비트코인: 개인간 전자화폐 시스템

사토시 나카모토
satoshin@gmx.com
www.bitcoin.org

초록. 순 개인과 개인간의 전자화폐는 한 집단에서 다른 곳으로 금융기관을 거치지 않고 직접 온라인 지불을 가능하게 할 것이다. 디지털 서명 기술이 일부 해결해주지만, 믿을 수 있는 제 3자가 이중지불을 방지해야 한다면 그 주요한 장점은 사라지게 된다. 우리는 이 논문에서 P2P 네트워크를 이용한 이중지불 문제의 해결 방법을 제안하고자 한다. 계속 진행되고 있는 암호화 기반 작업증명 과정의 연쇄 상에서 네트워크 시간 및 거래를 암호화하여 기록을 생성하게 되면 작업증명 과정을 되풀이하지 않는 한 바꿀 수 없게 된다. 가장 긴 체인은 각 사건 순서를 입증해주기도 하며, 가장 많은 컴퓨팅 파워가 입증했다는 뜻이기도 하다. 노드들에 의해 제어되는 컴퓨터 전력의 과반수가 협력하여 네트워크를 공격하지 않는 한, 그들은 가장 긴 체인을 생성하며 네트워크 공격자를 능가하게 될 것이다. 이러한 네트워크는 최소한의 구조를 필요로 한다. 각 노드들은 자발적으로 그 네트워크를 떠나거나 다시 합류할 수 있고, 어떤 일이 벌어졌는지에 대한 입증으로 가장 긴 작업증명 체인을 받아들이는 노드들의 메시지가 최대한 공유된다.

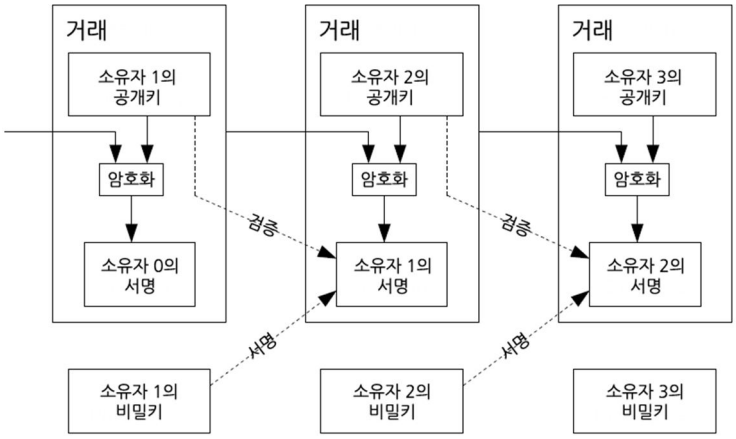
1. 서론

인터넷에서의 상거래는 거의 금융기관을 제 3자 신용기관으로 하는 전자지불 방식에 전적으로 의존하게 되었다. 대부분의 거래에 시스템은 충분히 작동하고 있지만, 여전히 신용기반 모델이라는 내재적인 약점을 갖고 있다. 완전히 취소 가능한 거래는 사실상 불가능한데, 금융기관은 거래 상의 분쟁을 중재하는 일을 피할 수 없기 때문이다. 이러한 중재 비용은 결국 거래 수수료를 올리고, 실질적인 최소 거래금액을 제한하여 소액 거래의 가능성을 막는데다가, 회수가 불가능한 서비스에까지도 번복 가능한 지불을 하게 만들어 더 많은 비용을 발생시킨다. 즉, 지불 번복을 위해 더 많은 신용을 요구하게 된다. 상업자들은 불필요한 더 많은 정보를 요구하여 고객을 귀찮게 만들고 경계하게 된다. 일정한 비율로 가짜 지불이 되는 것은 불가피한 현실이다. 이러한 비용과 지불의 불확실성은 사람이 직접 물리적으로 화폐를 지불하여 피할 수 있으나, 신용기관 없이 통신상으로 지불하는 방법은 존재하지 않는다.

