비트코인: 개인 대 개인 전자 화폐 시스템

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

Translated in Korean from https://bitcoin.org/bitcoin.pdf
by Philemon, Baucis

초록. 완전한 개인 대 개인 방식의 전자 화폐를 이용하면 한쪽에서 다른 쪽으로 직접 온라인 결제를 할 수 있다. 금융기관을 통할 필요가 없다. 디지털 서명은 부분적인 해결책을 제공하지만, 이중지불을 막기 위해 여전히 신뢰받는 제3자가 필요하다면 주된 이점들이 사라지고 만다. 우리는 이중지불 문제에 대한 해결책으로 개인 대 개인 네트워크 사용을 제안하고자 한다. 이 네트워크는 계속 진행 중인 해시 기반의 작업증명 체인 위에 거래를 해싱한 타임스탬프를 찍는다. 해당 기록은 작업증명을 다시 하는 게 아닌 이상 변경할 수 없다. 가장 긴 체인은 서명된 사건들의 순서를 입증하는 동시에 이 체인이 가장 큰 CPU 파워를 가진 풀에서 만들어졌음을 증명한다. 네트워크 공격에 협력하지 않는 노드들이 CPU 파워의 과반수를 통제하는 한, 그들이 가장 긴 체인을 만들고 네트워크 공격자보다 앞설 것이다. 이 네트워크 자체는 최소한의 구조만 있으면 된다. 메시지는 최선형 모델을 기반으로 전파된다. 그리고 노드들은 자신이 네트워크를 떠나 있을 때 진행되었던 가장 긴 작업증명 체인을 받아들이는 방식을 통해 마음대로 네트워크를 떠났다가 다시 합류할 수 있다.

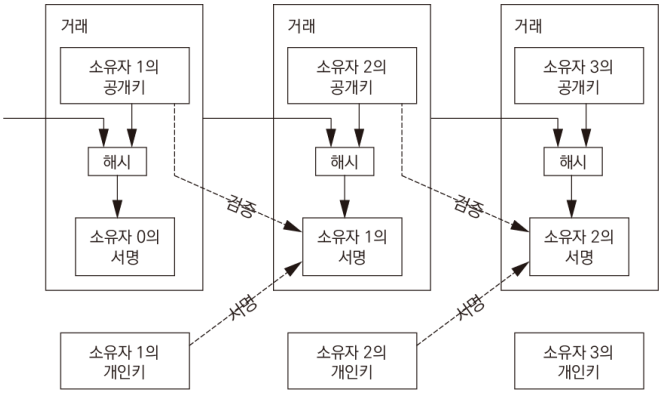
1. 서론

지금까지의 인터넷 상거래는 전자 결제를 처리하기 위해, 신뢰받는 제3자 역할을 하는 금융기관에 거의 전적으로 의존해 왔다. 이러한 시스템은 대부분의 거래에서 잘 작동하지만, 여전히 신뢰 기반 모델이 갖는 고유의 문제를 안고 있다. 이 시스템하에서 완전히 취소할 수 없는 거래란 사실상 있을 수 없는데, 금융기관은 거래상의 분쟁을 중재하는 역할을 해야만 하기 때문이다. 중재 비용은 거래 비용을 증가시키고, 이는 실질적인 최소 거래 금액을 제한하여 일상적인 소액 거래를 불가능하게 한다. 게다가 취소할 수 없는 서비스 제공에 대한 결제마저 취소할 수 있게 만듦으로써 더 큰 비용을 발생시킨다. 결제 취소의 가능성이 있게 되면 신뢰의 필요성이 커진다. 판매자는 실제로 필요한 수준보다 더 많은 정보를 고객에게 요구하면서 그들을 경계할 수밖에 없게 된다. 어느 정도의 사기는 불가피한 것으로 여겨진다. 직접 만나 실물 화폐를 사용하면 이러한 비용과 지불 불확실성을 피할 수 있다. 그러나 통신 채널에서 신뢰받는 기관 없이 결제할 수 있는 방법은 없다.

