Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
[redacted]
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. 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. We propose a solution to the double-spending problem using a peer-to-peer network. 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. 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. 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. The network itself requires minimal structure. 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.

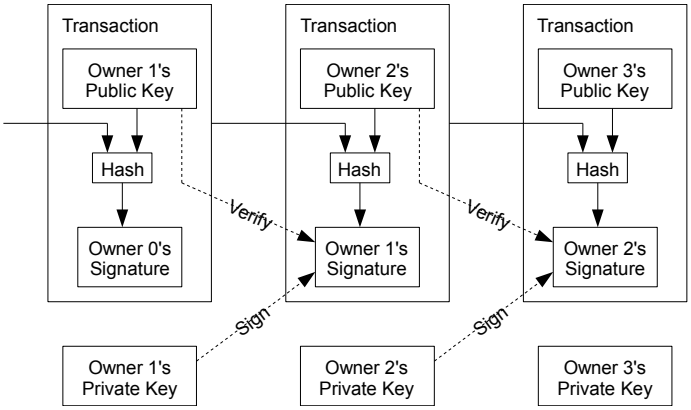
1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. 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. 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. 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.

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. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. 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. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

2. Transactions

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.

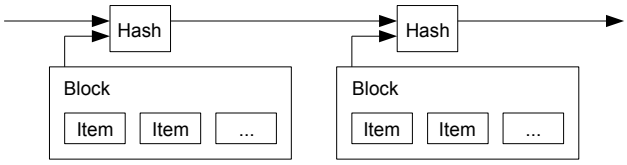


The problem of course is the payee can't verify that one of the owners did not double-spend the coin. A common solution is to introduce a trusted central authority, or mint, that checks every transaction for double spending. After each transaction, the coin must be returned to the mint to issue a new coin, and only coins issued directly from the mint are trusted not to be double-spent. 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.

We need a way for the payee to know that the previous owners did not sign any earlier transactions. For our purposes, the earliest transaction is the one that counts, so we don't care about later attempts to double-spend. The only way to confirm the absence of a transaction is to be aware of all transactions. In the mint based model, the mint was aware of all transactions and decided which arrived first. 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. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

3. Timestamp Server

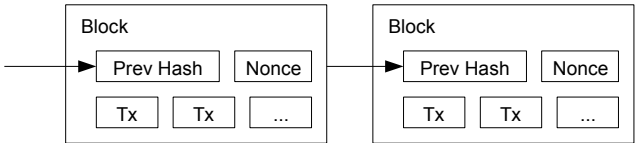
The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5]. The timestamp proves that the data must have existed at the time, obviously, in order to 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.



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

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. 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.



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. Proof-of-work is essentially one-CPU-one-vote. The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. 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. We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.

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.

5. Network

The steps to run the network are as follows:

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.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. 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. 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.

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. 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.

6. Incentive

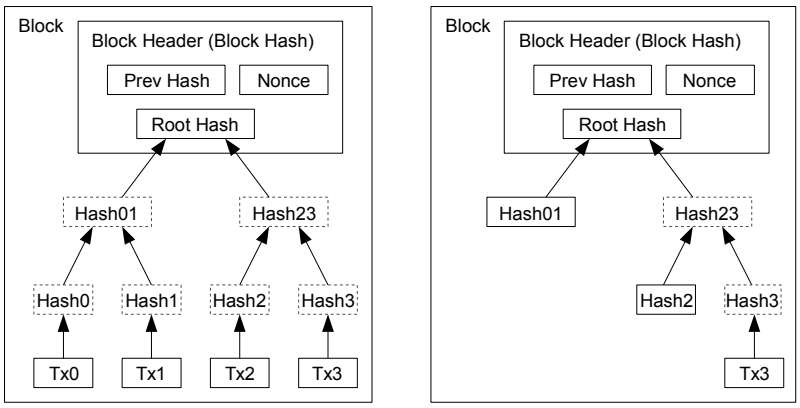
By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. 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.

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.

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.

7. Reclaiming Disk Space

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.



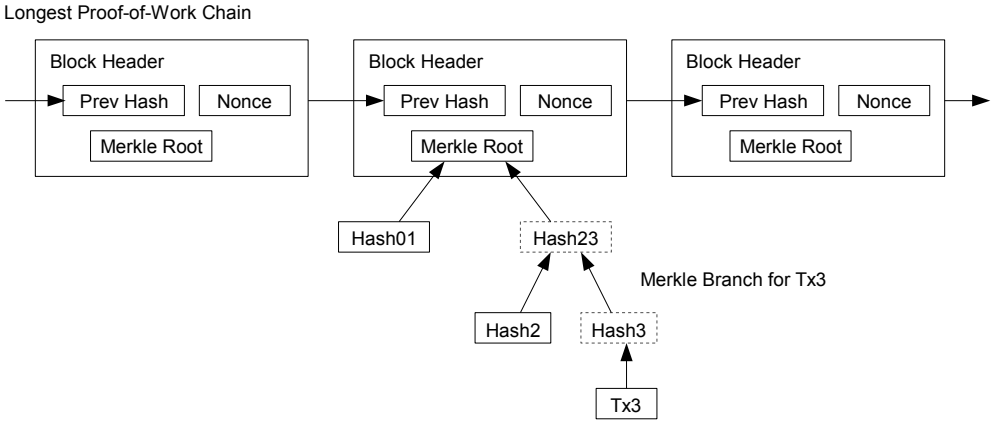
Transactions Hashed in a Merkle Tree

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.

8. Simplified Payment Verification

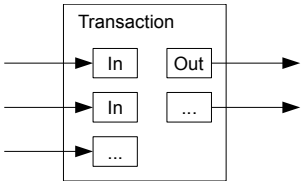
It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction 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.



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.

9. Combining and Splitting Value

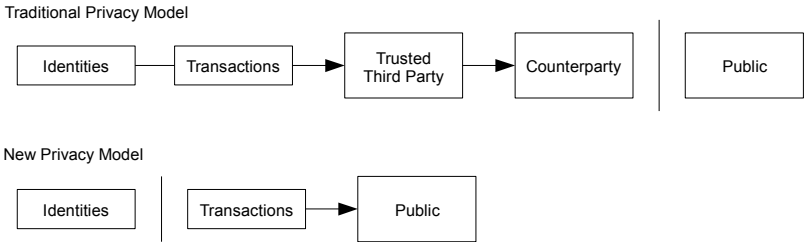
Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous 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.



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.

