비트코인: 개인-대-개인간 전자 화폐 시스템

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org
임민철 번역 ver.0.8
imc@live.co.kr
encodent.com

초록. 순수하게 개인 대 개인간의 전자 화폐 버전은 금융기관을 거치지 않고 한 쪽에서 다른 쪽으로 직접 전달되는 온라인 결제(payments)를 가능케 한다. 전자 서명은 부분적인 솔루션을 제공하지만, 만일 이중지불(double-spending)을 막기 위해 여전히 신뢰받는 제3자를 필요로 한다면 그 주된 이점을 잃게 된다. 우리는 개인 대 개인 네트워크를 사용해 이중지불 문제를 해결하는 솔루션을 제안한다. 이 네트워크는 거래를 해싱해 타임스탬프를 찍어서 해시 기반 작업증명(proof-of-work)을 연결한 사슬로 만들고, 작업증명을 재수행하지 않고서는 변경할 수 없는 기록을 생성한다. 가장 긴 사슬은 목격된 사건의 순서를 증명할 뿐아니라, 그것이 가장 광대한 CPU 파워 풀에서 비롯했음을 증명하기도 한다. CPU 파워 과반을 통제하는 노드가 네트워크를 공격하기 위해 협력하지 않는 한, 이들은 가장 긴 사슬을 만들어내며 공격자를 압도한다. 이 네트워크 스스로는 최소한의 구조만을 요구한다. 메시지는 최선의 노력을 다해 퍼져나가고, 노드는 의사에 따라 네트워크를 떠나거나 최장의 작업증명 사슬을 그들이 없는 사이에 벌어진 일의 증거로 채택해 재합류할 수 있다.

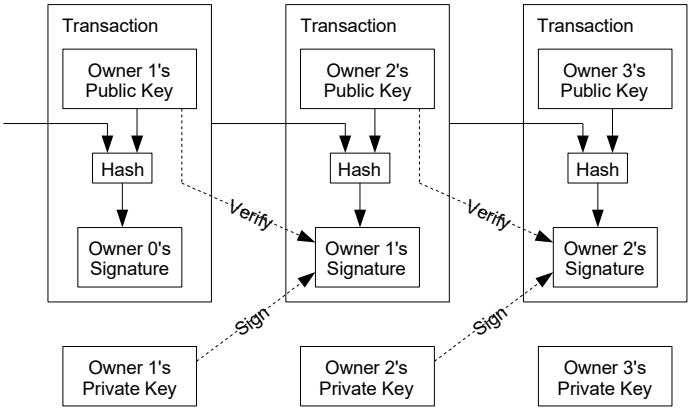
1. 서론

인터넷 기반 상거래는 전자 결제를 처리할 신뢰받는 제3자 역할을 거의 전적으로 금융기관에 의존해 왔다. 이 시스템은 대다수 거래에 충분히 잘 동작하지만, 여전히 신뢰 기반 모델의 태생적 약점을 극복하지 못한다. 금융기관은 분쟁 중재를 피할 수 없기에, 완전한 철회불가 거래는 실제로 가능하지 않다. 중재 비용은 거래 비용을 높이고, 실거래 최소 규모를 제한하며 소소한 일상적 거래 가능성을 가로막고, 철회불가 서비스를 위한 철회불가 결제 능력의 상실에서 더 큰 비용이 발생한다. 철회가능성을 위해 신뢰 확산이 요구된다. 상거래자(Merchants)는 필요 수준보다 더 많은 정보를 위해 그들을 괴롭히는 고객을 경계해야 한다. 사기의 일정 비율은 불가피한 것으로 간주된다. 이런 비용과 결제 불확실성은 직접 물리적 통화(currency)를 사용시 회피될 수 있지만, 신뢰(받는 제3)자 없이 통신채널로 결제를 수행할 방법은 존재하지 않는다.

