기

블록체인은 특정 기간 축적된 거래 내역을 저장한 블록을 연결한 것으로, 프라이빗 혹은 퍼블릭 사용자 간(P2P) 네트워크에서 일어나는 거래 정보가 네트워크에 참여하는 참가자 모두에게 공유되는 공공 거래 장부이다. 기존 시스템의 경우, 은행이나 증권 회사들에서 거래내역을 담은 장부를 안전하게 보관하는 중앙 집중형 데이터베이스를 가지고 있지만 블록체인은 분산 데이터베이스의 한 형태로 네트워크 참가자가 공동으로 거래 내역을 기록하고 관리하게 되며 수시로 거래 내역에 대해 검증을 하여 시스템에 대한 안정성을 보장하게 된다. 이러한 블록체인은 현재 비트코인, 이더리움 등의 가상화폐에서 거래할 때 발생할 수 있는 해킹을 막는 보안 기술로 사용되고 있으며, 현재 은행권을 중심으로 하여 블록체인 기반의 독자적인 전자 화폐 개발이 진행되고 있고[1], 스마트 계약 서비스, 사물인터넷 간 금융 거래 시스템[2] 등이 개발되고 있다. 하지만 블록체인에서 사용되는 트랜잭션에 대한 서명 알고리즘은 향후 개발될 양자 컴퓨터 환경에서의 알고리즘에 대해 취약점을 가지고 있다. 따라서 현재 블록체인의 서명 알고리즘에 대해 양자 컴퓨팅에 대한 대응 방안이 필요하다.

양자 컴퓨팅 환경에서 어떠한 트랜잭션에 대해 안전한 서명을 하기 위해서는 포스트 양자 암호 알고리즘(Post Quantum Cryptography) 적용이 필요하다. 포스트 양자 암호 알고리즘은 여러 종류가 있지만, 그중에서도 서명 알고리즘이면서 블록체인 구조를 크게 바꾸지 않고 적용 가능한 알고리즘으로 해시 기반 서명 기법(Hash-based Signature Scheme)이 있다. 해시 기반 서명 기법은 단일 서명 알고리즘인 OTS(One-Time Signature)와 머클 트리를 이용한 서명 알고리즘인 MSS(MerkleTree Signature Scheme)가 존재하며, 머클 트리를 이용한 서명 알고리즘을 이용하면 기존 블록체인에서 사용되는 머클 트리를 활용하여 블록체인 기존 시스템을 크게 변경하지 않고 해시

기반 서명 기법을 적용할 수 있다.

1.2 논문 기여

현재 비트코인 기준 블록체인에서는 트랜잭션별 서명 알고리즘으로 ECDSA를 사용하고 있다. 하지만 ECDSA(Elliptic Curve Digital Signature Algorithm) 전자 서명 알고리즘은 타원곡선 기반의 암호 알고리즘으로 이산대수 문제인 DLP(Discrete Logarithm Problem)에 기반을 두고 있으며, 이산대수 문제는 양자 컴퓨터 알고리즘 중 쇼어 알고리즘(Shor's algorithm)[3]에 의해 효율적인 시간 내에 공격이 가능하다는 문제점을 가지고 있다. 이러한 문제로 인해 기존 블록체인 시스템은 양자 컴퓨팅 환경에서 취약점을 가지게 된다.

따라서 이를 해결하기 위해 포스트 양자 암호 알고리즘 중 하나인 해시 기반 서명 기법을 적용하면 양자 컴퓨팅 환경에서의 취약점을 해결할 수 있다. 해시 기반 서명 기법은 오직 해시 함수만으로 이루어진 암호 알고리즘으로 해시 함수의 단방향성 특징에 의한 안전성에 기반을 두고 있어 현재 알려진 양자 컴퓨터를 활용한 알고리즘에 대해 강인하다. 본 논문에서는 해시 기반 서명 기법을 블록체인에 적용하되, 블록체인의 현재 구조를 활용할 수 있는 형태의 해시 기반 서명 기법을 적용한 블록체인을 제안한다. 이는 양자 컴퓨팅 환경에서의 공격에 강인하면서, 기존의 ECDSA보다 빠른 성능과 높은 안전성을 가진다.

1.3 논문 구성

본 논문은 총 5장으로 구성된다. 1장에서는 본 논문의 전반적인 개요인 연구 동기, 논문 기여 내용, 논문 구성에 관해 설명하고, 2장에서는 본 연구에 대한 배경지식으로 블록체인과 포스트 양자 암호인 해시 기반 서명 기법에 대해 설명하고, 각각에 대한 관련 연구들을 설명한다. 3장에서는 제안하는 해시 기반 서명 기법을 적용한 블록체

인의 구조에 관해 설명하며, 4장에서는 기존의 블록체인과 제안하는 해시 기반 서명
기법 기반 블록체인과의 성능을 비교 분석한다. 마지막으로 5장에서는 본 연구에 대해
결론을 내린다.

2 관련 연구

2.1 블록체인

본 장에서는 블록체인의 개념, 등장배경, 블록 구조, 블록체인 관련 연구에 대해서 알아본다.

최근 비트코인, 이더리움과 같은 전자 화폐가 주목을 받으면서 전자 화폐 보안 기술 중 하나인 블록체인에 대한 관심도가 높아지고 있다. 기존의 금융 거래 방식은 거래 장부를 한곳으로 모아 관리하며, 신뢰할 수 있는 제3의 기관(은행, 금융 회사 등)인 TTP(Trusted Third Party)를 설립하여, 거래자는 해당 기관을 사이에 두고 신뢰성 있는 거래를 하였다. 하지만 이러한 거래 방식은 중앙 집중형 방식으로 제3의 기관이 해킹당하는 경우나 해당 기관 자체에서 거래를 조작하는 경우 큰 문제가 발생할 수 있다. 반면, 블록체인을 이용한 거래 방식의 경우 모든 거래 장부를 네트워크 참여자들에게 공유하고, 분산하여 관리하기 때문에 기존의 중앙 집중형 방식의 단점을 보완할 수 있다.

블록체인 특징은 분산된 공공 거래 장부이기 때문에 중앙 서버가 따로 필요 없으며, 이로 인해 서버 구축 및 관리 비용이 절감된다. 그리고 거래를 할 경우 거래내역이 블록에 기록되는데, 일정 시간마다 거래 내역들이 블록에 저장되며 이러한 블록들이 연결되어 블록체인이 형성되게 된다. 블록체인을 사용하게 되면, 거래 발생시 거래내역이 각 네트워크 참여자의 컴퓨터에 저장되게 되고 각 참여자의 승인을 받아야 하며, 수시로 검증이 이루어지기 때문에 해킹이 어렵다. 특히 하나의 거래 정보를 수정하게 되면 그 거래가 포함된 블록을 수정해야 되고, 블록을 수정하게 되면 블록과 연결된 블록체인 전체를 수정해야 하므로 거래 기록을 조작하는 것은 매우 어렵다.

블록체인의 거래 과정에 대해 간단히 나타내면 아래와 같다.

- 1) 거래자 A가 거래자 B에게 특정 금액 송금

- 2) 해당 거래 정보가 담긴 블록 생성
- 3) 네트워크 내 모든 참여자에게 블록 전송
- 4) 모든 참여자가 해당 블록에 대해 검증을 시도(거래의 타당성 검증)
- 5) 검증을 통과한 블록을 기존 블록체인에 연결
- 6) 거래자 B에서 송금 완료

2.1.1 블록체인 구조

블록체인은 블록의 연결된 형태이이며, 블록의 구조는 아래와 같다.

블록 구조에 관해 설명해보면, 블록은 크게 4가지로 구성되어있다. 전체 블록 크기 를 나타내는 4 bytes의 블록 크기 필드, 총 80 bytes의 블록 헤더 필드, 블록에 저장된 트랜잭션의 개수를 나타내고 1~9bytes의 가변 크기를 가진 트랜잭션 개수 필드, 그리고 블록에 저장된 트랜잭션들을 저장하는 트랜잭션 필드로 구성되어있다.

블록 헤더의 구성에 대해 자세히 알아보면, 먼저 4 bytes로 구성된 버전 필드는 블록체인 소프트웨어나 사용하는 프로토콜에 대한 업그레이드 버전을 나타낸다.

32 bytes로 구성된 이전 블록 헤더 크기 필드는 블록체인에서 현재 블록 이전의 블록 헤더(80bytes)부분을 해시한 값이 저장된다. 이때, 비트코인의 블록체인 기준으로 SHA-256 해시 함수를 사용하여 이전 블록 헤더를 해싱하게 된다.

32 bytes로 구성된 머클 루트 필드는 해당 블록에 저장된 모든 트랜잭션을 이용하여 머클 트리를 구성하고 그 트리의 루트 값을 저장한 것으로 블록이 가지고 있는 모든 트랜잭션에 대한 무결성을 검증하거나, 특정 거래가 해당 블록에 포함이 되는지 아닌지를 확인할 수 있다. 머클트리는 이진 해시 트리로 자식 노드 값 두 개를 연결하고 그 값을 해싱하여 부모 노드를 구하는 방식으로 해시 트리를 생성하게 된다. 머클 트리를 이용한 머클 루트값 계산 예시는 아래와 같다.

4 bytes로 구성된 타임스탬프 필드는 블록의 생성 시간을 나타내는 필드로 유닉스



Figure 1: 블록 구조

기준일로부터 초 단위로 계산한 값이 저장된다.

각각 4 bytes로 구성된 난이도 목표 필드와 논스 필드는 채굴(Mining)이라는 과정에서 사용된다. 먼저 채굴에 관해 설명하면, 채굴은 전자 화폐를 생산하며, 블록에 대한 트랜잭션의 유효성을 검증하고, 블록을 블록체인에 연결하는 것이다. 채굴을 수행하기 위해 수많은 채굴자가 참가하게 되고, 수많은 채굴자 중 합의 메커니즘을 통해 보상으로 코인을 얻고, 블록을 생성하여 블록체인에 연결하는 채굴자를 선정하게 된다. 이러한 합의 메커니즘은 비트코인에서 작업의 증명(Proof Of Work)을 통해 이루어지며, 작

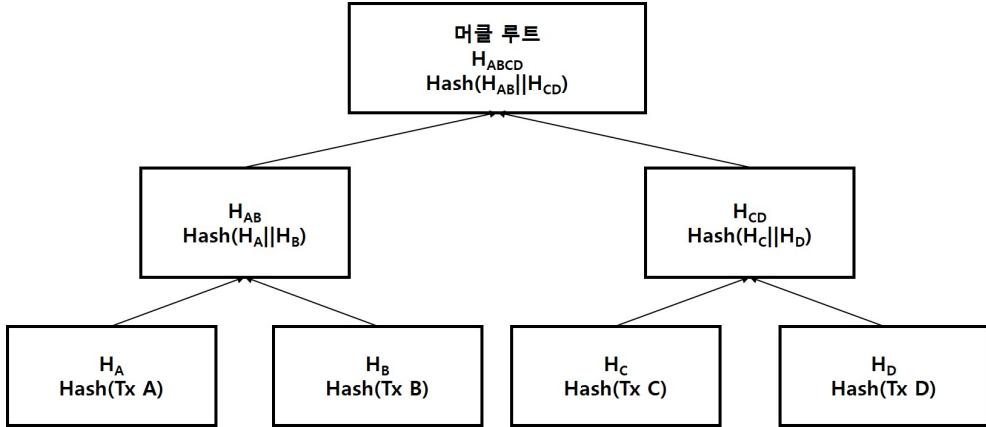


Figure 2: 머클 트리를 이용한 머클 루트 계산

업의 증명 문제로 hash cash 문제를 사용하게 된다. hash cash 문제는 hash 함수의 비가역성 특징을 이용한 문제로 논스 필드 값을 변수 범위로 두고 특정 입력값에 대한 해시값을 구해 그 값이 난이도 목표 필드 값의 범위 안에 드는지 확인한다. 이후 해당 난이도 목표에 맞는 입력값을 찾은 경우 hash cash 문제를 푼 것으로 한다. 예를 들어, "bitcoin blockchain"이라는 기본 문구가 있을 때, 채굴자는 "bitcoin blockchain1"부터 시작하여 "bitcoin blockchain2", "bitcoin blockchain3", ... 등 논스 필드의 값 범위까지 값을 증가시키며 기존 문구 뒤에 연결해 입력값으로 사용하여 해시값을 구하게 되며, 특정 난이도 값(ex. 0x100...000이하인 값을 찾아라)에 해당하는 입력값을 찾게 되면 hash cash 문제를 해결하게 된다[4].