10. Privacy

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.



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.

11. Calculations

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.

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.

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]:

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

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

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.

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.

The receiver generates a new key pair and gives the public key to the sender shortly before signing. 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. Once the transaction is sent, the dishonest sender starts working in secret on a parallel chain containing an alternate version of his transaction.

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:

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

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:

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

Rearranging to avoid summing the infinite tail of the distribution...

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

Converting to C code...

```
#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.

```
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%...

```
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

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. 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. 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. 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.

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.

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
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. 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. We propose a solution to the double-spending problem using a peer-to-peer network. 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. 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. 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. The network itself requires minimal structure. 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.

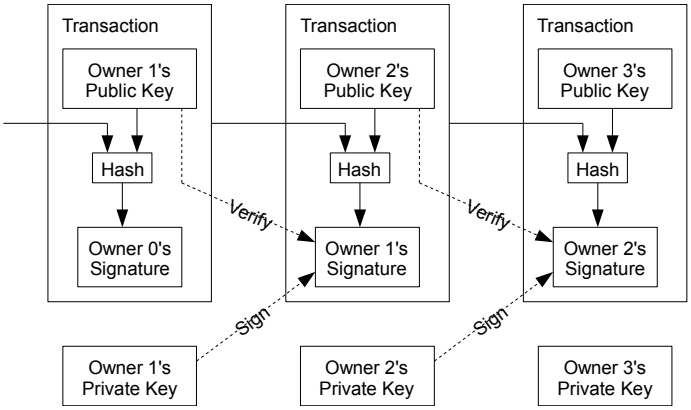
1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. 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. 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. 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.

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. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. 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. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

2. Transactions

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.

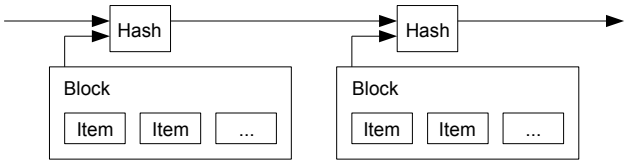


The problem of course is the payee can't verify that one of the owners did not double-spend the coin. A common solution is to introduce a trusted central authority, or mint, that checks every transaction for double spending. After each transaction, the coin must be returned to the mint to issue a new coin, and only coins issued directly from the mint are trusted not to be double-spent. 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.

We need a way for the payee to know that the previous owners did not sign any earlier transactions. For our purposes, the earliest transaction is the one that counts, so we don't care about later attempts to double-spend. The only way to confirm the absence of a transaction is to be aware of all transactions. In the mint based model, the mint was aware of all transactions and decided which arrived first. 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. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

3. Timestamp Server

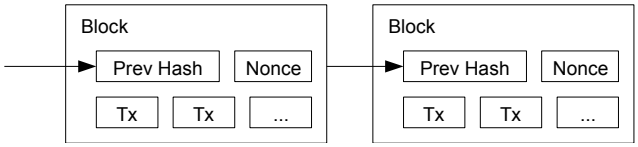
The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5]. The timestamp proves that the data must have existed at the time, obviously, in order to 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.



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

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. 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.



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. Proof-of-work is essentially one-CPU-one-vote. The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. 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. We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.

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.

5. Network

The steps to run the network are as follows:

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.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. 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. 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.

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. 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.

6. Incentive

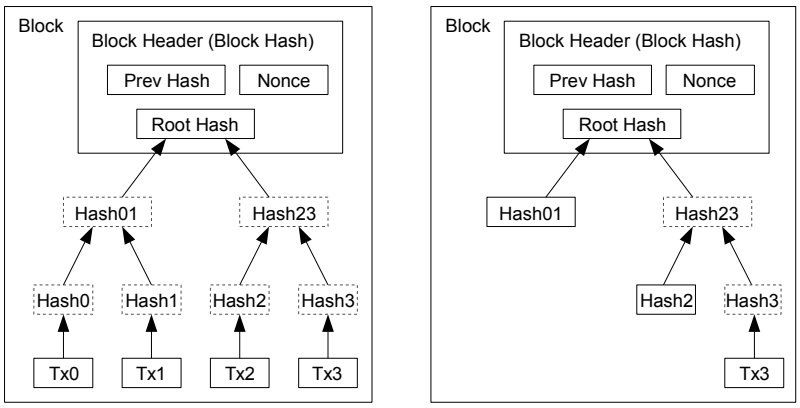
By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. 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.

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.

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.

7. Reclaiming Disk Space

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.



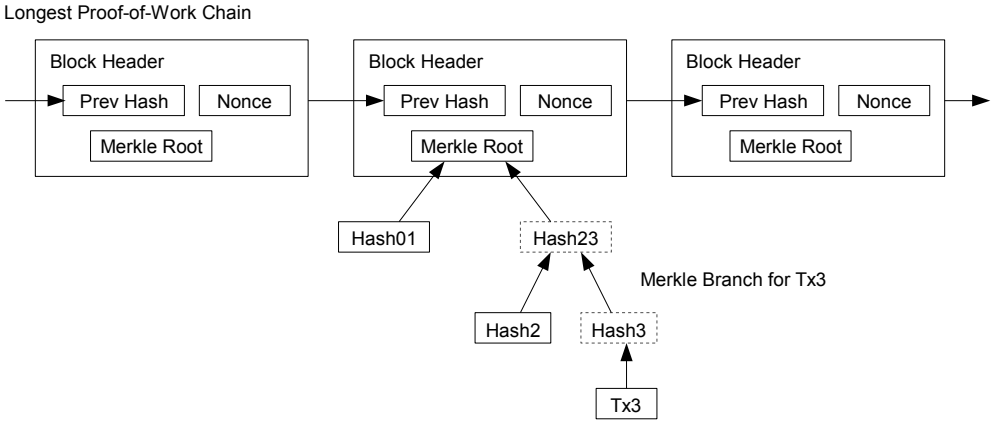
Transactions Hashed in a Merkle Tree

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.

8. Simplified Payment Verification

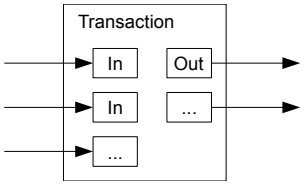
It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction 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.



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.

9. Combining and Splitting Value

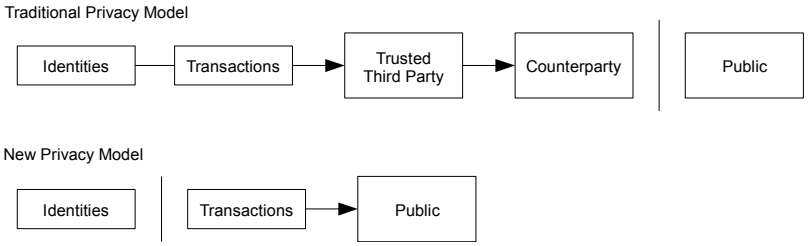
Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous 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.



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.

10. Privacy

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.



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.

11. Calculations

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.

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.

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]:

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

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

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.

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.

The receiver generates a new key pair and gives the public key to the sender shortly before signing. 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. Once the transaction is sent, the dishonest sender starts working in secret on a parallel chain containing an alternate version of his transaction.

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:

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

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:

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

Rearranging to avoid summing the infinite tail of the distribution...

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

Converting to C code...

