

비트코인: P2P 전자화폐 시스템

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

Translated in Korean from bitcoin.org/bitcoin.pdf
by Kyungwon Park

초록. 순전히 개인 대 개인(P2P) 방식의 전자화폐를 이용하면 금융기관을 통하지 않고도 한 사람과 다른 사람 간의 온라인 지급을 실현할 수 있다. 이를 위해 디지털 서명이 그 해결책의 일부가 될 수 있지만 이중지불이 발생하는 것을 방지하기 위해 여전히 신뢰할 수 있는 제3자가 있어야 한다면 중요한 이점을 놓치고 말 것이다. 따라서 우리는 P2P 네트워크를 이용한 이중지불 해결책을 제안하고자 한다. 이 네트워크에서는 해시 기반의 작업증명 체인에 거래를 해시화하고 타임스탬프를 날인 함으로써 해당 작업증명을 다시 수행하지 않고는 변경할 수 없는 기록을 생성한다. 가장 긴 체인은 거래 기록의 연결을 증명할 뿐만 아니라 이 체인이 가장 큰 CPU 파워 풀에서 생성되었음을 증명한다. 네트워크를 공격하려는 노드가 CPU 자원의 과반수를 통제하지 않는 한 네트워크는 가장 긴 체인을 생성하고 공격자를 막을 수 있을 것이다. 이 네트워크 자체는 최소한의 구조를 가진다. 메시지는 가능한 최대한의 한도 내에서 각 노드에 브로드캐스트되며 노드는 자유롭게 네트워크를 떠났다가 이탈한 사이 발생한 거래에 대한 증명으로 가장 긴 작업증명 체인을 받아들임으로써 네트워크에 재합류할 수 있다.

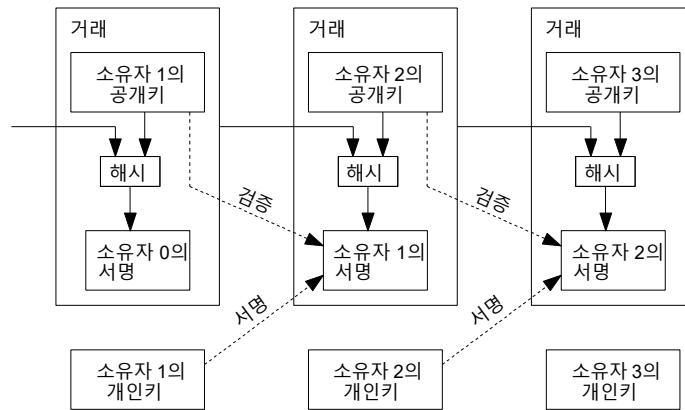
1. 서론

전자상거래는 전자 지급을 처리하는 데에 있어서 신뢰할 수 있는 제3자 역할을 하는 금융기관에 의존해 왔다. 이러한 시스템은 대부분의 거래에서 충분히 잘 동작하지만 신뢰 기반 모델에 내재한 취약점을 완전히 극복하지 못했다. 신뢰 모델에서 금융기관은 분쟁을 중재해야 하는 상황을 필연적으로 마주하기 때문에 거래의 번복 가능성을 완전히 차단하는 것은 사실상 불가능하다. 중재 비용이 발생하면 거래 비용이 증가해 실제 거래할 수 있는 최소 금액이 제한되고 소액의 일상 거래가 불가능해지며 한 번 제공되면 회수할 수 없는 서비스에 대한 결제의 번복을 막을 수 없게 되어 더 큰 손실이 발생한다. 결제가 번복될 가능성이 남아 있으면 신뢰할 수 있는 제3자의 필요성이 더욱 커진다. 판매자는 귀찮게 불필요한 정보까지 요구하면서 소비자를 경계하게 된다. 어느 정도의 사기 가능성은 불가피한 것으로 인정된다. 이러한 비용과 지불의 불확실성은 두 사람이 직접 만나 실물 화폐를 사용하면 피할 수 있으나 온라인에서 신뢰할 수 있는 제3자 없이 지불할 수 있는 메커니즘은 존재하지 않는다.

