Bitcoin: A Peer-to-Peer Electronic Cash System

비트코인: P2P 전자화폐시스템

Satoshi
Nakamoto
satoshin@gmx.c
om
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution.

전자화폐의 순수 P2P 버전은 금융기관을 통하지 않고 일방당사자로부터 다른 당사자로 직접 보내는 온라인 지불수단를 허용할 것이다.

Digital signatures provide part of the solution, but the main benefits are lost if a **trusted third party** is still required to prevent double-spending.

디지털 서명은 해결책의 일부를 제공하지만 이중사용(이중지출)을 방지하는데 **제 3 신뢰기관**(신뢰할 수 있는 서드파티)이 지속적으로 필요 하다면 (전자화폐의 P2P 의) 주요 이점들은 잃게 된다.

We propose a solution to the **double-spending** problem using a peer-to-peer network.

우리는 P2P 네트워크를 사용하여 **이중사용** 문제에 대한 해결책을 제안한다.

The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the **proof-of-work**.

네트워크는 거래(들)를 해시기반 **작업증명**의 지속적인 연결고리로 해시하는 것으로 거래(들)에 타임스탬프를 찍어 작업증명의 재작업 없이는 변경될 수 없는 레코드를 형성한다.

The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power.

가장긴 연결고리는 발생한 이벤트들의 순서에 대한 증명 역할을 할 뿐만 아니라 그것이 CPU 파워의 가장 큰 풀로 부터 왔다는것을 증명한다.

As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers.

CPU 파워의 대부분이 네트워크를 공격하기 위해 협력하지 않는 노드들에의해 제어 되는한 그들은 가장긴 연결고리를 생성하여 공격자들을 능가 할것이다.

The network itself requires minimal structure.
네트워크 그 자체는 최소한의 구조를 요구한다.

Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.
메시지들은 **최선의 노력 (모집주선-best effort) 기반**으로 브로드 캐스트(공개) 되고 노드들은 마음대로 네트워크를 떠나고 그들이 없는 동안 일어났던것의 증명으로 가장 긴 작업증명 연결고리를 받아들임으로써 다시 합류할 수 있다.

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments.
인터넷 상거래는 대부분 전자지불을 처리하는 제 3 신뢰기관으로서 역할을 하는 금융기관에 전적으로 의존 하고 있다.

While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model.
시스템이 대부분의 거래에 대해서 충분히 잘 동작하는 반면 여전히 신뢰기반 모델의 타고난 약점으로 부터 고통 받고 있다.

Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes.
금육기관들은 분쟁을 중재하는것을 피할 수 없기 때문에 완전히 취소불가능한(되돌릴 수 없는) 거래들은 실제적으로 가능하지 않다.

The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non- reversible services.
중재비용은 거래비용을 증가시기 때문에 최소 실거래 규모를 제한 하고 작고 가벼운 거래(small casual transaction)에 대한 가능성을 차단한다. 그리고 취소불가능한 서비스들에 대해서 취소불가능한 지불하는 기능을 없애는데 더 많은 비용이 든다. (의역 : 취소불가능한 서비스에 대해서 취소 가능하게 하는 것이 비용이 더 많이 든다.)

With the possibility of reversal, the need for trust spreads. Merchants must be wary of their customers, hassling them for more information than they would otherwise need. A certain percentage of fraud is accepted as unavoidable.
취소의 가능성 때문에 신뢰에 대한 필요성이 확산된다. 상인들은 그들(고객)이 필요로 하는것 보다 더 많은 정보를 위해 그들(상인들)을 괴롭히는 고객들을 조심해야 한다. 사기의일정 비율은 피할 수 없는것으로 받아들여 진다.

These costs and payment uncertainties can be avoided in person by using physical currency, but no mechanism exists to make payments over a communications channel without a trusted party.
이러한 비용과 지불 불확실성은 물리적인 통화를 사용하는 사람에게는 피할 수 있는 것이지만 신뢰기관이 없는 통신 체널을 통해 지불 할 수 있는 메카니즘은 존재하지 않는다

What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party.
필요한것은 신뢰 대신에 암호학적 증명에 기반을 둔 전자지불 시스템 으로 거래 의사가 있는 어떤 두 당사자가 제 3 신뢰기관에 대한 필요성 없이 서로 직접 거래하는것을 허용하는 것이다.

Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers.
취소하는것이 계산적으로 실행불가능한 거래는 사기로부터 판매자들을 보호할 것이고 통상적인 3 자거래(escrow)메카니즘이 구매자들을 보호하기 위해 쉽게 구현될 수 있다.

In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions.
이 논문에서 우리는 거래들의 시간순서(연대순)에 대한 계산적(컴퓨터를 사용한) 증명을 생성하는 하는 P2P 분산 타임스탬프 서버를 사용하여 이중사용(지출)문제를 위한 솔루션을 제안한다.