이러한 문제는 바로 신용보다는 암호화 기술에 기반한 전자지불 시스템을 이용하여 자발적인 두 거래자가 제 3자인 신용기관 없이도 직접적인 거래를 가능하게 만든다. 전산적으로 번복이 불가능한 송금은 판매자를 가짜 지불로부터 보호할 수 있으며, 구매자는 일반 에스크로 방식을 통해 보호받을 수 있다. 이 논문에서 우리는 거래들의 시간 순서를 전산적으로 입증하게 만들도록 하는 P2P 분산 네트워크 기반 타임스탬프 서버를 이용하여 이중지불 문제를 방지하는 해법을 제안하고자 한다. 이 시스템은 악의적으로 협력하는 노드 그룹보다 정직한 노드들이 더 많은 컴퓨팅 파워를 총체적으로 제어하는 한 안전하다.

2. 거래

우리는 전자 화폐를 디지털 서명의 연속으로 정의한다. 각 암호키 소유자들은 그 전까지의 거래 내역에 다음 소유자의 공개키를 덧붙인 뒤에 자신의 비밀키로 암호화하는 디지털 서명을 하고 넘긴다. 돈을 받는 사람은 서명 소유자들의 체인과, 서명들을 검증할 수 있다.

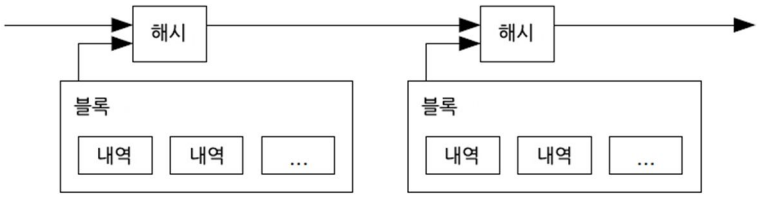


문제의 과정은 돈을 받는 사람은 소유자들 중 한 명이 이중지불을 하지 않았는지 검증할 수가 없는 상황에서 발생한다. 공통적인 해법은 각 거래가 이중지불이 되었는지 신용해주는 중앙기관을 도입하는 것이다. 각 거래 후에, 그 화폐는 다시 새로운 화폐로 찍어내기 위해 중앙기관으로 회수되어야 하고, 이중지불이 아니란 걸 믿을 수 있도록 중앙기관에서만 직접 화폐를 발행하여 쓰도록 한다. 이러한 방법의 문제는 화폐 시스템 전체가 바로 은행 같은 중앙기관에 모든 거래 내역이 거쳐가도록 하는 방법에 의존하게 된다는 것이다.

결국 돈을 받는 사람이 이전 소유자가 그 전에도 어떤 거래에도 서명을 하지 않았는지를 확인할 방법이 필요하다. 그러려면 가장 먼저 일어난 거래 내역을 찾기만 해도 그 이후에 이중지불을 시도했는지 확인할 필요가 없게 된다. 거래 내역이 하나라도 비어있는지 확인하는 유일한 방법은 모든 거래 내역을 살펴보는 것이다. 바로 찍어낸 화폐를 기반으로 한 모델에서는 모든 거래를 확인하고 어느 것이 먼저 이뤄졌는지를 결정하면 된다. 신용기관을 통하지 않고도 이런 방법을 가능하게 하기 위해서는, 모든 거래가 공개적으로 알려져야 하고[1], 참여자들이 시간 순서에 따라 단일 거래내역으로 수용하는 시스템이 필요하다. 돈을 받는 사람은 매 거래 시마다, 과반수 이상의 노드들이 최초의 거래라고 인정해주는 시간 증명이 필요하게 된다.

3. 타임스탬프 서버

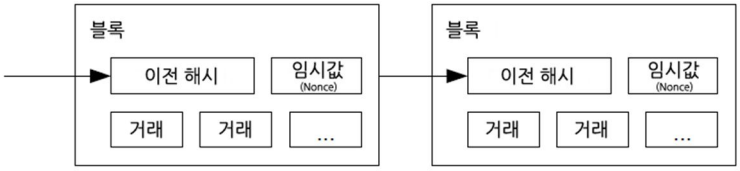
우리가 제안하는 해결 방법은 타임스탬프 서버에서 시작된다. 타임스탬프 서버는 시간 내역이 기록된 항목들의 블록 해시를 취합하고, 신문이나 유즈넷 포스트처럼 그 해시를 널리 발행하는 역할을 한다[2~5]. 이러한 타임스탬프 내역은 해시에 포함될 수 있도록 그 시간에 데이터가 명백히 존재했다는 것을 입증한다. 각 타임스탬프 내역은 이전 타임스탬프로부터 받은 해시 내역을 포함시킴으로써 보강하는 체인을 형성한다.