필요한 것은 신뢰 대신 암호학적 증명(cryptographic proof)에 기반해, 거래 의사가 있는 두 당사자가 신뢰받는 제3자를 필요로 하지 않고 서로 직접 거래하게 해주는 전자 화폐 시스템이다. 전산적으로 철회가 불가능한 거래는 사기로부터 판매자를 보호하고, 통상적인 제3자 예치(escrow) 방법은 구매자를 보호하기 위해 쉽게 구현될 수 있다. 이 논문에서, 우리는 시간순으로 거래의 전산적 증거를 생성하는 개인 대 개인간 분산 타임스탬프 서버를 사용한 이중지불 문제의 솔루션을 제안한다. 이 시스템은 정직한 노드가 공격을 하려고 협력하는 노드 그룹보다 총체적으로 더 많은 CPU 파워를 통제하는 한 보안상 안전하다.

2. 거래

우리는 디지털 서명의 사슬로써 전자적 화폐(electronic coin)을 정의했다. 각 소유자는 화폐를 송금할 때 먼저번 거래 내역 및 다음 소유자 공개키의 해시값에 전자적으로 서명을 하고 이 정보를 이 화폐의 끝에 첨가한다. 수금자(payee)는 소유권(ownership)의 사슬을 검증하기 위해 해당 서명을 검증할 수 있다.

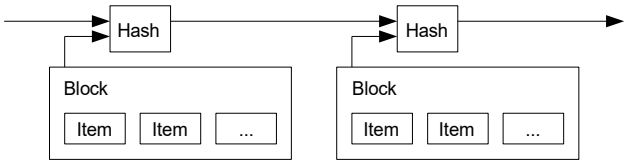


이 과정상 문제는 수금자가 소유자 가운데 누군가가 화폐를 이중지불하지 않았는지 검증할 수 없다는 점이다. 통상적인 솔루션은 신뢰받는 중앙통제기관(trusted central authority)이나 조폐국(mint)을 두고 모든 거래에서 이중지불 여부를 점검하는 것이다. 거래를 마칠 때마다 이 화폐는 조폐국으로 회수돼 새로운 화폐로 발행돼야하고, 조폐국에서 직접 발행된 화폐만이 이중지불되지 않았다고 신뢰받는다. 이 솔루션의 문제는 전체 통화체계(the entire money system)의 운명이 은행처럼 모든 거래가 거쳐가야하는 조폐국 운영 회사에 달려 있다는 점이다.

우리에게는 먼저번 소유자가 이전에 어떤 거래에도 서명하지 않았음을 수금자에게 알릴 수단이 필요하다. 이런 목적에서 우리는 가장 앞선 거래 하나를 인정하고, 이후 이중지불 시도에는 신경쓰지 않는다. 그런 (이중지불된) 거래가 없음을 확인할 유일한 방법은 모든 거래를 인식하고 있는 것뿐이다. 조폐국 기반 모델에서, 조폐국은 모든 거래를 인식했고 최초로 받은 거래를 (승인 대상으로) 결정했다. 신뢰받는 (제3)자 없이 이 방식을 실현하려면, 거래는 공개적으로 알려져야 하고[1], 노드들이 거래를 받는 순서의 단일 이력에 합의하는 시스템이 필요하다. 수금자는 매 거래시 그게 첫 수금이라는 것에 노드 다수가 동의했음을 증명해야 한다.

3. 타임스탬프 서버

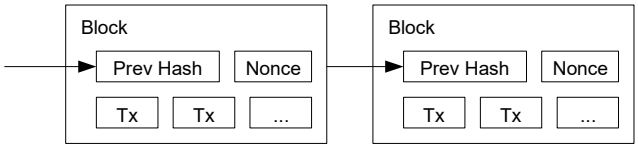
우리가 제안하는 솔루션은 타임스탬프 서버로 시작한다. 타임스탬프 서버는 타임스탬프가 찍힌 항목 블록의 해시를 가져가 그 해시를 신문이나 유즈넷 게시물[2-5]처럼 널리 배포하는 식으로 작동한다. 이 타임스탬프는 그 데이터가, 명백히, 해시(과정)에 들어가기 위해 해당 시각부터 존재했음을 증명한다. 각 타임스탬프는 그 해시 안에 먼저번 타임스탬프를 포함하고, 그에 앞선 것들을 하나씩 연장하는(reinforcing the ones) 타임스탬프가 찍힌 사슬을 생성한다.