우리에게는 신뢰가 아닌, 암호학적 증명을 기반으로 하는 전자 결제 시스템이 필요하다. 이 시스템에서는 거래를 원하는 두 당사자가 신뢰받는 제3자 없이 직접 거래할 수 있다. 전산상 취소할 수 없는 거래는 판매자를 사기로부터 보호해 주고, 구매자를 보호할 수 있는 통상적인 에스크로 방식은 쉽게 구현될 수 있다. 이 논문에서는 이중지불 문제에 대한 해결책으로, 거래의 발생 순서를 컴퓨터 계산으로 증명하는 개인 대 개인 분산 타임스탬프 서버를 사용하는 것을 제안한다. 이 시스템은 정직한 노드들이 공격자 노드 그룹보다 더 많은 CPU 파워를 통제하고 있는 한 안전하다.

2. 거래

우리는 전자 코인을 디지털 서명의 체인으로 정의한다. 코인의 각 소유자는 이전 거래 및 송금받을 사람의 공개키를 해싱한 값에 디지털 서명을 하고, 이를 코인의 끝부분에 추가함으로써 다음 소유자에게 코인을 송금한다. 코인을 받은 사람은 소유권의 체인을 검증하기 위해 해당 서명을 검증할 수 있다.

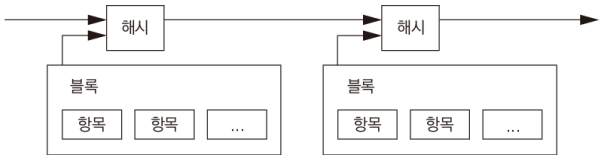


물론 코인을 받은 사람은 이전 소유자 중 누군가가 코인을 이중지불하지 않았는지 검증할 수 없다는 문제가 있다. 이에 대한 통상적인 해결책은 신뢰받는 중앙기관, 즉 화폐 발행처를 두고 그들이 모든 거래에 대해 이중지불 여부를 확인하게 하는 것이다. 한 번 거래된 코인은 새 코인을 발행하기 위해 화폐 발행처로 회수되어야 한다. 그리고 오직 발행처에서 직접 발행한 코인만이 이중지불되지 않은 코인이라는 신뢰를 얻는다. 이와 같은 해결책의 문제는 통화 시스템 전체의 운명이 화폐 발행처의 운영 조직에 달려 있다는 것인데, 마치 은행처럼 모든 거래가 이들을 거쳐야만 하기 때문이다.

우리에게는 이전 소유자들이 이전의 어느 거래에도 서명하지 않았다는 사실을, 돈을 받는 사람에게 알릴 방법이 필요하다. 이를 위해 가장 먼저 일어난 거래만 인정하고 그 후의 이중지불 시도는 무시하기로 한다. 어떤 거래가 없었다는 것을 확인할 수 있는 유일한 방법은, 모든 거래 내역을 알고 있는 것뿐이다. 화폐 발행처 기반 모델에서는 발행처가 모든 거래 내역을 알고 있었고 어떤 거래가 가장 먼저인지를 결정했다. 신뢰받는 기관 없이 이 과정을 수행하려면 모든 거래는 공개되어 있어야 한다[1]. 그리고 거래들이 수신된 순서에 대한 단일 기록에 참가자들이 동의할 수 있는 시스템이 필요하다. 돈을 받는 사람은 거래할 때마다 해당 거래가 가장 먼저 수신된 거래라고 노드의 과반수가 동의했다는 증거를 필요로 한다.

3. 타임스탬프 서버

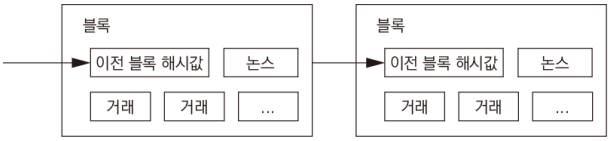
우리가 제안하는 해결책은 타임스탬프 서버로 시작한다. 타임스탬프 서버는 타임스탬프를 찍을 항목들로 이루어진 블록의 해시값을 취하고, 그 해시값을 신문이나 유즈넷 포스트처럼 널리 퍼뜨리는 방식으로 작동한다[2-5]. 그 타임스탬프는 해당 데이터가 해시에 포함되려면 그 시점에 반드시 존재할 수밖에 없었음을 증명한다. 각 타임스탬프는 이전의 타임스탬프를 해시에 포함하면서 체인을 만든다. 이러한 구조 덕분에 이후에 추가되는 타임스탬프가 이전의 타임스탬프들을 견고하게 만든다.