따라서 필요한 것은 신뢰 대신 암호화 증명에 기반한 전자 지급 시스템으로, 이를 통해 신뢰할 수 있는 제3자에 의존하지 않고 거래 의사를 가진 두 당사자가 직거래를 할 수 있게 해 주는 것이다. 거래를 전산적으로 철회할 수 없도록 만들면 판매자들을 사기로부터 보호하고 통상적인 에스크로 메커니즘을 간편하게 구현하여 구매자를 보호할 수 있을 것이다. 이 논문에서 우리가 이중지불 문제에 대한 해결책으로 제시하는 것은 거래가 발생한 시간 순서의 계산 증명을 생성하는 P2P 분산 타임스탬프 서버이다. 이 시스템에서 정직한 노드가 통제하는 CPU 자원이 공격자 집단에 속한 노드보다 많으면 이 시스템은 안전하다.

2. 거래

우리는 전자 코인을 디지털 서명의 체인이라 정의한다. 각 소유자는 이전 거래와 다음 소유자의 공개키에 대한 해시에 디지털 서명을 하고 이를 마지막에 추가하여 코인을 전달한다. 소유권의 체인을 검증하고 싶으면 수취인은 이 디지털 서명을 검증하면 된다.

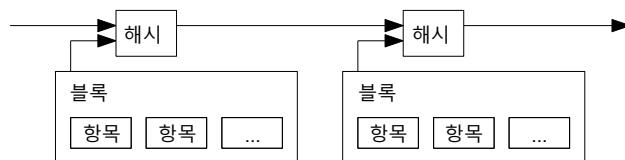


물론 문제는 수취인이 이전 소유자들이 이중지불을 했는지 검증할 수는 없다는 것이다. 통상적인 해결책은 조폐국 같이 신뢰할 수 있는 중앙 권한 기관을 도입하여 모든 거래의 이중지불을 검사하는 것이다. 한 번 거래에 사용된 화폐는 새로운 화폐의 발행을 위해 조폐국으로 회수되어야 하며 조폐국에서 직접 발행한 화폐만 이중지불되지 않은 것으로 신뢰할 수 있다. 이 방법의 문제점은 은행처럼 모든 거래가 조폐국 운영사를 거쳐야 한다는 점에서 전체 통화 시스템의 운명이 특정 회사에 좌지우지된다는 점에 있다.

우리는 이전 소유자가 앞서 어떤 거래에도 서명하지 않았다는 점을 수취인이 알 수 있는 방법이 필요하다. 우리의 목적을 위해서는 첫 번째 거래가 중요하기 때문에 그 이후의 이중지불 시도는 무시하기로 한다. 어떤 거래가 없었음을 확인하는 유일한 방법은 모든 거래를 확인하는 것뿐이다. 조폐국 기반 모델에서는 조폐국이 모든 거래를 기록하고 어떤 거래가 먼저인지 판단한다. 신뢰할 수 있는 제3자 없이 이를 달성하려면 모든 거래가 반드시 공개되어야 하며[1], 거래 순서를 담은 단일 기록에 참여자들이 동의할 수 있는 시스템이 필요하다. 수취인에게 필요한 것은 매 거래마다 과반수의 노드들이 이 거래가 첫 사용한 것임에 동의하는 증명이다.

3. 타임스탬프 서버

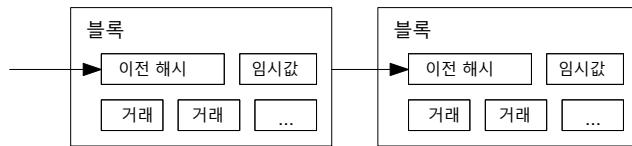
우리가 제안하는 해법은 타임스탬프 서버로 시작한다. 타임스탬프 서버의 동작 방식은 타임스탬프를 날인할 항목이 담긴 블록의 해시를 계산하고 이를 신문이나 유즈넷(Usenet) 포스트처럼 널리 공표하는 식이다[2-5]. 타임 스템프는 그 데이터가 해시에 포함되기 위해 해당 시점에 명백히 존재했음을 증명할 수 있다. 각각의 타임스탬프는 해시에 이전 타임스탬프를 포함해 체인을 형성하면서 이후에 추가되는 타임스탬프는 이전의 타임스탬프를 보강하게 된다.