4. 작업증명

개인 대 개인 기반으로 분산 타임스탬프 서버를 구현하기 위해 우리는 신문이나 유즈넷 게시물 대신 애덤 백의 해시캐시(Adam Back's Hashcash)와 유사한 작업증명 시스템[6]을 사용할 필요가 있다. 작업증명은 0 비트(zero bits) 여러 개로 시작하는, SHA-256같은 해시된 값 스캐닝을 수반한다. 이에 필요한 평균 작업은 요구되는 0비트 수에 따라 지수적이며 단일 해시를 실행함으로써 확인될 수 있다.

타임스탬프 네트워크용으로, 우리는 블록의 해시에 필요한 0비트를 주는 값이 발견될 때까지 블록 안에 임시값을 증분(incrementing a nonce)하는 것으로 작업증명을 구현했다. 작업증명을 만족하는 데 한 번 CPU 동작(CPU effort)이 쓰였으면, 그 블록은 그 작업을 재수행하지 않고는 변경될 수 없다. 거기에 나중 블록이 연결되는만큼, 블록을 변경하기 위한 재수행 작업은 그 뒤 모든 블록까지 포함한다.



작업증명은 다수결(majority decision making)의 대표성 문제도 해결한다. IP주소당 1표에 기반한 다수 조건이라면 누구든지 많은 IP를 할당할 수 있는 이에 의해 장악될(subverted) 수 있다. 작업증명은 기본적으로 CPU당 1표다. 다수의사는 최다 작업증명 동작이 투입된 가장 긴 사슬로 대표된다. 만일 다수 CPU 파워가 정직한 노드에 의해 통제된다면, 가장 정직한 사슬이 가장 빠르게 늘어나 다른 경쟁 사슬을 압도(outpace)할 것이다. 과거 블록을 변경하려면 공격자는 그 블록과 그 뒤를 잇는 모든 블록의 작업증명을 재수행해야 하고 그러면서 가장 정직한 노드들의 작업을 따라잡아 앞질러야 한다. 뒤에서 우리는 이어지는 블록이 추가될수록 더 느린 공격자가 따라잡을 확률이 지수적으로 감소함을 보이겠다.

시간이 지날수록 노드를 구동하는 하드웨어의 속도 증가와 변화하는 관여도(interest)를 보상하기 위해, 작업증명 난도(difficulty)는 시간당 평균 블록 수에 따른 평균 목표치를 조정해 결정된다. 그것들(블록)이 너무 빨리 생성되면 난도가 증가한다.

5. 네트워크

네트워크를 실행하는 단계는 다음과 같다:

1) 새로운 거래가 모든 노드에 브로드캐스트된다.
2) 각 노드가 새로운 거래를 블록에 수집한다.
3) 각 노드가 그 블록에 맞는 난도의 작업증명 찾기에 나선다.
4) 노드가 작업증명을 찾은 시점에, 그는 모든 노드에게 그 블록을 브로드캐스트한다.
5) 노드는 모든 거래가 유효하며 아직 지불되지 않았다는 조건에 맞을 경우에만 그 블록을 승인한다.
6) 노드는 블록 승인을 표현하기 위해 먼젓번 해시로 승인된 블록의 해시를 사용해 사슬 안에 다음 블록을 생성한다

노드는 항상 가장 긴 사슬을 정확한 것으로 간주하고 그걸 잇는 작업을 지속한다. 만일 두 노드가 동시에 다음 블록의 상이한 버전을 브로드캐스트하면, 어떤 노드는 그 중 하나 또는 다른 것을 먼저 받을 수 있다. 이 경우 그들이 먼저 받은 것을 작업하지만, 다른 branch도 저장해 그게 더 길어질 경우에 대비한다. 이 동점(tie)은 다음 작업증명이 발견되면서 깨지고 한 branch가 더 길어지며; 다른 branch를 작업하던 노드는 그 뒤 (작업 대상을) 더 긴 것으로 전환한다.

