비트코인: 개인간 전자화폐 시스템 (P2P Electronic Cash System)

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

Translated in Korean from bitcoin.org/bitcoin.pdf
by Seungwon (Eugene) Jeong 정승원 - blockchainstudio.info

초록. 순수한 P2P(peer-to-peer) 방식의 전자화폐는 금융기관을 거치지 않고도 온라인 지불을 한 쪽에서 다른 쪽으로 직접 보낼 수 있게 해준다. 전자서명이 부분적 해결책을 제공하지만 만약 이중지불(double-spending) 문제를 방지하기 위해 신뢰할 수 있는 제 3 자를 필요로 한다면 주요 장점들이 사라지고 만다. 우리는 P2P 네트워크를 이용하여 이중지불 문제에 대한 해결책을 제시하려 한다. 이 네트워크는 거래(transaction)들을 해시(hash) 기반의 작업증명(proof-of-work) 체인(chain)에 해싱하여 타임스탬프(timestamp)를 찍어, 그 작업증명을 다시 하지 않고는 변경할 수 없는 기록을 만든다. 가장 긴 체인은 목격한 이벤트들의 순서에 대한 증명뿐만 아니라 그것이 가장 큰 컴퓨팅(CPU) 파워 풀에서 나왔음에 대한 증명 역할도 한다. 과반수(majority)의 컴퓨팅 파워가 네트워크 공격에 협력하지 않는 노드(node)들에 의해 제어된다면 그들이 가장 긴 체인을 생성하고 공격자들을 앞지를 것이다. 이 네트워크 자체는 최소한의 구조만을 요구한다. 메세지는 최대한 가능한 한도 내에서 (on a best effort basis) 브로드캐스트(broadcast)되며 노드들은 원한다면 네트워크를 떠났다가 가장 긴 작업증명 체인을 그들이 없던 사이에 일어난 일들에 대한 증명으로 받아들여 다시 합류할 수도 있다.

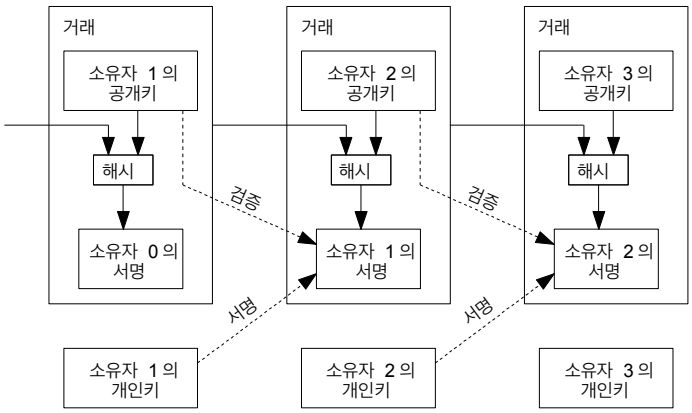
1. 서론

전자 상거래는 전자지불을 처리함에 있어서 신뢰할 수 있는 제 3 자 역할을 하는 금융기관들에 거의 전적으로 의존해 왔다. 이러한 시스템은 대부분의 거래에는 충분히 잘 동작하지만, 신뢰 기반 모델의 태생적인 약점을 여전히 지닌다. 금융기관들은 분쟁 중재를 피할 수 없기 때문에 완전히 취소(철회) 불가능(non-reversible)한 거래는 사실상 불가능하다. 중재의 비용은 거래 비용을 증가시키며, 이는 현실적으로 사용 가능한 최소 거래 규모에 제한을 만들어 작은 일상적인 거래를 불가능하게 하며, 또한 취소 불가능한 서비스에 대한 취소 불가능한 지불을 할 수 없음에서 오는 더 큰 비용이 있다. 철회 가능성으로 인해 더 큰 신뢰수준이 필요하다. 판매자는 굳이 필요 없었을 정보까지 귀찮게 요구하며 고객들을 경계할 수밖에 없게 된다. 일정 수준의 사기(fraud)는 불가피한 것으로 인정된다. 이러한 비용들과 지불의 불확실성은 물리적 화폐(physical currency)를 직접 만나(in person) 사용한다면 해결할 수 있지만 신뢰할 수 있는 제 3 자가 없이도 온라인상에서 지불을 가능하게 하는 메커니즘은 존재하지 않는다.