```
#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.

```
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%...

```
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

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. 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. 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. 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.

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.

2018 National Seminar

Bitcoin Glossary:
2018 Annual National Seminar

Address - A Bitcoin address is similar to a physical address or an email. It is the only information you need to provide for someone to pay you with Bitcoin. An important difference, however, is that each address should only be used for a single transaction. Typically consists of between 26 and 35 alphanumeric characters.

Altcoin - A form of cryptocurrency that has the same decentralized, peer-to-peer principles as bitcoin, but which uses its own blockchain and has its own rules of operation. Altcoin is the term used to describe those digital currencies that do not have as big a market capitalization or do not have the recognition of the current incumbent cryptocurrencies such as bitcoin, litecoin and dogecoin.

ASIC - ASIC stands for application specific integrated circuit, which is a specialized silicon chip that performs just one task. In the digital currency space, these chips process SHA-256 in order to mine bitcoin and validate transactions.

ASIC Miner - An ASIC Miner is the hardware that houses the chip of the same name. You put them into your Internet connection via a modem or wireless mode. Bitcoin is independent of your desktop computer.

Bit - Bit is a common unit used to designate a sub-unit of a bitcoin - 1,000,000 bits is equal to 1 bitcoin (BTC or ฿). This unit is usually more convenient for pricing, tips, goods and services.

Bitcoin / BTC (shorthand) - A form of digital currency created in 2009, that is created and distributed on a peer-to-peer basis. It has no central bank - transactions are conducted directly between individuals. Bitcoin is the most popular kind of cryptocurrency.

Bitcoin Index - The live bitcoin news bitcoin index is a weighted average index that shows the value of one bitcoin versus one single unit of currency of each of the majors in the Forex space – EUR, USD, JPY, GBP and AUD.

Bitcoin Whitepaper - Written by Satoshi Nakamoto in 2008, it describes the original plan and protocol for Bitcoin.

BitPay - BitPay is a payment processing company and software that allows merchants such as eBay, Amazon and other online shopping channels to accept bitcoin as payment for its goods and services.

Block - A block is a record in the block chain that contains and confirms many waiting transactions. Roughly every 10 minutes, on average, a new block including transactions is appended to the block chain through mining.

Block Reward - This term refers to the "reward" that the Miner receives for successfully hashing a transaction block.

Blockchain - A digital file distributed to everyone participating in a crypto-currency network. The blockchain acts as a kind of general ledger, keeping track of all the transactions that happen in the network. Everyone can look at the blockchain to see what transactions have happened on the network, and the blockchain is sealed using cryptography so that no one can tamper with it.

Cold Storage – A security measure for Bitcoin that is disconnected from the internet. Could be a paper wallet [see below], USB stick or hardware wallet.

Confirmation - Confirmation means that a transaction has been processed by the network and is highly unlikely to be reversed. Transactions receive a confirmation when they are included in a block and for each subsequent block. Even a single confirmation can be considered secure for low value transactions, although for larger amounts like $1,000 US, it makes sense to wait for 6 confirmations or more. Each confirmation exponentially decreases the risk of a reversed transaction.

Cryptocurrency - The broad name for digital currencies that use blockchain technology to work on a peer-to-peer basis. Cryptocurrencies don't need a bank to carry out transactions between individuals. The nature of the blockchain means that individuals can transact between each other, even if they don't trust each other. The cryptocurrency network keeps track of all the transactions and ensures that no one tries to renege on a transaction.

Cryptography - Cryptography is the branch of mathematics that lets us create mathematical proofs that provide high levels of security. Online commerce and banking already uses cryptography. In the case of Bitcoin, cryptography is used to make it impossible for anybody to spend funds from another user's wallet or to corrupt the block chain. It can also be used to encrypt a wallet, so that it cannot be used without a password.

Dogecoin - An altcoin first started as a joke in late 2013. Dogecoin, which features a Japanese fighting dog as its mascot, gained a broad international following and quickly grew to have a multi-million dollar market capitalization.

Double Spend - If a malicious user tries to spend their bitcoins with two different recipients at the same time, this is double spending. Bitcoin mining and the block chain are there to create a consensus on the network about which of the two transactions will confirm and be considered valid.

Exchange - An exchange is exactly how it sounds, somewhere where account holders can exchange one digital currency for another or a Fiat currency for a digital currency.

Faucet - When an individual or team of individuals develop a digital currency, they may pre-mine a certain amount before release and give these pre-mined coins away. This is called a faucet.

FIAT - A Fiat currency is a traditional paperback currency that is regulated by an organization such as the central bank. Examples include the Euro, the US dollar and the Australian dollar.

Bitcoin Glossary:
2018 Annual National Seminar

National Seminar

Genesis Block - The very first block in the block chain of any digital currency.

Hash - A cryptographic hash is a mathematical function that takes a file and produces a relatively short code that can be used to identify that file. A hash has a couple of key properties: It is unique. Only a particular file can produce a particular hash, and two different files will never produce the same hash. It cannot be reversed. You can't work out what a file was by looking at its hash. Hashing is used to prove that a set of data has not been tampered with. It is what makes bitcoin mining possible.

Hash Rate - The hash rate is the measuring unit of the processing power of the Bitcoin network. The Bitcoin network must make intensive mathematical operations for security purposes. When the network reached a hash rate of 10 Th/s, it meant it could make 10 trillion calculations per second.

Microtransaction – The ability to pay for things in very small sums thanks to the fact that Bitcoin may be extended to 8 decimal places. Microtransactions are especially important to Bitcoin casinos by providing players the ability to deposit and gamble fractions of Bitcoins.

