프로그래밍 bit coin

밑바닥부터 시작하는 비트코인

파이썬으로 직접 구현 하며 배우는 비트코인

<송재준>

Programming Bitcoin by Jimmy Song. Copyright 2019 Jimmy Song, 978-1-03149-9

11/25 월 ~ 12/10 화 (1회독 100%)

※ 주피터노트북 첨부파일 확인

㈜주 가 드

김재원

**목차**

[**목차 2**](#_Toc184650747)

[**1. 유한체 4**](#_Toc184650748)

[**2. 타원 곡선 7**](#_Toc184650749)

[**3. 타원곡선 암호 8**](#_Toc184650750)

[**4. 직렬화 11**](#_Toc184650751)

[**5. 트랜잭션 13**](#_Toc184650752)

[**6. 스크립트 15**](#_Toc184650753)

[**7. 트랜잭션 검증과 생성 18**](#_Toc184650754)

[**8. P2sh 스크립트 20**](#_Toc184650755)

[**9. 블록 22**](#_Toc184650756)

[**10. 네트워크 25**](#_Toc184650757)

[**11. 단순 지급 검증 27**](#_Toc184650758)

[**12. 블룸 필터 30**](#_Toc184650759)

[**13. 세그윗 32**](#_Toc184650760)

[**14. 고급 주제 및 다음 단계 34**](#_Toc184650761)

## 1. 유한체

타원 곡선 암호를 이해하기 위한 기본 수학 개념.

타원 곡선 암호는 트랜잭션 작동 방법의 핵심 알고리즘이며 전자 서명과 검증에 사용됨.

트랜잭션은 비트코인에서 더 이상 쪼갤 수 없는 이동의 단위

* 1. 현대 대수 학습

슈노어 서명, 기밀 트랜잭션, 첨단 비트코인 기술에 대해서 이해할 예정

* 1. 유한체 정의

유한체란 아래의 성질을 만족하는 2개의 연산자를 가진 집합.

이 집합의 원소 수가 유한하다는 특징 가짐

1. a와 b가 집합에 속해 있으면 a+b, a\*b도 집합 안에 있다.

2. 집합에 0으로 표기하는 원소가 있고 집합 내 다른 원소 a와 + 결과는 a다.

3. 집합에 1로 표기하는 원소가 존재하고 집합 내 다른 원소 a와 \* 결과는 a다.

4. 집합에 a와 +연산결과가 0이 되게하는 b가 있고 -a로 표기한다.

5. 0이 아닌 a에 대해 a\*b=1이 되게하는 b가 존재하고 a^-1로 표기한다.

위수(p) : 유한한 집합 크기

* 1. 유한 집합 정의하기

위수 11의 유한체

F11={0,1,2,3,4,5,6,7,8,9,10}

위수 17의 유한체

F17={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}

위수 983의 유한체

F983={0,1,2,…982}

특징

1. 위수는 항상 가장 큰 숫자 원소보다 하나 더 많다.

2. 유한체는 매번 위수가 소수이다.

3. 유한체는 반드시 소수이거나 소수의 거듭제곱을 위수로 가져야 한다.

1.3.1. 파이썬으로 유한체 코딩하기

# 코드 참조

* 1. 나머지 연산

1.4.1. 파이썬으로 나머지연산 코딩하기

# 코드 참조

* 1. 유한체 덧셈과 뺄셈

덧셈에 대해 닫혀 있다는 뜻의 정의는 다음과 같다.

a+fb∈F19, 인데 이 식은

a+fb=(a+b)%19 라는 나머지 연산을 사용해 정의가 가능하다.

예를 들면

7+f8=(7+8)%19 = 15

이를 일반화 하면,

a+fb=(a+b)%p이고 역원도 이런 식으로 정의한다.

-fa=(-a)%p

1.5.1. 파이썬으로 유한체 덧셈과 뺄셈 코딩하기

# 코드 참조

* 1. 유한체 곱셈과 거듭제곱

F19에서

5\*f3=5+ f5+ f5=15

1.6.1. 파이썬으로 곱셈 코딩하기

# 코드 참조

* 1. 유한체 나눗셈

<<페르마의 소정리>>

{1,2,3,…,p-2,p-1}={n%p,2n%p,…(p-2)%p,(p-1)%p}

1=n^(p-1) % p

## 2. 타원 곡선

2.1. 정의

타원곡선을 배워서 3장에서 타원곡선 암호를 실습한다.

y^2=x^3+ax+b

구체적으로 비트코인에서 사용되는 타원곡선은 secp256k1이며 다음과 같은 방정식이다.

Y^2=x^3+7

일반 정규식은 y^2=x^3+ax+b이다.

2.2. 파이썬으로 타원곡선 코딩하기

2.3. 두 점의 덧셈

라인, 평행, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

수학적 용어로서의 점 덧셈 = 비선형 연산

2.4. 점 덧셈 성질

1. 항등원 존재

2. 교환법칙 성힙

3. 결합법칙 성립

4. 역원 존재

I+A=A가 되는 I를 무한 원점이라고 부른다.