필요한 것은 신뢰 대신 암호학적 증명에 기반한 전자지불 시스템으로 이는 신뢰할 수 있는 제 3 자 없이도 그 어떤 두 사람이 직접 거래하는 것을 가능하게 해준다. 취소가 계산상(computationally) 거의 불가능한 거래는 판매자를 사기로부터 보호해주며, 구매자를 보호해 줄 수 있는 통상적인 에스크로(escrow) 메커니즘은 쉽게 구현될 수 있다. 이 논문에서 우리는 거래의 시간적 순서에 대한 계산적 증명을 생성하는 P2P 분산 타임스탬프 서버를 이용하여 이중지불 문제에 대한 해결책을 제시하고자 한다. 이 시스템은 정직한 노드들의 컴퓨팅 파워의 합이 그 어떤 공격자 그룹 노드들의 것들보다 크다면 안전하다.

2. 거래 (Transactions)

우리는 전자화폐(electronic coin)를 전자서명의 체인으로 정의한다. 각 소유자는 이전거래와 다음 소유자의 공개키(public key)에 대한 해시에 전자서명을 한 것을 마지막에 추가하여 코인을 전달한다. 수취인은 소유권의 체인을 검증하고 싶다면 이 전자서명들을 검증하면 된다.

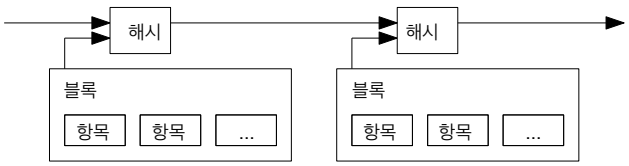


물론 문제는 수취인이 이전 소유자들이 이중지불을 했는지를 검증할 수는 없다는 것이다. 통상적인 해결책은 신뢰할 수 있는 중앙 기관이나 조폐국(mint)을 도입하여 그들이 모든 거래에 대해 이중지불을 검사하도록 하는 것이다. 한번 거래가 끝난 코인은 새 코인의 발행을 위해 조폐국으로 회수되어야 하며 조폐국에서 직접 발행한 코인들만이 이중지불되지 않은 것으로 신뢰할 수 있다. 이 해결책의 문제점은 전체 통화 시스템의 운명이 이 조폐국을 운영하는 회사에 의존해야 한다는 것이다. 모든 거래가 그들을 통해야만 하는 것이다. 마치 은행처럼.

우리는 이전 소유자들이 그 어떤 앞선 거래에도 서명하지 않았다는 것을 수취인이 알 수 있는 방법이 필요하다. 우리의 목적에는 최초의 거래만이 중요하기에 그 이후의 이중지불 시도는 신경 쓰지 않는다. 어떤 거래가 존재하지 않았다는 것을 확인할 수 있는 유일한 방법은 모든 거래를 다 확인하는 방법뿐이다. 조폐국 기반 모델에서는 조폐국이 모든 거래를 보관하여 어떤 거래가 먼저인지를 판단한다. 제 3 자 없이 이를 달성하려면 거래는 반드시 공개적으로 알려져야 하며 [1], 우리는 참가자들이 거래의 순서에 대한 단일 기록에 동의할 수 있는 시스템이 필요하다. 수취인은 매 거래마다 과반수의 노드들이 그것이 첫 사용이라고 동의해주는 증명을 필요로 한다.

3. 타임스탬프 서버 (Timestamp Server)

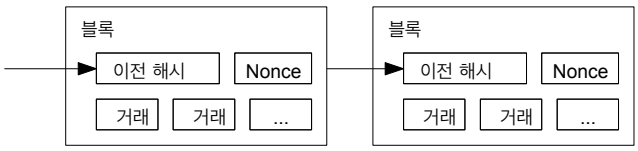
우리가 제안하는 해결책은 타임스탬프 서버로 시작한다. 타임스탬프 서버는 타임스탬핑을 할 항목들의 블록에 대한 해시를 계산해 이를 신문이나 유즈넷(Usenet) 포스트처럼 넓게 퍼뜨리는 방식으로 동작한다 [2-5]. 타임스탬프는 해당 시점에 그 데이터가 해시 계산에 들어가기 위해서 명백히 존재했음을 증명해준다. 각 타임스탬프는 이전 타임스탬프를 해시에 포함하여 각각의 추가 타임스탬프가 이전의 타임스탬프들을 보강 증명하는 체인을 형성한다.