4. 작업증명

P2P 기반의 분산 타임스탬프 서버를 구현하기 위해 우리는 신문이나 유즈넷 포스트 같은 방식이 아니라 애덤 백의 해시캐시[6]와 유사한 작업증명 시스템을 이용할 필요가 있다. 작업증명은 SHA-256 같은 알고리즘으로 해시되면 여러 개의 0 비트로 시작하는 값을 찾는 작업을 포함한다. 평균 작업량은 요구되는 0 비트의 개수에 따라 기하급수적으로 증가하나 검증은 단 한 번의 해시 계산으로 가능하다.

우리는 타임스탬프 네트워크를 위해 블록의 해시가 요구되는 0 비트를 가진 값을 찾을 때까지 임시값을 증가시켜 작업증명을 구현한다. 이 같은 연산 작업으로 작업증명 조건을 충족하면 이 작업을 되풀이하지 않고서는 블록을 변경할 수 없게 된다. 블록들은 서로 체인으로 연결되기 때문에 블록을 변경하려면 뒤이은 모든 후행 블록의 작업증명을 다시 수행해야 하는 것이다.



작업증명은 다수결에서 대표성을 선정하는 문제도 해결할 수 있다. IP 주소당 한 표를 부여하는 방식에 따라 과반수 결정이 정해지는 경우 많은 IP 주소를 할당할 수 있는 누군가가 과반수의 결정을 공격할 수 있다. 따라서 작업증명은 근본적으로 CPU 하나당 한 표를 부여한다. 다수결에 따른 결정은 가장 긴 체인으로 대표되며 이는 작업증명에 가장 큰 수고를 들인 체인이다. 정직한 노드가 과반수의 CPU 자원을 통제하면 이들이 생성한 정직한 체인이 가장 빨리 길어져 다른 경쟁 체인을 앞서게 된다. 과거의 블록을 조작하려면 공격자는 해당 블록과 그 뒤를 이은 모든 후속 블록의 작업증명을 다시 수행한 다음 정직한 노드들의 작업을 따라잡고 추월해야 한다. 뒤에서 우리는 블록이 추가될수록 속도가 느린 공격자들이 정직한 노드를 따라잡을 가능성에 기하급수적으로 낮아진다는 점을 다룰 것이다.

시간이 지남에 따라 향상되는 하드웨어 속도와 노드 운영에 대한 관심의 변화를 보정하기 위해 작업증명 난도는 매 시간 생성되는 블록의 평균 수를 기반으로 한 이동평균에 따라 결정된다. 블록이 생성되는 속도가 너무 빠르면 난도가 상승한다.

5. 네트워크

네트워크가 동작하는 과정은 다음과 같다.

- 1) 신규 거래들이 모든 노드에 브로드캐스트된다.
- 2) 각 노드는 신규 거래들을 하나의 블록에 모은다.
- 3) 각 노드는 각자의 블록에 대해 작업증명을 수행한다.
- 4) 한 노드가 작업증명을 마치면 자신의 블록을 모든 노드에 브로드캐스트한다.
- 5) 노드들은 블록에 포함된 거래가 모두 유효하고 앞서 사용된 적이 없는 경우에만 블록을 승인한다.
- 6) 노드들은 체인의 다음 블록을 생성하는 작업에서 수용한 블록의 해시를 이전 해시로 사용하여 브로드캐스트받은 블록의 승인을 표현한다.

노드는 항상 가장 긴 체인을 올바른 것으로 간주하고 계속해서 이를 확장해 나간다. 만약 두 노드가 동시에 서로 다른 블록을 브로드캐스트하면 둘 중 하나의 블록을 먼저 받은 노드가 생길 수 있다. 이 경우 노드는 먼저 받은 블록을 작업하되 자신의 블록이 더 짧아지는 경우에 대비해 두 번째로 수신한 블록의 갈래도 보관한다. 다음 작업증명이 완료되어 어느 한 쪽의 갈래가 더 길어지면 짧은 갈래를 작업하던 노드는 길이가 더 긴 갈래의 체인으로 전환하게 된다.