2.4. 점 덧셈 코딩

## 3. 타원곡선 암호

메시지 서명과 검증에 필요한 기본 알고리즘을 이야기하는데 서명과 검증은 비트코인 동작에서 중추에 해당한다.

3.1. 실수체에서 정의된 타원곡선

실수체와 유한체는 다음과 같은 동일한 성질이 있다.

1. a 와 b가 집합의 원소이면 a+b와 a\*b도 집합의 원소이다.

2. 0이라는 원소가 존재하고 그 원소는 a+0=a 성질을 만족한다.

3. 1이라는 원소가 존재하고 그원소는 a\*1=a라는 성질을 만족한다.

4. a가 어떤 집합의 원소이면 -a라는 원소또한 그 집합에 존재하며 그 원소는 a+(-a)=0의 성질을 만족한다.

5. 0이 아닌 a가 집합의 원소이면 a^-1이라는 원소또한 집합에 존재하며 그 원소는 a\*a^-1=1이라는 성질을 만족한다.

이 성질들로 재밌는 점이 하나 있다. 함수의 변수와 계수들이 어떤 체에서 정의되었는지 상관없이 타원곡선에서 점 덧셈의 성질이 성립한다.

3.2. 유한체에서 정의된 타원 곡선 코딩하기

3.3 유한체에서 정의된 타원 곡선 위 두 점의 덧셈

유한체 방정식에서 두 점의 덧셈은 가능하다.

3.4. 유한체에서 정의된 점 덧셈 코딩하기

3.5. 타원 곡선 위 점의 스칼라 덧셈

(170,142)+(170,142)=2\*(170,142)

이것이 바로 스칼라 곱셈이다.

반면 스칼라 나눗셈은 계산하기 쉽지 않은데 이를 이산 로그 문제라고 하며 이 성질이 타원곡선 암호 방법의 원리이다.

스칼라 곱셈의 성질은 스칼라 값을 계속 증가시키다보면 무한 원점에 도달한다는 것이다.

무한 원점에 도달하면 다음과 같은 군을 얻을 수 있는데 n은 유한한 특정 값이므로 유한군이라고 한다.

{G,2G,3G,…nG} nG=0

3.6. 스칼라 곱셈 특징

공개키 암호 기법에 스칼라 곱셈이 활용되는 이유는 타원곡선에서 스칼라 곱셈 연산이 어렵기 때문이다.

3.7. 스칼라 곱셈으로 생성된 군의 성질

실제 공개키 암호에 쓰이는 것은 유한순환군이다.

유한체에서 정의된 타원곡선 위 한 점에 스칼라 값을 곱해서 유한순환군을 설명한다.

이 점을 생성점이라 부른다.

3.7.1. 항등원의 존재

0+A=A

에서 항등원을 0으로 하고 이를 무한원점이라고 한다.

3.7.2. 닫혀 있음

3.7.3. 역원의 존재

3.7.4. 교환법칙

3.7.5. 결합법칙

3.8. 스칼라 곱셈 코딩하기

3.9. 비트코인에서 사용하는 타원 곡선

실전에서는 작은 소수를 사용하지 않는다. 컴퓨터로 탐색할 수 있기 때문이다.

이 비트코인에서 사용하는 타원 곡선의 특징은 다음과 같다.

1. 방정식이 상대적으로 간단하다. 매우 큰 a와 b를 가지고 있다.

2. 유한체의 위수인 p가 2^256에 매우 가깝다.

3.10. 공개키 암호

3.11. 서명 생성과 서명 검증

3.11.1. 디지털 서명 알고리즘

목표물의 위치를 화살촉에 새기는 방법

3.11.2. 서명 검증

32바이트 길이의 서명해시를 자신이 보장한다는 증명서와 같다.

## 4. 직렬화

클래스의 객체들을 네트워크로 보내거나 영구 저장장치에 저장할 때 직렬화가 필요하다.

4.1. 비압축 SEC(Standards for Efficient Cryptography) 형식

ECDSA 공개키(타원곡선암호)를 직렬화하는 표준안이 존재하는데 이를 SEC형식(Standards for Efficient Cryptography)이라 한다.

SEC(Standards for Efficient Cryptography)는 아주 작은 오버헤드만 요구한다. 500을 빅엔디언으로 표기하면 01f4라 하지만, 리틀엔디언으로 표기하면 f401이다.

4.2. 압축 SEC(Standards for Efficient Cryptography) 형식

x좌표를 갖는 근은 2개 y,-y이라는 사실을 알 수 있다. 이 사실을 이용한 것이 압축 sec이다.

33바이트만 차지하므로 공간 절약이 된다.

4.3. DER 서명 형식

1. 0x30바이트로 시작한다.

2. 서명의 길이를 붙인다. 보통 0x44나 0x45가 된다.

3. r 값의 시작을 표시하는 표식 바이트로 0x02를 붙인다.

4. 빅엔디언 정수로 r값을 표현한다. 결과의 첫 번째 바이트가 0x80보다 크거나 같으면 00을 앞에 붙인다.

5. s값의 시작을 표시하는 표식 바이트로 0x02를 붙인다.

6. 빅엔디언 정수로 s값을 표현한다.