4. 작업증명 (Proof-of-Work)

분산 타임스탬프 서버를 P2P 방식으로 구현하기 위해서, 우리는 신문이나 유즈넷 포스팅 방식이 아닌 Adam Back 의 해시캐시(Hashcash [6])와 비슷한 작업증명 시스템을 사용할 필요가 있다. 작업증명은 SHA-256 같은 것을 사용해 계산된 해시가 일정 개수의 0 으로 시작하는 것을 찾는 작업을 포함한다. 소요되는 평균적인 작업은 요구되는 0 의 개수에 따라 지수적으로(exponential) 증가하지만, 검증은 한 번의 해시 계산으로 가능하다.

우리는 타임스탬프 네트워크를 위해서, 블록의 해시가 요구되는 일련의 0 을 포함하는 것을 찾을 때까지 블록 내의 nonce 를 증가시키는 것으로 작업증명을 구현한다. 일단 연산 작업을 통해 작업증명을 완성했다면 이 작업을 다시 하지 않고서는 블록은 변경될 수가 없다. 블록들이 체인으로 연결되기 때문에 하나의 블록을 변경하기 위해서는 그 이후 모든 블록들에 대한 작업증명을 다시 해야 한다.



작업증명은 다수결 의사결정에 있어서 투표자(representation) 선정 문제도 해결해 준다. 과반수가 한 IP 당 한 표 방식으로 결정된다면 많은 IP 를 할당받는 것이 가능한 사람에 의해 공격받을 수가 있다. 작업증명은 근본적으로 한 CPU 당 한 표 방식이다. 다수결에 의한 결정은 가장 긴 체인으로 나타내어지며 이는 가장 많은 작업증명 노력이 들어간 체인이다. 과반수의 컴퓨팅 파워가 정직한 노드에 의해 제어된다면 정직한 체인이 가장 빨리 길어져 다른 경쟁 체인들을 앞서게 될 것이다. 이전의 한 블록을 수정하기 위해서는, 공격자는 해당 블록과 그 이후의 모든 블록들의 작업증명을 다시 해야 하고 또 정직한 노드들의 작업을 따라잡고 추월해야 한다. 우리는 뒤에서 느린 공격자가 따라잡을 확률이 블록들이 더해짐에 따라 지수적으로 감소함을 보일 것이다.

시간의 흐름에 따른 하드웨어 속도 증가나 노드 운영에 대한 관심의 변화를 보정하기 위해 작업증명의 난이도(difficulty)는 정해진 시간당 평균 블록 수를 목표로 하는 이동 평균(moving average)에 의해 결정된다. 너무 빨리 생성된다면 난이도는 증가하게 된다.

5. 네트워크 (Network)

네트워크가 동작하는 단계는 다음과 같다:

1) 새로운 거래들이 모든 노드에게 브로드캐스트된다.
2) 각 노드는 새로운 거래들을 블록에 수집한다.
3) 각 노드는 그 블록에 대한 어려운 작업증명을 수행한다.
4) 어떤 노드가 작업증명을 마쳤으면 해당 블록을 모든 노드에게 브로드캐스트한다.
5) 노드들은 해당 블록의 모든 거래들이 유효하고 이미 사용되지 않았을 경우에만 그 블록을 승인(accept)한다.
6) 노드들은 해당 블록의 해시를 이전 해시로 사용하여 다음 블록을 체인에 만드는 것으로 해당 블록의 승인을 표현한다.

노드들은 항상 가장 긴 체인을 올바른 것으로 간주하며 그 체인을 확장해 나간다. 만약 두 노드가 서로 다른 다음 블록을 동시에 브로드캐스트했다면 노드들마다 서로 다른 블록을 먼저 받게 될 수가 있다. 이런 경우에는 먼저 받은 것에 작업을 해나가지만 다른 블록을 포함한 체인이 더 길어지는 것에 대비하여 다른 브랜치(branch)도 보관을 한다. 이 동점 상황(tie)은 다음 작업 증명이 발견되어 한 브랜치가 더 길어지면 깨지게 되고, 다른 브랜치에서 작업하던 노드들은 더 긴 브랜치로 전환하게 된다.