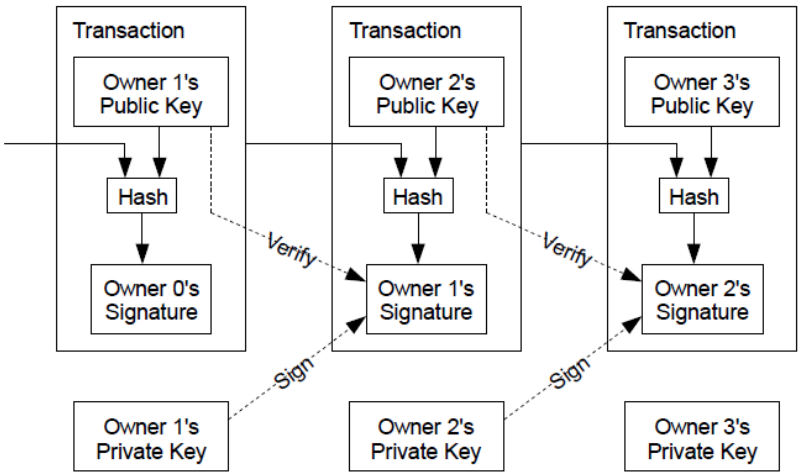
The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.
시스템은 정직한 노드들이 임의의 협력하는 공격자 노드들의 그룹보다 집단적으로 더 많은 CPU 파워를 제어하는한 안전하다.

2. Transactions

거래

We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.

우리는 전자코인을 디지털 서명들의 연결고리(체인)로 정의 한다. 각 소유자는 코인을 이전 거래의 해시와 다음 소유자의 공개키에 디지털적으로 사인하고 동전의 끝에 이러한것을 붙여서 다음으로 전송한다. 수취인은 소유권 체인을 확인하기 위해 서명을 확인 할 수 있다.



The problem of course is the payee can't verify that one of the owners did not double-spend the coin.

과정의 문제는 수취인이 소유자들 중 한 사람이 코인을 이중사용 하지 않았다는 것을 확인 할 수 없다는 것이다.

A common solution is to introduce a trusted central authority, or mint, that checks every transaction for double spending.

일반 솔루션은 이중사용에 대해 모든 거래를 체크하는 신뢰할 수 있는 중앙기관 이나 조폐국(화폐 주조소)을 도입하는 것이다.

After each transaction, the coin must be returned to the mint to issue a new coin, and only coins issued directly from the mint are trusted not to be double-spent.

매 거래 이후 코인은 새로운 코인을 발행하기 위해 조폐국 으로 돌아가야만 하고 오직 조폐국으로 부터 직접 발행된 코인만이 이중사용 되지 않았다고 신뢰 할 수 있다.

The problem with this solution is that the fate of the entire money system depends on the company running the mint, with every transaction having to go through them, just like a bank.

이 솔루션이 가진 문제는 전제 화폐 시스템의 운명이 은행처럼 모든 거래가 거쳐가야만 하는 조폐국을 운영하는 회사에 달려 있다는 것이다.

We need a way for the payee to know that the previous owners did not sign any earlier transactions.

우리는 이전 소유주가 어떤 더 이전 거래에 사인하지 않았다는 것을 수취인에게 알리는 방법이 필요하다.

For our purposes, the earliest transaction is the one that counts, so we don't care about later attempts to double-spend.

우리의 목적에서, 최초 거래가 가장 중요한 것이고 우리는 이중사용을 하기 위한 이후 시도에 대해서 신경 쓰지 않는다

The only way to confirm the absence of a transaction is to be aware of all transactions.

거래가 없었다는 것을 확신하는 유일한 방법은 모든 거래를 알고 있는 것이다.

In the mint based model, the mint was aware of all transactions and decided which arrived first.

조폐국 기반 모델에서 조폐국이 모든 거래를 알고 있었고 첫번째 도착했던 거래를 식별 했다.

To accomplish this without a trusted party, transactions must be publicly announced [1], and we need a system for participants to agree on a single history of the order in which they were received.

신뢰 기관 없이 이것을 달성하기 위해 거래는 공개적으로 알려 져야 하고[1] 우리는 참가자들이 거래가 도착한 순서에 대한 하나의 이력에 동의 하는 시스템이 필요하다.

The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

수취인은 매 거래 시간마다 노드들의 대다수가 그것이 처음 도착했다는 것에 동의 한다는 증명이 필요하다.

3. Timestamp Server

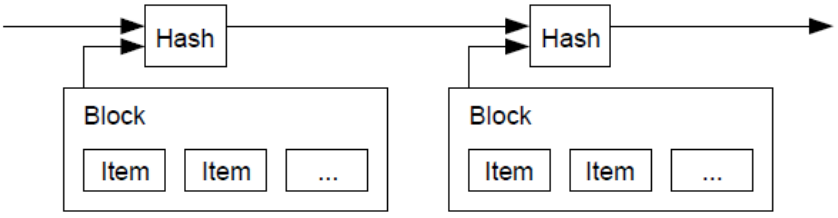
3. 타임스탬프 서버

The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5].