일반적으로 hash cash 문제를 푸는데 걸리는 시간은 10분 정도 소요되며, 이는 곧 블록 생성 시 걸리는 시간이 된다. 따라서 대개 블록 생성이 10분 걸린다고 하는 말은 hash cash 문제를 해결하는데 걸리는 시간이 10분 정도라는 의미이다. 최근 컴퓨팅 성능이 향상됨에 따라 hash cash 문제를 빨리 풀게 되는데, 이러한 경우 난이도 목표 필드 값을 수정하여 hash cash 문제 푸는 시간을 10분 정도로 다시 조정시킨다.

2.1.2 트랜잭션 구조

트랜잭션의 경우 해당 네트워크 참가자 누구나 생성 가능하며, 거래 생성 후 전자 서명을 하게 되면 해당 트랜잭션은 유효해진다. 이후 전자 화폐의 송금에 필요한 정보가 추가로 저장되고, 해당 트랜잭션은 블록체인에 연결되기 전까지 주변 노드들로 전달되게 된다. 이때 전송되는 트랜잭션들은 노드들에 의해 검증이 되어 유효화되고, 만약 검증이 실패할 경우, 거절 정보가 거래 생성자에게 돌아오게 된다.

비트코인 기준 블록에 저장되는 트랜잭션의 구조는 아래와 같다.

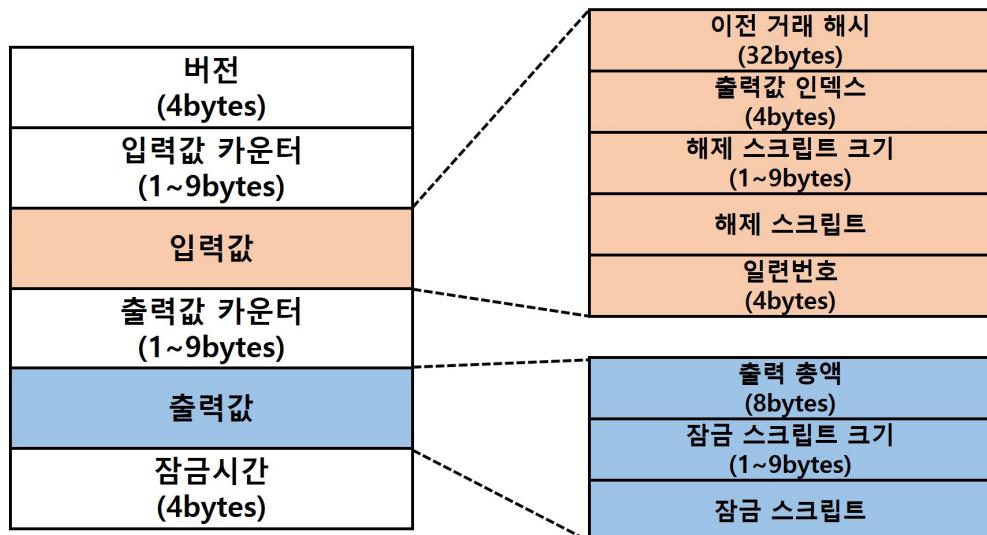


Figure 3: 트랜잭션 구조

트랜잭션의 경우, 입력값이라 불리는 출발지부터 출력값으로 불리는 도착지까지의 거래 내역을 인코딩한 데이터 구조이다. 트랜잭션에서 사용되는 화폐 단위는 UTXO(Unspent Transaction Outputs) 즉, 소비되지 않은 출력값으로 잔액 없이 오직 출력값만 존재하며, 비트코인의 경우, 0.00000001비트코인 단위인 1사토시 단위의 값을 표현할 수 있다. 먼저, 트랜잭션은 4 bytes의 버전 필드에 해당 트랜잭션이 따라야 하는 규정 정보가 저

장되어있다.

1~9bytes 가변 크기의 입력값 카운터는 트랜잭션의 입력값 개수를 나타내며, 가변 크기의 입력값 필드는 한 개 이상의 입력값 정보가 저장된다.

다음 1~9bytes 가변 크기의 출력값 카운터는 트랜잭션의 출력값 개수를 나타내며, 가변 크기의 출력값 필드는 한 개 이상의 출력값 정보가 저장된다.

마지막으로 4 bytes의 잠금 시간 필드는 트랜잭션이 블록체인에 추가되는 시간을 나타내며, 0은 바로 수행이 가능한 트랜잭션을 나타낸다.

트랜잭션의 입력값과 출력값은 또다시 여러 개의 필드로 나뉘어 진다. 먼저, 입력값의 경우, 32 bytes의 거래 해시 필드가 존재하며, 해당 필드는 소비될 UTXO를 가진 이전 트랜잭션에 대한 주소가 저장된다. 즉, 해당 트랜잭션의 입력값 출처가 저장된다.

4 bytes의 출력값 인덱스는 소비될 UTXO의 인덱스를 표시한다.

1~9bytes 가변 길이의 해제 스크립트 크기 필드는 아래의 가변 길이를 가진 해제 스크립트 크기를 나타내며, 해제 스크립트의 경우, 출력값에 저장된 잠금 스크립트의 전자 서명을 검증하기 위한 스크립트 정보가 저장되어있다. (ex. 공개키의 무결성 검증을 위한 공개키 해시값)

마지막으로, 4 bytes의 일련번호 필드는 앞서 설명한 트랜잭션에 대한 잠금 시간이 되지 전 거래를 중단하기 위해 사용되며, 현재 비트코인에서는 사용되지 않는 필드 값이다.

출력값의 경우, 8 bytes의 출력 총액 값이 저장되며, 단위는 사토시 단위이다.

1~9bytes 가변 길이의 잠금 스크립트 크기 필드는 아래의 가변 길이를 가진 잠금 스크립트 크기를 나타내며, 잠금 스크립트의 경우, 트랜잭션에 대한 전자 서명과 서명 검증에 필요한 공개키값이 저장되어있다[4].

위의 방법처럼 트랜잭션에 대한 서명을 검증하면서 트랜잭션에 대한 인증 및 무결성을 검증하게 되고, 이후 추가로 트랜잭션에 대한 검증을 트랜잭션과 연결된 트랜잭

션들의 출력값을 통합하면서 이뤄진다.

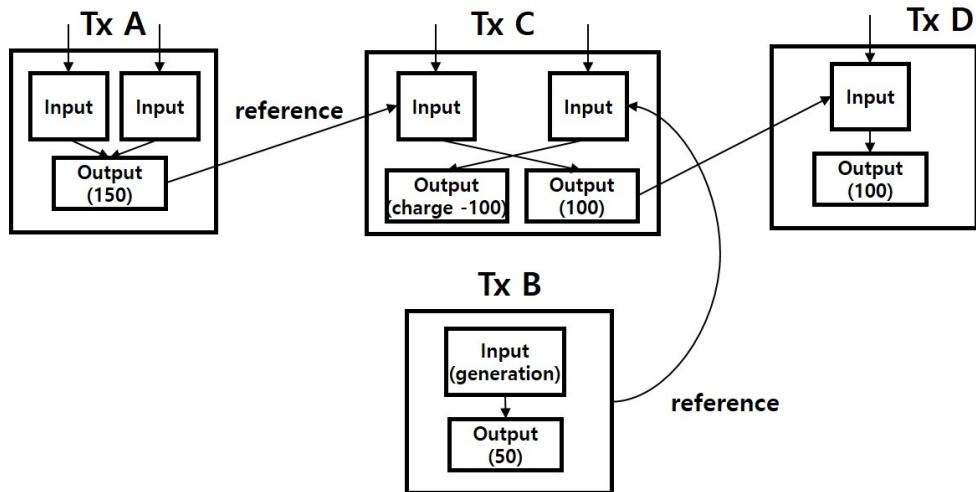


Figure 4: 전자 화폐 거래 예시

위의 예시의 경우, 거래자 C가 거래자 D에게 100BTC를 주는 과정이다. 이때 트랜잭션 C는 C가 만든 트랜잭션이라 가정한다. C의 경우, 이전 거래내역에 A로부터 150BTC를, B로부터 50BTC를 받은 내역이 있으며, 이에 대해 총 200BTC중 100BTC는 D에게 보내고, 나머지 100BTC는 자기 자신에게 청구하게 된다. (비트코인의 경우 적립의 개념이 없고 모두 출력값만 있어야 함) 이때, A의 경우는 이전 거래로 부터 150BTC가 들어온 내역이 있을 것이며, B의 경우는 입력값이 없으므로, 채굴을 통해 50BTC를 얻은 경우이다. 이처럼 이전 거래 내역들을 참조하면서 전체 통화 거래량을 분석함으로써 트랜잭션을 검증한다.

2.1.3 블록체인 관련 연구 및 활용 사례

최근 전자 화폐에 대한 관심도가 높아지면서 블록체인과 관련된 연구도 함께 활발히 진행되고 있다. 현재 국/내외의 금융권에서는 독자적인 분산 원장 기술을 개발하고

있으며, 통합 컨소시엄을 형성하여 핀테크 관련 스타트업을 진행 중이다. 그뿐만 아니라, 블록체인 기술 관련, 암호화 화폐 관련 특허를 등록하는 등 금융권에서 블록체인 기술에 대비해 다양한 활동이 진행되고 있다[1].

블록체인 기술은 금융권뿐만 아니라, 다른 서비스에서도 웹이나 애플리케이션 형태로 활용되고 있다. 기존 블록체인은 단순히 전자 화폐 거래에 사용되었지만, 지금은 전자 화폐 용도 외에도 부동산 시장에서 부동산 서비스(부동산 매물)에 적용하여 부동산 구매, 판매, 부동산 구매 판매를 관리하는 부동산 업자, 부동산 서비스를 광고하는 광고업자까지 서로 연결짓고, 부동산 서비스에 대한 신뢰성을 향상하는데 블록체인을 사용하고 있다[5].

그리고 블록체인을 이용하여, 감염병이 의심되는 환자에 대한 진료 정보를 상위 기관에 자동으로 공유하여 보고가 빠지는 경우를 대비하거나, 감염병 환자 정보를 안전하게 공유하기 위해 분산된 블록체인 시스템을 활용하기도 한다[6].

또한, 블록체인을 이용하여 사물인터넷(IoT, Internet Of Things) 디바이스에 대한 인증 스킴을 제안하는 연구도 있는데, 해당 연구에서는 인증에 필요한 암호화 모듈이 없거나, 공개키 암호를 활용할 수 없는 사물인터넷 디바이스에 대해 해시 기반 서명 기법 중 하나인 램포트 서명을 활용하여 저성능 사물인터넷 디바이스에서도 동작이 가능한 블록체인 기반 디바이스 인증 스킴을 제안하였다[7].

블록체인 기술 자체에 대한 연구 사례도 존재하는데, 비트코인에서 사용되는 합의 알고리즘인 PoW(Proof of Work)에 대해 PoW 대신 PoS(Proof of Stake), DPoS(Delegated Proof of Stake) 알고리즘에 대해 설명하고 있다. PoS의 경우, 지분 증명 알고리즘으로 PoW와 같이 채굴자의 계산 능력 대신 전자 화폐의 보유량에 따라서 합의 결정권을 얻게 되는 방식이다. 이 방식의 경우, Nothing-at-stake 문제 즉, 포크가 발생하였을 때, 두 개의 포크에 동시에 베팅할 수 있기 때문에 문제가 발생할 수 있다. 하지만 전자 화폐를 가지고 있으면, 이자의 개념으로 전자 화폐를 얻을 수 있어 채굴이 필요 없다. DPoS의

경우, 모든 노드가 투표를 통해 상위 101개의 신뢰할 수 있는 노드를 선정하게 되고, 선택된 대표 노드가 블록을 생성하는 방식이다. 만약 악의적 노드가 있는 경우, 다시 투표를 통해 네트워크에서 추방할 수 있어 안전한 블록체인을 유지하는 데 유용하다[8].

그리고 블록체인의 채굴에 대해 이기적인 채굴을 하는 경우 생기는 문제점을 제시하고, 이러한 이기적인 채굴을 방지하기 위한 대응기법으로 경쟁이 되는 블록이 생겼을 때 무작위로 블록을 선택하게 하는 랜덤 블록 채택 방법, 분기(fork)를 발생시킨 채굴자에 대해 처벌을 하는 분기된 블록 처벌 규칙 적용 방법, 타임스탬프를 활용하여 타임스탬프의 신선도에 따라 블록을 채택하는 신선한 블록 선호 방법을 설명하여 기존 블록체인에서 발생 가능한 취약점에 대해 분석하는 연구도 활발히 이루어지고 있다[9].