새 거래들의 브로드캐스트가 꼭 모든 노드에 도달할 필요는 없다. 많은 노드에 도달하기만 한다면 머지않아 그들은 블록에 포함되게 된다. 블록 브로드캐스트는 메시지 유실에도 견딜(tolerant) 수 있다. 어떤 노드가 블록을 받지 못했다면 다음 블록을 받았을 때 해당 블록이 빠진 것을 깨닫고 다시 요청할 것이다.

6. 인센티브 (Incentive)

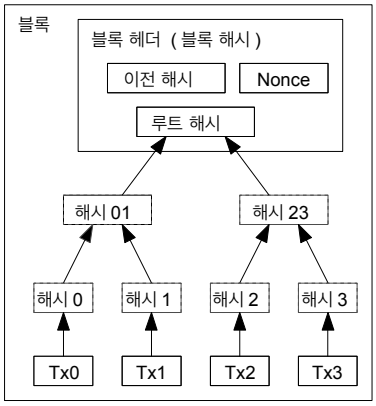
관례상, 블록의 첫 거래는 해당 블록을 생성한 사람이 소유하게 되는 신규 코인을 생성하는 특별한 거래가 된다. 이는 노드들에게 네트워크를 유지할 인센티브(incentive)를 제공하고, 코인들을 발행할 중앙 기관이 없기 때문에, 코인들이 최초로 유통(circulation)될 방법을 제공한다. 일정량의 새로운 코인이 꾸준히 추가되는 것은 금을 유통하기 위해 자원을 소비하는 금 채굴자와 유사하다. 우리의 경우 컴퓨팅 시간과 전기가 소비되는 자원인 것이다.

인센티브는 거래 수수료로도 제공될 수 있다. 거래의 결과값이 입력값보다 작다면 그 차이가 해당 거래를 포함하는 블록에 인센티브 값으로 추가되는 거래 수수료인 것이다. 미리 정해진 수량의 코인이 모두 발행되고 나면 인센티브는 거래 수수료 체제로 전환되고 인플레이션으로부터 자유로워(inflation free)진다.

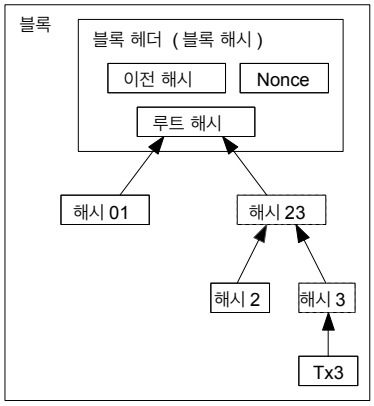
이 인센티브는 노드들이 정직함을 유지하는 것을 도와줄 수 있다. 만약 욕심 많은 공격자가 다른 모든 정직한 노드보다 더 큰 컴퓨팅 파워를 가질 수 있다면 그는 지불한 것들을 훔쳐서 사람들을 속이는 것과 새로운 코인들을 생성하는 것 중 선택을 해야 한다. 그는 시스템과 자신의 재산의 유효성을 해치기보단 다른 모든 사람들보다 많은 신규 코인을 지급하는 이 시스템의 규칙에 따라 행동하는 것이 더 이득이 됨을 깨닫게 될 것이다.

7. 저장 공간 재확보 (Reclaiming Disk Space)

코인 안의 최종 거래가 충분히 많은 블록들에 묻히게 되면 그 이전에 사용된 거래들은 저장 공간을 절약하기 위해서 폐기될 수 있다. 블록의 해시를 망가뜨리지 않고 이 문제를 해결하기 위해서, 거래들은 머클 트리(Merkle Tree) 안에 해시되어 저장되며 [7][2][5], 루트(root)만 해당 블록의 해시에 포함된다. 그리하여 오래된 블록들은 트리의 가지들을 잘라냄으로써 압축될 수 있다. 중간의 해시들은 저장될 필요가 없다.