우리가 제안한 솔루션은 타임스탬프 서버로 시작한다. 타임스탬프 서버는 타임스탬프가 찍히는 아이템들의 블록에 대해 해시를 하고 그 해시를 신문이나 유저넷 포스터 처럼 널리 공개하는것으로 동작한다.

The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.

타임스탬프는 데이터가 명확하게 해시에 들어가기 위해 그시간에 존재 하고 있어야 한다는것을 증명한다. 각 타임스탬프는 그 해시에 이전 타임스탬프를 포함하고 각 추가적인 타임스탬프가 그 전에 있던것들을 보강하는것으로 하나의 연결고리를 형성한다.



4. Proof-of-Work

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof- of-work system similar to Adam Back's Hashcash [6], rather than newspaper or Usenet posts.

P2P 에 기반한 분산 타임스탬프 서버를 구현하기 위해 우리는 신문이나 유저넷 포스트(Usenet Post)들 보다는 아담 백의 해시캐시(Adam Back's Hashcash)와 유사한 작업증명(Proof-of-work)을 사용하는것이 필요할 것이다.

The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits.

작업증명은 해시될 때 얼마간의 제로비트들로 시작하는 SHA-256 과 같은 값에 대한 스캐닝을 수반한다.

The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

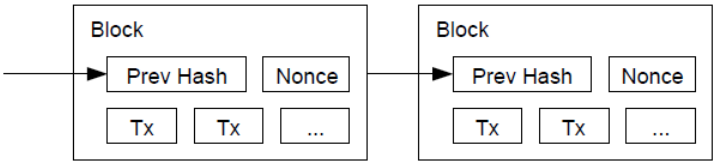
필요한 평균 작업은 요구되는 제로비트의 수에 있어 기하급수적이고 하나의 해시를 실행함으로서 확인될 수 있다.

For our timestamp network, we implement the proof-of-work by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits.

타임스탬프 네트워크의 경우 우리는 블록의 해시가 필요한 제로비트들을 주는 값이 발견될때 까지 블록에 있는 임시값(Nonce: number used once 의 약어, 한 번만 사용되는 임의의 수, 비표)을 증가 시킴으로서 작업증명을 구현한다.

Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.

CPU 의 노력(CPU effort)이 작업증명을 충족하는데 소모되어 왔다면 블록은 재작업 없이는 변경될 수 없다. 이후 블록들이 그 뒤에 연결되기 때문에 블록을 변경하는 작업은 그 뒤에 있는 모든 블록들을 재작업하는것을 포함할 것이다.



The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs.

작업증명은 또한 다수의사결정(다수결)에 있어 대표자를 결정하는 문제를 해결한다. 다수가 한 IP 주소당 한 투표에 기반을 뒀다면 많은 IP 들을 할당할 수 있는 누군가에의해 뒤집어 질 수 있다.

Proof-of-work is essentially one-CPU-one-vote.
작업증명은 근본적으로 한 CPU 당 한 투표 이다.

The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it.
다수결은 그것에 투자된 최고의 작업증명 노력을 가진 가장 긴 연결고리로 나타난다.

If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains.
만약 CPU 파워의 다수가 정직한 노드에 의해 제어 된다면 정직한 연결고리는 빠르게 성장하고 어떤 경쟁 노드들도 앞지를 것이다.

To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes.
지난 블록을 수정하기 위해 공격자는 그 블록과 그 블록 뒤에 있는 모든 블록들의 작업증명을 재작업 해야만 하고 그런 이후에 정직한 노드들의 작업을 따라잡고 능가해야만 할것이다.

We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.
우리는 나중에 더 느린 공격자가 따라잡는 확률이 그 다음의 블록들이 추가됨에 따라 기학급수적으로 사라지는것을 보여줄 것이다.

To compensate for increasing hardware speed and varying interest in running nodes over time, the proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases.
작업증명의 난이도는 시간당 블록들의 평균 개수를 대상으로하는 이동평균에 의해 결정되는데 이는 시간이 흐름에 따라 노드들을 실행하는데 있어 증가하는 하드웨어 속도와 변동하는 이자를 조정하기 위한 것이다. 만약 그들(블록들)이 너무 빨리 생성되면 난이도는 증가한다.

5. Network
5. 네트워크

The steps to run the network are as follows:

네트워크를 실행하는 단계는 다음과 같다

1) New transactions are broadcast to all nodes.
2) Each node collects new transactions into a block.
3) Each node works on finding a difficult proof-of-work for its block.
4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
5) Nodes accept the block only if all transactions in it are valid and not already spent.
6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

1) 네트워크 거래들은 모든 노드들에게 전파(공개) 된다.(뿌려진다)
2) 각 노드는 하나의 블록으로 새로운 거래들을 수집한다.
3) 각 노드는 그 블록에 대해 어려운 작업증명을 찾는것에 착수한다.
4) 한 노드가 작업증명을 찾았을 때 그 노드는 모든 노드들에게 그 블록을 전파(공개)한다.
5) 노드들은 블록에 있는 모든 거래들이 유효하고 이미 사용되지 않았을때 만 그 블록을 받아 들인다. (승인한다.)
6) 노드들은 이전 해시로서 받아들여진 (승인된) 블록의 해시를 사용 하여 연결고리에 다음블록을 생성하는 작업을 함으로서 블록의 받아들임(승인)을 표현한다.