이러한 블록체인 기술은 웹상에서 오픈소스로 공개되어 있으며, 일반 사용자도 쉽게 블록체인을 경험할 수 있는 서비스를 제공해주는 사이트도 존재한다. 특히, IBM의 PaaS(Platform As A Service) 클라우드 서비스인 Bluemix에서 Blockchain API를 통해 블록체인 서비스를 경험할 수 있으며, 직접 트랜잭션을 생성하고, 블록을 생성하는 등 전반적인 블록체인 기술을 사용할 수 있다[9].

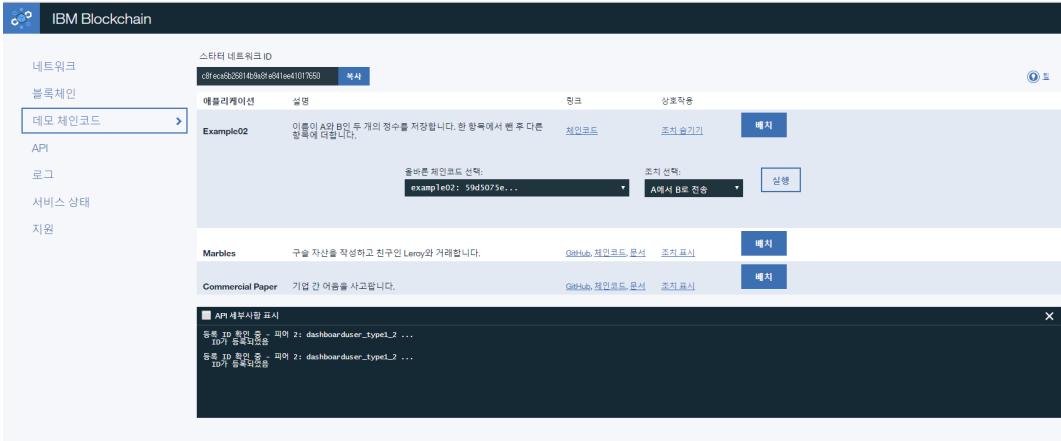


Figure 5: IBM Bluemix Blockchain

2.2 해시 기반 서명

본 장에서는 포스트 양자 암호 알고리즘 등장 배경, 포스트 양자 암호 알고리즘 종류, 해시 기반 서명기법에 대해 알아본다.

2.2.1 포스트 양자 암호 알고리즘 등장 배경

최근 양자 컴퓨터 개발 및 발전에 따라 양자 컴퓨터를 이용한 알고리즘인 쇼어 알고리즘과 그루버 알고리즘을 통한 인수분해 문제나 이산대수 문제 기반 공개키 암호 알고리즘들을 다행식 시간 안에 풀 수 있게 된다는 점이 강조되면서 포스트 양자 암호 알고리즘의 중요성이 대두되고 있다. 그뿐만 아니라, 포스트 양자 암호 알고리즘과 관련하여 2006년 PQCrypto 학회, SAFEcrypto, Crest Crypto-Math 등의 프로젝트 연구 등이 이루어지고 있으며, 미국의 NIST(National Institute of Standards and Technology)에서는 포스트 양자 암호 알고리즘에 대한 공모 사업 발표를 하여 포스트 양자 암호 알고리즘에 대한 관심도가 높아졌고 관련 연구가 활발히 진행 중이다. 대표적인 양자

컴퓨터를 이용한 양자 알고리즘에 대해 알아본다.

1. 쇼어 알고리즘

쇼어 알고리즘은 1994년 피터 쇼어가 제안한 알고리즘으로 소인수 분해를 빠르게 처리할 수 있다. 쇼어 알고리즘은 양자 푸리에 변환(Quantum Fourier Transformation)과 modular exponentiation을 이용해 인수 분해 문제를 빠르게 풀게 되는데 N 비트에 대해 걸리는 시간이 exponential time에서 $O(\log^3 N)$ 으로 단축된다. 쇼어 알고리즘의 아래와 같은 단계로 진행된다[3, 12].

- 1) N보다 작은 m 선택한다.
 - 2) m과 N 최대공약수를 계산한다.
 - 3) $\gcd(m, N) \neq 1$ 이라면 알고리즘은 종료되고, 아니라면 다음 단계로 넘어간다.
 - 4) $m^i \bmod n$ 의 주기 P를 계산한다.
 - 5) P가 홀수면 '1)' 단계로 돌아가고, 짝수면 다음 단계로 넘어간다.
 - 6) $m^{P/2} + 1 \equiv 0 \bmod N$ 이면 '1)' 단계로 돌아가고, 아니면 다음 단계로 넘어간다.
 - 7) $\gcd(m^{P/2}-1, N)$ 를 계산한다.
 - 8) 알고리즘을 종료한다.
- '2)' 단계가 양자 푸리에 변환에 해당하는 단계로, 해당 단계를 빨리 풀 수 있으므로 빠르게 소인수 분해 문제를 해결할 수 있다.

쇼어 알고리즘은 특정 주기를 찾는 알고리즘이며, 쇼어 알고리즘을 활용하면, 숨은 부분군 문제(HSP, Hidden Subgroup Problem)를 효율적인 시간내(다항시간 내(Polynomial Time))에 해결할 수 있다. 숨은 부분군 문제의 알고리즘으로는 Factoring, Discrete Logarithm Problem, Order of Permutation 등이 있는데. 이는 쇼어 알고리즘을 통해 인수 분해 문제, 이산 대수 문제를 효율적으로 풀 수 있고 이는 인수분해 문제를 이용한 RSA, 이산 대수 문제를 활용한 ECC 암호 알고리즘을 효율적인 시간 안에 풀

수 있다는 의미가 된다. 따라서 쇼어 알고리즘의 등장은 기존에 존재하는 공개키 암호 알고리즘들에게 큰 위협이 될 수 있다.

2. 그루버 알고리즘

그루버 알고리즘은 1996년 로브 그루버가 제안한 알고리즘으로 정렬이 되지 않은 데이터베이스에서 특정 데이터를 찾아내는 알고리즘으로, N개의 데이터에 대해 원하는 데이터 하나를 찾는데 $O(N)$ 의 탐색이 필요하다면, 그루버 알고리즘을 이용하면 $O(N^{1/2})$ 의 검색만으로 특정 데이터를 찾아낼 수 있다. 그루버 알고리즘은 Hadamard gate를 이용한 중첩 상태 적용과 Quantum Oracle 함수를 이용해 특정 데이터를 찾을 수 있도록 한다. 이때, Quantum Oracle은 시스템의 상태에 따라 위상 변환을 시키고, 올바른 값에 대해 진폭의 위상을 바꾸게 된다. 이후 Diffusion Transform을 이용하여 위상 변화를 시켜 찾고자 하는 데이터를 찾아내게 된다[11, 12].

그루버 알고리즘을 이용하면, 기존의 대칭키 암호 알고리즘인 AES, LEA, HIGHT 와 해시함수인 SHA, LSH 등에 대해 보다 효율적인 시간 내에 공격이 가능하다. 따라서 이에 대해 대칭키 암호 알고리즘의 경우, 보다 큰 키 값이 필요하고, 해시 함수의 경우, 보다 큰 메시지 다이제스트값이 필요하다.

이러한 양자 알고리즘들은 IBM Quantum Experience를 통해 실제 IBM에서 개발한 5 qubit의 양자 프로세서에 자신이 만든 프로그램을 실행시켜 볼 수 있다[13].

양자 알고리즘들은 현대 암호 알고리즘에 큰 영향을 미치고 있다. 쇼어 알고리즘은 공개키 암호 알고리즘에, 그루버 알고리즘은 대칭키 암호 알고리즘과 해시 함수에 영향을 미치며, 대칭키 암호 알고리즘의 경우, 키 길이 증가가 필요하고, 해시 함수의 경우는 출력 길이의 증가가 필요하다. 그리고 인수분해 문제 혹은 이산대수 문제를 기반으로 하는 공개키 암호 알고리즘은 더 이상 안전하지 않다.

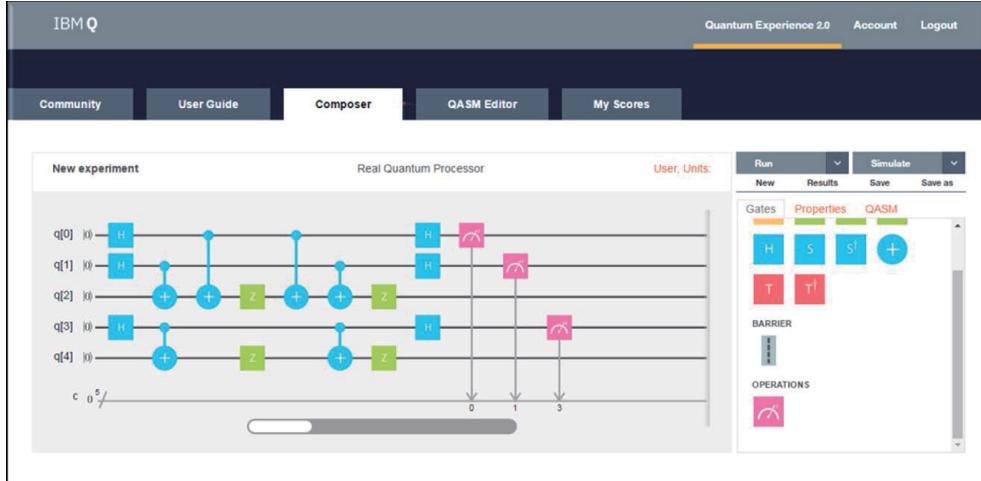


Figure 6: IBM Quantum Experience

2.2.2 포스트 양자 암호 알고리즘

앞서 알아본 바와 같이 포스트 양자 암호 알고리즘은 기존의 인수 분해 문제나 이산 대수 문제에 기반을 두지 않는, 또 다른 어려움에 기반을 둔 암호 알고리즘이어야 한다. 이에 대해 NP-hard 문제를 이용하거나, 해시 함수의 단방향성을 활용하는 등 여러 연구가 진행되고 있다.

대표적인 포스트 양자 암호 알고리즘은 크게 5가지가 있다.

- 다변수 기반 암호 알고리즘(Multivariate-based Cryptography) : NP-hard 문제인 유한체내에서의 다변수 함수를 푸는 것을 이용한 암호 알고리즘(ex. Rainbow Signature)
- 코드 기반 암호 알고리즘(Code-based Cryptography) : NP-hard 문제인 유한체 내에서의 다변수 함수를 푸는 것을 이용한 암호 알고리즘(ex. McEliece)
- 격자 기반 암호 알고리즘(Lattice-based Cryptography) : Lattice 상에서 문제를

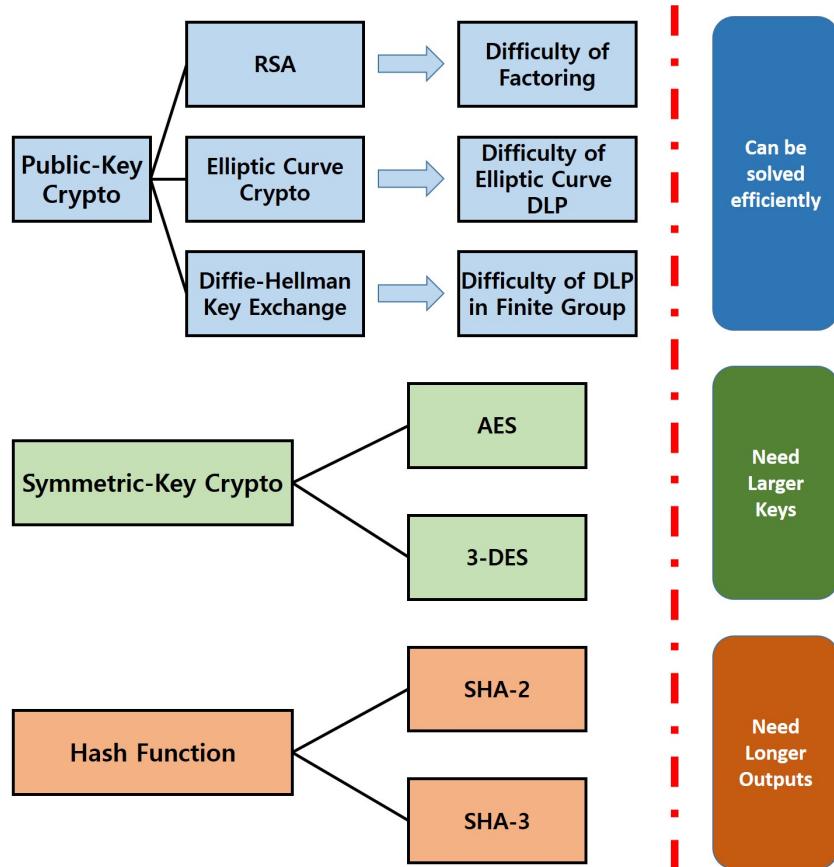


Figure 7: 양자 컴퓨팅 환경에서의 현대 암호 안전성

푸는 것. 행렬의 곱에 대해 특정한 오류를 추가하게 되면 곱하는 행렬을 구하기 힘들다는 것을 활용한, NP-hard 문제를 기반으로한 암호 알고리즘(ex. NTRU, Frodo)

- Isogeny 기반 암호 알고리즘(Isogeny-based Cryptography) : Order가 동일한 두 타원곡선에 대해 두 곡선 사이에 존재하는 isogeny를 구하는 것이 NP-hard 문제임을 이용한 암호 알고리즘
- 해시 기반 서명 기법(Hash-based Signature Scheme) : 해시 함수의 안전성을

바탕으로 하는 전자 서명 기법(MSS, Sphincs)

2.2.3 해시 함수

해시 함수는 임의의 길이를 가진 문자열을 고정된 길이의 이진 문자열로 매핑해주는 단방향 함수로 입력값이 약간만 변하더라도 그 결과값은 전혀 다른 값을 출력하게 된다. 따라서 해시 함수의 결과값으로부터 입력값을 유추하기 힘들다. 이때, 해시 함수의 결과값은 해시값 혹은 메시지 디제스트라 한다. 해시 함수의 대표적인 특성은 아래와 같다.