새로운 거래 브로드캐스트가 반드시 모든 노드에게 도달할 필요는 없다. 브로드캐스트는 많은 노드에 도달하는만큼 곧 한 블록 안에 들어간다. 블록 브로드캐스트는 또한 누락된 메시지에 내성을 갖는다. 만일 노드가 블록을 받지 못하면 그는 다음 블록을 받을 때 누락된 것을 알아차리고 그걸 요청한다.

6. 인센티브

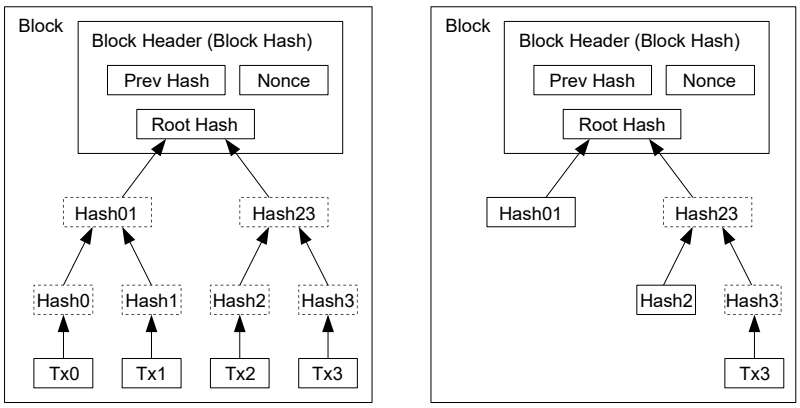
관례상 블록 안의 첫 거래는 블록을 만든 이의 몫이 될 새 화폐로 시작하는 특별한 거래다. 이는 화폐를 발행하는 중앙기관 없이, 노드가 네트워크를 지원할 인센티브를 더해 주며 초기에 발행한 화폐를 유통할 방법을 제공한다. 새 화폐 일정량을 꾸준히 추가하는 것은 금 채굴자가 유통하는 금을 추가하기 위해 자원을 소비하는 것과 유사하다. 우리의 경우 소비되는 것은 CPU 시간과 전기다.

이 인센티브는 또 거래 수수료(transaction fees) 재원이 될 수 있다. 만일 거래에서 도출된 가치가 투입된 가치보다 작다면, 그 차이가 거래를 포함한 블록의 인센티브 가치에 더해질 거래 수수료다. 한 번 선결된 화폐 수가 유통되면, 이 인센티브는 모두 거래 수수료로 전환돼 인플레이션에서 완전히 자유로워질 수 있다.

이 인센티브 노드들이 계속 정직하길 유도하는 데 도움을 줄 수 있다. 만일 탐욕스러운 공격자가 모든 정직한 노드보다 더 많은 CPU 파워를 모을 수 있다면, 그는 그걸 자신의 결제를 도로 훔쳐 사람들을 속이는 데 쓰는 것, 또는 새로운 화폐를 만들어내는 데 쓰는 것 사이에서 선택해야 한다. 그는 규칙대로 움직이는 게 더 이득임을 알게 돼 있는데, 규칙은 그에게 다른 모두의 몫을 합친 것보다, 시스템과 그가 보유한 부의 유효성을 해치는 것보다 더 많은 새 화폐를 베푼다.

7. 디스크 공간 회수

화폐 안의 최종 거래가 충분한 블록에 묻히면, 그 전에 지불된 거래는 디스크 공간을 절약하기 위해 폐기될 수 있다. 블록의 해시를 깨지 않고 이걸 촉진하기 위해, 거래는 그 블록의 해시 안에 포함된 root만으로, 머클트리(Merkle Tree)[7][2][5]로 해시된다. 그러면 오래된 블록은 트리의 가지branches를 걷어냄으로써 작아질 수 있다. 내부 해시는 저장될 필요가 없다.