Mining - The act of producing units of a cryptocurrency (such as bitcoins) through some kind of effort. The effort is required so that people can't just create infinite amounts of the digital currency, which would devalue it. In bitcoin, mining requires computing power. Here is a detailed description of how mining works. Bitcoin mining is the process of making computer hardware do mathematical calculations for the Bitcoin network to confirm transactions and increase security. As a reward for their services, Bitcoin miners can collect transaction fees for the transactions they confirm, along with newly created bitcoins. Mining is a specialized and competitive market where the rewards are divided up according to how much calculation is done. Not all Bitcoin users do Bitcoin mining, and it is not an easy way to make money.

Mt. Gox – one of the first Bitcoin exchanges that began liquidating after more than 850,000 of its users' Bitcoins were lost or stolen – an amount equal to more than $450,000,000 at the time.

Output - When a bitcoin transaction takes place, the output refers to the destination address used in the transaction.

Paper Wallet - Some people prefer to store their bitcoin in the paper wallet – a form of cold storage – in order to improve security. The term simply refers to a printed sheet of paper that holds a number of public bitcoin addresses and corresponding private keys.

P2P - Peer-to-peer refers to systems that work like an organized collective by allowing each individual to interact directly with the others. In the case of Bitcoin, the network is built in such a way that each user is broadcasting the transactions of other users. And, crucially, no bank is required as a third party.

Private Key - A private key is a secret piece of data that proves your right to spend bitcoins from a specific wallet through a cryptographic signature. Your private key(s) are stored in your computer if you use a software wallet; they are stored on some remote servers if you use a web wallet. Private keys must never be revealed as they allow you to spend bitcoins for their respective Bitcoin wallet.

Proof of Work [PoW] - Proof of work simply refers to the output of any efforts to mine bitcoin. In the bitcoin block chain, the hashing of a block takes time and effort, meaning the hash block can be considered proof of work.

Public key - The public key is a string of digits and letters (your bitcoin address). When hashed with a corresponding string known as a private key it digitally signs and online communication.

Satoshi – A Bitcoin "cent", the smallest form of Bitcoins. One Bitcoin is equal to 1 million Satoshis.

Satoshi Nakamoto – the creator of Bitcoin and the author of the original Bitcoin whitepaper and code. His real identity is unknown to the world.

Silk Road – An underground website, as part of the "dark web", that was essentially a black market online. One could purchase illegal drugs, organs or hire assassins online. The site used cryptocurrencies such as Bitcoin and was shut down in 2013 by the FBI.

SHA-256 - Every digital currency must have a cryptographic function that dictates how the hash is constructed. In bitcoin, SHA-256 is this function, and is used as the basis for hash creation (*i.e.* bitcoin's proof of work).

Signature - A cryptographic signature is a mathematical mechanism that allows someone to prove ownership. In the case of Bitcoin, a Bitcoin wallet and its private key(s) are linked by some mathematical magic. When your Bitcoin software signs a transaction with the appropriate private key, the whole network can see that the signature matches the bitcoins being spent. However, there is no way for the world to guess your private key to steal your hard-earned bitcoins.

Transaction Fee - Some transactions that occur in the bitcoin block chain contain transaction fees. These transaction fees are paid to the miner that hashes the block in question.

Wallet - A Bitcoin wallet is loosely the equivalent of a physical wallet on the Bitcoin network. The wallet actually contains your private key(s) which allow you to spend the bitcoins allocated to it in the block chain. Each Bitcoin wallet can show you the total balance of all bitcoins it controls and lets you pay a specific amount to a specific person, just like a real wallet. This is different from credit cards where you are charged by the merchant.

**This glossary contains terminology and explanations of concepts relevant to various emerging technologies. The purpose of the glossary is to inform the reader of the most commonly used vocabulary terms in the cyber world. This glossary was compiled from various sources readily available on the Internet.*

To receive updates on future events and other Commission activities, visit us on Twitter @TheUSSCgov, or subscribe to e-mail updates through our website at www.ussc.gov. For guidelines questions, call our Helpline at 202.502.4545, and to request training, email us at training@ussc.gov.

The United States Sentencing Commission, an independent agency in the judicial branch of the federal government, was organized in 1985 to develop a national sentencing policy for the federal courts. The resulting sentencing guidelines provide structure for the courts' sentencing discretion to help ensure that similar offenders who commit similar offenses receive similar sentences.

Blockchain & Money

Class 5

September 20, 2018

Class 5 (9/20): Study Questions

- How does Bitcoin record transactions? What is unspent transaction output (UTXO)? What is script code embedded in each Bitcoin transaction and how flexible a programming language is it?

- As many design features pre-date Bitcoin, what was the novel innovation of Santoshi Nakamoto?

- Who is Satoshi Nakamoto? (Only kidding a bit.)