- 역상 저항성(preimage resistance) : 입력을 모르는 해시값 y 가 주어졌을 때, $H(x) = y$ 가 되는 x 를 찾는 것은 계산적으로 어렵다.(one-wayness)
- 제2역상 저항성(2nd preimage resistance) : 입력 값 x 에 대해, $H(x) = H(x')$ 이 되는 x' 를 찾는 것은 계산적으로 어려워야 한다.(weak collision-resistance)
- 충돌 저항성(collision resistance) : $H(x) = H(x')$ 인 서로 다른 두 입력 x 와 x' 을 찾는 것은 계산적으로 어려워야 한다.(strong collision-resistance)

이러한 해시함수의 종류로는 SHA-1, SHA2(SHA-224/SHA-256/SHA-384/SHA-512), SHA3(SHA3-224/SHA3-256/SHA3-288/SHA3-384/SHA3-512), LSH(LSH-224/LSH-256/LSH-384/LSH-512)등이 있다. 특히, SHA(Secure Hash Algorithm)-2 해시 함수는 2001년 미국 표준 기술 연구소 NIST에서 발표된 알고리즘으로, 2002년에 정식 표준으로 지정되었다. 2004년에는 해시값 길이를 조정한 SHA-224/384가 추가되었으며, 현재 SHA-2 계열의 해시 함수가 가장 널리 사용되고 있고, 블록체인에서도 해시 함수로 SHA-256을 사용하고 있다[14].

SHA-2 계열의 압축 함수는 아래와 같은 연산으로 구성된다.

Table 1: SHA-2 계열 해시 함수 특성

SHA-2	출력 크기 (bit)	블록 크기 (bit)	최대 메시지 길이(bit)	라운드 수	구성 연산	보안 강도 (bit)
SHA-224	224	512	$2^{64}-1$	64	AND, XOR, ROT, ADD, OR, SHR	112
SHA-256	256	512	$2^{64}-1$	64	AND, XOR, ROT, ADD, OR, SHR	128
SHA-384	384	1024	$2^{128}-1$	80	AND, XOR, ROT, ADD, OR, SHR	192
SHA-512	512	1024	$2^{128}-1$	80	AND, XOR, ROT, ADD, OR, SHR	256

SHA-2 해시 함수의 압축 함수를 구성하는 연산은 아래의 식으로 구성된다.

$$Ch(E, F, G) = (E \& F) \oplus (\neg E \& G) \quad (1)$$

$$Ma(A, B, C) = (A \& B) \oplus (A \& C) \oplus (B \& C) \quad (2)$$

$$\sum 0(A) = (A >>> 2) \oplus (A >>> 13) \oplus (A >>> 22) \quad (3)$$

$$\sum 1(E) = (E >>> 6) \oplus (E >>> 11) \oplus (E >>> 25) \quad (4)$$

2.2.4 해시 기반 OTS(One-Time Signature)

해시 기반 One-Time Signature는 해시 기반 서명 기법 중 하나의 비밀키/공개키 쌍으로 하나의 문서에 대해 서명이 가능한 알고리즘이다. 해시 기반 One-Time Signature

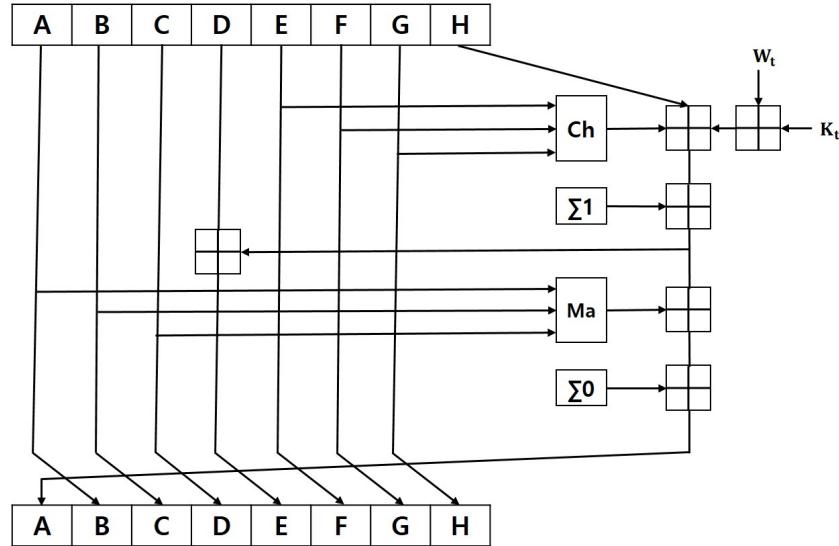


Figure 8: SHA-2 해시 압축 함수 블록 다이어그램

의 종류로는 서명하고자 하는 문서의 다이제스트 값을 인덱스로 사용하여 서명하는 Lamport-Diffie One-Time Signature, Lamport-Diffie One-Time Signature의 큰 서명 사이즈를 줄이기 위해 특정한 Winternitz Parameter를 정하고, 그 값에 따라 키와 서명의 크기를 압축하는 Winternitz One-Time Signature, Winternitz One-Time Signature에서, 더욱더 서명 사이즈를 줄이고자 어떠한 랜덤값을 bit masking 하여 서명 값을 줄인 $W - OTS^+$, Random Subset을 바탕으로 서명을 생성하는 HORS(Hash to Obtain Random Subset), 그리고 그것의 확장 버전인 HORSE(HORS Extended)까지 여러 가지의 해시 기반 One-Time Signature가 있다[15, 16, 17].

1. Lamport-Diffie One-Time Signature

Lamport-Diffie One-Time Signature는 1979년 Leslie Lamport가 제안한 서명 기법이다. 기본적으로 LD-OTS(Lamport-Diffie One-Time Signature)는 $n \times 2n$ -bit 크기의

서명키 X와 검증키 Y를 사용한다.

$$X = (x_{n-1}[0], x_{n-1}[1], \dots, x_1[0], x_1[1], x_0[0], x_0[1]) \in_R \{0, 1\}^{(n, 2n)} \quad (5)$$

$$Y = (y_{n-1}[0], y_{n-1}[1], \dots, y_1[0], y_1[1], y_0[0], y_0[1]) \in_R \{0, 1\}^{(n, 2n)}, \quad (6)$$

$$y_j = f(x_i[j]), 0 \leq i \leq n-1, j = 0, 1 \quad (7)$$

이러한 서명키 X와 검증키 Y를 생성하는 과정에서 one-way function(블록체인에서는 SHA-256)을 $2n$ 회 호출하게 된다.

Lamport-Diffe One-Time Signature 생성 과정은 어떠한 문서 M에 대해서 one-way function인 g를 이용해 메시지 다이제스트 $g(M) = d = (d_{n-1}, \dots, d_0)$ 와 검증키 X를 사용해 아래의 서명 값을 생성한다.

$$\sigma = (x_{n-1}[d_{n-1}], \dots, x_0[d_0]) \in 0, 1^{(n, n)} \quad (8)$$

Lamport-Diffe One-Time Signature 검증 과정의 경우, 검증키 Y, 단방향 함수 f를 사용하여 아래의 조건을 만족하는지 확인한다.

$$(f(\sigma_{n-1}, \dots, f(\sigma_0)) = (y_{n-1}[d_{n-1}], \dots, y_0[d_0]) \quad (9)$$

두 값이 같으면 서명 검증에 성공한 것으로 간주된다[18].

2. Winternitz One-Time Signature

앞서 설명한 Lamport-Diffie One-Time Signature의 경우, 효율적인 서명을 생성 하지만, 서명의 크기가 서명키, 검증키와 마찬가지로 매우 크다는 단점을 가지고 있다. 이러한 단점을 보완하기 위해서 Winternitz One-Time Signature에서는 메시지 다이제

스트값의 일부 비트에 대해 동시에 서명하는 방식을 사용하여, 서명키, 검증키, 서명의 크기를 전반적으로 압축시켰다[19]. Winternitz One-Time Signature의 키 쌍생성 과정은 먼저 동시에 서명하고자 하는 비트의 수를 나타내는 특정 상수 값인 Winternitz Parameter(w)를 2 이상의 수로 선택하고, 아래의 수식과 같이 t_1 , t_2 , 그리고 t 값을 계산한다.

$$t_1 = \lceil n/w \rceil \quad (10)$$

$$t_2 = \lceil (\lfloor \log_2 t_1 \rfloor + 1 + w)/w \rceil \quad (11)$$

$$t = t_1 + t_2 \quad (12)$$

Winternitz One-Time Signature의 서명키 X와 서명키 Y는 아래의 식을 통해 생성된다.

$$X = (x_{n-1}, \dots, x_1, x_0) \in \{0, 1\}^n \quad (13)$$

$$Y = (y_{n-1}, \dots, y_1, y_0) \in \{0, 1\}^n, y_i = f^{2^w-1}(x_i), 0 \leq i \leq t-1 \quad (14)$$

서명키 X의 x_i 는 uniformly random 한 값으로 설정되며, 키 생성 과정에서의 단방향 함수 f 의 수행횟수는 $t(2^w-1)$ 회이다. 그 결과로 t^*n -bit 길이의 서명 키와 검증키가 생성된다.

Winternitz One-Time Signature의 서명 생성 과정은 어떠한 문서 M에 대한 메시지 다이제스트값을 단방향 함수 g를 통해 구한다. 생성된 다이제스트 $g(M) = d = (d_{n-1}, \dots, d_0)$ 에 대해 그 길이가 winternitz parameter(w)에 의해 나누어질 수 있도록 0의 bit 값을 값 앞에 패딩 한다. 패딩 과정을 거친 후 메시지 다이제스트 d는 t_1 개의 스트링 ($d = b_{t-1} \| \dots \| b_{t-t_1}$, $b_i \in \{0, 1, \dots, 2^w-1\}$)으로 나누고, 다이제스트 d에 대한 checksum을 아래의 식을 통해 계산한다.