4. 작업증명

P2P를 기반으로 하는 분산 네트워크 타임스탬프 서버를 구현하기 위해서는 신문이나 유즈넷 포스트 대신 Adam Back's Hashcash[6]와 비슷한 작업증명 시스템을 이용할 필요가 있다. 작업증명에는 SHA-256과 같은 알고리즘으로 다수의 0비트들로 시작되는 암호화 해시값을 찾는 과정이 포함된다. 평균적으로 이러한 작업에 드는 시간은 연속되는 0비트의 요구 개수에 따라 지수적으로 증가하며, 암호화 해시를 한번 수행하는 것으로 확인할 수 있다.

타임스탬프 네트워크에서는 작업증명의 방법으로 블록 해시 결과가 0비트들을 갖도록 하는 해시값을 찾을 때까지 블록에 임시값(nonce)을 증가시켜가는 과정을 구현한다. CPU가 노력한 결과가 한번 작업증명 조건에 도달하게 되면, 그 블록은 다시 과정을 번복하지 않는 한 고정된다. 그 다음 블록들이 체인을 형성함으로써, 하나의 블록을 변경하기 위해서는 그 블록을 포함한 다음 모든 블록들에 대해 작업증명 과정을 다시 수행해야 하게 된다.



작업증명은 또한 다수결에 의한 의사결정 과정에서 대표자를 결정하는 문제를 해결한다. 다수가 한 IP주소당 한번의 투표를 할 수 있는 시스템 기반으로 결정된다면, IP주소를 많이 확보하는 방법으로 누구나 시스템을 뒤엎어버릴 수 있다. 그러나 작업증명은 본질적으로 한 개의 CPU당 한 번의 투표를 하는 구조이다. 다수의 결정은 가장 긴 체인을 나타내며, 이는 가장 많은 작업증명에 노력이 투입된 것이 된다. 컴퓨팅 파워의 과반수가 정직한 노드들에 의해 제어되고 있다면, 정직한 체인이 가장 빠르게 늘어나, 경쟁 체인을 압도하게 될 것이다. *(역자 주: 실제로는 Emin Gün Sirer, Ittay Eyal에 의해 전체 컴퓨팅 파워의 과반수인 50% 이상이 아니라 25% 이상만 점유해도 된다고 밝혀졌는데, 현재 전체 순위 1~2위의 마이닝 풀 집단은 25% 이상에 달하고 있음.)* 과거의 블록을 수정하기 위해서는 공격자는 수정할 블록과 그 이후에 이어진 모든 블록에 대해 작업증명 과정을 번복한 다음에 이어서 다른 정직한 노드들이 이루고 있는 체인보다 더 빠른 속도로 따라잡아 추월해야 한다. 느린 공격자의 추격 가능성은 블록들이 이어서 추가될 수록 지수적으로 감소하는 것에 대해 뒤에서 언급하기로 한다.

시간이 흐름에 따라 하드웨어 속도 증가와 노드들의 참여도 증가율을 보상하기 위해서, 작업증명의 난이도는 시간당 평균 블록 생성 수를 기준으로 하는 이동평균을 타깃으로 결정한다. 블록이 너무 빠르게 생성되면 난이도는 급증한다.

5. 네트워크

네트워크의 동작은 다음과 같은 과정으로 이루어진다:

1) 새로운 거래 내역이 모든 노드에 알려진다.
2) 각 노드들은 새로운 거래 내역을 블록에 취합한다.
3) 각 노드들은 그 블록에 대한 작업증명을 찾는 과정을 수행한다.
4) 어떤 노드가 작업증명을 성공적으로 수행했을 때, 모든 노드에게 그 블록을 전송한다.
5) 노드들은 그 블록이 모든 거래가 이전에 쓰이지 않고 유효한 경우에만 승인한다.
6) 노드들은 자신이 승인한 블록의 해시를 이전 해시로 사용하여 다음 블록을 생성하는 과정을 통해 그 블록이 승인되었다는 의사를 나타낸다.

노드들은 항상 가장 긴 체인을 옳은 것으로 간주하며 그 체인이 계속 확장하도록 작업을 수행한다.