4.4. 비트코인 주소 및 WIF 형식

4.4.1. Base58 부호화

맨 마지막에 체크섬을 붙이면 실수 탐지 기능을 더할 수 있다 이를 Base58부호화라고 한다.

지불 방법의 3가지 고려사항

1. 공개키의 가독성

2. 공개키의 길이

3. 보안성

4.4.2. 비트코인 주소 형식

메인넷 주소는 0x00으로 시작하고, 테스트넷 주소는 0x6f로 시작한다.

연속으로 2개의 해시함수를 적용하는 방법을 hash160이라고 한다.

Hash256의 첫 4바이트를 체크섬이라고 한다.

4.4.3. 비밀키의 WIF(wallet import format) 형식

비트코인의 비밀키는 256비트로 표현되는 숫자이다.

종이지갑에서 소프트웨어 지갑으로 이동시킬 때 비밀키를 이동하게 되는데 WIF형식을 사용한다.

아래는 비밀키를 WIF형식(wallet import format) 으로 만드는 방법이다.

1. 메인넷 비밀키는 0x80으로 시작하고 테스트넷 비밀키는 0xef으로 시작한다.

2. 비밀키를 32바이트의 빅엔디언으로 표현한다.

3. 만약 공개키를 압축 SEC로 표현하면 2번 결과 뒤에 0x01을 추가한다.

4. 1번의 접두 바이트와 2번의 빅엔디언의 비밀키, 3번의 접미 바이트를 연결한다.

5. 4번 결과를 hash256으로 해시하고 그 결과에서 첫 4바이트를 체크섬으로 취한다.

6. 4번 결과 뒤에 5번 절차에서 구한 체크섬을 붙이고 Base58로 부호화한다.

4.5. 비트코인에서 빅엔디언/리틀엔디언 변환

어디에서 리틀엔디언/빅엔디언을 사용하는지에 대한 규칙은 없다.

5장 이후로 리틀엔디언을 많이 사용한다.

## 5. 트랜잭션

5.1 트랜잭션 구성 요소

트랜잭션은 한 엔터티에서 다른 엔터티로의 가치 이동을 표현한다.

엔터티 : 스마트 계약

트랜잭션은 네 가지 요소로 구성된다.

1. 버전 : 트랜 잭션의 버전을 의미

2. 입력 : 사용할 비트코인을 정의

3. 출력 : 비트코인이 어디로 가는 지 종착지를 정의

4. 록타임 : 트랜잭션의 유효시점을 규정

5.2. 버전

비트코인의 입력은 이전 트랜잭션의 출력을 가리킨다.

증명 부분은 타원곡선 서명 알고리즘을 이용한다.

varints라는 형식으로 입력 개수를 표현한다.

Varints는 가변 정수의 약자로 0에서 2^64 - 1사이의 정숫값을 바이트로 표현하는 방법이다.

5.3. 입력

5.3.1. 스크립트 파싱

5.4. 출력

받는 사람이 여러 명일 때 한 명 한 명에게 여러 번 보내기보다 한 번의 트랜잭션으로 모든 사람에게 보낼 수 있다.

출력은 2개의 하부 필드로 구성되어 있다. 비트코인 금액과 잠금 스크립트.

5.5. 록타임

록타임은 트랜잭션 전파 후 실행을 지연시키는 방법을 제공한다.

빈번한 거래상황을 위해 고안되었다.

록타임의 주요 문제는 록타임에 도달했을 때 트랜잭션의 수신자가 트랜잭션이 유효한지 확신할 수 없다는 점이다.

5.6. 트랜잭션 직렬화 코딩하기

5.7. 트랜잭션 수수료

트랜잭션의 수수료가 0이라면 채굴자가 트랜잭션을 블록에 포함시킬 동기가 사라진다.

수수료는 채굴자에게 주는 인센티브이다.

5.7.1. 수수료 계산

## 6. 스크립트

비트코인을 잠근다 = 누군가에게 비트코인을 준다

해제한다 = 내가 받은 비트코인을 소비한다.

잠금/해제 메커니즘을 다른 말로 “스마트 계약”이라고 한다.

스크립트는 타원곡선 암호 알고리즘을 사용해 트랜잭션의 유효성을 검증한다.

6.1. 스크립트 기법

스마트 계약 : 블록체인 상 코인의 전송을 프로그램으로 기술한다.

스크립트 : 비트코인의 스마트 계약 언어

비트코인은 튜링 완전하지 않다.(어떤 계산 문제라도 적절한 알고리즘과 자원을 사용하면 해결이 가능하지 않다)

6.2. 스크립트 실행

스크립트는 프로그래밍 언어이다. 스택을 기반으로 명령어를 한번에 하나씩 처리한다.

명령어에는 두 종류가 있다. 하나는 원소이고 다른 하나는 연산자이다.

6.3. 주요 연산자

OP\_DUP는 간단히 스택 n이의 동일한 원소를 올린다.

6.4. 스크립트 파싱

스크립트 파싱: 스크립트 코드를 바이트 단위로 읽고 해석해, 명령어와 데이터 토큰을 구분하고 실행 가능한 형태로 변환하는 과정.