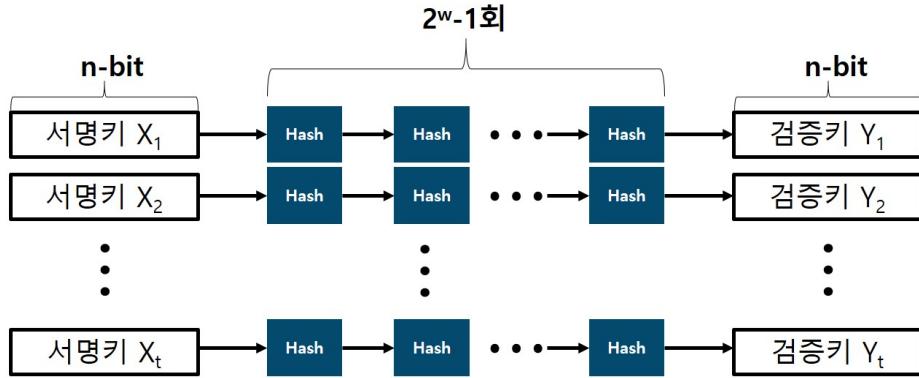


Figure 9: W-OTS 키 쌍 생성 과정

$$c = \sum_{i=t-1}^{t-1} (2^w - b_i) \quad (15)$$

계산된 checksum(c)는 winternitz parameter(w)로 나누어질 수 있도록 0의 bit를 c 값 앞에 패딩 한다. 패딩 과정 후 checksum(c)를 t_2 개의 스트링($c=b_{t_2-1}\|... \| b_0$)으로 나눈다. 문서 M에 대한 메시지 다이제스트값인 d와 checksum(c), 그리고 서명키 X를 이용해 아래의 수식에 맞춰 서명 값을 생성한다.

$$\sigma = (f^{b_{t-1}}(x_{t-1}), \dots, f^{b_1}(x_1), f^{b_0}(x_0)) \quad (16)$$

서명 생성과정은 키 생성과정과 마찬가지로 총 $t(2^w-1)$ 회 단방향 함수 f를 사용해 수행하며, 생성된 서명의 길이는 $t*n$ -bit 길이를 가진다.

Winternitz One-Time Signature의 검증 과정은 서명 값($\sigma=\sigma_{n-1}, \dots, \sigma_0$)과 앞서 구했던 메시지 다이제스트로부터 만든 bit string($b=(b_{t-1}, \dots, b_0)$)을 사용하여 아래의 수식 조건에 맞춰 서명 검증을 수행한다.

$$(f^{2^w-1-b_{r-1}}(\sigma_{n-1}), \dots, f^{2^w-1-b_0}(\sigma_0)) = (y_{n-1}, \dots, y_0) \quad (17)$$

좌변과 우변의 값이 동일 하다면, $\sigma_i = f^{b_i}(x_i)$ 조건을 만족하는것이기 때문에 결론적으로 아래의 수식과 같이 서명값의 검증 확인이 가능하다.

$$f^{2^w-1-b_i}(\sigma_i) = f^{2^w-1}(x_i) = y_i \quad (18)$$

Winternitz One-Time Signature 검증 과정에서의 단방향 함수 수행 횟수는, t개의 항목에 대해 모두 단방향 함수 f를 수행해야 하므로, 총 $t(2^w-1)$ 회의 단방향 함수 f를 수행한다. 그리고 Winternitz One-Time Signature의 security level은 안전한 PRF를 통해 키를 생성하였다고 가정한다면, $n - w - 1 - 2\log(tw)$ 이다[20].

앞서 살펴본 Lamport-Diffie One-Time Signature와 Winternitz One-TIme Signature의 서명키 크기, 검증키 크기, 서명 크기, 키 생성 시간은 아래의 표와 같다.

	LD-OTS	W-OTS
서명키 크기	$2bm$	$\sim 2bm/\log w$
검증키 크기	$2bm$	$\sim 2bm/\log w$
서명 크기	bm	$\sim 2bm/\log w$
키 생성 시간	$\sim 2m$	$\sim wm/\log w$

Table 2: LD-OTS와 W-OTS의 서명키/검증키 크기, 서명 크기, 그리고 키 생성 시간
(b: security level, w: Winternitz parameter, m: message length)

Winternitz One-Time Signature는 Lamport-Diffie One-Time Signature보다 작은 크기의 서명키/검증키, 그리고 서명을 사용하며, 키 생성 시간도 더욱 빠르다. 이러한 Winternitz One-Time Signature 특성 때문에 해시 기반 서명 기법에서 OTS로 많이 사용되고 있다[19].

2.2.5 해시 기반 MSS(MerkleTree Signature Scheme)

해시 기반 Merkle Tree Signature Scheme은 해시 기반 One-Time Signature와 달리 완전 이진 트리(Complete Binary Tree)의 일종인 머클 트리(Merkle tree)를 이용하여 하나의 비밀키/공개키 쌍으로부터 다중의 문서(머클트리의 높이가 H 라고 하였을 때, 2^H 개의 문서)를 서명/검증할 수 있는 구조로 구성되어있다. 해시 기반 Merkle Tree Signature Scheme의 종류로는 먼저, 가장 기본이 되는 머클 트리를 이용한 서명 기법인 MSS(Merkle Tree Signature Scheme)가 있고, 기존의 머클 트리에서 리프 노드에서는 $W - OTS^+$ 를 사용하고, 트리를 생성할 때에도 랜덤한 바이너리값을 bitmasking하여 트리를 생성하는 방식을 이용한 XMSS(eXtended Merkle Signature Scheme), XMSS에서 각각의 리프 노드에 대해 다시 Tree를 생성하여 문서의 개수에 상관없이 트리를 생성할 수 있는 방식인 $XMSS^{MT}$ (eXtended Merkle Signature Scheme Multiple Tree), 기존의 트리 형태의 서명과는 달리 현재 상태에 관계없이 트리를 구성할 수 있는 Stateless 방식의 해시 기반 서명 기법인 SPINCS 등이 있다[15, 16, 17].

그중에서도 가장 기본이 되는 Merkle Tree Signature Scheme에 관해 설명해보면, Merkle Tree Signature Scheme은 머클 트리 구조를 사용하여 트리의 높이 H ($H \in N, H \geq 2$)에 따라서 최대 2^H 개의 문서에 대해 서명과 검증을 할 수 있다. Merkle Tree Signature Scheme에서의 공개키(검증키)는 머클 트리의 루트에 위치하며, 2^H 개의 개인키(서명키)는 리프 노드에 위치하게 된다. 리프 노드에는 각 문서에 대한 서명키와 문서의 메시지 다이제스트 값($g(M_i), 0 \leq i \leq 2^H$)을 가지고 있다. 머클 트리의 이너 노드들은 각자 자신의 자식 노드의 연결(Concatenation)한 결과값에 대한 해시값 ($v_h[j] = g(v_{h-1}[2j] | v_{h-1}[2j + 1]), 1 \leq h \leq H, 0 \leq j \leq 2^{H-h}$)을 가진다.

머클트리의 서명키와 검증키 생성과정은 총 2^H 개의 리프노드 OTS에 대한 서명키/검증키 쌍을 생성하고, 총 $2^{H+1}-1$ 회의 해시 함수를 수행해야 한다. 이때, 머클 트리의

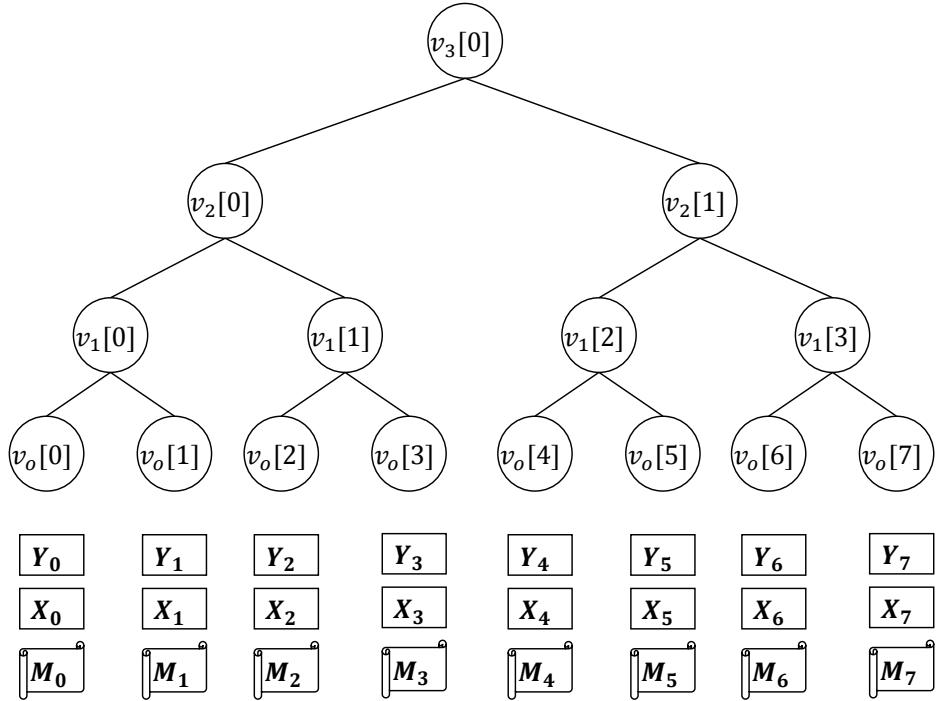


Figure 10: Merkle tree Signature Scheme 구조(트리 높이 3)

루트값인 공개키에 대해 효율적으로 계산할 수 있는 알고리즘으로 stack 구조를 사용하는 Treehash 알고리즘이 있으며, 연산 도중 최대 H개의 노드만 저장하여 알고리즘을 수행하기 때문에 메모리 사용 측면에서 효율성이 높은 알고리즈다.

Treehash 알고리즘에서는 노드 저장을 위해 스택을 사용하며, 스택의 POP, PUSH 연산을 이용한다. LEAFCALC(j) 연산은 j번째 리프 노드에 대한 연산으로 j번째 검증 키로부터 j번째 노드를 계산하는 과정이다.

Merkle Tree Signature Scheme에서의 서명 생성 과정은 리프 노드 상에서 생성된 One-Time Signature의 서명키를 사용하며, 문서 M에 대해 서명하기 위해 문서 M에 대한 메시지 다이제스트값을 구하고 서명키 $X_s (s \in \{0, \dots, 2^H - 1\})$ 를 이용해 One-Time Signature의 서명 σ_{OTS} 를 생성한다. 이때, Merkle Tree Signature에는 OTS 서명과

Algorithm 1: Treehash 알고리즘

Require: 머클트리 h
Ensure: R : 머클트리 루트값

- 1: **for** $j \in 0..2^H - 1$ **do**
- 2: j번째 리프 노드 계산 : $NODE_1 \leftarrow LEAFCALC(j)$
- 3: **while** $NODE_1.h = ROOT.h$ (스택의 최상위 노드와 동일한 높이를 가지는 경우) **do**
- 4: $NODE_2 \leftarrow STACK.pop()$
- 5: $NODE_1 \leftarrow g(NODE_2 \parallel NODE_1)$
- 6: $STACK.push(NODE_1)$
- 7: **end while**
- 8: **end for**
- 9: R : 스택에 저장된 단일 노드, $R \leftarrow STACK.pop(NODE_1)$
- 10: **return** R

OTS 서명 검증에 필요한 검증기 Y_s 를 포함하게 된다. 이후, 서명에 대한 검증을 수월하게 하려고, 검증자에게 리프 노드의 인덱스 값인 s 와 검증키인 머클 루트 값을 구하기 위한 인증 경로(AP, Authentication Path) $A_s = \{a_0, \dots, a_{H-1}\}$ 를 제공한다. 인증 경로상의 노드 h 는 인증 경로상에서 높이가 h 인 노드의 형제 노드를 나타내며, 인증 경로 인덱스 계산 공식은 아래와 같다.

$$a_h = v_h[\lceil (s/2^h) - 1 \rceil], \text{if } \lfloor s/2^h \rfloor \equiv 1 \pmod{2}, h = 0, \dots, H-1 \quad (19)$$

$$a_h = v_h[\lceil (s/2^h) + 1 \rceil], \text{if } \lfloor s/2^h \rfloor \equiv 0 \pmod{2}, h = 0, \dots, H-1 \quad (20)$$

예를 들어, 높이가 3인 Merkle Tree Signature Scheme에서 인덱스 s 가 3일 때, 서명 생성과정은 아래와 같다.

s 가 3인 경우에 Merkle Tree Signature Scheme의 서명값은 $\sigma_s = \{3, \sigma_3, Y_3, (a_0, a_1, a_2)\} = \{s, \sigma_s, Y_s, (a_0, \dots, a_{H-1})\}$ 이다.

Merkle Tree Signature Scheme의 서명값 검증 과정은 크게 2가지 단계로 나뉜다.

- 1) 검증자가 OTS 검증키인 Y_s 를 사용하여 OTS 서명 σ_{OTS} 를 검증한다.