만약 두 개의 노드가 서로 다른 버전의 다음 블록을 동시에 알리게 될 경우, 어떤 노드들은 둘 중 하나를 먼저 전달받게 된다. 이러한 경우 각 노드들은 자신이 먼저 받은 블록에 대해 작업을 수행하지만, 체인의 다른 갈래도 더 길어질 경우에 대비하여 저장해둔다. 체인의 어느 한쪽 갈래가 더 길게 생성되는 작업증명이 알려지면 체인 갈래의 길이는 더 이상 대등하지 않게 되고, 각 노드들은 체인이 더 긴 갈래로 작업을 전환한다.

새로운 거래 내역 알림이 꼭 모든 노드에까지 전달될 필요는 없으며, 많은 노드에 전달될수록 더 빨리 블록에 포함될 것이다. 블록 알림은 또한 누락되는 경우에도 취약하지 않다. 만약 한 노드가 블록을 받지 못했을 경우, 다음 블록을 받고 하나가 빠졌음을 알아차려 다시 요청해 받을 것이다.

6. 보상

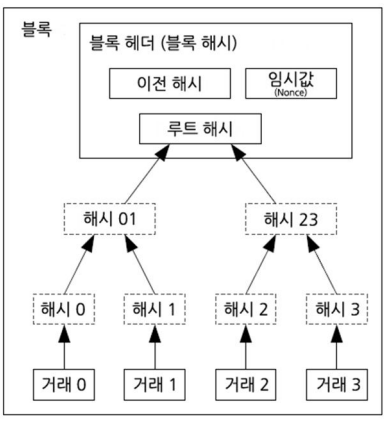
블록의 첫번째 거래 내역은 약속에 의해 최초 블록 생성자에게 새로운 돈을 소유할 수 있게 해주는 특별한 거래가 된다. 이렇게 하면 돈을 발행하는 중앙기관 없이도 네트워크를 구성하는 모든 노드들에게 보상을 지급하고, 유통될 돈을 처음에 배분하는 방법이 된다. 지속적인 일정량의 새 돈을 추가하는 건 금을 유통할 수 있게 광부들이 자원을 쏟는 것과 유사하다. 이 경우에는 컴퓨팅 자원과 전력이 소비된다.

보상에는 또한 거래 수수료가 될 수도 있다. 거래 내역에서 출력되는 돈이 입력되는 돈보다 적다면, 그 차액은 수수료처럼 작용하여 그 거래 내역을 포함하는 블록 생성의 보상 가치로 추가된다. 정해진 총량의 돈이 유통된 다음부터는, 보상은 거래 수수료만으로 이뤄지며 인플레이션으로부터 완전히 자유롭게 된다.

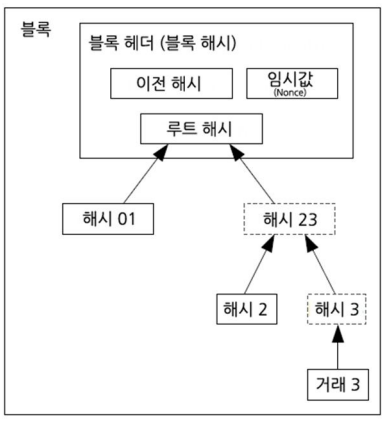
이러한 보상 시스템은 각 노드들이 정직하게 참여를 유지할 수 있도록 한다. 욕심을 낸 공격자가 다른 정직한 전체 노드들보다 더 많은 컴퓨팅 파워를 만들어낼 수 있다면, 그는 아마도 다른 사람들로부터 지불을 철회하여 돈을 사취하거나 새로운 돈을 생성하려 해야 할 것이다. 하지만 그런 식으로 다른 사람들이 엮이지 않고 시스템을 약화시켜 자신의 부를 축적하는 이기적인 방법보다는 약속대로 정직하게 시스템에 참여하는 것이 더 이득이라는 걸 알게 될 것이다.

7. 저장 공간 재확보

최근 거래 내역에 있던 돈이 충분히 많은 블록에 의해 묻히게 되면, 지나간 거래 내역은 저장 공간 확보를 위해 버려져도 된다. 블록 해시를 다시 헤집지 않고도 이를 수월하게 하기 위해서는, 거래 내역은 머클트리(Merkle Tree) 구조로 해시가 되며[7][2][5], 머클트리 구조의 루트 부분만 블록 해시에 포함되어야 한다. 오래된 블록은 트리 구조에서 가지를 쳐냄으로써 더 작아지게 되며, 하위 해시는 저장할 필요가 없게 된다.