잠금 스트립트와 해제 스크립트 모두 같은 방식으로 파싱된다. 파싱 시작 후 처음 읽은 한 바이트 값이 n이고 이 값이 0x01~0x4b범위의 값이면 n바이트 길이만큼 이어서 읽은 숫자를 한 원소로 간주한다. 그렇지 않으면 그 바이트 값은 opcode를 의미한다.

6.5. 잠금/해제 스크립트의 결합

스크립트 객체는 실행 명령어 집합이다. 스크립트를 실행하기 위해서는 잠금 스크립트와 해제 스크립트를 결합해야 한다. 금고와 열쇠는 서로 다른 트랜잭션이다.

해제 스크립트는 잠금 스크립트로 잠긴 코인을 해제하기 때문에 2개의 스크립트를 하나로 만드는 방안이 필요하다.

6.5.1. 결합 스크립트 코딩하기

6.6. 표준 스크립트

비트 코인에는 많은 종류의 표준 스크립트가 있다. 비트 코인 주소도 일종의 스크립트 템플릿이다.

6.7. p2pk 스크립트

사토시가 소유한 것으로 보이는 대부분의 비트코인은 p2pk UTXO로 존재한다.

메시지 : z,

공개키 : P,

서명 : r, s

6.7.2 스택 원소의 저장 형식

스택의 원소는 항상 같은 형태의 값이 아니라 어떤 때는 int형 정수, 어떤 때는 DER 서명, SEC 공개키 같은 bytes 배열을 쓴다.

6.8. p2pk 스크립트의 문제점

P2pk 스크립트는 누구나 비트코인을 공개키로 보내고 비밀키 소유자만 생성할 수 있는 서명으로 구성되어 있어 직관적으로 이해가 된다.

문제 1. 공개키 길이가 너무 길다. 압축은 33바이트, 비압축은 65바이트다. 문자 인코딩에서 제어 문자처럼 화면으로 표시하지 못하는 문자가 있다.

문제 2. 공개키의 긴 길이가 미묘한 문제를 일으킨다. 풀 노드에 부담을 준다.

문제 3. 공개키는 누구나 볼 수 있는 잠금 스크립트 안에 있어 해킹의 위험이 있다.

6.9. p2pk 스크립트의 문제 해결

P2pkh는 두 가지 장점이 있다.

1. 짧은 주소를 사용한다.

2. sha 256과 ripemd 160 해시함수에 의한 추가적인 보호 장치 추가

6.9.1 p2pkh 스크립트

6.10. 비표준 스크립트

6.10.1. 연산자의 유용성

연산함수는 비트코인 네트워크에서 더 이상 사용되지 않는다. 오피코드의 사용은 네트워크가 취약하게 만든다.

6.10.2. 해시충돌 발견 상금 이벤트

## 7. 트랜잭션 검증과 생성

7.1. 트랜잭션 검증

트랜잭션 검증이란 트랜잭션을 수신한 후 모든 노드에서 트랜잭션이 네트워크 규칙에 부합하는지 확인하는 과정.

7.1.1. 입력 비트코인 존재 확인

이중 지불을 방지하기 위해 노드는 각 입력이 가리키는 비트코인이 존재하고 사용 가능한지 확인한다.

비트코인의 입력이 가리키는 UTXO (Unspent Transaction Output)를 추적하면 이중 지불 여부를 확인할 수 있다.

7.1.2. 입력과 출력 비트코인 합계 확인

노드는 입력 비트코인의 합이 출력 비트코인의 합보다 크거나 같은지 확인한다.

5장에서 배운 수수료 계산 메서드를 통해 마이너스 수수료가 아닌지 확인합니다.

7.1.3. 서명 확인

가장 까다로운 부분 중 하나이다. 다중 서명된 UTXO (Unspent Transaction Output)를 소비하면 서명이 하나 이상이다.

트랜잭션을 단계별로 변형해야 한다.

1단계 : 모든 해제 스크립트를 비운다.

2단계 : 삭제된 해제 스크립트 자리에 사용할 UTXO (Unspent Transaction Output)의 잠금 스크립트를 삽입한다.

3단계 : 해시 유형을 덧붙인다.

7.1.4. 전체 트랜잭션 검증

7.2. 트랜잭션 생성

존재하고 사용가능한 UTXO (Unspent Transaction Output)가 필요하다.

7.2.1. 트랜잭션 설계

트랜잭션을 설계하기 위해서는 다음 기본 질문에 답할 수 있어야 한다.

1. 비트코인을 어느 주소로 보내고자 하는가?

2. 어느 UTXO (Unspent Transaction Output)를 사용할 수 있는가?

3. 얼마나 빨리 트랜잭션이 처리되기를 원하는가?

7.2.2. 트랜잭션 구성

7.2.3. 트랜잭션 해제 스크립트 생성

잠금 스크립트 안에 해시된 공개키에 대응하는 비밀키를 갖고 있다면 z에 대한 DER 형식의 서명을 생성할 수 있다.

7.3. 테스트넷 트랜잭션 생성 및 전파

## 8. P2sh 스크립트

8.1. 다중서명

비밀키 하나로 트랜잭션을 생성하면 위험하다.