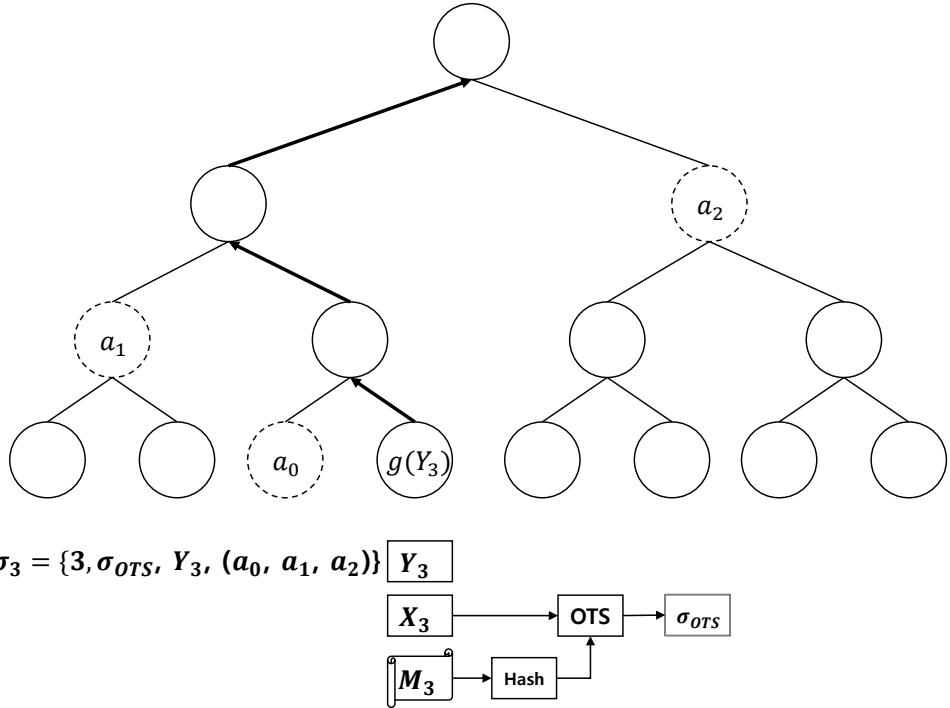


Figure 11: Merkle tree Signature Scheme 서명 생성 과정(트리 높이 3)

2) 검증자가 s 번째 리프 노드인 $g(Y_s)$ 로부터 머클 트리의 루트까지 전송 받은 인증 경로를 이용하여 검증을 하게 되며, 그 검증 과정은 아래와 같다.

$$p_h = g(a_{h-1} \parallel p_{h-1}), \text{ if } \lfloor s/2^{h-1} \rfloor \equiv 1 \pmod{2}, h = 0, \dots, H, p_0 = g(Y_s) \quad (21)$$

$$p_h = g(p_{h-1} \parallel a_{h-1}), \text{ if } \lfloor s/2^{h-1} \rfloor \equiv 0 \pmod{2}, h = 0, \dots, H, p_0 = g(Y_s) \quad (22)$$

인덱스 s 는 인증 패스 노드들의 순서를 정하는 데에 사용되고, 위의 식을 통해 최종적으로 얻게 되는 p_H 가 머클 트리의 루트값과 같은 경우, 검증키 Y_s 에 대한 검증이 최종적으로 마무리 된다[21, 22].

Merkle Tree Signature Scheme에서는 리프 노드의 OTS 알고리즘으로 여러 다양한

OTS 알고리즘이 사용될 수 있다. (ex. LD-OTS, W-OTS, $W - OTS^+$ 등)

2.2.6 해시 기반 서명 기법 관련 연구 및 활용 사례

최근 IBM의 5 qubit 양자 컴퓨터 개발 및 다른 양자 컴퓨터 관련 연구가 활발히 이루어짐에 따라 더불어 양자 컴퓨팅에 대비한 포스트 양자 암호 알고리즘 관련 연구가 활발히 진행되고 있다. 해시 기반 서명 기법 관련하여, 여러 연구가 진행되고 있는데, 먼저, 해시 기반 서명 기법의 동향과 관련하여 해시 기반 서명의 전반적인 동향에 관해 연구가 이루어지고 있다[23, 24]. 또한, 기존의 해시 기반 서명 기법을 개선할 수 있는 알고리즘들이 연구되기도 하는데, 기존 알고리즘인 MSS(Merkle tree Signature Scheme)에서 개인키 크기와 키 쌍생성 시간, 서명 생성 시간을 줄인 CMSS[25], 기존의 XMSS, GMSS 등의 해시 기반 서명 트리 알고리즘에 대해 multi buffer 기법을 적용해 키 생성 시간이나 서명 생성 및 검증 시간을 단축한 연구도 있다[26]. 그리고 대부분의 해시 기반 서명 기법은 stateful한 특성인 상태 기반의 연산을 통한 메모리 소모 증가 및 연산의 순차적 실행을 보완할 수 있는 stateless 한 해시 기반 서명 기법 연구가 이루어 지고 있으며, 2015년 stateless 한 해시 기반 서명 기법인 SPHINCS가 제안되었다[27].

이러한 해시 기반 서명 기법을 사물인터넷 환경에 적용하려는 노력도 계속되고 있다. 8-bit 마이크로프로세서 상에서 RSA, ECDSA를 대체하고자 MSS를 적용한 사례[28], AVR-ATxmega 계열 보드에서의 인증 경로 계산 연산 고속화를 통한 해시 기반 서명 기법 성능 향상 연구[29], ARM 계열 보드에서 stateless 한 해시 기반 서명 알고리즘인 SPHINCS 알고리즘 적용 연구가 진행되었다[30].

3 제안 기법

많은 사람이 이용하고 있는 비트코인, 이더리움 등의 전자 화폐 서비스의 블록체인에서는 트랜잭션에 대한 서명 알고리즘으로 타원 곡선 기반 전자 서명 기법인 ECDSA(Elliptic Curve Digital Signature Algorithm)를 사용한다. 비트코인의 경우, secp256k1 타원곡선을 사용하고 있으며, ed25519, secp256r1 타원 곡선 등 대부분 트랜잭션에 대한 서명 알고리즘으로 ECDSA 전자 서명 알고리즘을 사용하고 있다.

그런데, 최근 양자 컴퓨터에 개발이 활발해 지면서 양자 컴퓨팅으로 인한 현대 암호의 불안함이 커지고 있다. 특히, 현재 사용되는 ECDSA 전자 서명 기법은 쇼어 알고리즘을 통해 효율적인 시간내(다항시간 내(Polynomial Time))에 암호를 풀 수 있어 기존의 블록체인 서비스가 위협해질 수 있다. 특히, 일부 전자 화폐 시스템에서는 하나의 개인 키만으로 수십 개의 거래에 대해 서명을 하는 등 보안에 대해 안일하게 생각하고 서비스를 이용하는 경우도 있다. 따라서 해당 장에서는 기존의 블록체인 구조에 대해 양자 알고리즘에 내성을 가진 포스트 양자 암호 알고리즘 중 하나인 해시 기반 서명 기법을 적용하여 양자 컴퓨팅 환경에 강인한 블록체인 시스템을 제안한다.

3.1 트랜잭션 서명 기법

기존의 블록체인은 양자 알고리즘(쇼어 알고리즘)에 취약한 ECDSA 전자 서명 알고리즘을 사용하고 있는 경우가 많다. 이에 대해 양자 알고리즘에 강인한, 포스트 양자 암호 알고리즘 중 해시 함수 특성을 기반으로 안전성을 보장하는 해시 기반 서명 기법을 적용하는 것을 제안한다. 많은 해시 기반 서명 기법이 있지만, 그중에서도 하나의 서명키로 하나의 검증키만을 생성하게 하는 OTS 계열 중 서명의 길이가 짧은 편에 속하는 Winternitz One-Time Signature를 적용하여 트랜잭션에 대해 서명하고 검증을 수행한다.

3.2 머클 트리 생성 및 머클 서명 생성 기법

기존 블록체인에서는 채굴자가 블록을 생성하게 되었을 때, 블록 내의 모든 트랜잭션을 리프 노드로 두고 머클 트리를 생성하여 머클 루트값을 블록 헤더에 저장해 둔다. 이러한 구조는 해시 기반 서명 기법 중 Merkletree Signature Scheme과 매우 유사한 구조로 기존 블록체인의 구조를 이용해 트랜잭션에서 서명한 W-OTS와 더불어 MSS를 함께 적용하여 트랜잭션 검증 및 전체 블록에 대한 무결성 검증을 함께 수행할 수 있도록 해시 기반 서명 기법 트리 구조를 제안하며, 이후 머클 서명 값에 대한 검증이 빠르게 이루어질 수 있도록, 머클 트리를 생성하고 난 다음 인증 경로(Authentication Path) 값을 모든 트랜잭션에 추가하여 머클 서명값을 저장시킨다.

이에 대한 전반적인 구조는 아래와 같다.

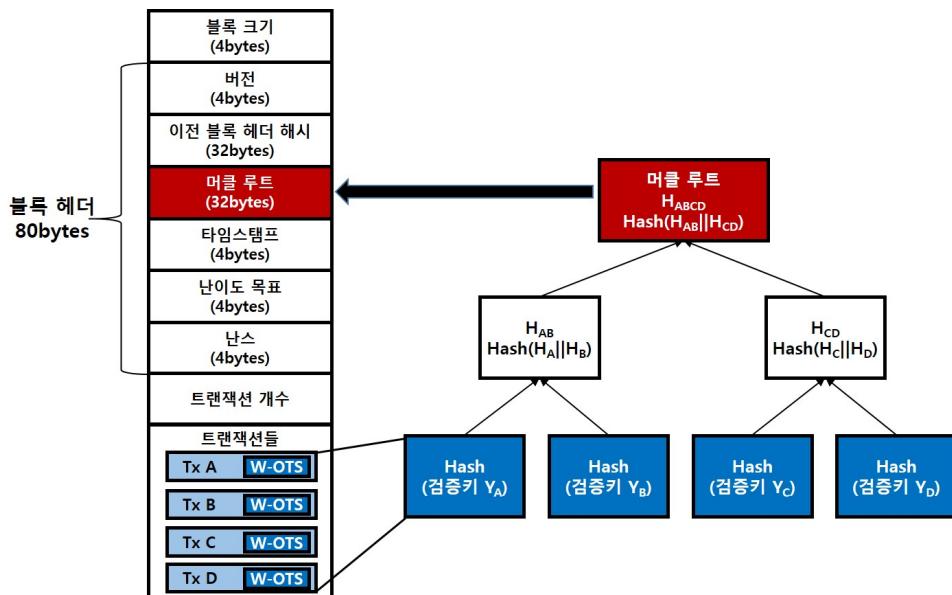


Figure 12: 제안 기법의 머클 트리 생성 과정

앞서 제안한 트랜잭션 서명 기법을 통해 각 트랜잭션은 W-OTS로 서명이 된 상태에서, 트랜잭션별 검증키에 대한 해시한 값을 리프 노드로 하는 머클 트리를 생성한다.

그리고 머클 루트값이 구해지면 머클 루트값을 블록 헤더의 머클 루트 필드에 저장한다.

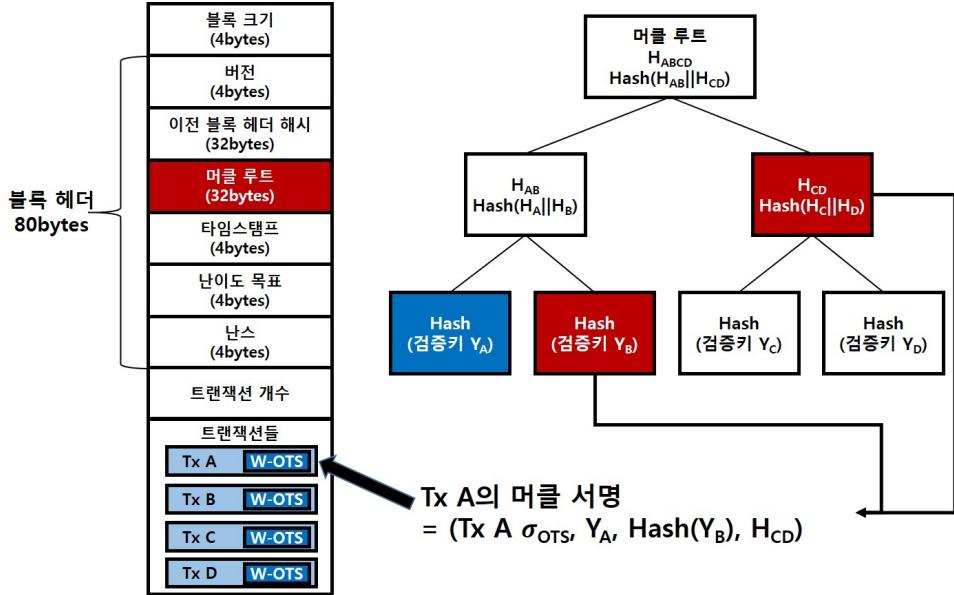


Figure 13: 제안 기법의 머클 서명 생성 과정

이후, 머클 서명을 구하게 되는데, 해당 트랜잭션의 W-OTS 서명, 검증키, 그리고 트랜잭션에 따른 인증 경로에 해당하는 값을 묶어 머클 서명값으로 하고, 해당 트랜잭션에 저장한다.