새로운 거래의 브로드캐스트가 반드시 모든 노드에 도달해야 하는 것은 아니다. 많은 노드에 도달하기만 하면 새로운 거래가 블록에 포함될 것이다. 메시지가 누락된 상황도 블록 브로드캐스트에 문제가 되지 않는다. 어떤 노드가 블록을 받지 못했다면 다음 블록을 받을 때 누락된 블록이 있다는 것을 알고 이를 다시 요청할 것이다.

6. 보상

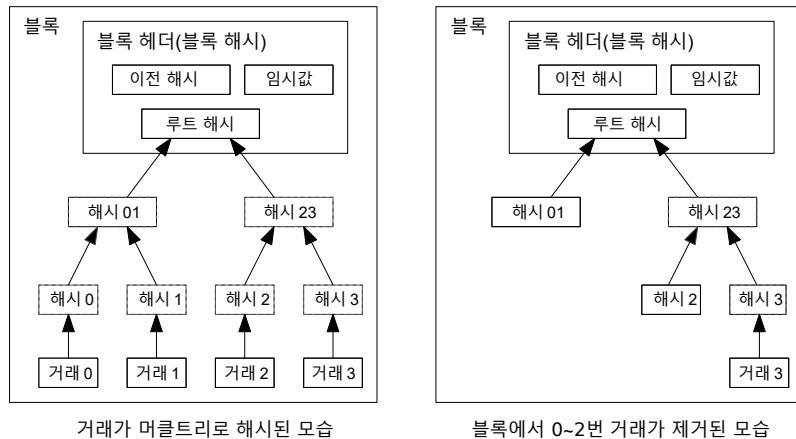
관례상 블록의 첫 거래는 해당 블록을 생성한 사람에게 신규 코인이 주어지는 특별한 거래이다. 이는 네트워크를 유지하는 대가로 노드가 받는 보상이자 코인을 최초로 배분하는 방법이 될 것이다. 이 시스템에서는 코인을 발행하는 중앙 기관이 없기 때문이다. 새로운 코인을 일정량씩 꾸준히 추가하는 것은 금 채굴자들이 자원을 투입하여 금의 유통량을 늘리는 것과 비슷하다. 우리의 경우 투입되는 자원은 CPU 연산 시간과 전력이다.

거래 수수료도 보상이 될 수 있다. 거래의 결과값이 입력값보다 작으면 그 차액은 해당 거래가 담긴 블록의 보상 가치에 추가되는 거래 수수료가 된다. 코인의 정해진 수량이 모두 시장에 유통되면 보상 체제는 거래 수수료 기반으로 완전히 전환되고 인플레이션으로부터 완전히 자유롭게 된다.

보상은 노드들이 부정을 저지르는 것을 막을 수 있을 것이다. 탐욕스러운 공격자가 정직한 노드보다 더 많은 CPU 자원을 끌어모을 수 있으면, 공격자는 연산 자원을 사용해 지급한 것을 다시 훔치는 사기를 칠 것인지 아니면 새로운 코인을 발행할 것인지 선택의 기로에 놓일 것이다. 공격자는 다른 사람들보다 더 많은 코인을 발행할 수 있는 기회를 제공하는 시스템 규칙을 지키는 것이 시스템과 자기 재산의 유효성을 손상시키는 것보다 이익이라는 점을 알게 될 것이다.

7. 디스크 공간 재확보

코인에 담긴 마지막 거래 위로 블록이 충분히 쌓이면 앞선 거래들은 공간 절약을 위해 버려질 수 있다. 블록의 해시를 유지하면서 이를 실현하기 위해 거래는 머클트리 구조로 해시되어[7][2][5] 루트만 해당 블록의 해시에 포함된다. 이를 통해 트리의 가지를 잘라내어 오래된 블록을 압축할 수 있다. 내부 해시는 저장할 필요 없다.