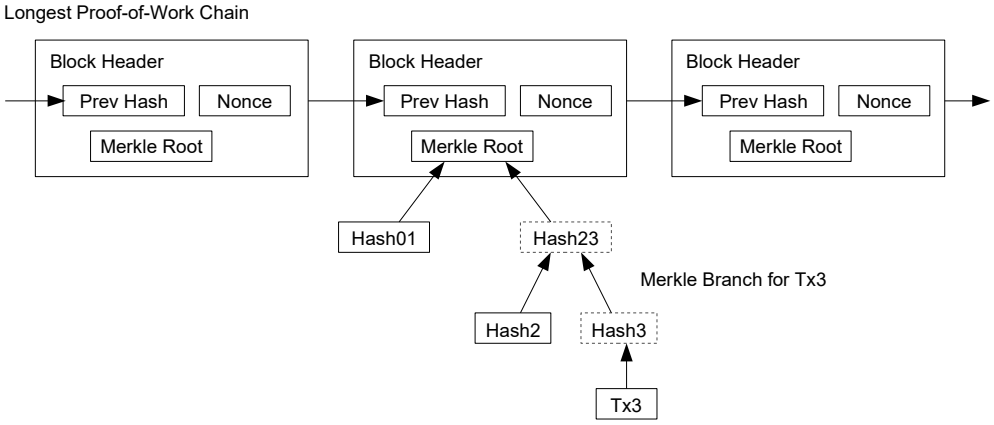


Transactions Hashed in a Merkle Tree　　After Pruning Tx0-2 from the Block

거래가 없는 블록 헤더는 약 80바이트가 된다. 블록이 10분마다 만들어진다고 가정하면, 80바이트 * 6 * 24 * 365 = 연간 4.2MB다. 2008년부터 통상적으로 판매되는 RAM 2GB짜리 컴퓨터 시스템, 그리고 현재 연간 1.2GB씩 성장을 예측하는 무어의 법칙으로 보면, 만일 블록 헤더가 메모리에 보존돼야 한다더라도 저장 공간은 문제가 되지 않는다.

8. 간소화한 결제 검증

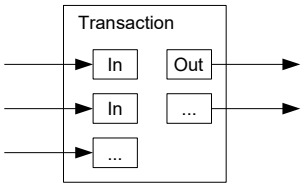
결제 검증은 전체 네트워크 노드를 구동하지 않고도 가능하다. 사용자는 그가 가장 긴 사슬을 가졌다고 확신할 때까지 네트워크 노드를 조회하게 해 주면서, 해당 거래를 타임스탬프가 찍힌 블록으로 연결한 머클 branch를 얻게 해 줄, 가장 긴 작업증명 사슬의 블록 헤더 사본을 갖고 있기만 하면 된다. 그는 자신의 거래를 검사할 수는 없지만 그걸 사슬 안의 장소로 연결함으로써, 네트워크 노드가 그걸 받아들인 것과, 이후 그게 받아들여졌음을 확인한 뒤 추가된 블록을 볼 수 있다.



이처럼, 네트워크를 제어하는 노드가 정직한 한 검증은 믿을만하지만, 만일 네트워크가 공격자에 의해 과점된다면 더 취약해진다. 네트워크 노드가 거래를 자체 검증할 수 있긴 하지만, 간소화한 방법은 공격자가 네트워크를 계속 과점할 수 있는 한 그가 조작한 거래에 의해 기만당할 수 있다. 이를 방어하기 위한 한 가지 전략은 네트워크 노드가 유효하지 않은 블록을 탐지시 그로부터 경고를 받아, 사용자의 소프트웨어가 그 온전한 블록을 내려받게 하고 경고된 거래에 그 불일치(inconsistency)를 확인하도록 하는 것이다. 수금이 빈번한 비즈니스는 아마도 여전히 더 독립적인 보안과 더 빠른 검증을 위해 그들의 자체 노드를 구동하길 원할 것이다.