머클 트리에 해시된 거래내역들

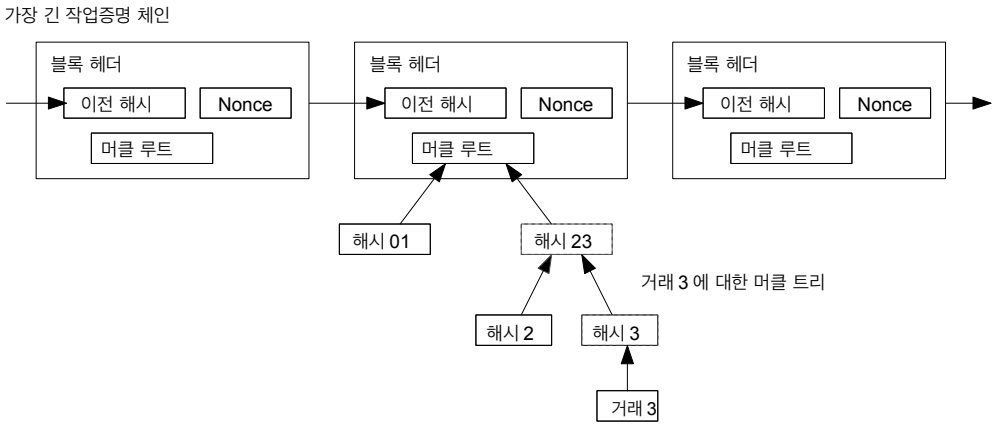


거래내역 0-2 를 블록에서 제거한 후의 모습

거래내역이 없는 블록 헤더의 크기는 80 bytes 정도이다. 매 10 분마다 블록을 생성한다면 1 년에 80 bytes * 6 * 24 * 365 = 4.2MB 정도가 된다. 2008 년 현재 보통 2GB 정도의 램을 장착한 컴퓨터가 팔리고 있고, 해마다 1.2GB 가 증가하리라 예측하는 무어의 법칙(Moore's Law)에 따른다면 심지어 블록 헤더들이 메모리상에 보관되어야만 한다고 하더라도 저장 공간은 문제가 되지 않을 것이다.

8. 간소화된 지불 검증 (Simplified Payment Verification)

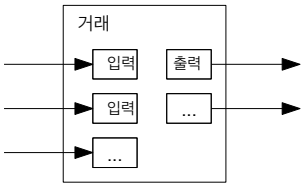
전체 네트워크 노드를 구동하지 않고도 지불을 검증하는 것이 가능하다. 사용자는 단지 가장 긴 작업증명 체인(이는 그가 가장 긴 체인을 가졌다고 확신할 때까지 네트워크 노드들에게 조회를 해서 얻을 수 있다)의 블록 헤더들의 사본만 유지하고, 또 해당 거래를 그것의 타임스탬프가 포함된 블록과 연결해주는 머클 브랜치를 얻기만 하면 된다. 해당 사용자는 스스로 거래를 체크하지는 못하지만, 그것을 체인의 한 부분에 연결함으로써 네트워크 노트가 해당 거래를 받아들이는지 확인할 수 있으며, 그 이후에 추가되는 블록들이 네트워크가 그걸 받아들이는지 추가로 확인해주게 된다.



이처럼 이 검증 방식은 정직한 노드들이 네트워크를 통제하는 한 신뢰할 수 있지만, 네트워크가 공격자에 의해 장악(overpower)된다면 보다 취약해지게 된다. 네트워크 노드들이 스스로 거래들을 검증할 수 있지만, 이 간단한 방법은 공격자가 계속해서 네트워크를 장악할 수 있는 한 공격자의 조작된 거래로 속는 것이 가능하다. 이를 막을 하나의 전략은 네트워크 노드들로부터 유효하지 않은 블록이 발견되었을 때 경고를 받아, 사용자의 소프트웨어가 불일치를 확인하기 위해 전체 블록과 경고를 받은 거래내역을 다운로드하도록 하는 것이다. 빈번한 지불을 받는 사업자들은 아마도 보다 독립적인 보안과 빠른 검증을 위해 자체 노드를 운영하기를 원할 것이다.

9. 금액 병합과 분할 (Combining and Splitting Value)

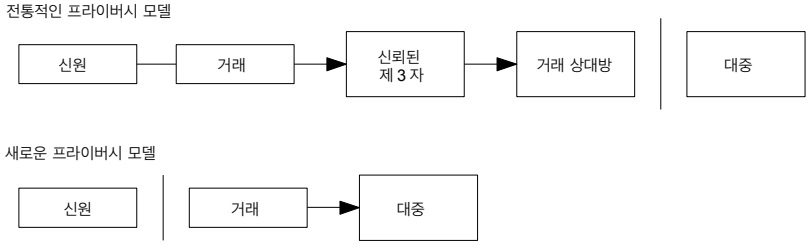
비록 코인들을 개별적으로 다루는 것이 가능하긴 하지만 한 송금의 모든 잔돈을 개별적인 거래로 만드는 것은 거추장스러운 일이다. 금액을 나누거나 합치는 것이 가능하도록, 한 거래는 여러 개의 입력과 출력을 포함한다. 보통은 더 큰 이전의 거래로부터 오는 단일 입력 또는 작은 금액들로 구성된 여러 개의 입력, 그리고 최대 두개의 출력이 존재한다. 하나는 지불을 위한 것이고, 다른 하나는 만약 있다면 보낸 사람에게 돌아갈 거스름돈이다.