4. 작업증명

개인 대 개인을 기반으로 분산화된 타임스탬프 서버를 구현하려면, 신문이나 유즈넷 포스트 방식이 아니라 애덤 백의 해시캐시[6]와 같은 작업증명 시스템을 사용해야 할 것이다. 작업증명은 특정 값을 찾는 과정을 포함하는데, 그 값이 SHA-256 같은 해시 함수로 해싱되었을 때 해시값은 여러 개의 0비트로 시작한다. 작업증명에 소요되는 평균 시간은 요구되는 0비트의 개수에 따라 기하급수적으로 증가하지만, 검증은 한 번의 해시 함수 실행만으로 가능하다.

타임스탬프 네트워크에서는 필요한 0비트 개수를 만족시키는 블록의 해시값을 찾을 때까지 블록 내에 있는 논스를 증가시키는 방식으로 작업증명을 한다. 일단 CPU 자원을 사용해서 이 작업증명 조건이 충족되었다면, 작업증명을 다시 하지 않는 이상 해당 블록을 변경할 수 없다. 이후에 생성되는 블록들이 체인으로 연결되기 때문에, 하나의 블록을 변경하려면 이후에 연결된 모든 블록에 대한 작업증명을 다시 해야 한다.



작업증명은 다수결에 의한 의사결정에서 대표자 결정의 문제도 해결한다. 과반수의 기준이 한 IP 주소당 한 표 방식에 기반한다면, 많은 IP를 할당할 수 있는 누군가가 시스템을 전복할 수 있다. 그러나 작업증명은 본질적으로 한 CPU 당 한 표 방식이다. 가장 긴 체인이 과반수의 결정을 대표하고, 이는 가장 많은 작업증명 노력이 투여되었음을 반영한다. 정직한 노드들이 CPU 파워의 과반수를 통제한다면, 이 정직한 체인은 가장 빠른 속도로 길어지며 어떤 경쟁 체인보다 앞설 것이다. 어떤 공격자가 과거의 블록을 수정하려면 해당 블록뿐만 아니라 그 이후에 생성된 모든 블록의 작업증명을 다시 해야 한다. 그에 더해 정직한 노드들의 작업을 따라잡고 추월해야만 한다. 후속 블록이 계속 추가되면서 더 느린 속도의 공격자가 따라잡을 확률은 기하급수적으로 감소한다. 이는 뒤에서 자세히 살펴볼 것이다.

시간이 지날수록 하드웨어 속도가 빨라지고 노드 구동에 대한 관여도도 변한다. 이를 보정하기 위해, 작업증명의 난이도는 시간당 평균 블록 수의 이동 평균에 의해 결정된다. 블록이 너무 빨리 생성되면 작업증명의 난도는 높아진다.

5. 네트워크

네트워크를 구동하는 단계는 다음과 같다:

1) 새로운 거래 내역이 모든 노드에게 전파된다.
2) 각 노드는 새로운 거래 내역들을 블록에 취합한다.
3) 각 노드는 그 블록에 해당하는 어려운 작업증명을 수행한다.
4) 어떤 노드가 작업증명을 마치면, 모든 노드에게 그 블록을 전파한다.
5) 노드들은 블록 안의 모든 거래가 유효하고 이전에 사용된 적이 없었을 경우에만 그 블록을 승인한다.
6) 노드들은 블록을 승인했다는 의사를 드러내기 위해, 해당 블록의 해시값을 이전 해시값으로 사용하여 그다음 블록을 생성한다.

노드들은 언제나 가장 긴 체인을 올바른 체인으로 간주하고, 그 체인을 확장해 나간다. 만약 두 노드가 동시에 서로 다른 버전의 다음 블록을 전파한다면, 어떤 노드들은 어느 한쪽을 먼저 받게 될 수 있다. 이 경우, 먼저 받은 블록을 토대로 작업하되, 다른 쪽이 더 길어질 경우에 대비해 다른 쪽 가지도 저장해둔다. 이와 같은 동점 상황은 다음 작업증명의 답이 발견되어 한쪽 갈래가 길어지면 끝난다. 그러면 다른 쪽 가지에서 작업하던 노드들은 더 긴 쪽으로 옮겨 작업한다.

