비트코인

백서 번역본

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

본 번역본은 가상자산 투자자들을 위한 참고용입니다. 본 번역본은 투자 권유를 목적으로 만들어지지 않았으며 원문을 단순히 번역한 것으로, 내용의 정확성은 원문 작성 주체에 달려 있으며 크립토닷컴 코리아는 투자 결과에 대하여 어떠한 책임도 지지 않습니다. 본 번역본의 내용에 대한 자세한 문의는 백서 원문을 참고하시길 바랍니다.

개요

완전히 개인 대 개인(P2P) 버전인 전자화폐는 금융기관을 통하지 않고 한 사람과 다른 사람 간의 온라인 지급이 가능하게 하는 것입니다. 디지털 서명(digital signature)은 그 해결책 의 일부로, 만약 이중지불(double-spending) 방지하기 위해 여전히 신뢰할만한 제3자(TTP, trusted third party)가 필요하게 된다면 큰 이점이 사라지게 될 것입니다. 그래서 우리는 P2P 네트워크를 이용해 이중지불 문제의 해결책을 제안하고자 합니다. 이 네트워크는 계속되는 해시 기반의 작업증명 체인(chain of hash-based proof-of-work)에 Transaction(거래)들을 해시화 하여 시점별로 기재함으로써, 해당 작업증명을 다시 수행하지 않고서는 변경할 수 없는 기록을 생성합니다. 가장 긴 체인은 일련의 거래 사건들을 증명할뿐 아니라, 이것이 가장 큰 CPU 파워 풀에서 생성했음을 증명합니다. 네트워크에 참여하는 과반수(majority) CPU 파워가 네트워크 를 공격하지 않는 데 협력하는 노드에 의해 통제되는 한 가장 긴 체인을 생성하고 공격자를 앞지를 것입니다. 이 네트워크 자체는 최소한의 구조를 요구합니다. 각 노드들에게는 최선의 노력 기반으로 메시지가 전송되며 노드들은 자신들이 네트워크를 떠나있을 때 생성된 가장 긴 작업증명 체인을 거래의 증명으로 받아들임으로써 마음대로 네트워크를 떠나거나 재합류할 수 있습니다.

1. 서론

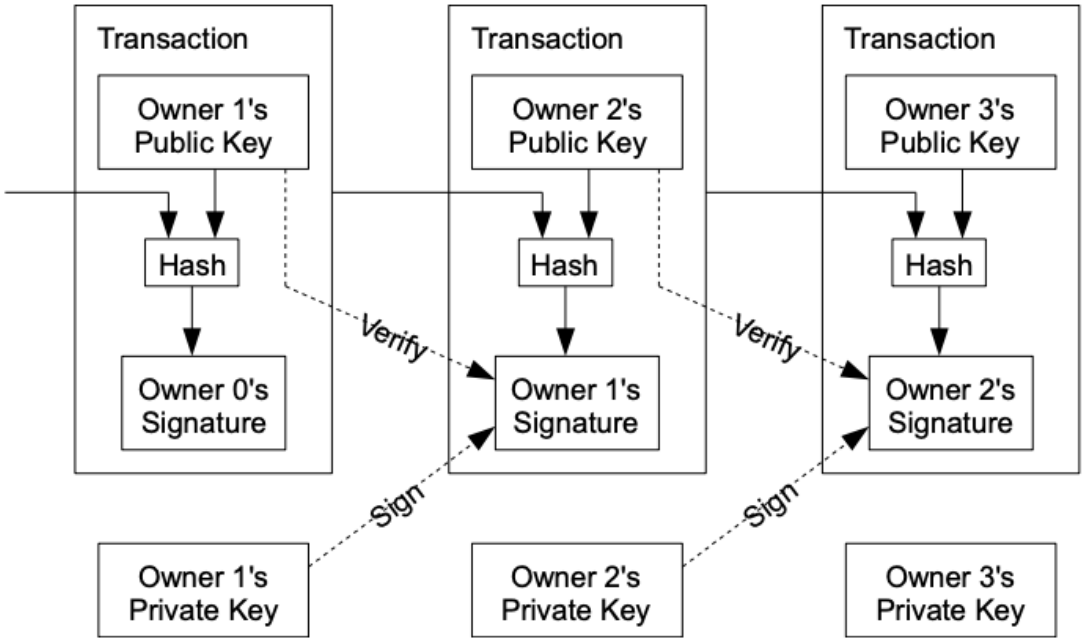
지금까지 인터넷상에서의 상행위는 전자적 지급(electronic payments)을 처리하는데 있어 거의 예외없이 신뢰할만한 제3자(TTP)라는 역할을 하는 금융기관들에 의존해 왔습니다. 이 시스템 은 대부분 거래에서는 충분히 잘 작동되지만, 신뢰 기반 모델(trust based model) 자체에 내 재하는 취약점에서 여전히 자유롭지 못합니다. 이 모델에서는 거래를 되돌리게 되면 금융기관이 이에 따른 분쟁을 중재하는 문제가 초래될 수 밖에 없기 때문에 거래를 완전히 되돌릴 수 없 도록(non-reversible transactions)하는 것이 실제로 불가능합니다. 중재 비용이 발생하기 때문 에 거래 비용이 증가하게 되고, 실무적으로는 최소 거래 금융을 제한할 수 밖에 없게 되고, 일 상적으로 일어나는 작은 규모의 거래도 일어나지 못하게 되고, 되돌릴 수 없게 되어 있는 서비스에 대한 지급 자체를 환불불가 조건으로 하지 못하는 등 많은 비용이 발생하게 됩니다. 지급 번복의 가능성이 있게 되면, 신뢰할 만한 당사자의 필요성이 더 커집니다. 상인들은 그 고객들을 더 조심해야 하고, 그들이 필요로 하는 것보다 더 많은 정보를 불필요하게 요구하면서 고객들 을 귀찮게 할 것입니다. 어느 정도의 사기 가능성은 피할 수 없을 것입니다. 이러한 비용과 지급의 불확실성은 직접 만나 실물 화폐를 사용해 거래하게 되면 피해 나갈 수 있겠지만, 신뢰할만한 당사자 없이 커뮤니케이션(온라인) 채널을 통해 지급을 수행할 수 있는 메커니즘은 존재하지 않습니다.

