비트코인: 개인 대 개인 전자 화폐 시스템

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

Translated in Korean from bitcoin.org/bitcoin.pdf
by Mincheol Im

초록. 전적으로 개인 대 개인 버전인 전자 화폐 Electronic Cash 는 금융기관을 거치지 않고 한쪽에서 다른 쪽으로 직접 전달되는 온라인 결제 payments 를 실현한다. 전자 서명은 솔루션 일부를 제공하지만, 이중 지급 double-spending 을 막기 위해 여전히 신뢰받는 제삼자 trusted third party 를 둬야 한다면 그 주된 이점을 잃는다. 우리는 개인 대 개인 네트워크를 사용해 이중 지급 문제를 해결하는 솔루션을 제안한다. 이 네트워크가 거래를 해시한 타임스탬프를 일련의 해시 기반 작업증명 proof-of-work 체인에 찍고, 이 작업증명을 재수행하지 않고서는 변경할 수 없는 기록을 생성한다. 가장 긴 체인은 목격된 사건의 순서를 증명하는 동시에 그게 가장 큰 CPU 파워 풀에서 비롯했음을 증명하는 역할을 한다. 이 네트워크를 공격하는 데 협력하지 않는 노드가 CPU 파워 대부분을 제어하는 한, 가장 긴 체인을 생성하며 공격자를 압도할 것이다. 이 네트워크 자체는 최소한의 구조만 요구한다. 메시지는 최선의 노력을 기반으로 전파되고 broadcast, 노드는 네트워크를 마음대로 떠났다가 재합류할 수 있으며, 자신이 빠진 사이에 일어난 일의 증명으로 가장 긴 작업증명 체인을 받아들인다.

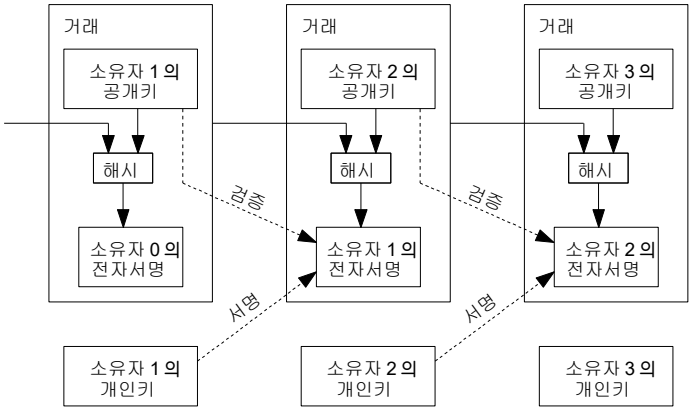
1. 서론

인터넷에서 상거래 commerce 는 전자 결제를 처리할 신뢰받는 제삼자 역할을 거의 전적으로 금융기관에 의존해 왔다. 이 시스템은 대다수 거래에 충분히 잘 동작하지만, 여전히 신뢰 기반 모델의 태생적 약점에 얽매인다. 금융기관이 분쟁을 중재해야 하므로, 실제로 완전히 비가역적인 non-reversible 거래는 가능하지 않다. 중재 비용이 거래 비용을 높여 실질적인 최소 거래 규모를 제한하고 일상적인 소액 거래 가능성을 가로막으며, 비가역 서비스에 맞는 비가역 결제 기능의 상실에서 더 큰 비용이 발생한다. 가역성 때문에 신뢰 수요 the need for trust 가 늘어난다. 상인 merchants 은 필요한 수준보다 정보를 더 얻어내려고 고객을 괴롭히지 않도록 신중히 처리해야 한다. 사기의 일정 비율이 불가피한 것으로 간주한다. 물리적 통화 currency 를 사용하는 대면 거래 외에, 신뢰받는 당사자 없이 통신 채널로 이런 비용과 결제 불확실성을 피할 방법은 존재하지 않는다.