한 거래가 여러 개의 거래들에 의존되고 또 이들은 다시 더 많은 거래들에 의존되는 팬아웃(fan-out)은 여기서 문제가 되지 않음을 주목할 필요가 있다. 한 거래내역의 완전한 독립(standalone) 사본을 추출해야 할 필요는 절대 없다.

10. 프라이버시 (Privacy)

전통적인 은행 모델은 정보 접근 권한을 거래 당사자들과 신뢰할 수 있는 제 3 자에게만 제공함으로써 일정 수준의 프라이버시를 제공한다. 모든 거래들을 공개적으로 알려야 하는 필요성은 이 방식을 불가능하게 하지만, 정보의 흐름을 다른 면에서 차단함으로써 프라이버시 보호가 여전히 가능하다. 바로 공개키를 익명으로 보존함으로써. 대중(public)은 누군가가 다른 누구에게 얼마를 보낸다는 걸 볼 수 있긴 하지만 해당 거래를 특정인과 연결(link)지을 수 있는 정보는 없다. 이는 "테이프(tape)"이라고도 부르는 시간과 각각의 거래의 규모는 공개되지만, 거래자들이 누구인지는 공개되지 않는 증권 거래소의 정보공개 수준과 비슷하다.



추가적인 보호책으로써 각각의 거래에 새로운 키 페어를 사용해서 공통된 소유자가 누구인지 연결지을 수 있는 여지를 막아야 한다. 여러 개의 입력을 가지는 거래의 경우에는 해당 입력들이 동일한 소유자의 것임이 드러날 수밖에 없어서 일부의 연결고리는 피할 수 없긴 하다. 위험성은 하나의 키의 소유자가 드러나면, 해당 연결고리는 다른 거래들까지 동일한 소유자의 것이라는 것을 드러내게 될 수가 있다.

11. 계산 (Calculations)

공격자가 정직한 체인보다 더 빠르게 다른 체인을 만드는 시나리오를 고려해보자. 심지어 이게 가능하더라도 이것이 시스템을 무차별적인 변경(예를 들자면 있지도 않은 금액을 만들어 낸다든지, 공격자에게 속한 적도 없던 돈을 취하게 한다든지)에 처하게 하지는 않는다. 노드들은 유효하지 않은 거래를 지불로써 받아들이지 않을 것이고 정직한 노드들은 이들을 포함한 블록을 받아들이지 않을 것이다. 공격자는 단지 그가 최근에 쓴 돈을 다시 받기 위해 자기 자신의 거래를 변경하는 것만 시도할 수 있다.

정직한 체인과 공격자의 체인 간의 경쟁은 이항 랜덤 워크(Binomial Random Walk)로 나타낼 수 있다. 성공 이벤트는 정직한 체인이 한 블록만큼 더 길어져 리드폭을 1 만큼 증가시키는 것으로, 실패 이벤트는 공격자의 체인이 한 블록만큼 늘어나 차이를 1 만큼 감소시키는 것으로 정의한다.

공격자가 주어진 열세로부터 따라잡을 확률은 도박사의 파산(Gambler's Ruin) 문제와 유사하다. 무한대의 신용도를 가진 도박사가 손실상태에서 시작해서 필요하다면 무한대의 수많은 시도를 하여 본전(breakeven)을 찾으려는 상황을 가정해보자. 언젠가는 본전에 도달할 확률, 즉 공격자가 정직한 체인을 언젠가는 따라잡을 확률은 다음과 같다 [8]:

p = 정직한 노드가 다음 블록을 찾을 확률
q = 공격자가 다음 블록을 찾을 확률
q_z = z 블록만큼 뒤처진 공격자가 언젠가는 따라잡을 확률

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

p > q 라는 가정하에서는 해당 확률은 공격자가 따라잡아야 할 블록의 수가 늘어남에 따라 지수적으로 감소한다. 이렇게 불리한 확률로 인해, 공격자가 초반에 운 좋게 치고 나가지 않는 이상 더 뒤처질 것이기 때문에 가능성은 거의 없다시피 줄어들고 만다.