이제 필요하게된 것은 신뢰 대신 암호화 증명을 기반으로 하는 전자 지급 시스템이고, 이는 신뢰받는 제3자 필요 없이 두 자발적인 당사자가 서로 직접 거래할 수 있도록 해 주는 것입니다. 거래 자체를 전산적으로(computationally) 되돌릴 수 없도록 할 수 있게 된다면 판매자들을 사기로부터 보호할 수 있을 것이고, 구매자들을 보호하기 위해 일상적인 에스크로 메커니 즘이 쉽게 실행될 수 있을 것입니다. 이 논문에서 우리가 지급수단의 중복사용 문제에 대한 해결 책으로 제시하는 것은, 제반 거래의 발생 시간 순서에 관한 전산적 증명을 생성하기 위해 분 산된 P2P 타임스탬프 서버를 이용하는 것입니다. 시스템을 정직하게 사용하는 노드들이 공격자 인 노드에 협력하는 그룹보다 더 많은 CPU 파워를 집단적으로 통제한다고만 하면 이 시스템 은 안전합니다.

2. Transactions (거래)

우리는 전자코인을 일련의 디지털 서명의 체인이라 정의합니다. 코인의 각 소유자는 이전의 트랜 잭션 및 다음 소유자의 공개키(public key)7)에 대한 암호화 해시값에 디지털 서명을

하여ㅍ(digitally signing a hash), 이를 코인의 끝부분에 추가하여 해당 코인을 다음 소유자에게 이전합니다. 코인을 받은 사람은 소유권의 연결고리가 맞는지 검증하기 위해 그 서명을 검증할 수 있습니다.



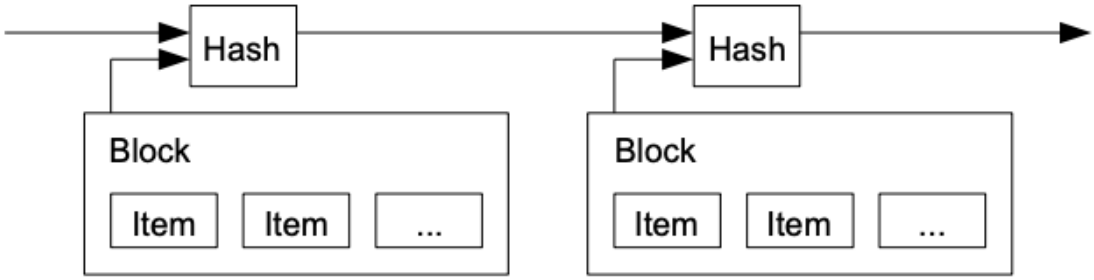
물론 문제는 코인 수령인이 소유자들 가운데 하나가 그 코인을 중복사용하지 않았다는 점을 확인할 수 없다는 점입니다. 일반적인 해결책은 조폐발행처(mint)처럼 신뢰할만한 중앙권한자를 도입하여 매 거래가 있을 때마다 이전에 사용한 적이 있는지 코인의 중복사용 여부를 점검하는 것입니다. 매 거래 후에, 그 코인은 반드시 새로운 코인으로 발행되게끔 발행처(mint)로 회수돼야 하고, 오로지 발행처가 직접 발행한 새 코인들만 중복사용된 것이 아니라는 신뢰가 부여됩니다. 이 방법의 문제점은 전체 화폐 시스템의 운명이 조폐발행처를 운영하는 회사에 의존 하게 된다는 것이고, 모든 거래가 마치 은행을 거치는 것처럼, 그 회사를 거쳐야 한다는 점입니다.

코인을 받는 사람을 위해서는 이전의 소유자들이 그와의 거래보다 앞선 어떤 거래들에 대해 서명을 하지 않았다는 점을 알 방법이 필요합니다. 우리의 이러한 목적을 위해 중요한 고려대상은 가장 먼저 있었던 거래인 것이고 따라서 우리는 그 후에 일어나는 중복사용 시도는 무시하기로 합니다. 어떤 거래가 없었음을 확정하는 유일한 방법은 모든 거래에 대해 아는 것입니다.
조폐발행처 기반 모델에서는 발행처가 모든 거래를 인지하고 어떤 거래가 먼저 도달했는지 결정했습니다. 하나의 신뢰 할만한 당사자 없이 이러한 목적을 달성하기 위해서는, 반드시 모든

거래가 공개적으로 알려져야 하며, 수취가 이루어진 트랜잭션들의 단일한 순서의 기록(single history of the order)에 대해 모든 참여자가 동의하는 시스템이 필요합니다. 수취인에게 필요한 것은 매 트랜잭션이 일어날 때마 다 그 트랜잭션은 가장 빨리 받은 트랜잭션이라는 점을 50% 이상의 노드들이(majority of nodes) 동의하고 있다는 증명입니다.