머클트리 구조로 해시된 거래 내역들

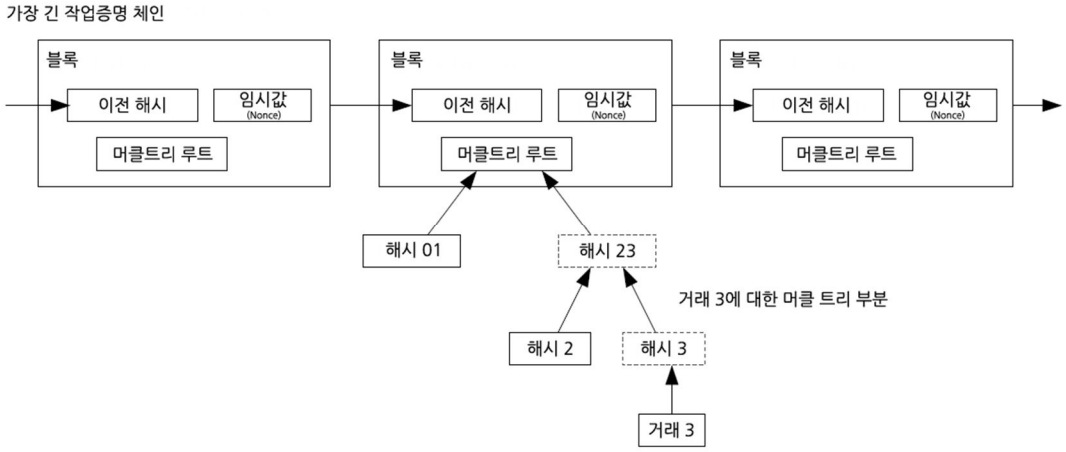


블록에서 거래 내역 0~2를 제거 후

거래 내역이 없는 블록 헤더는 약 80바이트 정도이다. 매 10분마다 블록이 생성된다고 가정할 경우, 80바이트 * 6 * 24 * 365 = 4.2MB가 매년 소요된다. 2008년 기준으로 시중에 판매되고 있는, 2GB 메모리가 장착된 컴퓨터 시스템과, 매년 1.2GB가 증가할 거라 예측한 무어의 법칙에 의하면, 블록 헤더가 메모리를 점유하고 있어야 하더라도 문제가 되지 않는다. *(역자 주: 무어의 법칙은 본문과 달리 18개월에 두 배가 된다는 예측이다. 즉, 매년 약 1.59배로 증가)*

8. 지불 입증 간소화

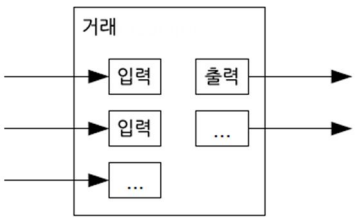
굳이 전체 네트워크 노드를 쓰지 않더라도, 돈이 지불된 사실을 입증하는 것이 가능하다. 사용자는 가장 긴 작업증명 체인의 블록 헤더의 사본만 갖고 있으면, 자신이 가장 긴 체인이라 확인할 때까지 네트워크 노드들에게 요청하고, 그 거래 내역이 기록된 블록에 연결된 머클트리 일부만 받아오면 된다. 그는 스스로 거래 내역을 확인할 수는 없고, 체인에 연결되어 네트워크 노드가 그것을 승인했는지, 이후에도 계속 블록이 추가로 확증되어 승인된 지로 알 수 있다.



이렇게 정직한 노드들에 의해 네트워크가 제어되는 한 거래 인증은 신뢰할 수 있지만, 공격자에 의해 네트워크가 압도당하면 취약하다. 네트워크 노드들은 스스로 거래 내역을 입증할 수 있지만, 단순히 공격자에 의해 과점된 네트워크로 거래를 조작하고 유지함으로써 무력화될 수 있다. 이러한 방법을 보호하는 전략으로는 사용자의 소프트웨어에서 블록 전체를 다운로드 받고 모순임이 확증된 유효하지 않은 블록을 발견했을 때 네트워크 노드들이 경고 알림을 받는 것이다. 잦은 지불을 받는 사업자의 경우에는 아마도 자체 노드에서 독립된 보안 체계와 빠른 인증 방법을 더 원할 것이다.