9. 가치 합치기와 나누기

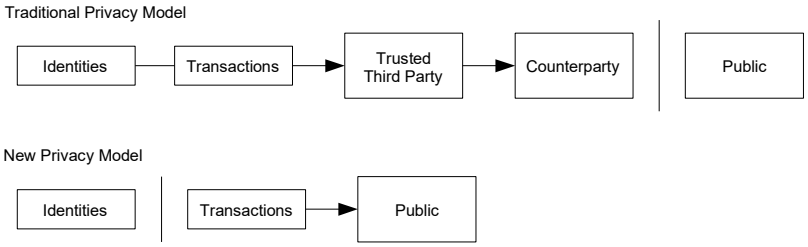
화폐를 독립적으로 다루는 것은 가능하더라도, 송금에 모든 푼돈(every cent)을 별도 거래로 만드는 건 무리한 일이다. 가치를 나누고 합칠 수 있도록, 거래는 다중 입출력을 포함한다. 일반적으로 입력은 더 큰 먼젓번 거래의 단일 입력 또는 더 작은 양을 결합한 다중 입력이며, 출력은 지불용 출력 하나와 만일 있다면 송금자(sender)에게 돌려줄 거스름돈 출력 하나, 이렇게 많아야 둘이다.



펼친 부채꼴(fan-out)처럼, 거래가 여러 거래에 의존하고 그 여러 거래가 더 많은 거래에 의존하는 것은 문제가 되지 않는다는 것에 주목해야 한다. 완전 독립된(standalone) 거래 내역 사본을 추출해야 할 필요는 전혀 없다.

10. 프라이버시

전통적인 은행 모델은 참여 당사자(the parties involved)와 신뢰받는 제3자에게 정보 접근을 제한함으로써 일정 수준 프라이버시를 달성한다. 이 방법은 모든 거래를 공개할 필요성에 따라 배제되지만, 공개키 익명성을 보존해 다른 장소에서 정보의 흐름을 끊는 걸로 여전히 프라이버시가 보장될 수 있다. 대중은 누군가가 다른 누군가에게 보내는 금액을 볼 수 있지만, 그 거래에 연결된 누군가에 대한 정보는 볼 수 없다. 이는 증권거래소에서 공개되는 정보 수준과 비슷하게, 개별 거래 시각과 규모를 나타내는 "테이프(tape)"는 공개되지만, 그 당사자가 누구인지 알지는 못하는 것이다.



추가 방화벽으로, 새로운 키 쌍이 각 거래마다 공통된 소유자와 연결을 유지하도록 사용돼야 한다. 어떤 연결은 다중입력 거래시 여전히 불가피하게 그 입력이 동일 소유자의 것임을 필연적으로 드러낸다. 만일 키 소유자가 공개되면, 연결은 다른 거래도 동일 소유자에게 속하는 것임을 노출할 위험이 있다.

11. 계산

정직한 사슬보다 더 빨리 대체 사슬을 만들어내려는 공격자의 시나리오를 고려해 보자. 만일 이런 시도가 성공한다 하더라도, 그게 아무것도 없는 곳에서 가치를 만들어내거나 공격자가 소유한 적도 없는 돈을 얻게 만드는 식으로 이 시스템을 무단 변경되도록 허용하진 않는다. 노드는 유효하지 않은 거래를 결제로 받아들이지 않으며, 정직한 노드는 그걸 포함하는 블록을 절대 받아들이지 않는다. 공격자는 오로지 자신의 거래에서 그가 최근 지출한 돈을 거둬들이는 것 하나만을 바꿀 수 있다.

정직한 사슬과 공격자 사슬간의 경주는 이항임의보행(Binomial Random Walk)으로 특징지을 수 있다. 성공 이벤트는 정직한 사슬이 그 우위(lead)를 +1만큼 늘리는 블록 하나를 연장한 것이고, 실패 이벤트는 공격자 사슬이 그 격차를 -1만큼 좁히는 블록 하나를 연장한 것이다.