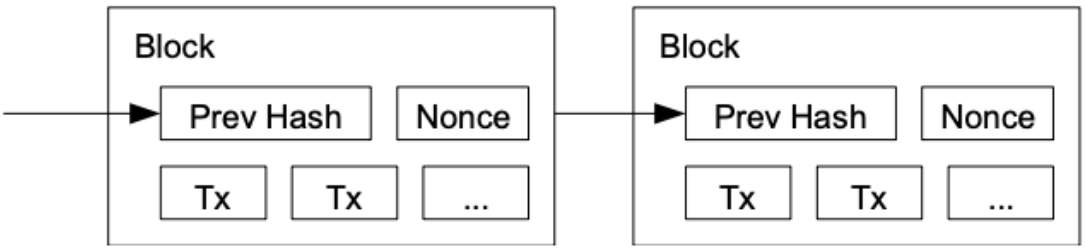
3. 타임스탬프 서버

우리가 제안하는 솔루션은 타임스탬프 서버로 시작합니다. 타임스탬프 서버는 타임스탬프가 찍힐 항목(item)들의 블록에 대한 해시를 취하고, 마치 신문이나 유즈넷 포스트에 뿌리듯이, 그 해시를 다른 노드들에 널리 공표하는 방식으로 작동합니다. 타임스탬프는 그 데이터 가 그 해시에 포함되기 위해, 그 시점에 명백하게 존재했다는 것을 증명하게 됩니다. 각 타임스탬프는 그 해시에 이전의 타임스탬프를 포함하게 되고, 체인을 형성하면서, 이후 추가되는 타임 스탬프는 그 앞의 타임스탬프들을 보강하게 됩니다.



4. Proof-of-Work (작업증명)

우리가 P2P 기반으로 분산된 타임스탬프 서버를 구현하려면, 신문이나 유즈넷 포스트 같은 방식보다는, 아담 백(Adam Back)의 Hashcash와 유사한 작업증명 시스템을 이용할 필요가 있습니다. 이 작업증명은, SHA-256과 같은 알고리즘으로 해시되었을 때 어떤 값을 탐색(스캐닝)하는 과정을 포함하는데, 이 해시값은 여러 개의 숫자 0비트로 시작합니다. 평균적인 작업증명 소요 시간은 요구되는 “0” bit의 개수에 따라 기하급수적으로 증가하며(exponential), 작업증명은 단 한 번의 해시를 수행함으로써 검증될 수 있습니다.



우리는 우리가 생각하는 타임스탬프 네트워크를 위해, 그 블록의 해시가 요구되는 “0”의 비트 자릿수에 해당하는 어떤 값을 찾을 때까지 그 블록 내에 임시값(nonce)을 증가시켜 가면서 작업증명을 실행하게 됩니다. 일단 CPU 자원이 해당 블록의 작업증명 조건을 충족하기 위해 사용 된 이후에는, 그 블록은 그 작업의 재실행 없이는 변경될 수 없게 됩니다. 이후에 생성되는 블록 들은 그 블록 뒤에 체인으로 연결되므로, 그 블록을 변경하는 작업에는 그 후에 생성되어 연결 된 모든 블록에 대한 작업증명을 다시 수행하는 것이 포함되는 것입니다.

또한 작업증명은 다수결에서 대표성을 결정하는 문제도 해결할 수 있습니다. 만약 과반수의 토대가 IP주소 하나당 하나의 의결권 체제를 기반으로 하고 있다면 과반수의 결정은 수많은 IP주 소를 할당할 수 있는 누군가에 의해 뒤집힐 수도 있게 됩니다. 그러므로 작업증명은 필수적으로 CPU 하나당 하나의 의결권을 가질 수밖에 없습니다. 과반수의 결정은 가장 긴 체인에 의해 대표 되며, 이는 투여된 가장 큰 작업증명 노력을 반영합니다. 만약 정직한 노드들이 CPU 자원의 과 반수를 통제할 수만 있다면 정직한 노드들이 통제하는 체인(the honest chain)은 경쟁 관계에 있을 수 있는 다른 어떤 체인보다 가장 빠른 속도로 앞서 나가며 길어질 것입니다. 어떤 공격자 가 과거의 블록을 수정하려면 그 블록 물론 이후에 연결된 모든 블록에 대한 작업증명을 다 시 수행해야만 하고 정직한 노드들에 의해 만들어진 작업속도를 따라잡고 체인의 길이를 넘어서 야만 합니다. 우리는 뒤에서 후속 블록들이 추가되기 때문에 속도가 느린 어떤 공격자가 따라잡을 가능성은 기하급수적으로 감소할 수 있다는 점을 보여줄 것입니다.

시간이 지나면서 네트워크에 참여하는 노드를 운영하면서 하드웨어 속도를 늘리고 이들의 다양해 지는 이해관계를 보상하기 위해 시간당 생성되는 블록의 평균 수에 맞게 조정되는 이동평균 (moving average)에 따라 작업증명 난이도가 결정될 것입니다. 블록이 너무 빠르게 생성된다고 하면 작업증명 난이도가 증가하게 됩니다.