필요한 것은 신뢰 대신 암호학적 증명 cryptographic proof 에 기반해, 거래 의사가 있는 두 당사자가 신뢰받는 제삼자를 찾지 않고 서로 직접 거래하게 하는 전자 결제 시스템이다. 전산적으로 computationally 철회 불가능한 거래가 사기로부터 판매자를 보호하고, 통상적인 제삼자 예치 escrow 방법이 구매자를 보호하기 위해 쉽게 구현될 수 있다. 이 논문에서, 우리는 거래 시간순의 전산적 증명을 생성하는 개인 대 개인 간 분산 타임스탬프 서버를 사용해 이중 지급 문제의 솔루션을 제안한다. 이 시스템은 정직한 노드가 공격자 노드의 협력 그룹보다 총체적으로 더 많은 CPU 파워를 제어하는 한 보안상 안전하다.

2. 거래

우리는 전자 화폐 electronic coin 를 디지털 서명의 체인으로 정의한다. 각 화폐 소유자는 자신에게 그 화폐를 보낸 직전 거래 명세 the previous transaction 및 그 화폐를 받는 다음 소유자의 공개키 the public key of the next owner 를 해시 처리한 값에 디지털 방식으로 서명하고 이를 화폐 끝에 추가해 다음 소유자에게 송금한다. 수취인 payee 은 그 서명을 검증해 화폐 소유권의 체인을 검증할 수 있다.

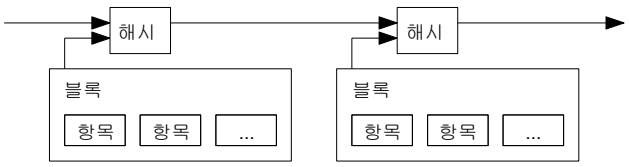


이 과정의 문제는 화폐 소유자 가운데 이중 지급하지 않은 한 사람을 수취인이 검증할 수 없다는 점이다. 통상적인 솔루션은 신뢰받는 중앙통제기관 trusted central authority 이나 조폐국 mint 을 세우고 모든 거래마다 이중 지급 여부를 점검하는 것이다. 거래를 마칠 때마다 이 화폐는 조폐국으로 회수돼 새로운 화폐로 발행돼야 하고, 조폐국에서 직접 발행된 화폐만이 이중 지급되지 않았다는 신뢰를 받는다. 이 솔루션을 적용 시 문제는 마치 은행처럼 모든 거래가 거쳐 가야 하는 조폐국 운영 기업에 전체 통화체계 money system 의 운명이 달려 있다는 점이다.

우리에게는 직전 화폐 소유자가 앞서 어떤 거래에도 서명하지 않았음을 수취인에게 알릴 수단이 필요하다. 이런 목적에 따라 우리는 가장 앞선 거래 하나만 인정하고, 뒤따르는 이중 지급 시도를 무시한다. 이중 지급 거래가 없음을 확인할 유일한 방법은 모든 거래를 인식하는 것뿐이다. 조폐국 기반 모델에서, 조폐국은 모든 거래를 인식했고 어느 거래가 최초로 도달한 것인지 결정했다. 신뢰받는 당사자 없이 이 방식을 달성하려면, 거래는 공개적으로 알려져야 하고[1], 우리에게는 참가자가 받은 거래 순서의 단일 이력에 합의하는 시스템이 필요하다. 수취인에게는 각 거래 당시 그게 최초로 받은 것이라고 노드 다수가 동의했다는 증명이 필요하다.

3. 타임스탬프 서버

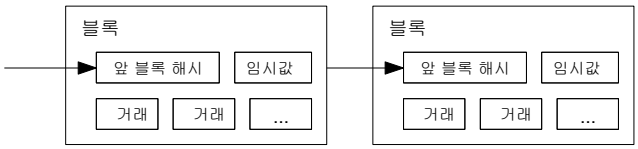
우리가 제안하는 솔루션은 타임스탬프 서버로 시작한다. 타임스탬프 서버는 타임스탬프가 찍힌 항목 블록의 해시값을 가져가고 그 값을 신문이나 유즈넷 게시물[2-5]처럼 널리 배포하는 식으로 작동한다. 이 타임스탬프는 그 데이터가 해시 처리에 들어가기 위해 명백히 그 시점에 존재했음을 증명한다. 각 타임스탬프는 그 해시 안에 직전 타임스탬프를 포함해 체인을 형성하며, 추가되는 각 타임스탬프가 그 이전 타임스탬프를 강화한다.