Class 5 (9/20): Readings

- *'Bitcoin's Academic Pedigree'* Narayanan and Clark

- *'Making Sense of Cryptoeconomics'* CoinDesk

Class 5 Overview

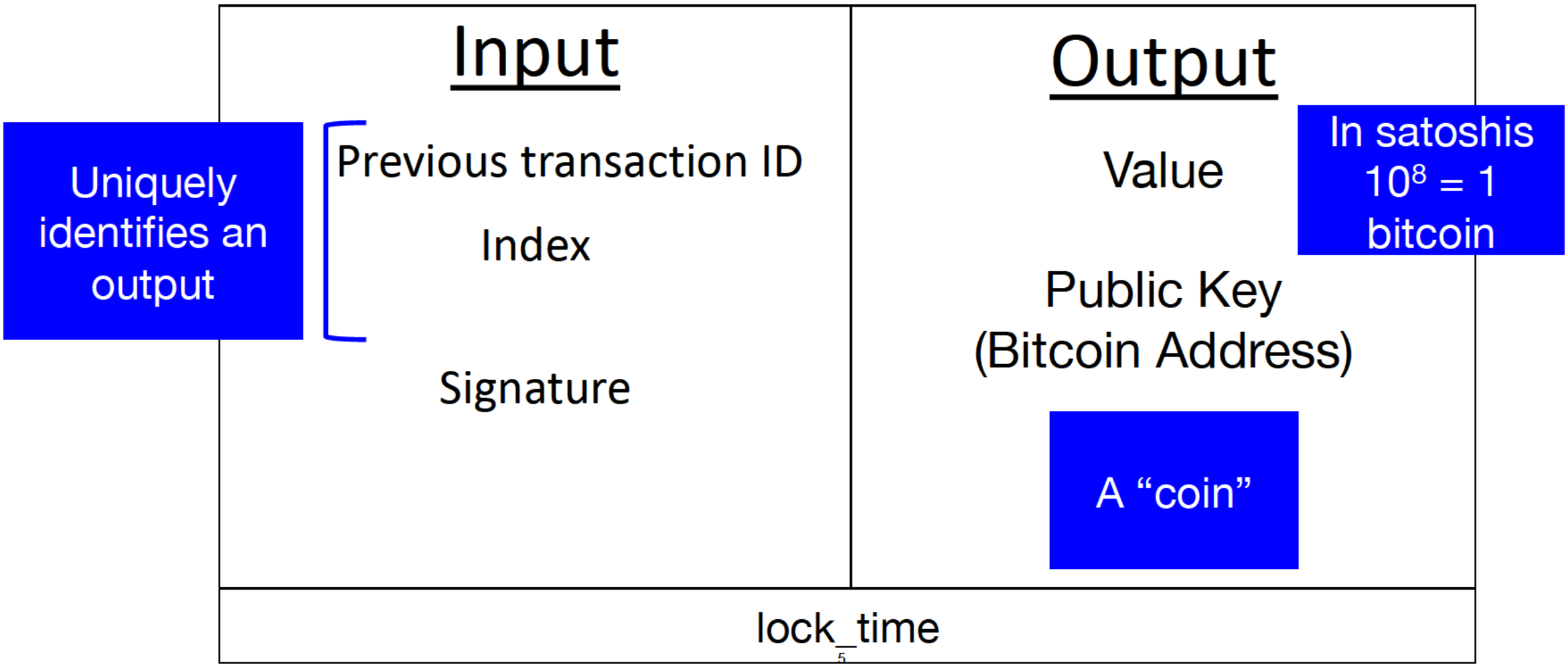
- Transaction Inputs & Outputs
- Unspent Transaction Output (UTXO)
- Scripting language

- Blockchain Design – Putting it All Together
- Bitcoin’s Academic Pedigree

- Who is Satoshi Nakamoto?

- Conclusions

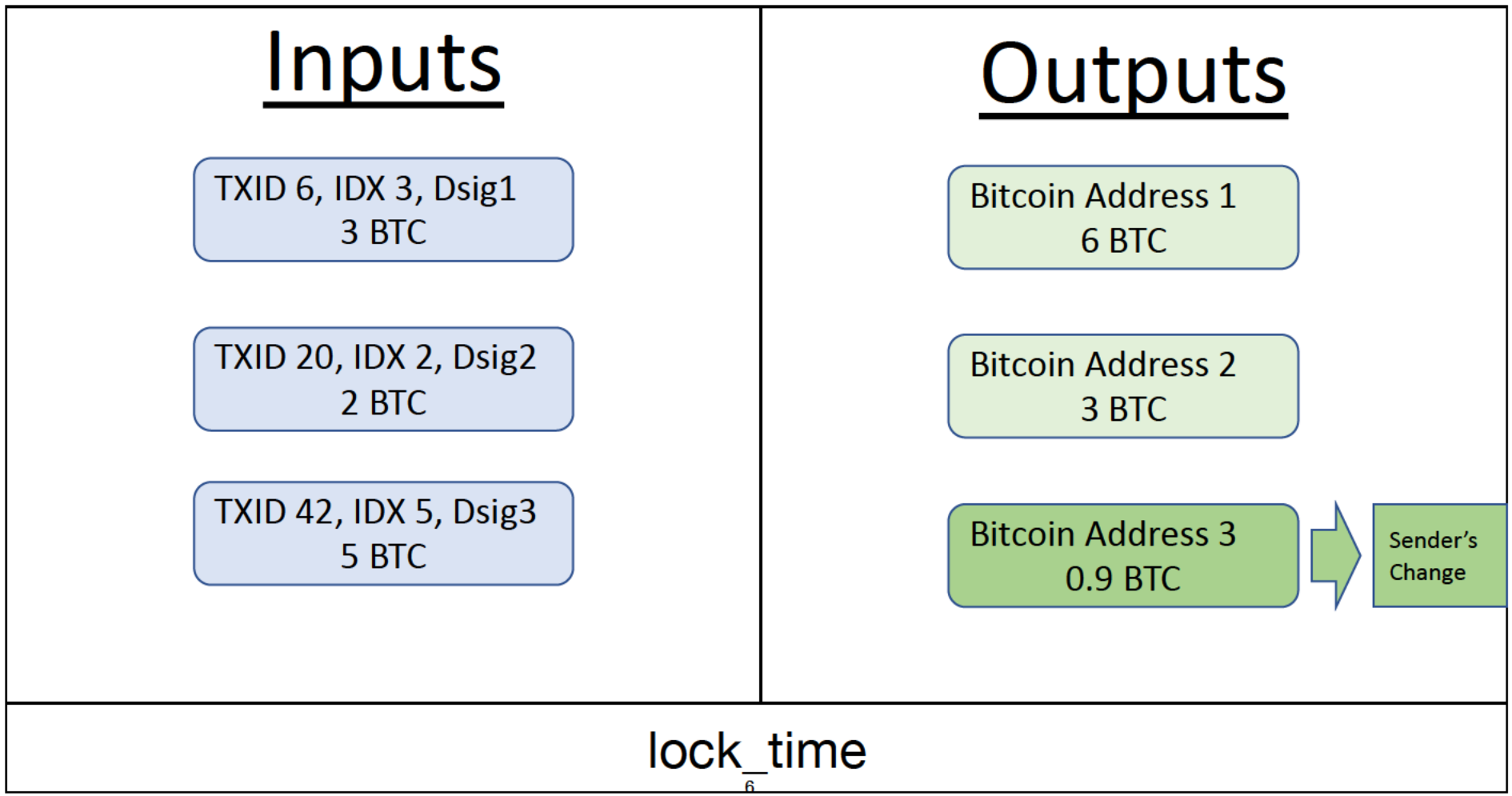
Transaction format



Transaction format

Multiple Inputs & Outputs

Inputs ≥ Outputs
Inputs – Outputs = Fees



Coinbase Transaction

Reward for Solving Proof of Work

- Only Input is the Coinbase Block Reward
- Reward halves (1/2s) every 210,000 blocks
  - Currently 12.5 Bitcoins per block
  - Originally 50 Bitcoin per block
- Output may not be used as a Transaction Input until another 100 Blocks
- Recorded as First Transaction in Merkle Tree
- May Include 100 bytes of arbitrary data
  - Used for Additional Nonce
  - Genesis Block included Headline from Financial Times:

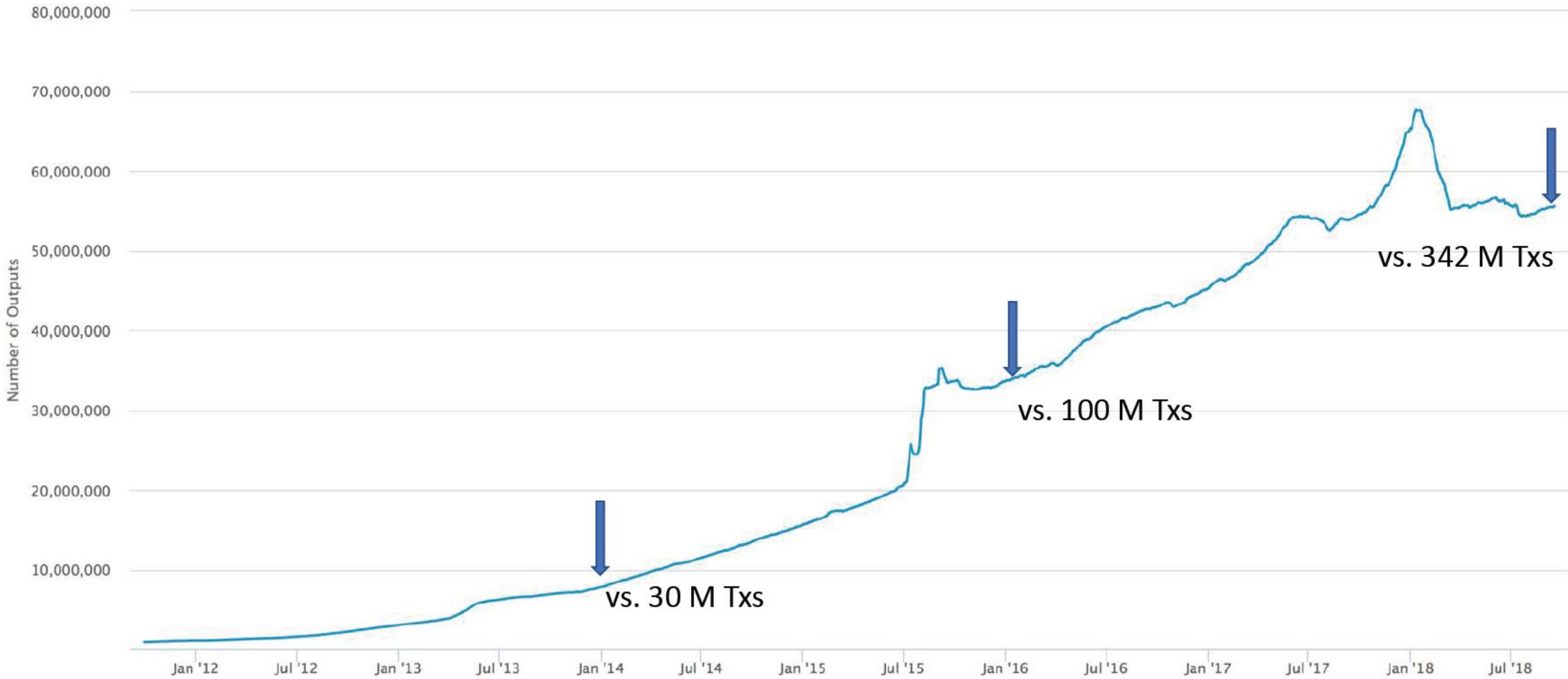
    ‘The Times 03/Jan/2009 Chancellor on brink of second bailout for banks’

Unspent Transaction Output (UTXO) Set

Bitcoin transaction outputs that have not been spent at a given time

- Contains All Currently Unspent Transaction Outputs
- Speeds up Transaction Validation Process
- Stored using a LevelDB database in Bitcoin Core called ‘chainstate’

Unspent Transaction Output (UTXO) Set



Source: Blockchain.com – 9/17/18

Courtesy of Blockchain Luxembourg S.A. Used with permission.

Bitcoin Script

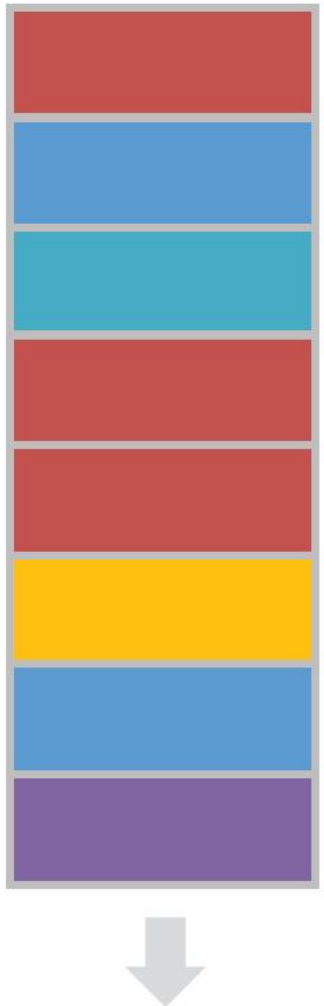
Programing Code used for Transactions

- Stack-based Code, with no Loops (not Turing-complete)
- Provides a Flexible Set of Instructions for Transaction Validation and Signature Authentication
- Most Common Script Types in UTXO:
  - Transaction sent to Hash of Bitcoin Address – ‘Pay-to-PubkeyHash’ (81%)
  - Transaction sent to Hash of Conditional Script – ‘Pay-to-ScriptHash’ (18%)
  - Transaction subject to Multiple Signatures – ‘M of N Multisig’ (0.7%)
  - Transaction sent to Bitcoin Address – ‘Pay-to-Pubkey’ (0.1%)

    (Source: Perez-Sola, Delgado-Segura, et al.)