Nodes always consider the longest chain to be the correct one and will keep working on extending it.
노드들은 항상 가장 긴 연결고리를 정확한 것으로 생각하고 그 것을 확장하는 작업을 지속할 것인다.

If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer.
만약 두 노드가 동시에 다음 블록의 다른 버전을 전파(공개) 한다면 어떤 노드들은 하나 또는 다른것을 먼저 받을 수 있다. 그런 경우 그들이 받았던 첫번째 블록에 대한 작업을 하지만 그것이 더 길게될 경우 다른 분기를 저장한다

The tie will be broken when the next proof- of-work is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one.
그 끈은 다음 작업증명이 발견되고 한 분기가 더 길어질때 깨질것이다; 다른 분기에 대한 작업을 했던 노드들은 그 이후에 더 긴것(블록)으로 바뀐다.

New transaction broadcasts do not necessarily need to reach all nodes. As long as they reach many nodes, they will get into a block before long.

새로운 거래 전파(공개)는 모든 노드들에 반드시 도달할 필요는 없다. 그들(새로운 거래의 전파(broadcasts))이 많은 노드들에 도달하는한 오래지 않아 그들은 한 블록안으로 들어 갈것이다.

Block broadcasts are also tolerant of dropped messages. If a node does not receive a block, it will request it when it receives the next block and realizes it missed one.
블록 전파(공개)는 또한 잃어버린 메시지에 대한 내성이 있다. 만약 노드가 블록을 받지 못한다면 그 노드는 다음 블록을 받고 잃어버린 블록을 인식 할때 해당 블록을 요청 할것이다.

6. Incentive
6. 인센티브 (장력책)

By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block.
관례상 블록에 있는 첫번째 거래는 그 블록의 생성자가 소유한 새로운 코인을 출발시키는 특별한 거래 이다.

This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them.
이것은 노드가 네트워크를 지원하기위해 인센티브를 추가하는 것으로 초기에 코인들을 유통으로 분배하는 방법을 제공한다. 왜냐면 코인을 발행하는 중앙 기관이 없기 때문이다.

The steady addition of a constant of amount of new coins is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended.
새로운 코인의 꾸준한 양의 안정적인 추가는 금 채굴자가 유통에 금을 추가하기 위해 자원들을 늘리것과 유사하다. 우리의 경우 소비되는 CPU 시간과 전력이다.

The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.
인센티브는 또한 거래 수수료로 투자될 수 있다. 만약 거래의 결과값이 입력값 보다 작으면 그 차이가 거래를 포함하고 있는 블록의 인센티브 값에 부가되는 거래 수수료이다. 한번 미리 정해진 코인의 수가 유통되었다면 인센티브는 전적으로 거래 수수료로 전환될 수 있고 완전하게 인플레이션에서 자유로울 수 있다.

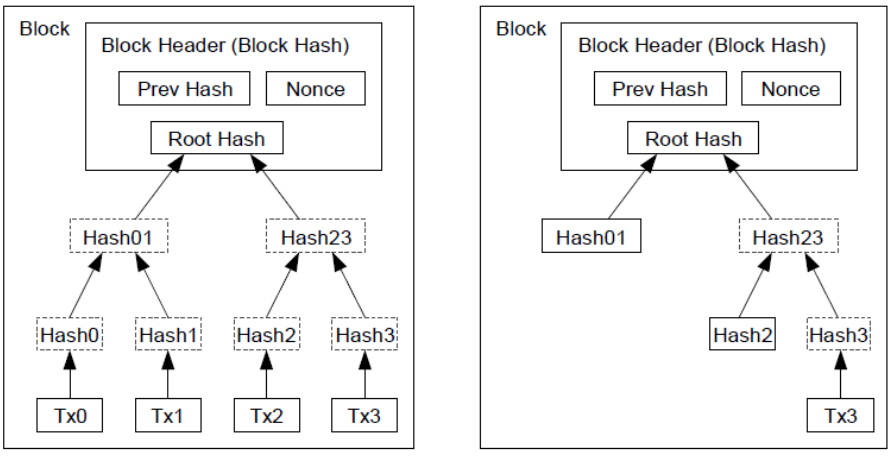
The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