4. 작업증명

개인 대 개인 기반으로 분산 타임스탬프 서버를 구현하기 위해 우리는 신문이나 유즈넷 게시물 대신 애덤 백 Adam Back 의 해시캐시 Hashcash[6]와 유사한 작업증명 시스템을 사용할 필요가 있다. 작업증명은 SHA-256 같은 해시 처리를 거쳐 결괏값이 0 비트 zero bits 여러 개로 시작하는 특정 값을 찾는 작업을 수반한다. 여기에 필요한 일반적인 연산 작업은 특정 값이 요구하는 0 비트 개수에 따라 기하급수적으로 증가하며 해시 함수 실행 한 번으로 검증할 수 있다.

타임스탬프 네트워크용으로, 우리는 블록의 해시에 필요한 0 비트를 주는 값이 발견될 때까지 블록 안에 임시값을 증분하는 incrementing 것으로 작업증명을 구현했다. 일단 CPU 자원 effort 이 작업증명을 충족하는 데 동원되면, 그 블록은 해당 작업을 재수행하지 않고는 변경될 수 없다. 나중에 생성된 블록이 그 뒤에 연결되므로, 그 블록을 변경하는 재수행 작업은 그 뒤 모든 블록을 생성하는 연산까지 포함한다.



작업증명은 다수결 majority decision making 의 대표성 문제도 해결한다. 수적 우위가 IP 주소 하나당 한 표에 기반하면 누구든지 많은 IP 를 할당할 수 있는 이에 의해 장악될 수 있다. 작업증명은 기본적으로 CPU 하나당 한 표다. 다수 의사는 최다 작업증명 동작이 투입된 가장 긴 사슬로 대표된다. 정직한 노드가 CPU 파워 대부분을 제어하면, 가장 정직한 체인이 가장 빠르게 늘어나 다른 경쟁 체인을 압도할 것이다. 과거 블록을 변경하려면 공격자는 그 블록과 그 뒤를 잇는 모든 블록의 작업증명을 재수행해야 하고 그러면서 정직한 노드의 작업을 따라잡아 앞질러야 한다. 우리는 체인에 후속 블록이 추가될수록 후발주자인 공격자에게 따라잡힐 확률이 기하급수적으로 감소한다는 점을 나중에 보일 것이다.

시간이 지남에 따라 노드 운영 하드웨어 속도가 상승하고 관여도 interest 가 변화하면 이를 보정하기 위해 작업증명 난도 difficulty 는 시간당 평균 블록 수를 목표로 하는 유동적인 평균 목표치로 결정된다. 블록이 너무 빨리 생성되면 난도가 높아진다.

5. 네트워크

이 네트워크를 실행하는 단계는 다음과 같다:

1) 새로운 거래가 모든 노드에 전파된다.
2) 각 노드가 새로운 거래를 블록에 수집한다.
3) 각 노드가 그 블록에 맞는 난도의 작업증명을 찾는다.
4) 노드가 작업증명을 찾으면 해당 블록을 모든 노드에 전파한다.
5) 노드는 모든 거래가 유효하며 아직 지급되지 않았을 때만 그 블록을 받아들인다.
6) 노드는 블록을 받아들였음을 나타내기 위해 앞서 받아들인 블록의 해시를 직전 해시로 사용해 체인의 다음 블록을 생성하는 작업을 수행한다.

노드는 항상 가장 긴 체인을 옳은 것으로 간주하고 그걸 잇는 작업을 지속한다. 동시에 두 노드가 다음 블록의 서로 다른 버전을 전파하면, 일부 노드가 그중 하나 또는 다른 것을 먼저 받을 수 있다. 이때 그들이 먼저 받은 것을 작업하되, 다른 분기 branch 도 보관해 그쪽이 더 길어질 때를 대비한다. 이 동수 tie 는 다음 작업증명이 발견될 때 깨져 한쪽 분기가 더 길어지며, 그러면 다른 분기를 작업하던 노드가 더 긴 분기로 전환한다.