5. 네트워크

네트워크는 다음과 같은 단계를 거쳐 작동하게 된다:

1) 새로운 트랜잭션(거래내역)들은 모든 노드에 전파된다.

2) 각 노드는 새로운 거래 내역들을 하나의 블록에 모은다.

3) 각 노드는 자신의 블록을 생성하기 위해 어려운 작업증명 작업을 수행한다.

4) 한 노드가 하나의 작업증명 해답을 찾게 되면, 그 블록을 모든 노드에 전파한다.

5) 노드들은 작업증명이 끝난 블록에 포함된 모든 거래가 유효하고 이전에 중복사용된 것이 아닌 때에만 그 블록을 수용한다.

6) 노드들은 자신이 해당 블록을 수용했다는 의사표시는, 자신이 수용한 블록의 해시값을 바로 직전의 해시값으로 사용하여 체인 내에 다음 블록을 생성하는 작업을 수행함으로써 표현한다.

노드들은 항상 가장 긴 체인을 올바른 체인으로 인정하고 그 가장 긴 체인을 확장하는 작업 을 계속해 나간다. 만약 작업증명 해답을 찾은 두 노드가 서로 다른 버전의 다음 블록을 동시에 전파하는 경우, 그 둘 중 어느하나를 먼저 받는 일부 노드들이 있을 수 있습니다. 이런경우 이들은 자신이 먼저 받은 블록을 토대로 작업을 수행하고, 다른 갈래의 블록은 그쪽이 더 길어 질 경우를 대비하여 저장해 둔다. 다음 작업증명의 해답이 찾아지면서 어느 한쪽 갈래가 더 길어지게 되면 이 두 갈래 묶음은 풀어지게 되고 다른 쪽 가지에서 작업을 수행하던 노드들 은 그 더 긴 체인 쪽으로 작업을 전환하게 됩니다.

새로운 트랜잭션의 전파라 해서 반드시 모든 노드에 도달될 필요가 있는 건 아닙니다. 새로운 트랜잭션이 많은 노드에 도달될수록 이것들은 어떤 블록이 길어지기 전에 그 블록에 포함될 것입니다. 블록 전파(알림)는 메시지를 빠뜨리고 못 받는 때에도 취약하지 않을 것입니다. 만약 어떤 노드가 어 떤 블록을 전달받지 못했다면 그 노드는 자신이 다음 블록을 받으면서 자신이 블록 하나를 놓쳤다 는 것을 알 수 있게 될 때 그 블록을 요청하게 될 것입니다.

6. Incentive (보상)

관례상, 하나의 블록 안에서 첫 번째 트랜잭션은 새 코인이 해당 블록을 생성한 자에게 주어지는 특별한 트랜잭션입니다. 이는 노드들이 네트워크를 지지하는 대가로 주어지는 보상이고, 이 시스템에 는 코인을 발행하는 중앙당국이 없으므로 유통될 코인들을 맨 처음 배분해 주는 방법이 될 것이 다. 새로운 코인을 일정량으로 꾸준히 추가해 나가는 것은 금 채굴자들이 자원을 소모해

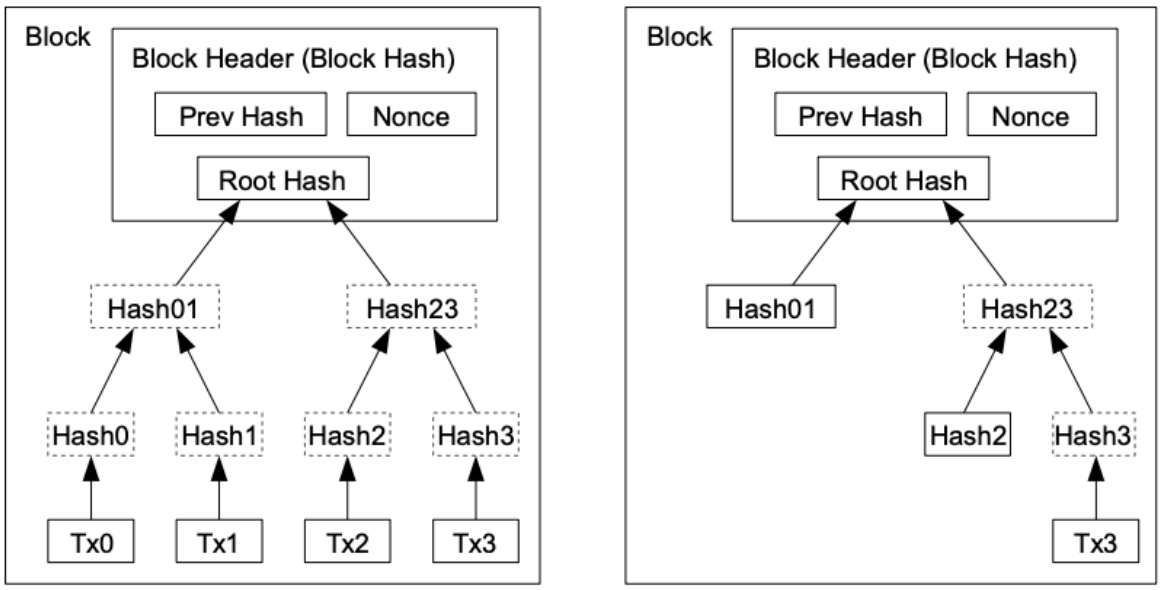
가며 유통되는 금의 양을 늘려 가는 것과 유사합니다. 우리의 경우 사용되는 자원은 CPU 연산 시간과 전력량입니다.