새로운 거래가 반드시 모든 노드에 전파될 필요는 없다. 새로운 거래들이 많은 노드에 도달하기만 한다면, 블록 안에 곧 포함될 것이다. 블록 전파는 메시지 누락이 일어나도 괜찮다. 만약 어떤 노드가 블록을 전달받지 못했다면, 다음 블록을 받을 때 누락을 알아채고 누락된 블록을 요청할 것이다.

6. 인센티브

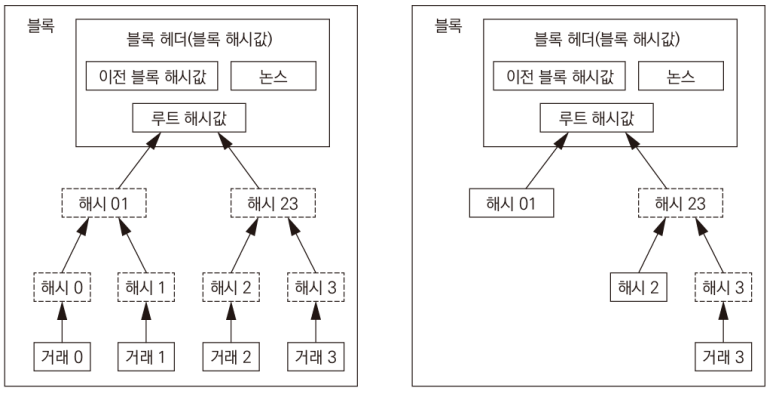
관례상 블록의 첫 거래는 해당 블록을 생성한 사람이 갖게 될 새 코인으로 시작하는 특별한 거래다. 이는 노드들에게 네트워크를 지속하게 할 인센티브를 주는 것이다. 또한 이 시스템에는 화폐를 발행하는 중앙기관이 없으므로, 이는 코인이 최초로 유통될 방법도 제공하는 것이다. 일정량의 새 코인이 꾸준히 추가되는 것은 금의 유통량을 늘리기 위해 광부들이 자원을 소비하는 것과 비슷하다. 우리의 경우, CPU 연산 시간과 전기를 소비한다.

거래 수수료도 작업증명에 대한 인센티브가 될 수 있다. 거래의 출력값이 입력값보다 작을 경우 그 차이는 거래 수수료가 된다. 이 수수료는 거래를 담고 있는 블록의 인센티브 값에 추가된다. 미리 정해진 코인 수량이 모두 시장에 풀리고 나면, 인센티브는 전부 거래 수수료로 전환된다. 이때가 되면 인플레이션에서 완전히 자유로워진다.

인센티브는 노드들이 계속 정직함을 유지하도록 장려할 것이다. 만약 탐욕스러운 공격자가 정직한 노드 전부보다 더 많은 CPU 파워를 가질 수 있다면, 자신의 지불을 철회하여 사람들을 속일지, 아니면 이 CPU 파워를 사용해 새 코인들을 생성할지 선택해야 한다. 그는 이 시스템과 자기 자산의 유효성을 훼손하기보다는 규칙대로 행동해 다른 모든 이들의 몫을 합친 것보다 많은 코인을 받는 것이 더 이득이 된다는 것을 알아야 한다.

7. 디스크 공간 재확보

코인의 가장 최근 거래가 충분히 많은 블록들에 묻히면, 이전에 사용된 거래들은 디스크 공간을 확보하기 위해 폐기될 수 있다. 블록의 해시값을 훼손하지 않고 이 작업을 진행하기 위해 거래들은 머클 트리[7][2][5]로 해싱되고, 트리의 루트 해시값만 블록 해시값에 포함된다. 오래된 블록들은 트리의 가지들을 잘라냄으로써 압축될 수 있다. 내부의 해시값들은 저장될 필요가 없다.