전파한 새로운 거래가 반드시 모든 노드에 도달할 필요는 없다. 많은 노드에 도달하는 한, 블록 안에 곧 들어갈 것이다. 블록 전파는 또한 메시지 누락을 허용한다. 노드가 어떤 블록을 받지 못하면 다음 블록을 받고 누락된 것을 알아차릴 때 그걸 요청한다.

6. 인센티브

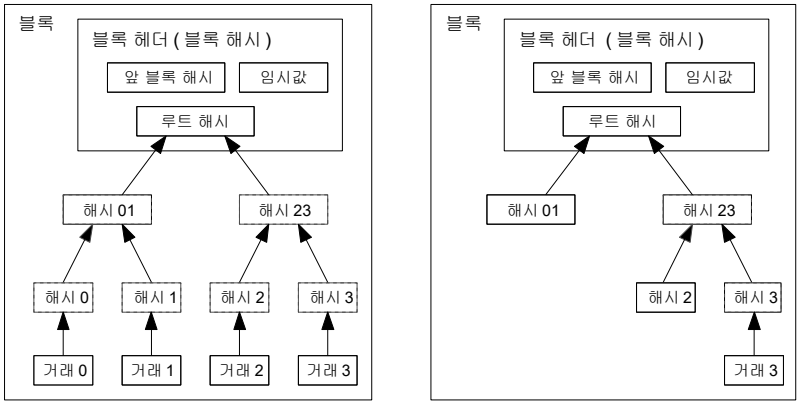
관례상 블록 안의 첫 거래는 해당 블록 창조자가 소유하는 새 화폐를 발행하는 특별한 거래다. 이는 노드가 네트워크를 지원할 인센티브를 더해 주며, 화폐를 발행하는 중앙기관 없이 초기에 화폐를 유통할 방법을 제공한다. 새 화폐를 일정량 꾸준히 추가하는 것은 금 채굴자가 유통하는 금을 추가하기 위해 자원을 소비하는 것과 유사하다. 우리 사례에서는 CPU 사용 시간과 전기가 소비할 자원이다.

이 인센티브는 거래 수수료 transaction fees 로 충당될 수도 있다. 어떤 거래에서 출금액 output value 이 입금액 input value 보다 작다면, 그 차이가 거래를 포함한 블록의 인센티브 금액에 더해질 거래 수수료다. 일단 정해 놓은 수만큼 화폐가 유통되면, 인센티브가 거래 수수료로 전면 전환되고 인플레이션이 완전히 없어질 수 있다.

이 인센티브는 노드가 계속 정직하게 행동하길 유도하는 데 도움을 줄 수 있다. 탐욕스러운 공격자가 모든 정직한 노드보다 더 많은 CPU 파워를 모을 수 있다면, 이것을 써서 자기 결제 금액을 도로 훔치고 사람들을 속일지 새로운 화폐를 만들어 낼지 선택해야 한다. 그는 이 시스템과 자기가 소유한 부의 유효성을 해치기보다, 다른 모두의 몫을 합친 것 이상으로 많은 새 화폐를 그에게 베푸는 이 규칙을 따르는 것이 더 이익임을 깨달아야 한다.

7. 디스크 공간 회수

일단 어떤 화폐의 최근 거래가 충분한 블록 아래에 묻히면, 그전에 지급된 거래는 디스크 공간을 절약하기 위해 폐기될 수 있다. 블록의 해시를 깨지 않고 이를 촉진하기 위해 거래는 머클트리 Merkle Tree[7][2][5]로 해시 처리되며, 그 루트 root 정보만 블록의 해시에 포함된다. 그러면 오래된 블록은 해당 트리의 분기를 쳐냄으로써 작아질 수 있다. 내부 해시는 저장될 필요가 없다.