Blockchain Technology

timestamped
append-only log



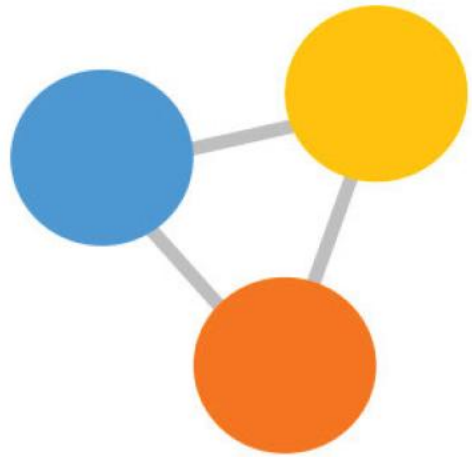
auditable database

Secured via cryptography

- Hash functions for **tamper resistance** and **integrity**
- Digital signatures for **consent**

Consensus for **agreement**

network consensus protocol



Addresses '**cost of trust**' (Byzantine Generals problem)

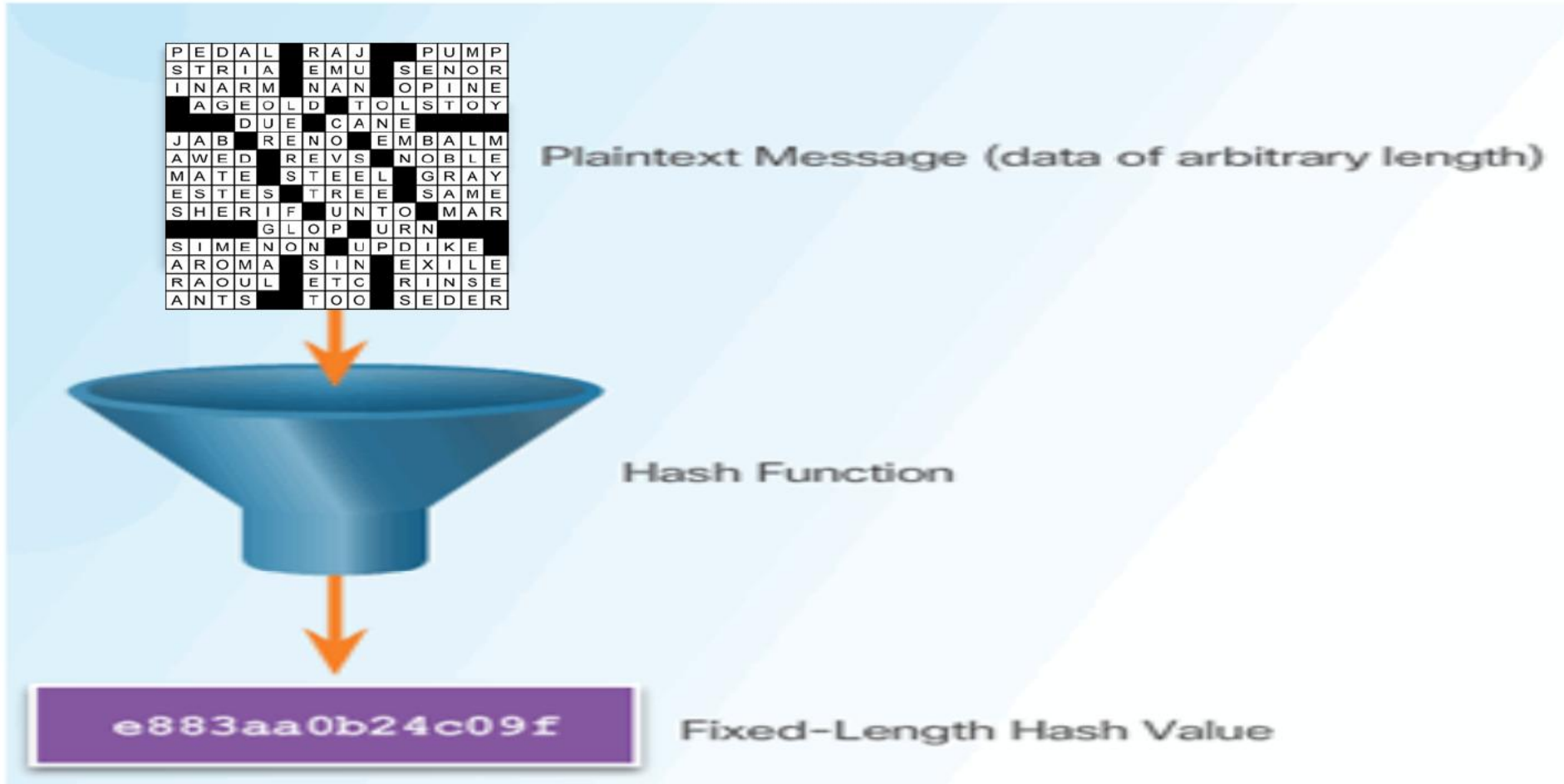
- Permissioned
- Permissionless

Bitcoin – Technical Features

- Cryptography & Timestamped Logs
  - Cryptographic Hash Functions
  - Timestamped Append-only Logs (Blocks)
  - Block Headers & Merkle Trees
  - Asymmetric Cryptography & Digital Signatures
  - Addresses
- Decentralized Network Consensus
  - Proof of Work
  - Native Currency
  - Network
- Transaction Script & UTXO
  - Transaction Inputs & Outputs
  - Unspent Transaction Output (UTXO) set
  - Scripting language