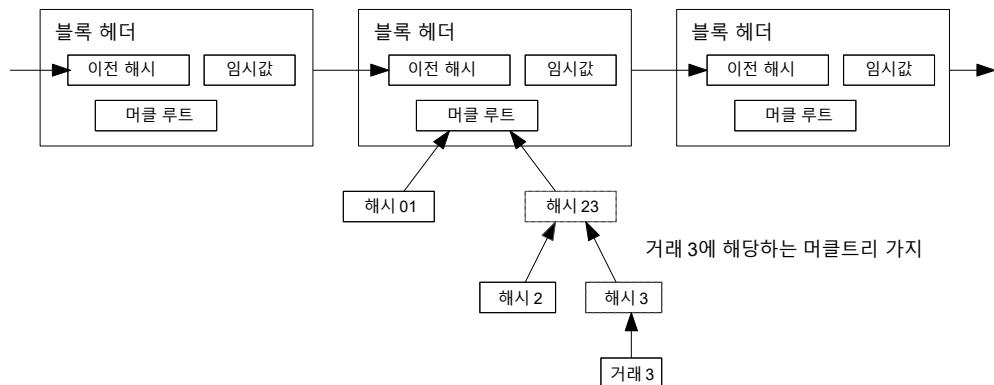


거래 내역이 비어 있는 블록 헤더의 용량은 약 80바이트이다. 블록이 10분마다 생성된다고 가정하면 연간 증가량은 $4.2\text{MB}(80\text{바이트} * 6 * 24 * 365 = 4.2\text{MB})$ 이다. 2008년 기준으로 시중에서 판매되는 컴퓨터의 램은 보통 2 GB이고 해마다 1.2GB씩 성장할 것으로 예측하는 무어의 법칙을 고려하면 블록 헤더가 반드시 메모리에 보관되어야 한다고 해도 저장 공간은 문제가 되지 않을 것이다.

8. 간소화한 지급 검증

네트워크의 모든 노드를 가동하지 않고도 지급을 검증할 수 있다. 사용자는 자신이 가장 긴 체인을 가지고 있다는 것을 확신할 때까지 네트워크 노드에게 질의하여 획득한 가장 긴 작업증명 체인의 블록 헤더 사본을 보유하고 해당 거래와 해당 타임스탬프가 날인된 블록을 연결하는 머클트리 가지를 보유하기만 하면 된다. 사용자 본인이 직접 거래를 확인할 수는 없지만 체인의 일부에 연결하여 네트워크 노드가 이를 승인하는지 확인할 수 있다. 이후에 블록이 추가로 연결되면 네트워크가 이를 승인하였음을 확신할 수 있다.

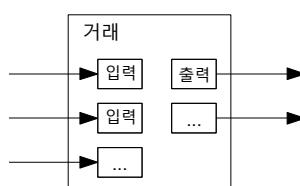
가장 긴 작업증명 체인



이처럼 정직한 노드가 네트워크를 통제하는 한 검증을 신뢰할 수 있지만 공격자가 네트워크를 장악하면 취약해질 수 있다. 네트워크 노드는 스스로 거래를 검증할 수 있지만 한편으로는 네트워크 장악을 유지할 수 있는 한 공격자는 조작한 거래로 이 간소화된 검증법을 속일 수 있다. 이를 막을 수 있는 한 가지 전략은 유효하지 않은 블록을 감지했다는 네트워크 노드의 경고를 받아들이고 사용자의 소프트웨어가 블록 전체와 경고를 받은 거래를 내려받게 하여 불일치를 확인하는 것이다. 지급이 빈번하게 일어나는 사업자들은 보안의 독립성을 높이고 검증의 신속성을 갖추기 위해 직접 노드를 운영하고 싶어 할 것이다.