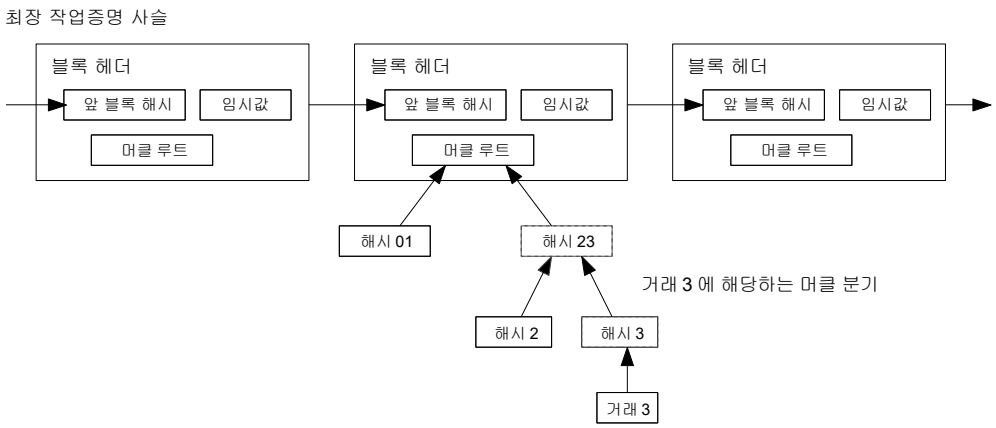


머클트리로 해시 처리된 거래 0~2 번 거래 분기를 쳐낸 블록

거래가 없는 블록 헤더는 약 80 바이트다. 블록이 10 분마다 생성된다고 치면 80 바이트 * 6 * 24 * 365 = 연간 4.2MB 다. 2008 년 기준 일반적으로 2GB RAM 을 탑재한 컴퓨터 시스템이 판매되고 무어의 법칙이 현재 연간 1.2GB 씩 성장을 예측하므로 블록 헤더를 메모리에 보존하더라도 저장공간은 문제가 되지 않는다.

8. 간소화한 결제 검증

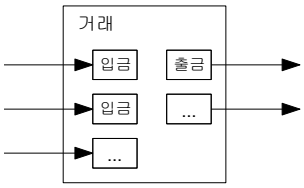
결제 검증은 온전한 네트워크 노드 full network node 를 구동하지 않고도 가능하다. 사용자는 자신이 가장 긴 체인을 가졌다고 확신할 때까지 네트워크 노드를 조회해 얻을 수 있는 가장 긴 작업증명 체인의 블록 헤더 사본을 유지하면서, 해당 거래를 타임스탬프가 찍힌 블록에 연결한 머클 분기를 얻기만 하면 된다. 그는 자신의 거래를 검사할 수는 없지만, 그걸 체인 내 어떤 장소에 연결해 해당 거래를 받아들인 네트워크 노드와 이후 네트워크가 거래를 받아들였음을 확인하고 추가된 블록을 볼 수 있다.



이처럼 정직한 노드가 네트워크를 제어하는 한 검증은 믿을만하지만, 공격자가 네트워크를 장악하면 더 취약해진다. 네트워크 노드가 거래를 자체 검증할 수 있긴 하지만, 간소화한 방법은 공격자가 네트워크를 계속 장악할 수 있는 한 조작한 거래로 속일 수 있다. 이를 방어하기 위한 한 가지 전략은 유효하지 않은 블록을 탐지한 네트워크 노드의 경고를 사용자 소프트웨어가 받아들여 온전한 블록과 경고 대상 거래 정보를 내려받고 이들 간 불일치 inconsistency 를 확인하도록 하는 것이다. 지불금을 자주 받는 기업은 아마도 더 독립적인 보안과 더 빠른 검증을 위해 여전히 자체 노드를 구동하길 원할 것이다.