인센티브는 노드들이 정직함을 유지하도록 독려하는것을 도울 수 있다. 만약 탐욕스런 공격자가 모든 정직한 노드들 보다 더 많은 CPU 파워를 모을수 있다면 그는 자신의 지불금을 다시 훔침으로서 사람을 속이는데 그것을 사용할지 새로운 코인들을 생산하는데 사용할지 선택 해야 할것이다. 그는 그것이 다른사람들이 결합 코인보다 더많은 새로운 코인으로 그에게 호의를 보여주는 그와 같은 규칙들로 플레이 하는것이 시스템과 그 자신의 부의 유효성을 훼손시키는 것 보다 더 이익이 된다는것 발견 해야 한다

7. Reclaiming Disk Space
7. 디스크공간 회수하기

Once the latest transaction in a coin is buried under enough blocks, the spent transactions before it can be discarded to save disk space. To facilitate this without breaking the block's hash, transactions are hashed in a Merkle Tree [7][2][5], with only the root included in the block's hash. Old blocks can then be compacted by stubbing off branches of the tree. The interior hashes do not need to be stored.

하나의 코인에 있는 마지막 거래가 충분한 많은 블록들에 묻혀 있다면 그전에 사용된 거래들은 디스크 공간을 절약하기 위해 폐기 될수 있다. 블록의 해시를 파괴하지 않고 이것을 가능(용이)하게 하기 위해 거래들은 블록의 해시에 오직 루트만 포함된 머클 트리(Merkle Tree)에 해시된다. 오래된 블록은 그 이후 트리의 가지를 잘라냄으로서 압축될 수 있다. 내부 해시들은 저장될 필요가 없다.



Transactions Hashed in a Merkle Tree

After Pruning Tx0-2 from the Block

A block header with no transactions would be about 80 bytes. If we suppose blocks are generated every 10 minutes, 80 bytes * 6 * 24 * 365 = 4.2MB per year. With computer systems typically selling with 2GB of RAM as of 2008, and Moore's Law predicting current growth of 1.2GB per year, storage should not be a problem even if the block headers must be kept in memory.

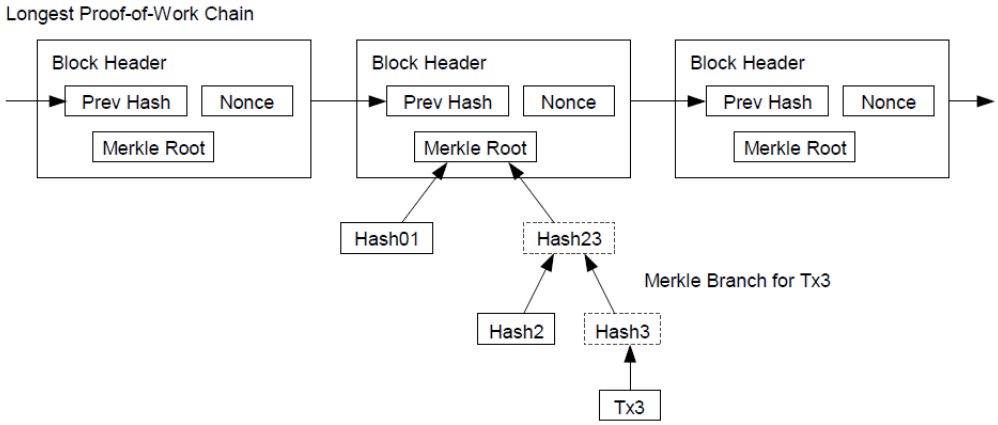
거래가 없는 블록 헤드는 약 80 바이트가 될것이다. 만약 블록들이 매 10 분 마다 생성된다고 가정하면 매년 80bytes * 6 * 24 * 354 = 4.2MB 가 된다. 2008 년 부터 2G 램으로 통상적으로 판매되는 컴퓨터 시스템과 매년 1.2G 의 현재 성장을 예측하는 무어의 법칙으로 보면 심지어 블록의 헤드들이 메모리에 보존되어야 한다고 하더라도 스토리지는 문제가 되지 않는다

8. Simplified Payment Verification

8. 단순화된 지불 검증

It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in. He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it.

전체 네트워크 노드를 실행하지 않고 지불을 검증하는것이 가능하다. 사용자는 가장 긴 연결고리를 가지고 있고 거래가 타임스탬프가 찍혀있는 블록에 연결되어 있는 머클 브랜치를 얻었다는 확신이 들때 까지 네트워크 노드들을 쿼리함으로서 얻을수 있는 가장긴 작업증명 연결고리의 블록의 헤드들의 복사본을 유지하는것만 필요하다. 사용자는 스스로 거래를 체크할 수 없지만 연결고리의 위치에 그것을 연결하는것으로 네트워크 노드가 그것을 받아들였는다는(승인했다는) 것을 볼 수 있고 그 뒤에 추가된 블록들은 네트워크가 그것을 받아 들인것(승인한것)을 더욱 확신 한다.