3.3 트랜잭션 검증 기법

기존 블록체인에서는 트랜잭션에 대해 검증을 할 때 먼저 트랜잭션에 저장된 서명을 검증하여 트랜잭션에 대한 인증 및 무결성을 확인하고, 이후 트랜잭션별 출력값을 통해 해당 거래 내역이 올바른 것인지 검증하게 된다.

트랜잭션의 서명을 검증하는 데 있어서 ECDSA를 사용하고 있으며, 이는 앞에서 설명한 바와 같이 양자 알고리즘에 취약점을 가지고 있으므로, 이를 W-OTS 서명 기법으로 대체한다. 그리고 앞서 머클 트리를 생성하면서 만들었던 머클 서명값이 블록 생성 후 트랜잭션에 저장되게 되는데, 이를 검증하려면, 먼저 트랜잭션에 대한 W-OTS

검증을 수행한다. 이로써 트랜잭션에 대한 인증 및 무결성은 보장된다. 게다가 머클 해시값에 대한 검증을 함께 저장된 인증 경로를 이용해 검증함으로써 가지고 있던 공개 키에 대한 검증(블록 생성 전 공개키가 그대로 유지가 되어있는지 무결성 확인)과 머클 해시 값 검증을 통한 전체 트랜잭션에 대한 검증, 그리고 검증하고자 하는 트랜잭션이 해당 블록에 포함되는지도 검증이 된다.

검증 과정은 아래와 같다.

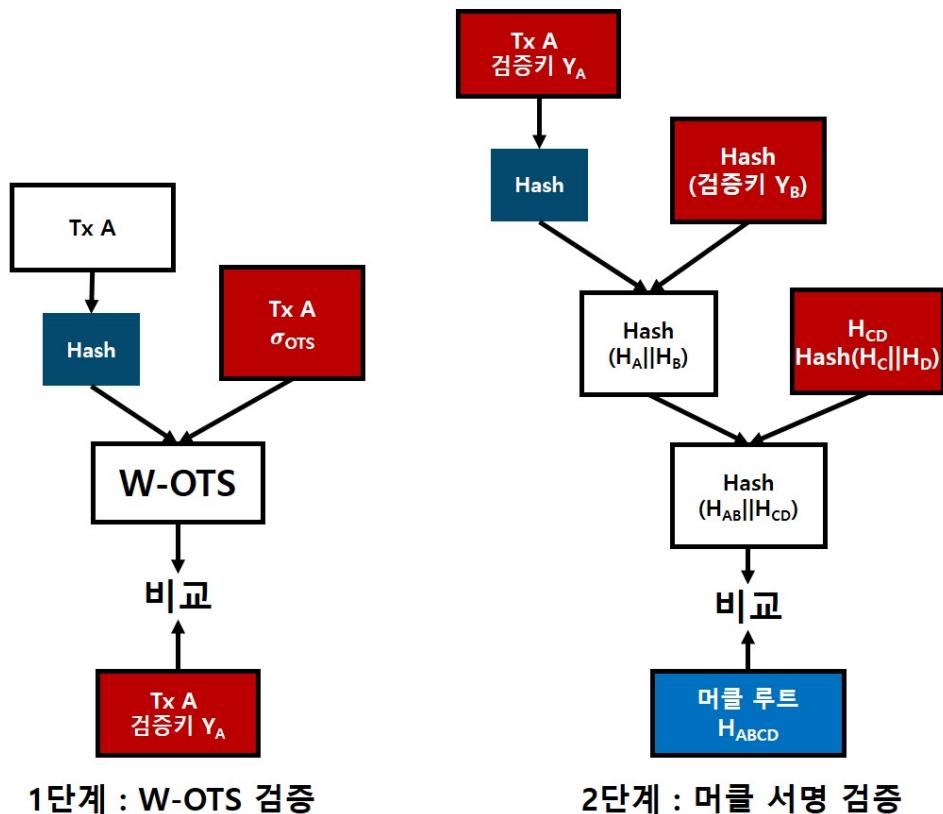


Figure 14: 제안 기법의 머클 서명 검증 과정

먼저, 머클 서명에서 σ_{OTS} 값과 검증키를 이용하여 W-OTS 검증을 수행하여, 트랜잭션에 대한 검증을 수행한다. 이후 검증키와 인증 경로 값들을 활용하여 머클 루트 값을 구하고, 블록 헤더에 저장된 머클 루트값과 비교 검증하여 검증키 검증 및 블록

전체에 대한 검증을 수행한다.

4 실험 결과

실험은 크게 3가지로, 먼저, 트랜잭션을 서명하기 위한 ECDSA, W-OTS의 키 생성 및 서명 생성 비교 실험, 두 번째는 머클 트리 생성 및 머클 서명 생성 실험, 마지막으로 ECDSA 서명 검증과 W-OTS 기반 MSS 서명 검증 비교 실험을 수행하였다.

4.1 실험 환경

본 논문에서의 실험 환경은 Windows 10 Pro OS 64-bit에서 진행되었고, Intel i7-4790 프로세서(3.60GHz), 16GB RAM을 갖춘 PC 환경에서 수행하였다. 구현 개발은 Java 버전 1.8.0_131, Eclipse Luna (4.4.2) 상에서 Java 언어를 이용하여 진행하였다. 실험 및 구현/개발 환경은 아래 표와 같다.

Operating System(OS)	Windows 10 Pro 64-bit
Processor	Intel(R) i7-4790 @3.60GHz
RAM	16GB
Java version	Java version 1.8.0_131
Development tool	Eclipse Luna (4.4.2)

Table 3: 실험 및 구현/개발 환경

위의 표에서 설명한 실험 및 구현/개발 환경에서 기존의 다양한 해시 기반 서명 기법에 대한 오픈 소스들이 존재하지만, 아직 제대로 된 기능을 제공하지 않고 있기 때문에 머클 트리 자료구조 및 머클 트리 서명 기법의 기본 구조(스켈레톤 소스 코드)를 제공하는 오픈 소스인 winternitz_merkletree[31]를 기반으로 W-OTS(Winternitz One-Time Signature) 기반의 MSS(Merkle tree Signature Scheme)를 구현하였다. 그리고 기존의 블록체인 서명 알고리즘 ECDSA secp256k1을 구현하기 위해, 오픈 소스 라이브러리인 bouncy castle 버전 1.57 라이브러리[32]를 활용하여 서명 알고리즘을 구현하였으며, 해시 함수의 경우, Java에서 제공하는 java.crypto 라이브러리의 SHA-2 해시 함수를 사용하여 구현하였다. 본 논문에서는 임의의 트랜잭션을 생성하고 그 트랜잭션에 대해

ECDSA, W-OTS 기반 MSS로 서명 및 검증 작업을 수행하였으며, W-OTS의 경우, Winternitz Parameter가 2, 3, 4인 경우에 대해 실험을 진행하였다.

4.2 실험 결과

4.2.1 트랜잭션 서명 생성

본 실험에서는 임의의 트랜잭션에 대해 ECDSA 키 생성, 서명 생성, 서명 검증 및 W-OTS 키 생성, 서명 생성, 서명 검증을 1000번 수행하고 평균을 내어 그 성능을 측정하였다. W-OTS의 경우, Winternitz Parameter 2, 3, 4의 경우에 대해 측정하였다.

실험 결과는 아래와 같다.

알고리즘		구분		
		키 쌍 생성	서명 생성	서명 검증
ECDSA		1.136	1.124	2.048
W-OTS	w=2	1.070	0.136	0.251
	w=3	1.060	0.168	0.537
	w=4	1.268	0.312	0.402

Table 4: ECDSA와 W-OTS간 서명 알고리즘 성능 비교(단위 : ms)

서명 알고리즘에 대한 키 생성 및 트랜잭션 서명 생성, 서명 검증에 대해 키 생성의 경우, ECDSA와 W-OTS의 성능 차이가 크게 나지 않지만, 나머지 성능측면에서 W-OTS가 ECDSA에 비해 훨씬 빠르다는 사실을 확인할 수 있다.

그뿐만 아니라, 기존의 ECDSA의 경우, 이산 대수 문제에 기반을 둔 공개키 암호 알고리즘으로써, 양자 알고리즘 중 쇼어 알고리즘에 대해 취약한 특성을 가지지만, W-OTS의 경우, 해시 함수의 비가역성에 의존한 전자 서명 알고리즘으로써, ECDSA와 달리 양자 알고리즘에 대한 내성이 존재한다. 보안강도의 경우, ECDSA P-256을 기준으로 128비트 보안 강도를 가지며, W-OTS의 경우, SKR(Second Key Resistance)는 약 235비트 보안 강도를, KCR(Key Collision Resistance)는 약 127비트 보안 강도를

가지며, 안전한 PRF(Pseudo Random Function)를 사용한다면 ECDSA에 뒤쳐지지 않는 보안 강도를 가지고 있다.

4.2.2 머클 트리 생성 및 머클 서명 생성

본 실험에서는 실제와 유사하게 실험하기 위해 2,048개의 트랜잭션[33]을 생성하여 해당 트랜잭션을 이용해 머클 트리를 생성하였다. 그리고 머클 트리를 생성한 다음, 추가로 머클 서명을 모든 트랜잭션에 생성하는 실험을 수행해 머클 서명을 모든 트랜잭션에 저장시키는 것이 기존 시스템에 어느 정도 영향을 미칠 수 있는 실험 하였다. 앞선 실험에서 가장 좋은 성능을 보였던 winternitz parameter가 2일 때를 기준으로 측정하였다.

구분	머클 트리 생성	머클 서명 생성
w=2	123.329	4.406

Table 5: 머클 트리 생성 및 머클 서명 생성(단위 : ms)

기존 시스템과 제안 기법 모두 머클 트리 생성은 같이 동작을 하게 된다. 하지만 머클 서명 값을 생성하여 모든 트랜잭션에 저장을 해야 한다는 것이 제안 기법의 문제점이었는데, 실험 결과 머클 트리 생성 시간보다 머클 서명 생성 시간은 매우 짧으므로, 제안 기법을 이용한 블록체인에서 머클 서명 생성을 추가한다 하더라도 성능 저하로 인한 문제는 없을 것으로 예상한다.

4.2.3 트랜잭션 서명 검증

본 실험에서는 트랜잭션에 대해 검증할 때, 임의의 트랜잭션에 대해 ECDSA 서명을 생성하고 그 서명을 검증하는데 걸리는 시간과 머클 서명에 대한 서명 검증에 걸리는 시간을 서로 비교하는 실험을 하였다.

기존 시스템과 비교하였을 때, 기존 블록체인은 트랜잭션에 대한 검증으로 ECDSA

알고리즘	구분	
ECDSA	2.048	
W-OTS	w=2	0.435
	w=3	0.537
	w=4	0.603

Table 6: 트랜잭션 서명 검증(단위 : ms)

를 이용한 검증 한 번만 하지만, 제안 기법의 경우, 1단계로 트랜잭션의 서명(W-OTS 서명)에 대한 검증을 수행하고, 2단계로 검증키와 인증 경로 값들을 이용해 머클 트리값을 구하여 다시 한번 검증을 하는 방식으로 단계적인 측면에서는 기존 블록체인이 빠를 것 같았으나, 제안 기법이 거의 4~5배 빠르게 서명에 대해 검증한다는 사실을 확인할 수 있었다. 그리고 보안성에서도 트랜잭션 서명 검증 시 W-OTS를 활용한 제안 기법은 기존의 ECDSA를 이용한 서명 검증과 유사한 보안 강도를 가지면서 양자 알고리즘에 대한 내성도 추가로 가진다. 그뿐만 아니라, 제안 기법은 트랜잭션의 검증과 더불어 검증키에 대한 검증, 트리 해시값에 대한 검증을 통해 전체 트랜잭션 무결성 검증, 그리고 대상 트랜잭션이 해당 블록에 포함되는지까지 확인할 수 있다.