9. 금액 value 합치기와 나누기

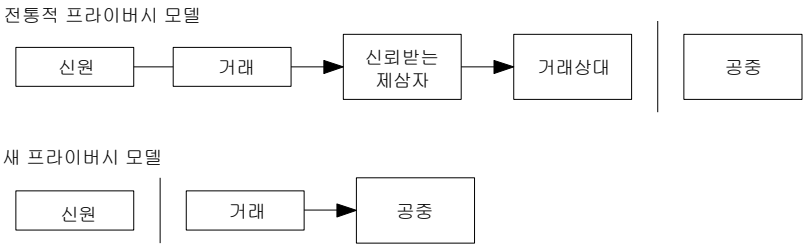
화폐를 개별적으로 다루는 것이 가능하더라도, 모든 푼돈 every cent 을 별도 이체 거래로 만드는 건 무리한 일이다. 금액을 나누고 합칠 수 있도록, 거래 정보는 복수의 입금과 출금을 포함한다. 일반적으로 액수가 더 큰 직전 거래의 단수 입금 또는 액수가 더 작은 거래를 결합한 복수 입금이 있을 것이고, 출금은 많아야 둘인데 하나는 지불금, 하나는 거스름돈이 있다면 이것을 발송인 sender 에게 돌려주는 것이다.



어떤 거래가 여러 거래에 의존하고 이러한 거래가 더 많은 거래에 의존하는 부채꼴 fan-out 구조는 여기서 문제가 되지 않는다는 점에 유의해야 한다. 거래 이력의 완전 독립형 사본을 추출해야 할 필요가 전혀 없다.

10. 프라이버시

전통적인 은행 모델은 관여 당사자 the parties involved 와 신뢰받는 제삼자의 정보 접근을 제한해 일정 수준 프라이버시를 달성한다. 모든 거래를 공개할 필요성에 따라 이 방식을 배제하되, 공개키 익명성을 보존해 다른 장소에서 정보의 흐름을 끊는 것으로 여전히 프라이버시가 유지될 수 있다. 공중 the public 은 누군가가 다른 누군가에게 보내는 금액을 볼 수 있지만, 그 거래에 연결된 누군가에 대한 정보를 볼 수 없다. 이는 증권거래소에서 공개되는 정보 수준과 비슷하게, 개별 거래 시각과 규모를 나타내는 "테이프 tape"는 공개되지만, 그 거래 당사자가 누구인지 알지는 못하는 것이다.



추가 보호 수단으로 거래마다 그것이 어떤 공통된 소유자에게 연결되지 않도록 새로운 키 쌍을 사용해야 한다. 여러 입금이 동일 소유자의 소유임을 부득이 드러내는 다중입금 거래에서 일부 연결은 여전히 불가피하다. 어떤 키 소유자가 드러나면 이 연결 때문에 동일 소유자의 다른 거래까지 드러날 위험이 있다.

11. 계산

우리는 정직한 체인보다 더 빨리 대체 체인을 생성하려는 공격자의 시나리오를 고려한다. 이런 시도가 성공한다고 하더라도, 이 시스템은 허공에서 가치를 만들어 내거나 공격자가 소유한 적도 없는 돈을 얻게 하는 무단 변경을 허용하지 않는다. 노드는 유효하지 않은 거래를 결제로 받아들이지 않으며, 정직한 노드는 그것을 포함하는 블록을 절대 받아들이지 않는다. 공격자가 할 수 있는 시도는 최근 쓴 돈을 회수하기 위해 자신의 거래 중 하나만을 바꾸는 것이다.

정직한 체인과 공격자 체인 간 경주는 이항 임의보행 Binomial Random Walk 으로 특징지을 수 있다. 성공 사건은 정직한 체인이 블록 하나를 연장해 그 우위 lead 를 +1 늘리는 것이고, 실패 사건은 공격자 사슬이 블록 하나를 연장해 그 격차를 -1 만큼 줄이는 것이다.