예를 들어 1억 달러어치 비트코인을 보유한 회사는 이 비밀키를 분실하거나 도난당하면 모든 자금을 잃어버리는데 이 비밀키를 “단일 실패 지점”이라 한다.

“다중서명”방식으로 이 문제를 해결할 수 있다.

8.2. OP\_CHECKMULTISIG 함수

M of n 다중서명에서 스텍에는 n이 제일 위에 있고 다음 n개의 공개키, 다음 m개의 서명, 마지막으로 off by one 버그를 위한 원소 1개가 있다.

8.3. 다중서명의 문제점

1. 서로 다른 많은 공개키를 가지고 있으므로 잠금 스크립트가 매우 길어진다.

2. p2pkh보다 5~20배 더 긴 출력이 된다.

3. 크기가 매우 크므로 이를 악용할 수 있다.

8.4. p2sh(pay to script hash) 스크립트

P2sh(pay to script hash) 스크립트는 긴 주소와 긴 잠금 스크립트 문제를 해결하는 방법이다. 긴 스크립트 대신 스크립트의 짧은 해시값을 제시하고 원래의 긴 스크립트는 해제 시 드러난다.

2 of 2 다중서명 잠금 스크립트를 리딤 스크립트라고 부른다. 리딤 스크립트라고 부르는 이유는 스크립트 해시값이 아닌 원래 스크립트를 필요할 때 가져오기 때문입니다.

해시값은 리딤 스크립트의 hash160 해시 값이고 리딤 스크립트는 잠금 스크립트와 같다.리딤 스크립트는 대응하는 p2sh 주소를 만들 때 함께 만들어져 사용자가 보관한다. 만약 리딤 스크립트를 분실한다면 비트코인을 읽어버리게 되는 것이다.

스크립트 그 자체가 아닌 리딤 스크립트의 해시값을 블록체인에 넣고 있으면 이러한 잠금 스크립트를 pay to script hash라고 부른다.

8.5. p2sh(pay to script hash) 스크립트 코딩하기

8.5.1. 다중서명 이외의 p2sh(pay to script hash)

P2sh(pay to script hash) 스크립트 장점은 리딤 스크립트 길이가 최대 520바이트까지 가능하다.

8.5.2. p2sh(pay to script hash) 주소

P2sh(pay to script hash) 스크립트에서 사용하는 주소의 계산 방법은 p2pkh(Pay-to-PubkeyHash) 주소를 구하는 방법과 유사하다. Hash160 해시값에 접두부 바이트를 붙이고 체크섬을 끝에 덧붙인다.

8.5.3. p2sh 서명 검증

P2sh 스크립트에서 까다로운 부분 중 하나는 서명을 검증하는 부분이다.

P2pkh의 경우 1개의 공개키로 1개의 서명을 검증하지만 여기서는 복수의 공개키와 그보다 같거나 적은 서명을 검증한다.

P2pkh 서명 검증 과정은 다음과 같다.

1 단계 : 모든 해제 스크립트를 지운다.

2 단계 : 삭제된 해제 스크립트 자리에 리딤 스크립트를 삽입한다.

3 단계 : 해시유형을 덧붙인다.

## 9. 블록

비트코인의 혁신은 블록을 통한 이중 지불 문제를 해결한 것이다. 블록은 트랜잭션 순서를 정하는 방법이라고 할 수 있다. 트랜잭션 순서 중 첫 번째 것만 유효하고 나머지는 무효로 간주한다.

이를 위해서 모든 노드가 모든 트랜잭션의 순서에 합의해야하고 노드 사이에 많은 통신 비용이 발생한다.

비트코인은 비용의 절충안으로 트랜잭션을 10분마다 정산한다. 이러한 트랜잭션의 묶음을 블록이라고 한다. 코인베이스 트랜잭션은 모든 블록의 첫 번째 트랜잭션을 의미한다.

9.1. 코인베이스 트랜잭션

트랜잭션의 출력에서는 보통 p2pkh 잠금 스크립트로 채굴자가 지정한 주소에 블록 보상으로 주어지는 비트코인과 블록 내 모든 트랜잭션 수수료를 잠가 놓는다.

트랜잭션 구조는 3 가지 특이 사항을 제외하고 다른 트랜잭션과 별반 다르지 않다.

1. 코인베이스 트랜잭션은 입력이 정확히 하나이다.

2. 하나의 입력은 이전 트랜잭션 해시값으로 32바이트의 0을 갖는다.

3. 하나의 입력은 이전 트랜잭션의 출력 번호로 ffffffff를 갖는다.

9.1.1. 해제 스크립트

코인베이스 트랜잭션은 입력에 이전 트랜잭션 출력이 없다. 입력의 해제 스크립트가 의미가 없으므로, 그 자리에는 무엇이 있을까?

채굴자가 정한다. 크기가 2바이트에서 100바이트를 넘지 않아야 한다.

9.1.2. BIP0034 제안서

코인 베이스 트랜잭션의 해제 스크립트의 첫 번째 원소를 규정하고 있다. 서로 다른 블록이여도 같은 코인베이스 트랜잭션id를 갖는 문제를 해결하고자 했다.

9.2. 블록 헤더

블록 헤더는 블록에 포함된 트랜잭션에 대한 메타 데이터이다.