9. 거래 금액의 결합과 분할

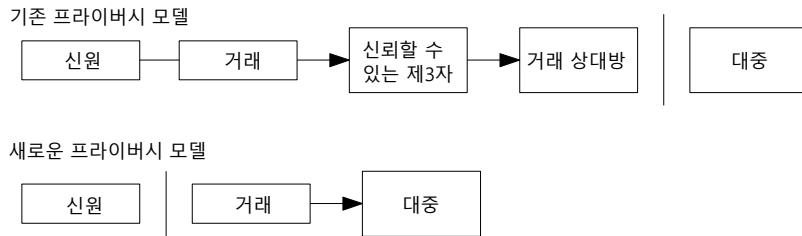
코인을 개별로 다룰 수도 있지만 송금 시 잔돈을 하나도 빠짐없이 별도의 거래로 처리하는 것은 번거로운 일이다. 거래 금액을 분할하고 결합하기 위해 거래는 복수의 입력과 출력을 포함한다. 보통은 이전의 더 큰 거래에서 나온 단일한 입력이 있거나 작은 금액을 합치는 다수의 입력이 존재하며 출력은 최대 두 개가 존재하는데 하나는 지급을 위한 것이고 다른 하나는 보낸 이에게 돌려 줄 게 있는 경우 거스름돈을 위한 출력이다.



부채꼴처럼 한 거래가 여러 거래를 기반으로 하고 또 이 여러 거래가 다시 더 많은 거래에 기반하는 것은 문제가 되지 않는다는 것에 주목해야 한다. 한 거래 내역의 완전하고 독립된 사본을 추출할 필요는 전혀 없는 것이다.

10. 프라이버시

기존 은행 모델은 정보에 액세스할 수 있는 주체를 관련 당사자와 신뢰할 수 있는 제3자로 제한하여 일정 수준의 프라이버시를 달성하고 있다. 모든 거래를 공개적으로 공표해야 하는 모델에서는 이 방식을 적용할 수 없지만 공개키를 익명으로 보관하여 정보의 흐름을 차단하는 방식으로 프라이버시를 지킬 수 있다. 대중은 누가 누구에게 얼마를 보내는지 알 수 있지만 이 거래를 특정인과 결부시킬 수 있는 정보는 알 수 없다. 이는 누구나 거래 시간과 거래량 정보를 담은 테이프(tape)를 볼 수 있지만 거래 당사자가 누구인지는 알 수 없는 증권 거래소의 정보 공개 수준과 비슷하다.



공통된 소유자로 연결되지 않도록 거래마다 새로운 키 쌍을 사용하면 방화벽을 더욱 강화할 수 있다. 다중 입력 거래는 동일한 소유자가 다수의 입력을 소유하고 있었다는 사실을 필연적으로 드러내기 때문에 연결고리를 완전히 끊는 것은 불가능하다. 리스크는 어떤 키의 소유자가 누구인지 밝혀지면 해당 소유자의 다른 거래도 드러날 수 있다는 것이다.

11. 계산

공격자가 정직한 체인보다 빠르게 대안 체인을 생성하려는 시나리오를 가정해 보자. 이 시도가 성공하더라도 공격자는 마음대로 가치를 생성하거나 단 한 번도 자신의 것이었던 적이 없는 돈을 가져가는 것과 같은 임의 조작을 시스템에 가할 수 없다. 노드는 유효하지 않은 거래를 지금으로 인정하지 않을 것이며 정직한 노드는 그러한 거래가 포함된 블록을 절대 인정하지 않을 것이다. 공격자는 단지 자신이 최근에 사용한 돈을 돌려받기 위해 자신의 거래 중 하나를 변경하려고 할 것이다.

정직한 체인과 공격자 체인의 경쟁은 이산적 랜덤 워크(Binomial Random Walk)로 특징지을 수 있다. 성공 이벤트는 정직한 노드가 한 블록씩 길어져 리드를 +1만큼 늘리는 것이고 실패 이벤트는 공격자 체인이 한 블록씩 길어져 갭을 -1만큼 줄이는 것이다.