또한, 트랜잭션 수수료도 작업증명에 대한 보상이 될 수 있습니다. 만약 어떤 트랜잭션의 작업증명 수행에 수반된 아웃풋 가치가 그 투입된 가치보다 적다면 그 차액은 트랜잭션 수수료로, 그 트랜잭션이 들어있는 블록의 보상 가치에 추가되는 것입니다. 일단 사전에 확정된 코인의 수량이 시장에 풀려 유통되게 된 이후에는, 블록생성에 대한 이 유인책은 전적으로 트랜잭션 수수료로 전환될 수 있고, 시장에 풀린 전체 코인 수량이 완전히 고정되기 때문에 인플레이션에 서 완전히 자유롭게(inflation free) 됩니다.

유인책은 노드들이 네트워크에 정직한 참여자로 남아있게 하는 역할을 할 것입니다. 만약 탐욕에 사로잡힌 공격자가 모든 정직한 노드들보다 더 많은 CPU 자원을 끌어모을 수 있게 된다면, 이 공격 자는 그 자원을 사람들을 속여 자신이 실행한 지급을 다시 몰래 훔쳐오는데 쓸 건지 새 코인을 생성하는데 쓸건지를 선택해야 할 상황을 맞을 것입니다. 공격자는 CPU 자원을 이용해 이 시스템의 신뢰성과 자신이 이룬 부의 타당성을 떨어뜨리기보다는, 다른 모든 사람을 합친 것보다 더 많은 새 코인을 생성할 기회를 줌으로써 본인에게 더 이익이 될, 시스템의 규칙을 지키는 것이 더 이익 이 된다는 점을 알게 될 것입니다.

7. 디스크 공간 재확보

일단 코인(디지털 서명의 체인)에 포함된 가장 최근의 트랜잭션이 충분히 많은 블록 밑으로 묻히게 되면, 그전에 사용된 트랜잭션은 디스크 저장공간을 절약하기 위해 버려질 수 있습니다. 이 블록의 해시를 깨뜨리지 않고 이것이 가능해지도록, 트랜잭션(거래내역)은 루트만 해당 블록의 해시에 포 함된 채로 머클 트리(Merkel Tree) 구조로 암호화(해시)됩니다. 오래된 블록들은 트리의 가지들을 잘라냄으로써 압축될 수 있고, 하위 해시들은 저장될 필요가 없습니다.



Transactions Hashed in a Merkle Tree

After Pruning Tx0-2 from the Block

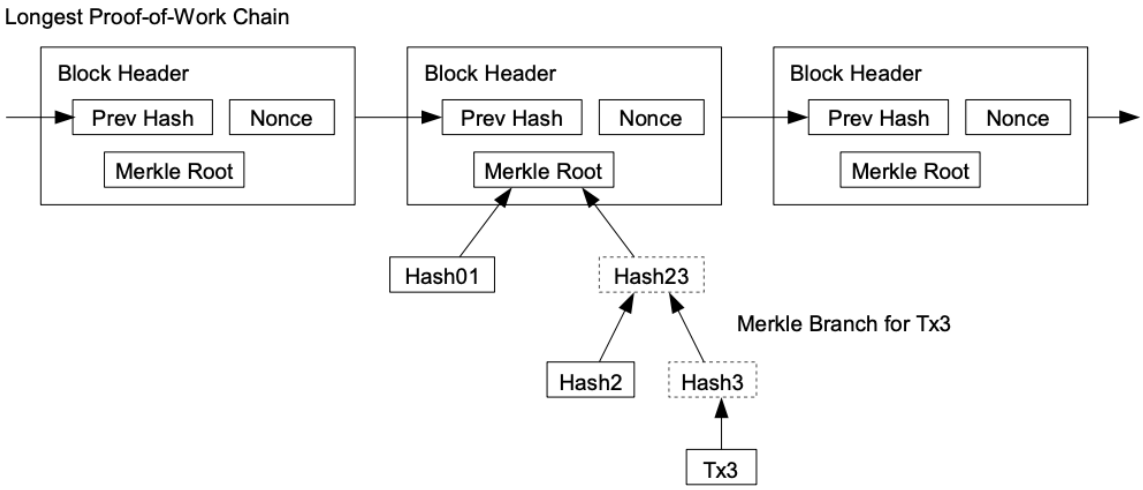
[설명]

(좌) 트랜잭션들이 머클트리 구조로 해시된 모습과 (우)블록에서 거래내역 0~2가 제거된 후 모습

트랜잭션이 들어있지 않은 블록헤더(block header)는 약 80바이트 정도입니다. 블록이 10분마다 생성된다고 가정하면, 1년에 4.2MB씩 늘어난다(80byte*6*24*365=4.2MB). 2008년 기준으로 대개 시중에서 판매되는 컴퓨터의 용량이 2GB 램이고, 매년 1.2GB씩 성장할 것으로 예측하는 무어의 법칙24)을 고려하면, 비록 블록헤더들이 메모리에 반드시 저장돼야 한다고 해도 저장공간 자체 는 크게 문제가 되지 않을 것입니다.