블록 헤더는 다음과 같이 구성된다.

1. 블록 버전

2. 이전 블록 해시값

3. 머클루트

4. 타임스탬프

5. 비트값

6. 논스값

트랜잭션과 달리 블록 헤더의 필드는 고정 길이이다. 80바이트이다.

9.2.1. 블록 버전

블록 버전은 블록을 생성하는 비트코인 소프트웨어 기능의 집합을 나타내고 초기에 채굴자에게 배포할 준비가 된 하나의 기능을 가리킨다.

블록 버전2는 소프트웨어가 BIP0034(Bitcoin Improvement Proposal)에 대한 준비가 되었음을 의미한다.

블록 버전3는 코인 베이스 트랜잭션에 블록 높이를 넣는 제안서이다.

블록버전 4는 소프트웨어가 BIP0065(Bitcoin Improvement Proposal)에 대한 준비가 되었음을 의미한다.

BIP0009(Bitcoin Improvement Proposal)는 한번에 서로 다른 기능의 준비 상황이 29개까지 표시되어 전파되는 방법이다.

9.2.2. 이전 블록 해시값

모든 블록은 바로 이전 블록의 해시값을 가지고 있다. 이 해시 값으로 이전 블록을 특정할 수 있는데 이러한 데이터 구조를 블록체인이라고 한다.

9.2.3. 머클루트

머클루트는 블록 내 순서에 따라 나열된 모든 트랜잭션을 32바이트 해시값으로 변환한 값이다.

9.2.4. 타임스탬프

타임스탬프는 유닉스 형식으로 표현된 4바이트 값이다.

9.2.5. 비트값

비트값은 블록에서 필요한 작업증명과 관련된 필드이다.

9.2.6. 논스값

Number used only ONCE의 줄임말로 작업증명을 위해 채굴자가 변경하는 값이다.

9.3. 작업증명

작업증명으로 탈중앙화 방식의 비트코인 채굴이 가능해지고 비트코인 보안이 유지된다.

따라서 작업증명을 ‘채굴’이라고 한다.

작업증명을 찾는다는 것은 특정한 조건을 만족하는 매우 희소한 숫자를 찾는 것이다.

9.3.1. 채굴자의 해시값 생성 방법

채굴의 재료가 되는 숫자 얻는 곳?

블록 헤더 필드 중 하나인 논스값에서 얻는다.

채굴자는 작업증명을 위해 논스값 외에 코인베이스 트랜잭션을 변경할 수 있다.

9.3.2. 목푯값

작업증명의 요구사항에 따라 비트코인의 모든 블록 헤더의 해시값이 특정 값보다 작아야 한다.

9.3.3. 난이도

서로 다른 목푯값을 쉽게 비교할 수 있도록 난이도라는 개념이 나왔다.

9.3.4. 작업증명 유효성 확인

블록 헤더의 hash256 해시값을 계산하고 이것을 리틀엔디언 정수로 읽어서 작업증명을 계산하는 것을 배웠다.

9.3.5. 난이도 조정

비트코인에서 2016블록의 각 가룹을 난이도 조정 기간이라고 한다.

## 10. 네트워킹

처음 4바이트는 항상 동일하며 네트워크 매직 바이트라고 한다.

P2P네트워크는 탈중앙화 특성상 매우 견고하다.

비트코인 네트워크는 브로드캐스트 네트워크라고 한다. 모든 노드는 자신이 아는 트랜잭션, 블록, 인접 노드 리스트를 전파한다.

10.1. 네트워크 메시지

비동기 네트워크 통신 프로그래밍에서 일반적으로 매직 바이트를 사용한다.

10.2. 페이로드 파싱

각 커맨드마다 서로 다른 페이로드를 구성한다.

첫 번째 필드는 네트워크 프로토콜 버전 정보를 갖고 있다.

다음 필드는 IP 주소로 IPv6, IPv4 또는 OnionCat으로 표현한다.

마지막 4바이트가 IP주소이다.

메인넷의 디폴트 포트번호는 8333이고 빅엔디언으로 쓰여 0x208d이다.

논스는 노드가 어떤 노드와 연결되었을 때 의미 없는 자신과의 연결을 식별하기 위해 사용한다.

10.3. 네트워크 핸드셰이크

네트워크 핸드셰이크는 노드 사이에 통신 링크를 처음 수립하는 과정이다. 방법은 다음과 같다.

1. A는 B와 연결하기 위해 version 메시지를 보낸다.

2. B는 version 메시지를 받고 verack 메시지로 응답한다. 그리고 바로 자신의 version 메시지를 보낸다.

3. A는 B가 보낸 verack와 version 메시지를 받고 verack 메시지로 응답한다.

4. B는 verack 메시지를 받고 이제 통신 링크가 형성되었다.

10.4. 네트워크 접속

네트워크 통신은 비동기 특성으로 인해 다루기 어렵다. 실험할 때는 네트워크 노드에 동기식으로 접속할 수 있다.

10.5. 블록 헤더 요청

어떤 노드가 처음 네트워크에 연결되고 입수해야 하는 가장 중요한 데이터는 블록 헤더이다.