공격자가 주어진 열세를 극복하고 정직한 노드를 따라잡을 확률은 도박꾼의 파산 문제와 유사하다. 무한대의 신용을 가진 도박꾼이 손실을 본 상태에서 본전을 찾기 위해 무제한으로 도박을 시도하는 상황을 가정해 보자. 도박꾼이 본전을 찾을 확률, 즉 공격자가 정직한 체인을 따라잡을 확률은 다음과 같이 계산할 수 있다[8].

$$p = \text{정직한 노드가 다음 블록을 찾을 확률}$$

$$q = \text{공격자가 다음 블록을 찾을 확률}$$

$$q_z = \text{공격자가 } z\text{개 블록 뒤처진 상태에서 따라잡을 확률}$$

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

$p > q$ 라 가정하면, 공격자가 따라잡아야 하는 블록 수가 늘어날수록 확률이 기하급수적으로 하락한다. 이렇게 불리한 확률로 인해 초반에 운 좋게 앞서 나가지 못하면 공격자는 점차 뒤쳐지며 이길 확률이 사실상 사라진다

이제 새로운 거래에서 전송자가 거래를 변경할 수 없음이 충분히 확실해질 때까지 수신자가 얼마나 기다려야 하는지 생각해 보자. 이때 전송자는 공격자로, 한동안 수신자가 지급을 받았다고 믿게 만든 다음 일정 기간이 지나면 이를 회수하려 한다고 가정한다. 회수가 발생하면 수신자는 경고 알림을 받지만 전송자는 그 전에 이미 모든 일을 마치려고 한다.

수신자는 새로운 키 쌍을 생성하고 서명 직전에 공개키를 전송자에게 전달한다. 이를 통해 전송자가 미리 준비한 블록의 체인을 지속해서 작업하다가 운 좋게 충분히 앞서나가는 데 성공하여 거래를 실행하는 것을 막을 수 있다. 거래가 보내지면 부정직한 전송자는 몰래 다른 버전의 거래가 포함된 체인을 병행해서 작업하기 시작한다.

수신자는 해당 거래가 블록에 추가되고 뒤이어 z 개의 블록이 연결될 때까지 기다린다. 수신자는 공격자의 작업이 얼마나 진행되었는지 모르지만 정직한 블록들은 블록당 평균 예상 시간이 걸렸다고 가정하여 공격자의 잠재적 진척도는 기댓값을 갖는 푸아송 분포를 따를 것이다.

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

이제 공격자가 여전히 따라잡을 수 있는 확률을 구하기 위해 공격자가 이뤄냈을 수 있는 각 진척율에 대한 푸아송 분포에 그 시점부터 따라잡을 수 있는 확률을 곱한다.

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

무한히 이어지는 분포의 꼬리를 계속해서 더하지 않도록 식을 다듬으면 다음과 같다.

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

이를 C 코드로 변환하면 다음과 같다.

```
#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;
}
```

몇 차례 식을 계산해 보면 z 값에 따라 확률이 기하급수적으로 하락하는 것을 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
```

P 가 0.1%보다 작다고 가정하면 다음과 같다.

$P < 0.001$

```
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 제안하였다. 우리는 먼저 소유권을 확실히 제어할 수 있는 디지털 서명에 기반한 코인의 프레임워크를 다루었으나 이중지불에 대한 해결책 없이는 불완전하다. 이를 해결하기 위해 우리는 작업증명을 통해 공개적인 거래 내역을 기록하는 P2P 네트워크를 제안했다. 이 네트워크에서 정직한 노드들이 CPU 자원의 과반수를 통제하면 공격자가 거래 내역을 조작하는 것은 전산적으로 불가능해진다. 이 네트워크는 구조화되지 않은 단순성 덕분에 견고하다. 노드는 조정이 거의 없어도 일사불란하게 동작한다. 메시지는 특정한 곳으로 보내지지 않고 최대한 가능한 만큼 전달되면 되기 때문에 노드를 식별할 필요가 없다. 노드는 원하면 언제든 네트워크를 떠났다가 이탈한 사이 발생한 거래의 증명으로 작업증명 체인을 받아들여 다시 네트워크에 합류할 수 있다. 노드는 유효한 블록은 연장하는 작업을 수행하여 받아들이고 유효하지 않은 블록은 작업을 거절하여 받아들이지 않는 등 자신의 CPU 자원으로 의사 표시를 한다. 필요한 규칙이나 보상은 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.