8. 간소화된 지급 확인

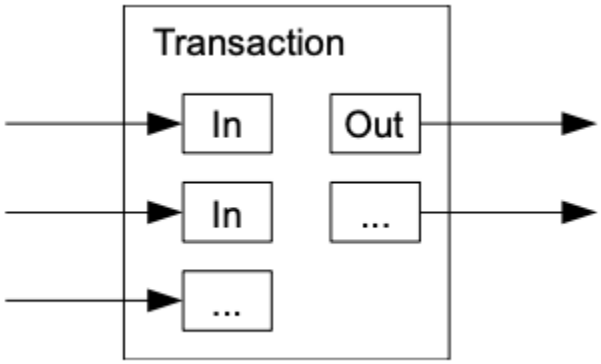
네트워크 전체 노드를 가동하지 않고서도 지급을 증명할 수 있습니다. 유저는 가장 긴 작업증명 체인 의 블록헤더들 사본만 갖고 있으면 되는데, 유저는 자신이 가장 긴 체인을 갖고 있다는 점에 확신이 들 때까지 네트워크상의 다른 노드들에 질의하면서 이를 얻을 수 있고, 해당 트랜잭션을 그 타 임스탬프가 들어있는 블록에 연결해 주는 머클트리 가지(Merkle branch)를 획득할 수 있습니다. 유저 는 자신이 직접 해당 트랜잭션을 확인할 수는 없지만, 해당 거래를 체인에 들어있는 어떤 장소로 연결해 봄으로써, 어떤 네트워크 노드가 이를 승인했음을 알 수 있고, 그 후에 블록들이 추가된 사실을 통해 네트워크가 이를 받아들였다고 확신할 수 있게됩니다.



위와 같이, 정직한 노드들이 네트워크를 통제하고 있는 한 검증은 신뢰할 수 있겠지만, 만약 이 네트워크가 정직한 노드들보다 우세한 자원을 가진 공격자에 의해 지배된다면 더 취약해질 수 있습니다. 네트워크 노드들은 스스로 트랜잭션들이 유효한지 검증할 수 있지만, 한편으로는 공격자가 계속해서 정직한 노드들보다 우세한 자원으로 이 네트워크를 지배할 수만 있다면 공격자가 꾸며낸 트랜잭션들에 의해 이 간소화된 방식이 농락당할 가능성이 충분합니다. 이러한 취약점으로부터 시스 템을 보호하기 위한 한가지 전략은, 유효하지 않은 비정상 블록을 감지한 네트워크 노드들이 보내주는 경고를 받아들이는 것입니다. 이 경고는 유저의 소프트웨어가 그 비상적인 부분을 확인할 수 있게 블록 전체와 경고 대상 트랜잭션을 내려받으라고 촉구하는 것입니다. 빈번하게 지급을 수취하 는 사업자들로서는 좀 더 독립적인 보안과 신속한 검증 체계를 갖추고자 자신의 노드를 운영하기를 원할 수도 있겠습니다.

9. 거래금액의 결합과 분할

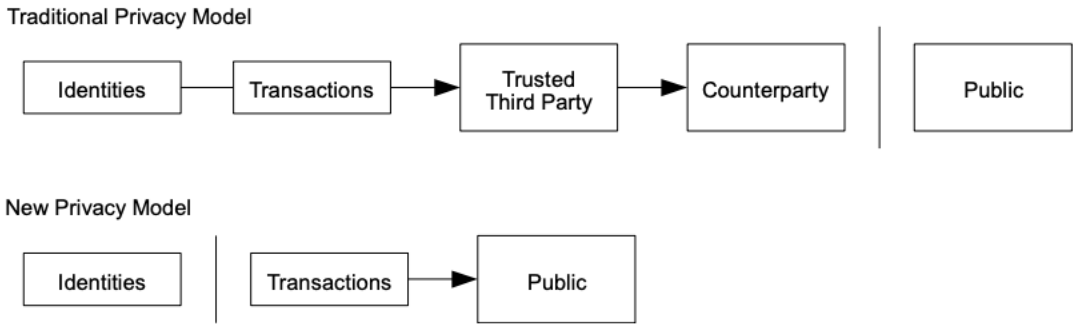
코인들을 개별로 다룰 수는 있겠지만, 매번 최소 단위를 이전할 때마다 트랜잭션을 분리하는 것은 복잡하고 불편할 것입니다. 이전하는 가치를 쪼개거나 합치기 위해, 트랜잭션들은 다수의 입력과 결 과물을 포함하게 됩니다. 보통은 이전의 더 큰 트랜잭션으로부터의 단일한 인풋이 있거나 좀 더 작 은 금액들을 결합하는 복수의 인풋이 있을 수 있고, 그리고 크게 보면 두 개의 아웃풋, 즉 지급을 위한 아웃풋 하나와, 만약 전송자에 되돌려 줄 게 있다면, 그 돌려줄 잔액에 관한 아웃풋이 있을 수 있을 것입니다.



트랜잭션 하나가 몇 개의 트랜잭션을 토대로 하고, 또 그러한 트랜잭션들은 더 많은 트랜잭션을 토대로 하는 때에도, 트랜잭션 정보가 널리 퍼지는 것(fan-out)은 문제가 되지 않는다는 점은 주목해야 합니다. 어떤 트랜잭션의 히스토리(정보)를 보여줄 완전하고 유일한 사본을 추출할 필요는 결코 없는 것입니다.