풀노드의 경우 여러 노드에 비동기적으로 병렬 요청하여 전체 블록을 빠르게 내려받는다.

라이트 노드는 블록 헤더를 내려받아 각 블록의 작업증명이 올바른지 검정한다.

Version 메시지의 페이로드와 마찬가지로 네트워크 프로토콜 버전 정보로 시작하여 블록 헤더, 블록 헤더 그룹갯수 등으로 구성되어 있다.

10.6. 블록 헤더 응답

노드를 생성하고 핸드셰이크로 노드와 연결하여 블록 헤더까지 요청할 수 있게 되었다.

## 11. 단순 지급 검증

머클루트가 무엇인지 살핀다. 이는 특정 트랜잭션의 블록 포함 여부를 알아내는 데 유용하며 이러한 과정을 단순 지급 검증이라 한다.

11.1. 단순 지급 검증의 필요성

전체 비트코인 블록체인 크기는 약 200기가로 저장하기 부담스럽다. 따라서 비트코인 지갑을 만들 수 있다.

11.2. 머클트리

머클트리는 포함증명을 효율적으로 탐색할 수 있도록 고안된 자료구조이다. 순서 리스트와 해시함수가 필요하다. 머클트리를 생성하기 위한 알고리즘은 다음과 같다.

1. 선정한 해시함수로 순서 리스트에 있는 모든 항목의 해시값을 구한다.

2. 만약 해시값이 정확히 1개면 작업을 완료한다.

3. 그렇지 않은 경우 만약 해시값 개수가 홀수면 마지막 해시값을 리스트에 추가한다.

( c.f ) 짝수 개수를 유지하는 이유 : 완전이진트리( ; 각 노드가 최대 2개의 자식 노드를 갖는 트리형태의 자료구조)를 유지해야 하므로

4. 순서대로 인접한 해시값을 2개씩 쌍으로 묶고 그의 해시값을 구해서 부모 자리에 위치시킨다.

5. 2번으로 돌아간다.

Caution. 머클 트리의 보안 취약점

해시 값 개수가 홀수 인 것이 보안의 취약점으로 꼽힌다.

11.3. 머클 부모

2개의 해시 값 중 왼쪽과 오른쪽 해시 값을 연결하여 붙인 값을 부모 해시 값이라고 한다.

H = 해시 함수

P = 부모 해시 값

L = 왼쪽 해시 값

R = 오른쪽 해시 값

부모 해시 값을 구하는 이유는 효율적으로 포함증명을 제공할 수 있기 때문이다.

11.4. 머클부모 레벨

리스트의 모든 해시 값에 대해 부모 해시 값을 구하는 것을 말한다.

11.5. 머클루트

머클 루트를 얻기 위해서 하나의 부모 해시 값을 얻을 때까지 머클 부모 레벨을 계산한다.

11.6. 블록에서 머클루트 계산하기

실제 계산 시 엔디언 문제를 조심해야 한다. 머클트리의 각 단말 노드를 리틀엔디언으로 읽는다. 이후 머클루트를 계산하고 최종 결과는 다시 리틀엔디언으로 표현된다.

11.7. 머클트리 활용하기

라이트 노드는 특정 트랜잭션이 블록에 포함됐는지 확인하기 위해 모든 트랜잭션을 알 필요가 없다.

11.8. 머클블록

풀 노드가 포함증명을 보낼 때 함께 보내는 두 가지 정보가 있다. 첫째는 머클트리의 구조에 대한 정보이고, 둘째는 머클트리에서 어떤 해시값이 어느 위치에 있는지에 대한 정보이다.

11.8.1. 머클트리 구조

머클트리는 단말 노드에서 루트 노드 방향으로 생성되므로 라이트 노드는 단말 노드의 개수만 알면 머클트리의 구조를 알 수 있다.

11.8.2. 머클트리 코딩하기

11.8.3. 머클블록 커맨드

11.8.4. 플래그 비트와 해시값 사용

플래그 비트는 깊이 우선 순회에서 방문하는 노드와 머클블록에 들어 있는 해시값을 대응시킨다.

플래그 비트 작성 규칙은 다음과 같다.

1. 노드의 해시값이 hashes 필드에 들어 있으면 플래그 비트는 0이다.

2. 라이트 노드가 계산해야 하는 내부 노드인 경우 플래그 비트는 1이다.

3. 블록 포함 여부가 궁금한 관심 트랜잭션에 해당하는 단말 노드이면 플래그 비트는 1이다.

## 12. 블룸 필터

풀 노드가 라이트 노드의 관심 트랜잭션을 아는 방법? 라이트노드가 풀 노드에게 주소를 알려준다. 또는 이를 포함한 더 많은 트랜잭션이 있는 상위집합 즉, 블룸 필터를 사용한다.

12.1. 블룸 필터의 정의

상위집합에 포함되는 트랜잭션을 선정하는 필터이다.

12.1.1. 다중 해시 함수 사용

해시함수를 여러 개 사용해서 블룸 필터를 구성한다면 비트 필드의 길이를 상당히 줄일 수 있다.

12.2. BIP0037 블룸 필터

블룸 필터를 정의하는 정보는 다음과 같다.