9. 금액의 결합과 분할

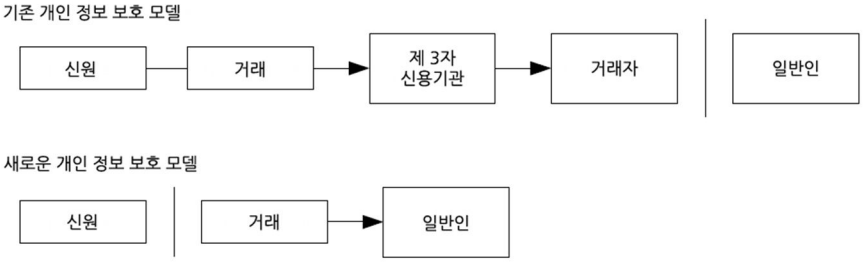
돈을 개별로 관리하는 것도 가능하겠지만, 거래에서 작은 개별 단위까지 굳이 나누어 다루는 것은 거추장스럽다. 금액을 나누고 합칠 수 있도록 거래 내역은 복수의 입력과 출력으로 이뤄진다. 대개는 큰 금액의 단일 입력이거나, 소액 모금을 위한 여러 입력이거나, 한 출력은 지불 다른 출력은 거스름돈 같은 두 개의 출력 방법이 될 것이다.



이를 팬아웃이라 하며, 거래가 여러 거래들에 기반하고 각 거래들이 다수에 의존하는 경우더라도 여기서는 문제가 되지 않는다. 거래 기록의 완전히 독립된 사본을 추출할 필요가 없다.

10. 개인 정보 보호

기존의 은행 모델은 당사자들과 제 3자 신용기관에 정보 접근 권한을 제한하여 개인 정보 보호가 어느 정도 가능하도록 했다. 모든 거래를 공개적으로 알려야 하는 중요성은 이러한 것을 불가능하게 하지만, 공개키를 익명으로 소유하도록 함으로써 정보의 흐름을 차단하고 개인 정보가 유지될 수 있다. 외부에서는 누가 다른 누군가에게 얼마를 보냈단 사실을 볼 수는 있으나, 그 거래 당사자들의 신분으로 연결되지 않으면 알 수가 없다. 이는 증시에서 공개되는, 시간과 거래 목록만 확인 가능한 거래 체결 정보 공개 수준과 비슷하다.



추가적인 안전장치의 일환으로, 매번 거래마다 새로운 공개키-비밀키 쌍을 사용하여 공통의 소유자에게 연결되도록 하면 된다. 여러 개의 입력을 갖는 거래의 경우에는 여전히 연결고리를 남기는 것이 불가피하게도, 그 입력들이 동일 소유자라는 사실을 공개된다. 소유자의 암호키가 공개되면 그러한 연관성이 다른 거래 내역에서도 동일인에 의한 것이라는 게 알려지게 된다.

11. 계산

공격자가 다른 체인의 갈래를 빠르게 생성하려고 시도하여 정직한 노드들의 체인을 앞질러 가장 긴 체인을 생성할 시나리오를 간주해보자. 이것이 성공한다 하더라도, 그렇게 가짜 금액을 만들거나 공격자 소유였던 적이 없는 돈을 사취하더라도, 임의의 시스템 변화 상태로는 남겨두지 않는다. 노드들은 유효하지 않은 거래를 지불로 승인하지 않을 것이며, 정직한 노드들은 그러한 거래 내역을 포함하는 블록을 절대 받아들이지 않을 것이다. 공격자는 오직 최근에 지불한 돈을 도로 되찾기 위해 그의 거래 내역 하나만 바꾸려고 시도할 수 있다.

정직한 노드들의 체인과 공격자의 체인 간의 경쟁은 이항랜덤워크(Binomial Random Walk)로 특정지을 수 있다. 정직한 체인이 하나의 블록을 성공적으로 생성하는 사건이 일어나면 +1, 실패하여 공격자의 체인이 블록을 하나 생성하는 사건이 일어나면 -1이라 한다.