이제 새로운 거래의 수취인이, 송금인이 거래를 변경하지 못할 것을 충분히 확신하려면 얼마 정도를 기다려야 할지 생각해보자. 송금인이 받는 사람으로 하여금 돈을 받았다고 일정 기간 믿게 만든 후 돈을 되돌려 받으려고 하는 공격자라고 가정해 보자. 그런 일이 일어나면 수취인에게 경고 메시지가 전달되겠지만 송금인은 이미 너무 늦었길 바라면서.

수취인은 새로운 키 페어를 생성하고 서명하기 바로 얼마 전에야 공개키를 송금인에게 전달한다. 이는 송금인으로 하여금 운 좋게 될 때까지 지속적인 시도를 해서 블록의 체인을 미리 준비하고 해당 거래를 실행하는 것을 방지해준다. 거래가 보내지고 나면 부정직한 송금인은 몰래 다른 버전의 거래를 담은 체인도 병행해서 작업하기 시작할 것이다.

수취인은 해당 거래가 블록에 포함되고 z 개의 블록이 추가로 연결되는 것을 기다린다. 그는 공격자가 얼마나 작업을 진척했을지 정확히는 모르지만, 정직한 블록이 블록당 평균 예상 시간만큼 걸린다고 가정했을 때, 공격자의 잠재적 진척도는 다음의 평균값을 가지는 푸아송 (Poisson) 분포를 따르게 된다:

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

이제 공격자가 여전히 따라잡을 수 있는 확률을 구하기 위해 우리는 공격자의 가능한 각각의 진척도에 대한 푸아송 확률에 그 시점부터 따라잡을 수 있는 확률을 곱한다:

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

분포 끝부분의 무한급수를 피하고자 식을 정리하고...

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

C 언어 코드로 변환하여...

```
#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;
}
```

어느 정도 결과를 돌려보면 z 에 따라 확률이 지수적으로 줄어듦을 알 수 있다.

```
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
```

0.1%보다 작은 P 에 대해 구해보면...

```
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 신뢰에 의존하지 않는 전자지불 시스템을 제안하였다. 우리는 소유권에 대한 확실한 제어가 가능한 전자서명에 기반한 통상적인 프레임워크의 코인에서 출발하였지만, 이는 이중지불 문제에 대한 해결책이 없다면 불완전하다. 이를 해결하기 위해, 정직한 노드들이 과반수의 컴퓨팅 파워를 가진다면 공격자가 이를 변경하는 것이 금새 계산적으로 비현실적이 되는 작업증명을 이용하여 거래들의 공개 이력을 기록하는 P2P 네트워크를 제안하였다. 이 네트워크는 구조적이지 않은(unstructured) 단순함에 있어서 견고(robust)하다. 노드들은 조직적인 조정(coordination)이 거의 없이도 다 함께 동작한다. 메시지들이 특정한 곳으로 보내져야 하는 것이 아니고 최대한 가능한 범위 내에서만(on a best effort basis) 전달되면 되기 때문에 노드들이 식별되어야 할 필요가 없다. 노드들은 원한다면 네트워크를 떠났다가 가장 긴 작업증명 체인을 그들이 없던 사이에 일어난 일들에 대한 증명으로 받아들여 다시 합류할 수도 있다. 그들은 컴퓨팅 파워를 이용해 투표(vote)를 하며, 이는 유효한 블록을 받아들이는 것은 그를 연장하는 것으로써, 유효하지 않은 블록을 거부하는 것은 거기에 추가로 작업함을 거절함으로써 표현한다. 어떤 필요한 규칙이나 인센티브는 이 합의 메커니즘(consensus mechanism)을 통해 시행(enforce)될 수 있다.

참고문헌

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In *20th Symposium on Information Theory in the Benelux*, May 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In *Journal of Cryptology*, vol 3, no 2, pages 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," *In Sequences II: Methods in Communication, Security and Computer Science*, pages 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 28-35, April 1997.

[6] A. Back, "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] R.C. Merkle, "Protocols for public key cryptosystems," In Proc. *1980 Symposium on Security and Privacy*, IEEE Computer Society, pages 122-133, April 1980.

[8] W. Feller, "An introduction to probability theory and its applications," 1957.