10. 거래자 비밀 보호

전통적인 뱅킹모델에서는 정보에 대한 접근을 관계 당사자들 및 신뢰받는 제3자(TTP)로 제한함으로써 일정 수준의 비밀보장(level of privacy)을 달성하고 있습니다. 이 방식은 모든 트랜잭션을 공개 적으로 공표할 필요성이 있는 모델에는 맞지 않지만, 이 모델에서도, 공개키들을 익명화하여 다른 곳에 보관하는 방식으로 그 정보의 흐름을 끊음으로서 프라이버시가 여전히 유지될 수 있습니다. 네트 워크에 참여하는 대중은 누군가가 어떤 금액을 누군가에게 전송하고 있다는 것을 볼 수 있지만, 이 트랜잭션을 특정인과 결부시키는 정보는 알 수 없습니다. 이러한 점은 증권거래소들이 공개하는 정 보의 수준과 유사한데, 증권거래소들은 개별 매매거래가 이루어진 시간과 매매거래의 규모에 관한 정보인 '거래체결 정보'(tape)를 공표할 뿐, 당해 거래의 당사자들이 누구인지 그 신분까지 알려주 지는 않습니다.



[설명]

(상) 기존의 프라이버시(거래자 신원정보 보호) 모델과 (하) 새로운 프라이버시 모델

추가적인 방화벽이 될 수 있는 것으로서, 각 트랜잭션이 어떤 한 명의 소유자로 연결되지 않도록 쌍을 이루는 두 개의 키(key pair, 즉 공개키와 비밀키)가 새롭게 사용될 필요가 있습니다. 복수로 입력되는 트랜잭션들의 경우 일부 연결고리는 여전히 피할 수 없고, 이 경우에는 필연적으로 그 입 력된 트랜잭션들은 동일한 소유자가 소유하고 있었다는 사실 정도는 드러나게 됩니다. 만약 어떤 암 호키의 소유자가 누군지 드러나게 될 때 그 리스크는, 그 연결로 인해 그 동일한 소유자에 속한 다른 거래들까지도 드러나게 할 수 있다는 것입니다.

11. 계산

우리는 한 공격자가 정직한 체인보다 빠른 다른 대체 체인을 생성하려는 시나리오를 생각해 본다. 이렇게 하는 것이 성공한다고 하더라도, 가치를 난데없이 날조해 내거나 결코 그 공격자의 것이 아닌 돈을 가져가게 할 정도로 이 시스템이 임의적인 변경에 노출돼 있다고는 할 수 없습니다. 노드들 은 유효하지 않은 트랜잭션을 지급으로 인정하지 않으려 할 것이고, 그리고 정직한 노드들은 그러 한 트랜잭션이 포함된 블록을 절대 받아들이지 않을 것입니다. 공격자가 유일하게 시도할 수 있는 것은 자신이 최근에 지급에 사용한 돈을 취소하고 회수하기 위해 자신의 트랜잭션 중 하나를 바꾸려는 것 뿐입니다.

정직한 체인과 공격자 체인간에 일어날 경쟁의 특성은 Binominal Random Walk와 같다고 볼 수 있습니다. 성공하는 이벤트는 정직한 체인이 블록 하나를 확장하면선 그 리드를 +1만큼 늘려 가는 것이고, 실패하는 이벤트는 공격자의 체인이 블록 하나를 확장하면서 그 갭을 -1로 줄이는 것이 될 것입니다.

공격자가 뒤진 상황(given deficit)에서 뒤따라 잡을 가능성은 도박꾼의 파산 문제(Gambler's Ruin problem)와 유사합니다. 무한정의 신용을 가진 어떤 도박꾼이 돈을 잃고 있는 적자 상태에서 손익분기점에 다다르기 위해 잠재적으로 무제한으로 게임을 시도하는 경우를 생각해 봅시다. 우리는 그가 손익분기점에 도달할 가능성, 혹은 어떤 공격자가 정직한 체인을 따라잡을 가능성(확률)을 아래와 같이 계산해 볼 수 있습니다:

p = probability an honest node finds the next block
q = probability the attacker finds the next block
q_z = probability the attacker will ever catch up from z blocks behind

$$q_z = \begin{cases} 1 & \text{if } p \le q \\ (q/p)^z & \text{if } p > q \end{cases}$$

p는 정직한 노드가 다음 블록을 찾을 확률이고,
q는 공격자가 다음 블록을 찾아낼 확률이며
qz는 공격자가 z 블록 뒤처진 상태에서 따라잡을 확률입니다.

우리가 공격자가 다음 블록을 찾아낼 확률(q)보다 정직한 노드가 다음 블록을 찾아낼 확률(p) 이 높은 상황(p>q)을 가정하면, 따라잡을 블록들의 수가 증가할수록 (z 블록 뒤처진) 공격자 가 따라잡을 확률은 기하급수적으로 감소하게 됩니다. 이는 공격자에게 불리한 것인데, 공격자가 운 좋게 한발 앞서 계속해 가지 않는 한, 그는 점점 더 뒤처지면서 점점 그의 기회는 줄어들면서 사라지게 됩니다.

이번에는 새로운 트랜잭션을 전송한자가 그 거래를 변경할 수 없다고 충분히 확신이 서기 까지에는 이 트랜잭션을 받은 수취인이 얼마나 오래 기다려야 할 것인지에 대해 생각해 봅시다. 여기서 전송자는 그 수취인이 당분간은 본인에게 전송자가 지급한 것이라고 믿게 하면서, 얼마의 시간이 지나면 그 거래를 바꿔 자신에게 돈이 되돌아오게끔 변경하려는 공격자라 가정해 봅시다. 수취인은 그러한 일이 발생하게 되면 알림 경보를 받게 될 것이지만, 전송자는 그런 알림이 충분히 늦게 일어나길 바랄 것입니다.