1. 비트 필드의 길이, 혹은 분류되는 그룹의 개수

2. 해시함수의 개수

3. 오탐지율에 영향을 주는 tweak 파라미터

4. 설계된 블룸 필터의 비트 필드 값

12.3. 블룸 필터 설정

라이트 노드가 블룸 필터를 생성하면 이를 풀 노드에게 알려야 한다. 포함증명을 보낼 때 풀노드가 이를 활용한다.

12.4. 머클블록의 입수

라이트 노드는 풀 노드에게 관심 트랜잭션이 포함된 머클블록을 요청하는 커맨드가 필요하다.

12.5. 관심 트랜잭션 입수

블룸 필터를 풀 노드로 전송한 라이트 노드는 특정 블록에 자신의 관심 트랜잭션의 포함 증명을 받을 수 있다.

## 13. 세그윗

세그윗은 비트코인 네트워크에서 소프트포크(하위 호환 업그레이드)이다. 세그윗이 가져온 변화는 다음과 같다.

1. 증가된 블록 크기

2. 트랜잭션 가변성 문제 해결

3. 체계적인 업그레이드가 가능한 세그윗 버전 식별 방식

4. 이차 해싱문제 해결

5. 오프라인 지갑 수수료 계산의 보안 강화

13.1. p2wpkh 스크립트

트랜잭션 가변성은 트랜잭션 의미는 유지하면서 트랜잭션 ID가 변경될 수 있는 성질을 말한다. 해제 스크립트의 데이터가 증인필드에 분리되어 있다.

13.1.1. 트랜잭션 가변성

트랜잭션 가변성은 트랜잭션 의미는 유지하면서 트랜잭션 ID가 변경될 수 있는 성질을 말한다.

라이트닝 네트워크에서 가장 작은 단위인 결제 채널을 만들 때 트랜잭션 ID가 변할 수 있다는 사실은 중요한 고려 사항이다.

트랜잭션 ID는 전체 트랜잭션 기반으로 계산된다. 트랜잭션 ID는 전체 트랜잭션의 hash256 해시값이다.

13.1.2. 가변성 문제

트랜잭션 가변성을 해결하기 위해 해제 스크립트 필드를 비우고 그 안에 있어야 할 데이터를 별도의 필드로 옮긴다.

13.2 p2wpkh 트랜잭션

세그윗을 이해하려면 세그윗 노드가 세그윗 노드와 예전 노드에 보내는 트랜잭션 모습을 살펴보는 것이 도움된다,

13.3 p2sh-p2wpkh 스크립트

p2wpkh는 Bech32라고 하는 새로운 주소 형식을 사용한다.

세그윗 스크립트가 p2sh 리딤 스크립트 안에 있으므로 네스티드 세그윗이라고 한다.

13.4 p2wpkh와 p2sh-p2wpkh 코딩하기

13.5 p2wsh 스크립트

복잡한 기능을 위해서는 더 유연한 스크립트가 필요하다. P2wsg스크립트이다.

13.6 p2sh-p2wsh 스크립트

13.7 p2wsh와 p2sh-p2wsh 코딩하기

## 14. 고급 주제 및 다음 단계

위 내용들은 배워야 할 기술에 있어 일부에 지나지 않는다. 오픈소스 커뮤니티에 기여하거나 다른 방법을 알아 본다.

14.1. 주제 제안

14.1.1. 비트코인 지갑

비밀키를 안전하게 관리하는 것은 쉽지 않다. 그래서 안전한 지갑을 설계하는 것은 생각보다 어렵다.

연상 단어 시드

256비트 시드를 베껴 쓰거나 옮기는 것은 쉽지 않다.

14.1.2. 결제 채널과 라이트닝 네트워크

결제 채널은 라이트닝 네트워크를 이루는 가장 작은 단위이다. 이들의 동작방법을 살피는 것은 좋은 주제이다.

14.2. 오픈소스 기여

1. Bitcoin Core

2. Libbitcoin

3. btcd

4. Bcoin

5. pycoin

6. Bitcoinj

7. BitcoinJS

8. BTCPay

14.3. 프로젝트 제안

14.3.1. 테스트넷 지갑

비트코인에서 보안은 가장 중요한 요소이다. 테스트넷의 지갑을 만들 때 고려 사항이 무척 많다. 많은 정보를 얻을 수 있다.

14.3.2. 블록 탐색기

Postgres나 MySQL같은 전통적인 DB를 사용하면 유용하다. 비트코인 코어에는 색인이 없으므로 주소 색인이 있으면 트랜잭션을 쉽게 조회할 수 있다.

14.3.3. 온라인 쇼핑몰

비트코인 기반 온라인 쇼핑몰을 만들면 많은 것을 배울수 있다. 벡엔드로 비트코인 결제 가능을 사용하면 된다.

14.3.4. 유틸리티 라이브러리

예시로 세그윗 서명해시의 BIP0143 직렬화를 코딩하며 프로토콜 프로그래밍에 익숙해진다.

14.3.5. 관련 기업 입사

특정 회사의 API 프로그래밍 경험을 해본다.

참고 자료

Programming Bitcoin by Jimmy Song. Copyright 2019 Jimmy Song, 978-1-03149-9