As such, the verification is reliable as long as honest nodes control the network, but is more vulnerable if the network is overpowered by an attacker. While network nodes can verify transactions for themselves, the simplified method

can be fooled by an attacker's fabricated transactions for as long as the attacker can continue to overpower the network. One strategy to protect against this would be to accept alerts from network nodes when they detect an invalid block, prompting the user's software to download the full block and alerted transactions to confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

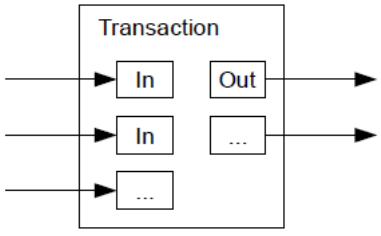
그와 같은것으로 검증은 정직한 노드들이 네트워크를 제어하는한 믿을만 하지만 만약 네트워크가 공격자에의해 장악된다면 더욱 취약 해진다. 반면에 네트워크 노드들은 그들 스스로 거래들을 검증할 수 있다. 간소화된 방법은 공격자가 지속적으로 네트워크를 장악할 수 있는 한 공격자가 꾸며낸 거래에 속을수 있다. 이것을 방지하는 한가지 전략은 네트워크 노드들이 잘못된 블록을 감지 했을 때 전체 블록과 불일치를 확인하는 경고된 거래들을 다운로드하는 사용자 소프트웨어를 뛰우면서 네트워크 노드들로 부터 오는 경고를 받아 들이는게 되는 것이다. 잦은 지불금을 받는 기업들은 더 독립적인 보안과 더 빠른 검증을 위해 아마도 아직 그들 소유의 노드들을 실행하기를 바랄것이다

9. Combining and Splitting Value

9. 결함 및 분할 값

Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.

비록 독립적으로 코인들을 다루는것이 가능하더라도 전송에 있어 매 센트(한푼 한푼)에 대해 개별 거래 만드는것은 다루기 힘들 것이다. 값이 분할되고 결합되는것을 허용하기 위해 거래들은 다중 입력과 출력을 포함한다. 일반적으로 더 큰 이전 거래로부터 하나의 입력 있거나 더 작은 양을 결합하는 다중 입력과 최대 두개의 출력이 있을 것이다. 하나는 지불을 위해서 그리고 하나는 만약있다면 거스름돈을 발신자에게 돌려 주기 위한것 이다.



It should be noted that fan-out, where a transaction depends on several transactions, and those transactions depend on many more, is not a problem here.

There is never the need to extract a complete standalone copy of a transaction's history.

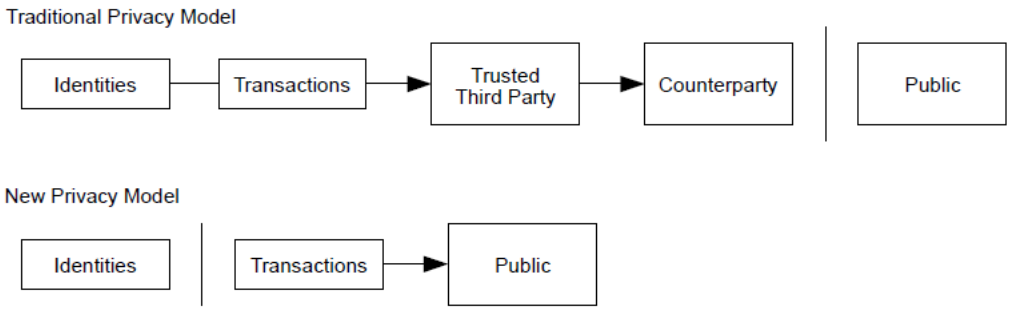
하나의 거래는 여러개의 거래들에 종속되고 그러한 거래들은 더 많은 거래들에 종속되는 팬 아웃은 여기서 문제가 되지 않는다는 것에 유의해야 한다. 거래 이력의 완전한 독립 복사본을 추출 할 필요가 절대 없다.

10. Privacy

10. 개인정보보호

The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to announce all transactions publicly precludes this method, but privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the "tape", is made public, but without telling who the parties were.

전통적인 은행 모델은 관련 당사자들과 제 3 신뢰기관에 대한 정보에 접근을 제한함으로서 개인 정보 보호 수준을 달성 한다. 모든 거래들을 알릴 필요성은 공개적으로 이 모델을 배제하지만 개인정보보호는 다른곳에서 정보의 흐름을 깨는것으로 계속 유지될 수 있다: 익명의 공개키들을 유지함으로서. 대중은 누군가 다른누군가에게 어떤양을 보내는것을 볼수 있지만 누군가에게 거래를 연결하는 정보는 없다. 이것은 증권거래소에 의해 발표된 정보 수준과 유사하다 여기서 개별거래들의 시간과 규모 즉 "테이프"는 세상에 알려지지만(is made public) 당자자들이 누구였는지는 말하지 않는다.



As an additional firewall, a new key pair should be used for each transaction to keep them from being linked to a common owner. Some linking is still unavoidable with multi-input transactions, which necessarily reveal that their inputs were owned by the same owner. The risk is that if the owner of a key is revealed, linking could reveal other transactions that belonged to the same owner.