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

p = 정직한 노드가 다음 블록을 발견할 확률
q = 공격자가 다음 블록을 발견할 확률
q_z = 공격자가 z블록 뒤에서부터 따라잡을 확률

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

p > q라 가정하면, 공격자가 따라잡아야 하는 블록 수가 늘어날수록 그럴 수 있는 확률은 지수적으로 감소한다. 그에게 주어진 조건상, 만일 그가 초기에 운좋게 앞으로 치고나가지 못한다면, 그의 기회는 그가 뒤쳐질수록 보이지 않을만큼 작아진다.

이제 발신자(sender)가 새로운 거래를 변경할 수 없다고 충분히 확신하기 전까지 수신자(recipient)가 얼마나 오래 기다려야할지 고려해 보자. 발신자가 자신이 지불했음을 수신자로하여금 한동안 믿게 한 다음, 시간이 좀 지나서 지불금을 회수하도록 만들려는 공격자라고 가정한다. 해당 수신자(receiver)는 그런 일이 생길 때 경고를 받겠지만, 발신자는 그게 늦기를 바란다.

수신자는 새로운 키 쌍을 생성하고 서명 직전에 발신자에게 공개키를 준다. 이는 발신자가 운좋게 충분히 앞설 때까지 계속 그 작업을 수행함으로써 미리 블록의 사슬을 준비하지 못하게 방지하고, 그 시점에 거래를 실행한다. 거래가 한 번 발신되면, 이 부정직한(dishonest) 발신자는 몰래 그의 거래를 대신할 버전으로 사슬 작업을 병행하기 시작한다.

수신자는 해당 거래가 블록에 추가되고 그 뒤에 z블록이 연결될 때까지 기다린다. 그는 공격자가 진척시킨 규모를 알지 못하지만, 정직한 블록이 예상되는 블록당 시간 평균치를 따른다고 가정하면, 공격자의 잠재적 진척도는 기대값을 갖는 푸아송 분포(Poisson distribution)가 될 것이다:

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

현재 공격자가 여전히 따라잡을 수 있는 확률을 얻기 위해, 그가 해당 시점부터 따라잡을 수 있는 확률로 만들어낼 각 진척 규모별 푸아송 밀도를 곱한다:

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

분포의 무한꼬리 합산을 피하도록 정리하고...

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

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, z에 따라 지수적으로 감소하는 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 P를 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 전자거래용 시스템을 제안했다. 강력한 소유권 통제를 제공하는 디지털 서명으로 만든 화폐(coins made from digital signatures)의 유력한 프레임워크로 시작했지만, 이는 이중지불 방지수단 없이는 불완전하다. 이를 해결하기 위해, 우리는 정직한 노드가 CPU 파워 대부분을 제어한다면 공격자가 전산적으로 변경하기가 금세 비현실적이 되는 작업증명을 사용해 공개된 거래 이력을 기록하는 개인 대 개인 네트워크를 제안했다. 이 네트워크의 견고함은 그 정형화하지않은 단순성(unstructured simplicity)에 있다. 노드는 거의 조정(coordination)없이 한 번에 모두 동작한다. 이들은 메시지가 경로를 지정받아 어떤 특정 위치로 가는 게 아니라 단지 최선의 노력을 다해 전달되면 그만이기 때문에 식별될 필요가 없다. 노드는 의지에 따라, 네트워크를 떠났다가 그가 없는 동안 벌어진 일의 증거로 작업증명 사슬을 받아들여 재합류할 수 있다. 이들은 CPU 파워를 사용한 투표로, 유효한 블록을 연장하는 작업을 통해 그걸 승인했음을 나타내고 유효하지 않은 블록에 대한 작업을 거부함으로써 그걸 기각한다. 어떤 필요 규칙과 유인이든 이 합의 작용(consensus mechanism)을 통해 집행될 수 있다.

References

[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.