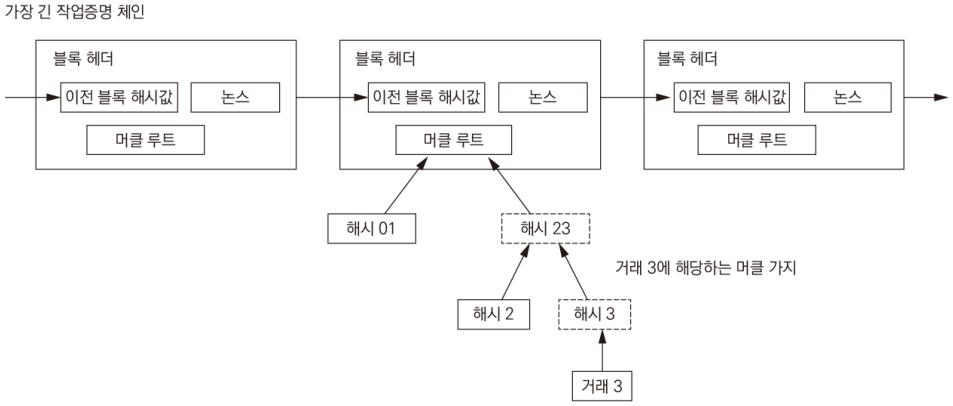


머클 트리로 해싱된 거래들　　블록에서 거래 0-2를 가지치기 한 후

거래 내역이 들어 있지 않은 블록 헤더의 크기는 약 80바이트다. 블록이 10분마다 생성된다고 가정하면, 1년에 80바이트 * 6 * 24 * 365 = 4.2MB 정도가 된다. 2008년 기준 2GB 램을 탑재한 컴퓨터가 일반적으로 판매되고 있고, 매년 1.2GB 증가를 예측하는 무어의 법칙을 고려하면, 블록 헤더들이 메모리에 보관되어야 한다고 해도 저장 공간은 문제 되지 않을 것이다.

8. 간소화된 지불 검증

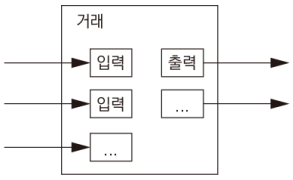
지불 검증은 풀 네트워크 노드를 구동하지 않아도 가능하다. 사용자는 가장 긴 작업증명 체인의 블록 헤더 사본만 갖고 있으면 된다. 이는 가장 긴 체인을 갖고 있다는 확신이 들 때까지 네트워크 노드들에게 요청하여 획득할 수 있다. 그리고 해당 거래를 타임스탬프가 찍힌 블록과 연결해 주는 머클 가지를 얻기만 하면 된다. 그가 직접 해당 거래를 확인할 수는 없지만, 그 거래를 체인의 한 부분에 연결함으로써 어떤 네트워크 노드가 그것을 승인했음을 알 수 있다. 그리고 그 후에 연결된 블록들이 네트워크가 해당 거래를 승인했음을 추가로 확인시켜 준다.



이러한 검증 방식은 정직한 노드들이 네트워크를 통제하는 한 신뢰할 수 있다. 그러나 공격자가 네트워크를 장악하면 이 검증 방식은 보다 취약해진다. 네트워크 노드들은 스스로 거래들을 검증할 수 있지만, 이 간소화된 방법에서는 공격자가 계속 네트워크를 장악할 수 있는 한 공격자의 조작 거래에 속을 수 있다. 이에 맞서는 한 가지 전략은 네트워크 노드들이 유효하지 않은 블록을 감지했을 때 보내는 경고를 받아들이는 것이다. 이 경고는 사용자의 소프트웨어가 블록 전체와 경고를 받은 거래를 다운로드하라고 촉구하여, 이들 간 불일치를 확인하게 한다. 결제 금액을 자주 받는 사업체들은, 더 독립적인 보안과 빠른 검증을 위해 여전히 자체 노드를 구동하기를 원할 수 있다.