추가 방화벽 처럼 새로운 키쌍은 공동 소유자에 연결되는것을 유지하기 위해 각각의 거래를 위해 사용되어야 한다. 어떤 연결은 다중입력 거래를 지속적으로 피할수 없다 그것은 그들의 입력들이 동일한 소유자에 의해 소유되었다는것을 필연적으로 들어내는 것이다. 위험은 키의 소유자가 공개되는것으로 연결은 동일한 소유자에 속하는 다른 거래를 공개 할 수 있다.

11. Calculations
11. 계산들

We consider the scenario of an attacker trying to generate an alternate chain faster than the honest chain. Even if this is accomplished, it does not throw the system open to arbitrary changes, such as creating value out of thin air or taking money that never belonged to the attacker. Nodes are not going to accept an invalid transaction as payment, and honest nodes will never accept a block containing them. An attacker can only try to change one of his own transactions to take back money he recently spent.
우리는 정직한 연결고리보다 더 빨리 다른 연결고리를 생성하는 시도를 하는 공격자의 시나리오를 고려한다. 비록 이것이 달성된다 할지라도 시스템을 예를 들어 허수(가짜)(out of thin air) 값을 생성한다거나 공격자에게 결코 속하지 않는 돈을 취하는등의 임의의 변경에 노출되도록 던져 두지는 않는다. 노드들은 지불로서 유효하지 않은 거래를 받아들지(허용하지) 않을 것이고 정직한 노드들은 결코 그들을 포함한 블록을 받아들이지(허용하지) 않은 것다. 공격자들은 다만 그가 최근 사용했던 돈을 되돌리기(되찾기) 위해 그 자신의 거래들중 하나를 변경하려고 시도 할수 있다.

The race between the honest chain and an attacker chain can be characterized as a Binomial Random Walk. The success event is the honest chain being extended by one block, increasing its lead by +1, and the failure event is the attacker's chain being extended by one block, reducing the gap by -1.
정직한 연결고리와 공격자 연결고리간의 경쟁은 이항 랜덤 워크(Binomial Random Walk)로서 특징 지을수 있다. 성공 이벤트는 +1 만큼 우위를 증가시키는 하나의 블록으로 확장되는 정직한 연결고리 이다. 그리고 실패 이벤트는 -1 만큼 갭을 줄이는 블록으로 확장되는 공격자 연결고리 이다.

The probability of an attacker catching up from a given deficit is analogous to a Gambler's Ruin problem. Suppose a gambler with unlimited credit starts at a deficit and plays potentially an infinite number of trials to try to reach breakeven. We can calculate the probability he ever reaches breakeven, or that an attacker ever catches up with the honest chain, as follows [8]:
주어진 적자를 따라 잡는 공격자의 확률은 도박꾼의 파멸문제(Gambler's Ruin problem)와 유사하다. 무제한 신용을 가진 도박꾼이 적자에서 시작하고 손익분기에 도달하려는 시도를 위해 잠재적으로 무한히 시행을 한다고 가정해보자. 우리는 그가

항상 손익분기에 도달하거나 공격자가 항상 정직한 연결고리를 따라 잡는 확률을 다음 [8]과 같이 계산할 수 있다.

p = probability an honest node finds the next block
q = probability the attacker finds the next block
q_z = probability the attacker will ever catch up from z blocks behind

p = 정직한 노드가 다음 블록을 발견할 확률
q = 공격자가 다음 블록을 발견할 확률
qz = 공격자가 z 블록들 뒤에서 부터 따라 (다음블록을) 잡을 확률

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

Given our assumption that $p > q$, the probability drops exponentially as the number of blocks the attacker has to catch up with increases. With the odds against him, if he doesn't make a lucky lunge forward early on, his chances become vanishingly small as he falls further behind.
p > q 라고 가정 하면 공격자가 따라잡아야 하는 블록의 수가 증가함에 따라 확률은 기하급수적(exponentially)으로 떨어 진다. 그에 대한 확율로 만약 그가 초기에 운좋게 앞으로 돌진하지 못한다면 그의 기회는 그가 더욱 뒤 처져 아주 작게 된다.

We now consider how long the recipient of a new transaction needs to wait before being sufficiently certain the sender can't change the transaction. We assume the sender is an attacker who wants to make the recipient believe he paid him for a while, then switch it to pay back to himself after some time has passed. The receiver will be alerted when that happens, but the sender hopes it will be too late.

우리는 지금 발신자가 새로운 거래를 변경할 수 없다고 충분히 확신하게 되기전까지 새로운 거래의 수신자가 얼마나 오래동안 기다릴 필요가 있는지 고려하고 있다. 우리는 그가 잠시동안 그(수신자)에게 지불 했다는것을 수신자가 믿게 하고 약간의 시간이 지난후에 그자신(발신자)에게 되돌려 주도록 거래를 전환하기를 원하는 발신자를 공격자라고 가정한다. 수신자는 그것이 일어 났을때 경고를 받게 될것이지만 발신자는 그것(수신자가 경고를 받는게 되는것)이 아주 늦게 되기를 바란다.