공격자가 주어진 열세를 따라잡을 확률은 도박꾼의 파산 Gambler's Ruin 문제와 유사하다. 도박꾼이 무제한의 신용을 갖고 열세로 시작해 손익분기 breakeven 에 도달하려는 시도를 잠재적으로 무한한 횟수에 걸쳐 시행한다고 가정해 보자. 우리는 그가 점차 손익분기에 도달할 확률, 다시 말해 공격자가 정직한 사슬을 따라잡을 확률을 다음과 같이 계산할 수 있다 [8]:

p = 정직한 노드가 다음 블록을 찾을 확률
q = 공격자가 다음 블록을 찾을 확률
q_z = 공격자가 블록 z 개 뒤에서 따라잡을 확률

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

$p > q$ 라는 가정에 따라 공격자가 따라잡아야 하는 블록 수가 늘어날수록 따라잡을 수 있는 확률은 기하급수적으로 떨어진다. 주어진 조건상, 그가 초기에 운 좋게 앞으로 치고 나가지 못한다면, 따라잡을 기회는 그가 더 뒤처지면서 보이지 않을 만큼 작아진다.

이제 발송인이 새로운 거래를 변경할 수 없다고 충분히 확신하기 전까지 수취인 recipient 이 얼마나 오래 기다려야 할지 고려해 보자. 우리는 발송인이 수취인에게 돈을 줬다고 한동안 믿게 한 다음 이 거래를 바꿔서 그 돈을 되가져가려는 공격자라고 가정한다. 수취인 receiver 은 이때 경고를 받겠지만, 발송인은 그 시점이 늦기를 바랄 것이다.

수취인이 새로운 키 쌍을 생성하고 서명 직전에 발송인에게 공개키를 준다. 이는 발송인이 운 좋게 충분히 앞설 때까지 계속 그 작업을 수행하고 그 시점에 거래를 실행해 블록의 체인을 미리 준비하지 못하게 방지한다. 일단 이 거래가 보내지면, 이 부정직한 발송인이 자기 거래의 대체 버전을 포함하는 병렬 체인에서 몰래 작업을 시작한다.

수취인은 해당 거래가 블록에 추가되고 그 뒤에 블록 z 개가 연결될 때까지 기다린다. 그는 공격자가 블록을 몰래 처리한 규모를 알지 못하지만, 정직한 블록이 블록당 예상 소요 시간 평균치를 따른다고 가정하면 공격자의 잠재적 작업 진척도는 기댓값을 갖는 푸아송 분포 Poisson distribution 가 될 것이다:

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

현재 공격자가 여전히 따라잡을 수 있는 확률을 얻기 위해, 우리는 공격자가 달성할 수 있는 각 진척도에 대한 푸아송 밀도 Poisson density 와 그가 해당 지점에서 따라잡을 수 있는 확률을 곱한다.

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

해당 분포의 무한한 꼬리를 합산하지 않게 재배열하고...

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

C 코드로 바꿔서...

```
#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;
}
```

결과를 실행하면, z 값에 따라 기하급수적으로 떨어지는 확률을 볼 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% 미만의 P 를 풀면...

```
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 전자 거래용 시스템을 제안했다. 강력한 소유권 제어를 제공하는 디지털 서명으로 만든 화폐라는 평범한 프레임워크로 시작했지만, 이는 이중 지급 방지 수단 없이는 불완전하다. 이를 해결하려고 우리는 작업증명을 사용해 공개 거래 이력을 기록하는 개인 대 개인 네트워크를 제안했다. 정직한 노드가 CPU 파워 대부분을 제어하면 해당 이력을 공격자가 바꾸기는 전산적으로 비현실적인 일이 된다. 이 네트워크의 견고함은 그 비정형적 단순함 unstructured simplicity 에 있다. 노드는 거의 조정 coordination 없이 한 번에 모두 동작한다. 메시지가 어떤 특정 위치로 경로를 지정받아 가는 게 아니라 단지 최선의 노력을 기반으로 전달되면 그만이기 때문에 식별될 필요가 없다. 노드는 마음대로 네트워크를 떠났다가 재합류할 수 있으며, 자신이 빠진 사이에 일어난 일의 증명으로 작업증명 체인을 받아들인다. 이들은 CPU 파워를 사용한 투표로 유효 블록을 확장하는 작업을 통해 승인을 나타내고 무효 블록 작업을 거부해 그것을 부인한다. 필요한 규칙과 인센티브는 어떤 것이든 이 합의 작용 consensus mechanism 을 통해 집행될 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.