공격자가 갖고 있지도 않은 금액으로 성공할 확률은 도박사의 파산 문제(Gambler's Ruin problem)와 비슷하다. 무제한의 신용을 가진 도박사가 적자 상태로 시작하여 거의 무제한의 게임을 시도하여 손익분기점에 도달한다 가정하자. 그가 손익분기점에 도달할 확률, 즉 공격자가 정직한 체인을 따라잡을 수 있는 가능성은 다음과 같이 계산할 수 있다[8]:

p = 정직한 노드가 다음 블록을 찾을 확률
q = 공격자의 노드가 다음 블록을 찾을 확률
q_z = z개의 다음 블록들을 빨리 찾을 확률

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

$p > q$ 라는 가정이 주어진다면, 공격자가 블록 증가를 따라잡을 수 있는 확률은 블록 수에 지수적으로 감소하게 된다. 공격자가 먼저 달려들어 운이 좋게 성공하지 못한다면, 가능성은 뒤로 갈수록 점점 희박해진다.

이번에는 새로운 거래에서 돈을 받는 사람 입장에서 송금자가 거래를 바꾸지 못할 거라는 것이 충분히 확증하려면 얼마나 기다려야 하는지 생각해보기로 한다. 송금자가 받는 사람이 일시적으로 돈을 받았다 믿게 만들고 일정 시간 뒤에 다시 돈을 자기 자신에게 되돌리게 시도하려는 공격자라 가정하자. 받는 사람은 그러한 일이 벌어지면 경고 알림을 받을 것이고, 공격자는 그게 늦기를 바랄 것이다.

받는 사람은 새로운 암호키 쌍을 생성하여 송금자에게 서명하기 직전에 공개키를 넘긴다. 이러한 방법은 공격자가 미리 충분한 시간 전에 가짜 블록 체인 생성을 준비해두어 가짜 거래를 성립시키는 것을 방지한다. 송금이 이뤄질 때, 정직하지 않은 송금자는 은밀히 그의 다른 거래 내역을 포함하는 체인을 동시에 생성을 시작한다.

돈을 받는 사람은 거래가 블록에 포함되고 추가로 z개의 추가 블록들이 연결될 때까지 기다린다. 그는 공격자가 얼마나 많은 작업을 진척시켰는지 정확히 모르지만, 정직한 블록들이 매 평균 시간 간격마다 생성될 것이고, 공격자의 잠재적 진행률은 포아송 분포의 기대값인 다음과 같을 것이다.

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

공격자가 그 순간에도 여전히 따라잡을 수 있는 확률을 계산하기 위해, 포아송 분포를 공격자가 그 시점에 블록 체인 생성을 따라잡을 매 진행률에 각각 곱하도록 한다.

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

분포의 무한급수를 더하지 않기 위해 식을 정리하면…

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

이를 C언어 코드로 구현하면…

```
#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 돌려보면 *z*값에 따라 지수적으로 확률이 감소함을 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
```

P가 0.1%보다 작을 경우를 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

지금까지 신용에 기반하지 않은 전자 거래 시스템을 제안하였다. 디지털 서명으로 이뤄진 일반적인 화폐 구조에서 출발했다. 소유권을 강력히 제어하는 방법을 제공하지만 이중지불을 방지할 방법이 없이는 완벽하지가 않다. 이러한 문제를 해결하기 위해, 과반수의 컴퓨팅 파워를 정직한 노드들이 제어한다면 계산상으로 공격자가 빠르게 조작할 수 없이 공개적으로 거래를 기록할 수 있도록 작업증명을 수행하는 P2P 네트워크를 제안하였다. 네트워크는 구조적이지 않은 단순함에서 믿을 수 있다. 노드들은 조직화할 필요도 없이 협력하도록 되어있다. 특정 위치에 메시지가 전달되지 않더라도 최선을 기반으로 전달되기만 하면 되기 때문에, 정체를 확인할 필요도 없다. 노드들은 자발적으로 네트워크를 떠났다가 합류할 수 있으며, 작업증명 체인을 그 동안에 벌어졌던 사실에 대한 증명으로 받아들이기만 하면 된다. 컴퓨팅 파워를 통해 의사결정을 하고, 유효한 블록에 대해서만 작업을 수행함으로써 유효하지 않은 블록들은 거부하게 됨으로써 거래 승인을 표시하게 된다. 이러한 합의 메커니즘을 위해 어떤 규칙이나 보상이 성립될 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.

원문

S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” http://bitcoin.org/bitcoin.pdf, 2009.

번역 - 츄이스