수취인은 새로운 (암호)키 쌍을 생성하고, 서명하기 직전에 그중 공개키를 전송자에게 보내주게 됩니다. 이렇게 하면, 전송자인 공격자가 앞선 시간에 운 좋게 충분히 앞서나가게 될 때까지 여러 블 록의 체인(chain of blocks)을 준비해 두었다가 그 시점이 되면 그 (가짜) 거래를 실행하는 일을 하지 못하게 방지하게 됩니다. 일단 트랜잭션이 전송되면, 부정직한 전송자는 몰래 자신의 거래를 대 체하는 다른 버전을 담은 다른 가지의 체인에서(on a parallel chain) 작업을 시작합니다.

수취인은 그 트랜잭션이 어떤 블록에 추가되고 그 뒤에 z개의 블록이 연결될 때까지 기다린다. 그 는 공격자가 작업을 얼마만큼 진척시켰는지는 정확히 알지 못하겠지만, 정직한 블록들은 블록당 평균 예상되는 시간이 걸릴 것이라고 가정할 것이기 때문에, 공격자의 잠재적 진척도는 다음 산식 에 의한 기댓값을 갖는 프와송 분포(Poisson distribution with expected value)가 될 것입니다:

$$\lambda = z\frac{q}{p}$$

이제 우리가 공격자가 아직도 따라잡을 수 있을 확률을 계산하기 위해, 공격자가 해냈을 만큼의 매 진척율을 보여주는 프와송 분포 밀도(Poisson density)에다 그 특정 시점부터 따라잡을 수 있는 확률을 곱해 봅시다:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} (q/p)^{(z-k)} & \text{if } k \le z \\ 1 & \text{if } k > z \end{cases}$$

분포의 무한급수 꼬리를 계속 더해 나가지 않도록 식을 바꿔 정리해 보면 다음과 같다:

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{(z-k)}\right)$$

이를 C언어 code로 변환해 보면 다음과 같다:

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```

계산을 몇 번 돌려보면, 그 확률은 z 값에 따라 기하급수적으로 감소하는 것을 볼 수 있습니다.

```
q=0.1
z=0     P=1.0000000
z=1     P=0.2045873
z=2     P=0.0509779
z=3     P=0.0131722
z=4     P=0.0034552
z=5     P=0.0009137
z=6     P=0.0002428
z=7     P=0.0000647
z=8     P=0.0000173
z=9     P=0.0000046
z=10    P=0.0000012

q=0.3
z=0     P=1.0000000
z=5     P=0.1773523
z=10    P=0.0416605
z=15    P=0.0101008
z=20    P=0.0024804
z=25    P=0.0006132
z=30    P=0.0001522
z=35    P=0.0000379
z=40    P=0.0000095
z=45    P=0.0000024
z=50    P=0.0000006
```

P가 0.1%보다 작은 경우를 대입하여 풀어보면 아래와 같은 결과가 나옵니다.

```
P < 0.001
q=0.10   z=5
q=0.15   z=8
q=0.20   z=11
q=0.25   z=15
q=0.30   z=24
q=0.35   z=41
q=0.40   z=89
q=0.45   z=340
```

12. 결론

우리는 (제3자의) 신뢰에 의존하지 않는 전자적 트랜잭션(거래)에 필요한 시스템을 제안하였습니다. 우리는 디지털 서명들로 만들어지는 코인들로 이루어질 실용적인 체재(framework)를 구상해 보았는 데, 이 시스템은 소유권에 대해 강력한 통제장치는 제공하지만, 중복지급(사용)을 방지하는 방법이 없다면 불완전한 것입니다. 우리는 이 문제를 해결하기 위해, 정직한 노드들이 CPU 자원의 과반수를 통제하는한 어떤 공격자가 바꾸려는 시도를 전산적으로 불가능해지도록 거래에 관한 공개적 내력(public history of transactions)을 빠르게 만들어내는, 그 공개적 내력을 기록하기 위해 작업증명을 사용하는 P2P 네트워크를 제안하였습니다. 이 네트워크는 그 구조화되지 않은 단순성 (unstructured simplicity)으로 인해 강력한 힘을 발휘하게 됩니다. 노드들은 약간의 조정(협의)만으 로 모두 한 번에 작동합니다. 노드들이 누군지 그 정체가 확인될 필요도 없습니다. 왜냐하면 메시지들이 어떤 특정 장소를 목표로 하여 보내지지 않고, 최선의 노력이라는 기반으로 전달되기만 하면 되기 때문입니다. 노드들은 자신들이 떠나 있을 때 어떤 일이 발생하였는지에 대한 증거로서 작업증명 체 인을 받아들이면 되므로, 자신의 의지에 따라 네트워크를 떠나 있거나 다시 합류할 수 있습니다. 노드들은 유효하지 않은 블록에 대한 작업은 거부함으로써 거절하고 유효한 블록들에 대해서는 그것을 확장하는 작업을 함으로써 이를 받아들인다는 의사표시를 하면서, 자신의 CPU 파워를 가지고 의 사결정에 참여합니다. 시스템을 유지하는데 필요한 제반 규칙과 유인책은 이 합의 메커니즘과 더불어 집행될 수 있습니다.