9. 금액 합치기와 나누기

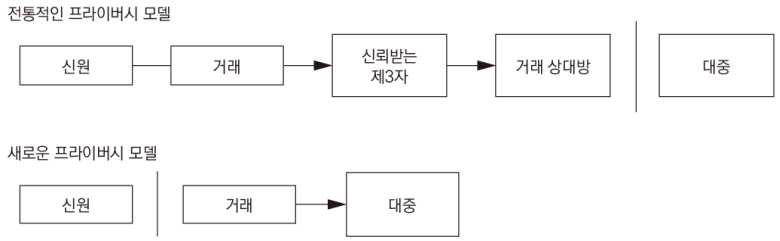
코인들을 하나하나 처리하는 것이 가능하기는 하지만, 모든 푼돈을 각각의 거래로 보내는 것은 거추장스러운 일일 것이다. 금액을 나누고 합칠 수 있도록, 거래는 여러 개의 입력과 출력을 포함한다. 일반적으로 입력은 이전 거래로부터 오는 큰 금액의 단일 입력이거나 작은 금액들로 구성된 여러 개의 입력이다. 출력은 많아야 2개 존재하는데 하나는 지불을 위한 것이고, 다른 하나는 거스름돈이 있을 경우 송금인에게 되돌려보내기 위한 것이다.



거래 하나가 여러 거래를 토대로 하고, 그 여러 거래가 더 많은 거래들을 토대로 하는 팬아웃은 여기서 문제 되지 않는다는 것에 주목해야 한다. 거래 이력의 완전히 독립된 사본을 추출해야 할 필요는 전혀 없다.

10. 프라이버시

전통적인 은행 모델은 관계 당사자들과 신뢰받는 제3자만 정보에 접근할 수 있게 제한함으로써 어느 정도 프라이버시를 달성한다. 모든 거래 내역은 공개되어야 하므로 이 방식은 불가능하다. 하지만 공개키를 익명으로 유지하는 방식으로 정보의 흐름을 다른 곳에서 끊음으로써 여전히 프라이버시를 지킬 수 있다. 대중은 누군가가 다른 누군가에게 얼마를 보내는지 알 수 있다. 하지만 해당 거래를 특정인과 연결하는 정보는 없다. 이는 증권 거래소에서 공개하는 정보의 수준과 비슷하다. 각 거래의 체결 시간과 규모는 공개하지만 거래 당사자의 정보는 밝히지 않는, '테이프'와 같은 것이다.



추가적인 방화벽으로, 거래가 어떤 공통된 소유자와 연결되는 것을 방지하기 위해 거래마다 새로운 키 쌍을 사용해야 한다. 여러 개의 입력으로 이루어진 거래에서는 일부 연결고리를 피할 수 없다. 해당 입력들이 동일한 소유자의 것이었음이 드러나기 때문이다. 어떤 키의 소유자가 드러나면, 이 연결고리가 동일 소유자의 다른 거래들도 드러낼 수 있는 위험이 있다.

11. 계산

공격자가 정직한 체인보다 더 빨리 대체 체인을 생성하려고 하는 시나리오를 생각해 보자. 만약 이런 시도가 성공하더라도, 난데없이 금액을 만들어 내거나 공격자 자신이 소유한 적 없던 돈을 갖게 되는 등의 근거 없는 변경에 시스템이 노출되지는 않는다. 노드들은 유효하지 않은 거래를 지불로 승인하지 않을 것이고 정직한 노드들은 그러한 거래가 포함된 블록을 절대 승인하지 않을 것이기 때문이다. 공격자가 할 수 있는 일이라고는 최근에 쓴 돈을 다시 받기 위해 자신의 거래 중 하나를 변경하려고 시도하는 것뿐이다.

정직한 체인과 공격자 체인 간의 경쟁은 이항 랜덤 워크의 특징을 지닌다. 이 경쟁에서 성공 사건은 정직한 체인이 블록 하나를 연장해 +1만큼 앞서는 것이고, 실패 사건은 공격자의 체인이 블록 하나를 연장해 그 격차를 -1만큼 줄이는 것이다.