The receiver generates a new key pair and gives the public key to the sender shortly before signing.
수신자는 새로운 키쌍을 생성하고 사인하기 바로전에 발신자에게 공개키를 제공한다.

This prevents the sender from preparing a chain of blocks ahead of time by working on it continuously until he is lucky enough to get far enough ahead, then executing the transaction at that moment.
이것은 그가 충분히 앞서 성공하는 행운이 있을때 까지 지속적으로 그것에 대한 작업(새로운 거래를 변경하는 작업)을 함으로서 사전에 블록의 연결고리를 준비하는 것으로 부터 발신자를 방지 한다음 그 순간에 거래를 실행한다

Once the transaction is sent, the dishonest sender starts working in secret on a parallel chain containing an alternate version of his transaction.
한번 거래가 전송되면 부정직한 발신자는 그의 거래의 대체버전을 담고 있는 병행 연결고리에 비밀리에 작업을 시작한다.

The recipient waits until the transaction has been added to a block and *z* blocks have been linked after it. He doesn't know the exact amount of progress the attacker has made, but assuming the honest blocks took the average expected time per block, the attacker's potential progress will be a Poisson distribution with expected value:

수신자는 거래가 한 블록에 추가되고 z 블록들이 그 뒤에 연결될 때 까지 기다린다. 그는 공격자가 만든 정확한 진행 양(블록) 모르지만 정직한 블록들이 블록당 평균 기대 시간이 걸렸다고 가정하면 공격자의 잠재적인 진행은 다음과 같은 기대값을 가진 포아송(Poisson) 분포를 이룰것 이다.

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

To get the probability the attacker could still catch up now, we multiply the Poisson density for each amount of progress he could have made by the probability he could catch up from that point:
공격자가 여전히 지금 따라 잡을수 있는 확률을 얻기 위해 우리는 그가 그 시점에서 부터 따라 잡을수 있는 확률로 만든 진행의 각 양에 대해 포아송 밀도를 곱한다.

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

Rearranging to avoid summing the infinite tail of the distribution...
분포의 무한한 꼬리를 합산하는것을 피하기 위한 재정리...

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

Converting to C code...

C 코드로 변환하기...

```
#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;
}
```

Running some results, we can see the probability drop off exponentially with z.

연속되는 일부 결과들에 대해서 우리는 z 에 따라 기하급수적으로 줄어드는 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
```

Solving for P less than 0.1%...

0.1% 보다 적은 P 값에 대한 풀이...

```
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. Conclusion

12. 결과

We have proposed a system for electronic transactions without relying on trust. We started with the usual framework of coins made from digital signatures, which provides strong control of ownership, but is incomplete without a way to prevent double-spending.

우리는 신뢰에 의존하지 않는 전자거래에 대한 시스템을 제안 했다. 우리는 소유권에 대한 강력한 제어를 제공하는 디지털 서명으로 만들어진 코인의 일반적인 프레임워크로 시작했지만 이중사용을 방지하는 방법없이는 불완전 하다.

To solve this, we proposed a peer-to-peer network using proof-of-work to record a public history of transactions that quickly becomes computationally impractical for an attacker to change if honest nodes control a majority of CPU power. The network is robust in its unstructured simplicity.
이것을 해결하기 위해 우리는 정직한 노드들이 CPU 파워의 대부분을 제어한다면 공격자가 변경하기에는 빠르게 계산적으로 불가능하게 되는 거래들의 공개 이력을 기록하기 위한 작업증명을 사용하는 P2P 네트워크를 제안 했다. 네트워크는 구조화 되지 않은 단순성에 강력함이 있다.

Nodes work all at once with little coordination. They do not need to be identified, since messages are not routed to any particular place and only need to be delivered on a best effort basis.
노드들은 약간의 조정으로 한번에 작업한다. 메시지들이 어떤 특별한 장소로 보내지지 않고 단지 최선의 노력 (모집주선- best effort) 기반으로 전달될 필요가 있기 때문에 그들은 식별될 필요가 없다.

Nodes can leave and rejoin the network at will, accepting the proof-of-work chain as proof of what happened while they were gone. They vote with their CPU power, expressing their acceptance of valid blocks by working on extending them and rejecting invalid blocks by refusing to work on them. Any needed rules and incentives can be enforced with this consensus mechanism.
노드들은 마음대로 네트워크를 떠나고 그들이 없는 동안 일어났던것의 증명으로 가장 긴 작업증명 연결고리를 받아들임으로써 다시 합류할 수 있다. 그들은 CPU 파워로 블록들을 확장하는데 노력을 들이면서 유효한 블록에 대한 허용을 표현하고 블록들에 노력을 들이는것을 거절함으로서 유효하지 않은 블록들을 허용하지 않는 투표를 한다. 필요한 모든 규칙들과 인센티브들(incentives)은 이런 합의 메카니즘으로 실행 될 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..