정리하면, 기존 블록체인의 경우, 트랜잭션에 대해 트랜잭션 생성 시 ECDSA 서명 생성을, 트랜잭션 검증 시 ECDSA 서명 검증을 이용한다. 이때 사용되는 곡선으로는 secp256k1 곡선으로 보안강도 128비트를 갖고 있다. 이에 대해 제안 기법의 경우, 트랜잭션 생성 시 W-OTS 서명 생성을, 트랜잭션 검증 시 W-OTS 및 MSS 서명 검증을 이용하는데, 해시 함수로 SHA-256을 사용함으로써 트랜잭션별 서명 생성 및 검증에 대한 W-OTS 보안강도는 SKR은 235비트 보안강도를, KCR에서는 127비트 보안강도를 가지며 이는 기존의 블록체인과 비교했을 때 떨어지지 않는 보안 강도를 보인다. 그리고 서명 검증 시 MSS를 이용함으로써, 기존의 ECDSA 서명 검증과는 달리 추가로 머클 루트값에 대한 무결성을 검증할 수 있으며, 이를 통해 전반적인 트랜잭션에 대한 무결성과 해당 트랜잭션의 블록에 대한 포함 여부도 함께 확인할 수 있다. 그뿐만 아니

라, 앞선 실험 결과에서 확인할 수 있듯이 기존의 트랜잭션에 대한 서명 생성 및 서명 검증 시간을 단축할 수 있으며, 양자 알고리즘인 쇼어 알고리즘에 취약한 ECDSA 대신 양자 알고리즘에 내성을 가진 해시 함수 기반의 해시 기반 서명 기법(W-OTS, MSS)를 이용함으로써 보다 양자 알고리즘에 강인한 블록체인 기법이 될 수 있다.

5 결론

최근 양자 컴퓨터 기술의 발전과 양자 알고리즘의 영향으로 포스트 양자 암호 알고리즘에 대한 연구가 활발히 이루어지고 있다. 하지만, 실생활에서 적용되는 암호 알고리즘들은 아직 이러한 양자 알고리즘에 취약한 경우가 대부분이다. 특히, 거래가 이루어지는 비트코인, 이더리움과 같은 암호화 전자 화폐 시장에서조차 아직 양자 알고리즘에 취약한 ECDSA 전자 서명 알고리즘을 이용하여 트랜잭션에 대해 서명을 하고 있다.

이를 보완하고자 본 논문에서는 기존 블록체인의 트랜잭션 서명 생성 및 검증 기법으로 ECDSA 전자 서명 알고리즘 대신 해시 기반 서명 기법을 적용하여 양자 컴퓨팅 환경에 강인한 블록체인 기법을 제시하였다. 그 결과, 해시 기반 서명 기법인 W-OTS(Winternitz One-Time Signature)기반의 MSS(Merkle tree Signature Scheme)를 적용함으로써 기존 기법에 뒤처지지 않는 보안 강도를 가지면서 양자 알고리즘에 강인 한 특성을 추가로 가지게 되었고, 성능 측면에서도 트랜잭션에 대한 서명 생성 시간이 기존 생성 시간에서 약 1/6로 단축되었으며, 트랜잭션에 대한 서명 검증 시간은 머클 서명 기준으로 기존 검증 시간의 약 1/5 정도로 단축되었다. 그뿐만 아니라, 기존 시스템에서 사용하는 머클 트리를 활용한 머클 서명 기법을 통해 트랜잭션에 대해 검증할 때마다 머클 루트 값을 비교하므로 트랜잭션 전체에 대한 무결성 검사 및 트랜잭션이 해당 블록에 포함되는지 확인도 함께 진행되어 보다 안전한 블록체인 시스템이 될 수 있을 것이다.

향후 연구 방향으로는 제안된 블록체인에서 해시 함수만으로 구성된 해시 기반 서명 기법이 적용되었기 때문에 해시 함수를 SHA보다 빠른 해시 함수로 적용할 경우, 그 성능이 향상될 것으로 예상하며, 현재 W-OTS의 경우 키 생성 알고리즘과 서명 알고리즘에 병렬적으로 처리할 수 있는 부분이 존재한다. 따라서 비록 키 쌍의 크기와 서명의

크기가 크더라도 서로 독립적으로 처리할 수 있는 부분이 있으므로 병렬 처리를 통한 성능 향상을 기대할 수 있을 것이다. 혹은 현재 W-OTS 기반의 MSS를 적용하였는데, W-OTS 대신 W-OTS⁺를 적용하고, MSS 대신 XMSS(eXtended Merkletree Signature Scheme)를 적용한다면, 더욱 작은 크기의 키 쌍과 서명 크기로보다 높은 보안 강도의 서명 알고리즘을 수행할 수 있을 것이다.

References

- [1] 한국인터넷진흥원. (2016). ”국내외 블록체인 기술 적용분야 및 사례 연구”, 한국인터넷진흥원, pp. 1-87.
- [2] 박병주, 이태진, 곽진. (2017). ”블록체인 기반 IoT 디바이스 인증 스킴”, 정보보호학회논문지, 27(2), pp. 343-351.
- [3] Shor, P. W. (1994). ”Algorithms for quantum computation: Discrete logarithms and factoring. In Foundations of Computer Science”, 1994 Proceedings., 35th Annual Symposium, pp. 124-134.
- [4] Andreas M.Antonopoulos. (2015). ”비트코인, 블록체인과 금융의 혁신”, 교려대학교 출판문화원
- [5] 오서영, 이창훈. (2017). ”부동산 시장의 신뢰성 향상을 위한 블록체인 응용 기술”, 한국전자거래학회지, 22(1), pp. 51-64.
- [6] 김태성, 김우진, 이도윤, 김일곤. (2016). ”블록체인 네트워크 기반에서 FHIR를 활용한 감염병 환자 진료 정보 공유 시스템”, 한국정보과학회 학술발표논문집, pp. 2053-2055.
- [7] 박병주, 이태진, 곽진. (2017). ”블록체인 기반 IoT 디바이스 인증 스킴”, 정보보호학회논문지, 27(2), pp. 343-351.
- [8] 이부형, 임연주, 이종혁. (2017). ”블록체인 플랫폼에서의 합의 알고리즘. 한국통신학회 학술대회논문집”, pp. 386-387.
- [9] 김소희, 양지연, 김윤정. (2015). ”비트코인 블록체인의 이기적인 채굴 방안 분석” 한국통신학회 학술대회논문집, pp. 422-423.

- [10] IBM. (2017). "IBM Blumix Blockchain-z7", <https://console.ng.bluemix.net/catalog/services/blockchain>. (2017-05-24 방문)
- [11] Grover, L. K. (1996). "A fast quantum mechanical algorithm for database search", In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pp. 212-219.
- [12] 한국인터넷진흥원. (2016). "양자컴퓨팅 환경을 고려한 현대암호 안전성 연구", 한국인터넷진흥원, pp.1-297.
- [13] IBM, (2017). "IBM Q experience", <https://quantumexperience.ng.bluemix.net/qx/user-guide>, (2017-05-24 방문)
- [14] National Institute of Standards and Technology. (2012). "Secure Hash Standard", FIPS PUB 1804
- [15] Umana, V. G. (2011). "Post-Quantum Cryptography", Post-Quantum Cryptography.
- [16] Bernstein, D. J., Buchmann, J., Dahmen, E. (Eds.). (2009). "Post-quantum cryptography" Springer Science and Business Media.
- [17] Hülsing, A., Gazdag, S. L., Butin, D., Buchmann, J. "Hash-based Signatures: An Outline for a New Standard", ISO 690
- [18] Lamport, L. (1979). "Constructing digital signatures from a one-way function", (Vol. 238). Palo Alto: Technical Report CSL-98, SRI International.

- [19] Billet, O., Robshaw, M. J., Peyrin, T. (2007, July). "On building hash functions from multivariate quadratic equations" In Australasian Conference on Information Security and Privacy pp. 82-95.
- [20] Buchmann, J., Dahmen, E., Ereth, S., Hülsing, A., Rückert, M. (2011). "On the security of the Winternitz one-time signature scheme" In International Conference on Cryptology in Africa pp. 363-378.
- [21] Merkle, R. C., Charles, R. (1979). "Secrecy, authentication, and public key systems"
- [22] Becker, G. (2008). "Merkle signature schemes, merkle trees and their cryptanalysis" Ruhr-University Bochum, Tech. Rep.
- [23] 박태환, 배봉진, 김호원. (2016). "해시 기반 서명 기법 최신 기술 동향 및 전망" 정보보호학회논문지, 26(6), pp. 1413-1419.
- [24] 박태환, 배봉진, 이가람, 김호원. (2016). "Stateful 해시 기반 서명 기법 동향 및 전망" 한국정보과학회 학술발표논문집, pp. 1121-1123.
- [25] Johannes Buchmann. (2006). "CMSS – An Improved Merkle Signature Scheme", Progress in Cryptology – INDOCRYPT, p.349-363.
- [26] Ana Karina D.S. de Oliveira. (2015). "An Efficient Software Implementation of the Hash-Based Signature Scheme MSS and Its Variants", LATINCRYPT 2015, pp.366-383.

- [27] Bernstein, Daniel J., et al. (2015). "SPHINCS: practical stateless hash-based signatures." Advances in Cryptology-EUROCRYPT 2015. Springer Berlin Heidelberg, pp. 368-397.
- [28] Sebastian Rohde. (2008). "Fast Hash-Based Signatures on Constrained Devices", CARDIS 2008, pp.104-117.
- [29] Thomas Eisenbarth. (2013). "A Performance Boost for Hash-based Signatures", LNCS 8260, pp.166-182.
- [30] Hülsing, Andreas, Joost Rijneveld, and Peter Schwabe. (2016). "ARMed SPHINCS Computing a 41KB signature in 16KB of RAM."
- [31] Cassius Pudozius. (2017). "winternitz merkletree", https://github.com/puodzicus/winternitz_merkletrie (2017-05-24 방문)
- [32] Bouncy Castle. (2017). "Bouncy Castle 1.57", https://www.bouncycastle.org/latest_releases.html (2017-05-24 방문)
- [33] BLOCKCHAIN. (2017). "Average Number Of Transactions Per Block", <https://blockchain.info/ko/charts/n-transactions-per-block?showDataPoints=true> (2017-05-24 방문)

A 약어

AP : AuthenticationPath

DLP : DiscreteLogarithmProblem

DPoS : DelegatedProofofStake

ECC : EllipticCurveDigitalCryptography

ECDSA : EllipticCurveDigitalSignatureAlgorithm

GCD : GreatestCommonDivisor

HBS : Hash – basedSignatureScheme

HSP : HiddenSubgroupProblem

HORS : HashtoObtainRandomSubset

HORSE : HashtoObtainRandomSubsetExtended

IoT : InternettofThings

KCR : KeyCollisionResistance

LD – OTS : Lamport – DiffieOne – TimeSignature

MSS : MerkletreeSignatureScheme

NIST : NationalInstituteofStandardsandTechnology

OTS : One – TimeSignature

PaaS : PlatformAsAService

PoW : ProofofWork

PoS : ProofofStake

PQC : PostQuantumCryptography

QFT : QuantumFourierTransformation

SHA : SecureHashAlgorithm

SKR : SecondKeyResistance

TPP : TrustedThirdParty

UTXO : UnspentTransactionOutputs

W – OTS : WinternitzOne – TimeSignature

XMSS : eXtendedMerkleTreeSignatureScheme

B 초록

In these days, the security level of existing cryptographic algorithms is needed to be increased by the rapid development of Quantum computer and Quantum algorithms such as Shor's algorithm and Grover's algorithm. Some of existing cryptographic algorithms have vulnerabilities against Quantum algorithm. Especially, Blockchain technologies such as Bitcoin and ethereum on Cryptocurrency market use the ECDSA Digital Signature Algorithm which has vulnerabilities against Quantum algorithm. In this paper, applying the hash-based signature scheme which is one of post-quantum cryptography on the existing blockchain. The proposed method is using W-OTS(Winternitz One-Time Signature) on generating and verifying the signature of transaction and MSS(Merkle tree Signature Scheme) to Merkle tree architecture on the blockchain. In the case of generating the signature of transaction based on the proposed method decreases time from $1/3$ to $1/9$, the proposed verification method on the signature of transaction decreases time about $1/5$ and it can verify not only the merkle signature, but also the integrity of all transactions and check out whether block includes the verified transaction.