공격자가 뒤처진 상황에서 따라잡을 확률은 도박꾼의 파산 문제와 비슷하다. 무한한 신용을 가진 어떤 도박꾼이 적자 상태에서 시작해서 손익분기점에 도달하기 위해 잠재적으로 무한히 게임을 시도한다고 가정해 보자. 도박꾼이 손익분기점에 도달할 확률, 즉 공격자가 정직한 체인을 따라잡을 확률은 다음과 같이 계산할 수 있다[8]:

p = 정직한 노드가 다음 블록을 찾을 확률

q = 공격자가 다음 블록을 찾을 확률

q_z = 공격자가 z 블록 뒤처진 상태에서 따라잡을 확률

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

$p > q$라는 가정하에, 공격자가 따라잡아야 하는 블록 수가 늘어날수록 이 확률은 기하급수적으로 감소한다. 공격자에게 불리한 확률 때문에, 그가 초반에 운 좋게 치고 나가지 못한다면, 점점 뒤처지면서 따라잡을 가능성이 희박해진다.

이제 새로운 거래에서 송금인이 거래를 변경할 수 없다고 수취인이 충분히 확신하려면 얼마나 오래 기다려야 할지 생각해 보자. 송금인이 공격자인 상황을 가정해 보자. 공격자는 수취인이 돈을 받았다고 잠시 믿게 하고, 어느 정도 시간이 흐른 후 그 거래를 자신에게 돈이 되돌아오는 거래로 바꾸려고 한다. 그런 일이 일어날 때 수취인은 경보를 받겠지만, 송금인은 경보 시점이 충분히 늦기를 바랄 것이다.

수취인은 새로운 키 쌍을 생성하고 서명 직전에 송금인에게 공개키를 전달한다. 이렇게 하면 송금인이 충분히 앞설 만큼 운이 좋을 때까지 계속 작업하여 블록의 체인을 미리 준비해 두었다가, 앞서게 된 바로 그 시점에 거래를 실행하는 것을 방지할 수 있다. 거래가 전송되면, 부정직한 송금인은 자신의 거래를 대체하는 버전을 포함한 병렬 체인에서 몰래 작업을 시작한다.

수취인은 거래가 블록에 추가된 뒤 z개의 블록이 추가로 연결될 때까지 기다린다. 그는 공격자가 얼마나 작업을 진행했을지 정확히는 모른다. 하지만 정직한 블록들이 생성되는 데 블록당 평균 예상 시간만큼 걸린다고 가정하면, 공격자의 잠재적 작업 진척도는 다음과 같은 기댓값을 갖는 푸아송 분포가 될 것이다:

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

공격자가 이 시점에서 여전히 따라잡을 수 있는 확률을 구하기 위해, 그가 진행했을 만큼의 각 작업량당 푸아송 밀도 함수에, 그 지점으로부터 따라잡을 수 있는 확률을 곱해보자:

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

분포에서 무한급수 꼬리를 계속 더하지 않도록 식을 정리하고...

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

C언어 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;
}
```

결과를 실행하면, 확률은 z 값이 증가함에 따라 기하급수적으로 감소한다는 것을 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 작은 경우를 풀어보면...

```
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 거래 시스템을 제안했다. 우리는 디지털 서명으로 만들어지는 코인이라는 통상적인 틀에서 출발했다. 이 시스템에서는 소유권을 강력하게 통제할 수 있지만, 이중지불을 방지하는 해결책 없이는 불완전하다. 이 문제를 해결하기 위해, 우리는 거래들의 이력을 공개적으로 기록하는 작업증명 기반의 개인 대 개인 네트워크를 제안했다. 정직한 노드들이 CPU 파워의 과반수를 통제하는 한, 어떤 공격자가 해당 거래 이력을 변경하는 것은 계산상 순식간에 불가능해진다. 네트워크는 비조직적인 단순함으로 인해 견고하다. 노드들은 조율이 거의 없이도 동시에 작동한다. 그들은 식별될 필요가 없다. 메시지가 특정 위치로 가는 것이 아니라, 최선형 모델을 기반으로 전송되기만 하면 되기 때문이다. 노드들은 자신이 네트워크를 떠나 있을 때 진행되었던 작업증명 체인을 받아들이는 방식을 통해 마음대로 네트워크를 떠났다가 다시 합류할 수 있다. 이들은 CPU 파워로 투표하는데, 유효한 블록은 해당 블록을 연장함으로써 수락하고, 유효하지 않은 블록은 연장을 거부함으로써 거절한다. 필요한 규칙과 인센티브는 어느 것이든 이 합의 메커니즘을 통해 시행될 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.