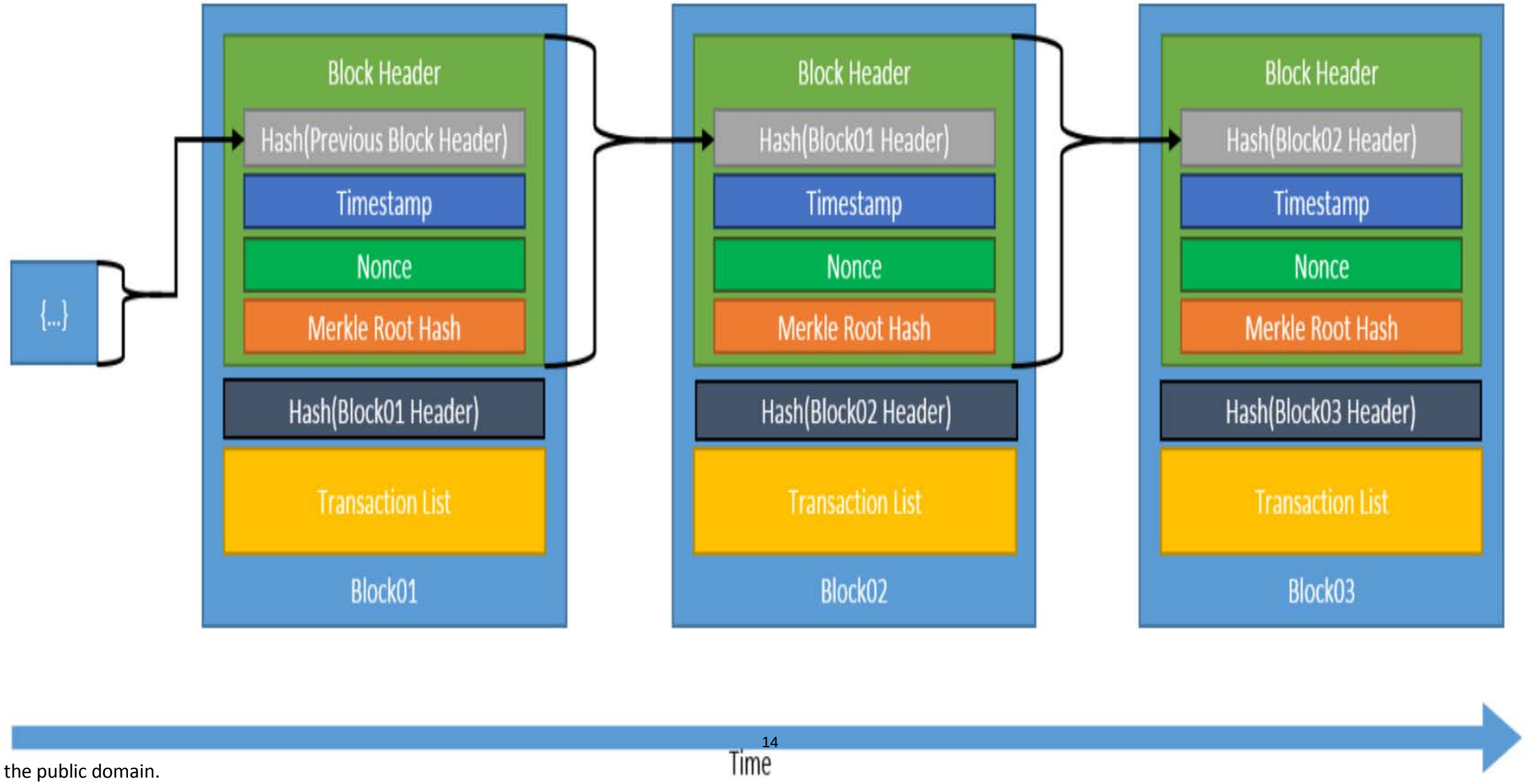
Cryptographic Hash Functions

One-Way Data Compression

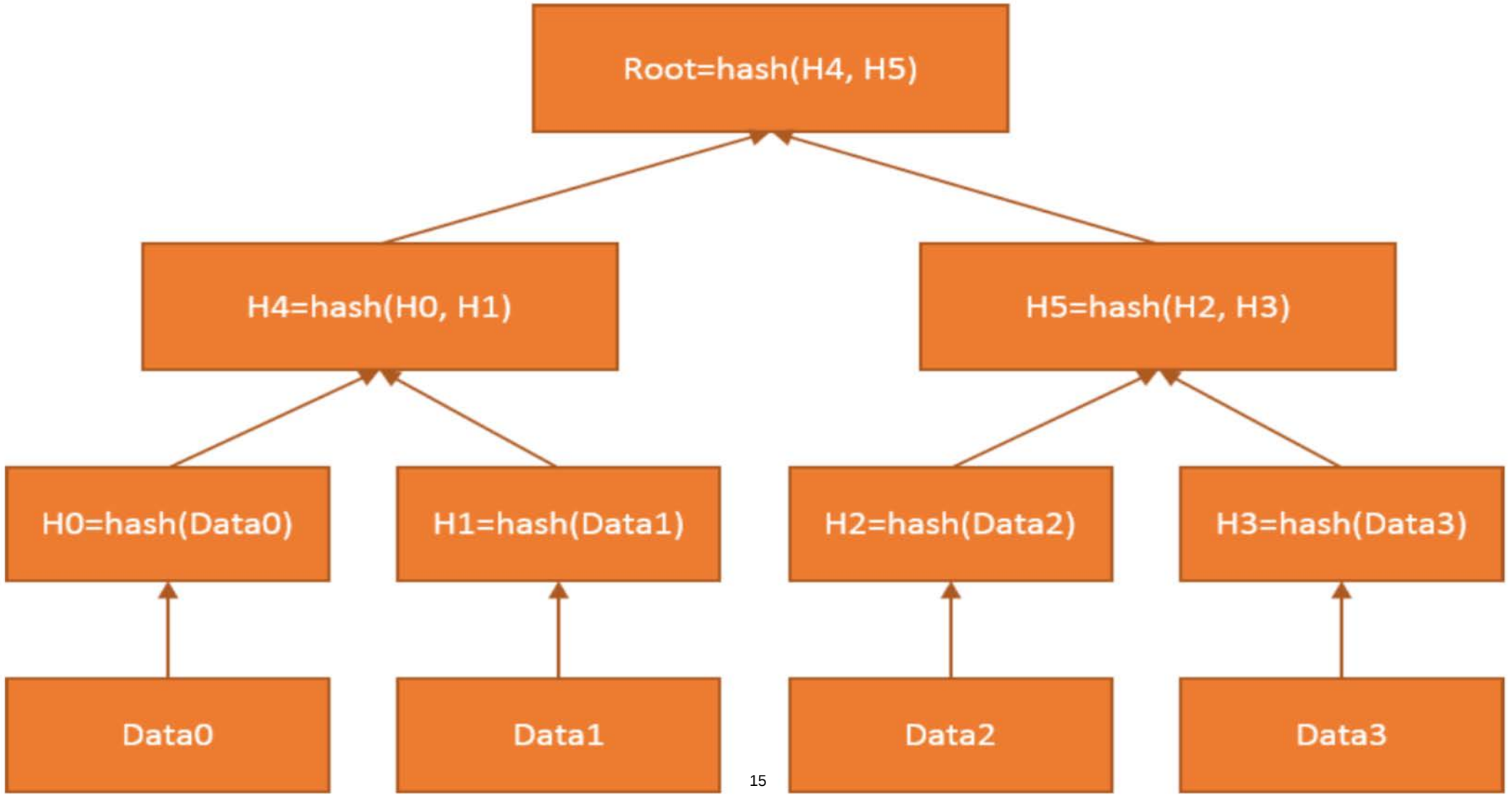


Data Commitment

Timestamped Append-only Log - Blockchain



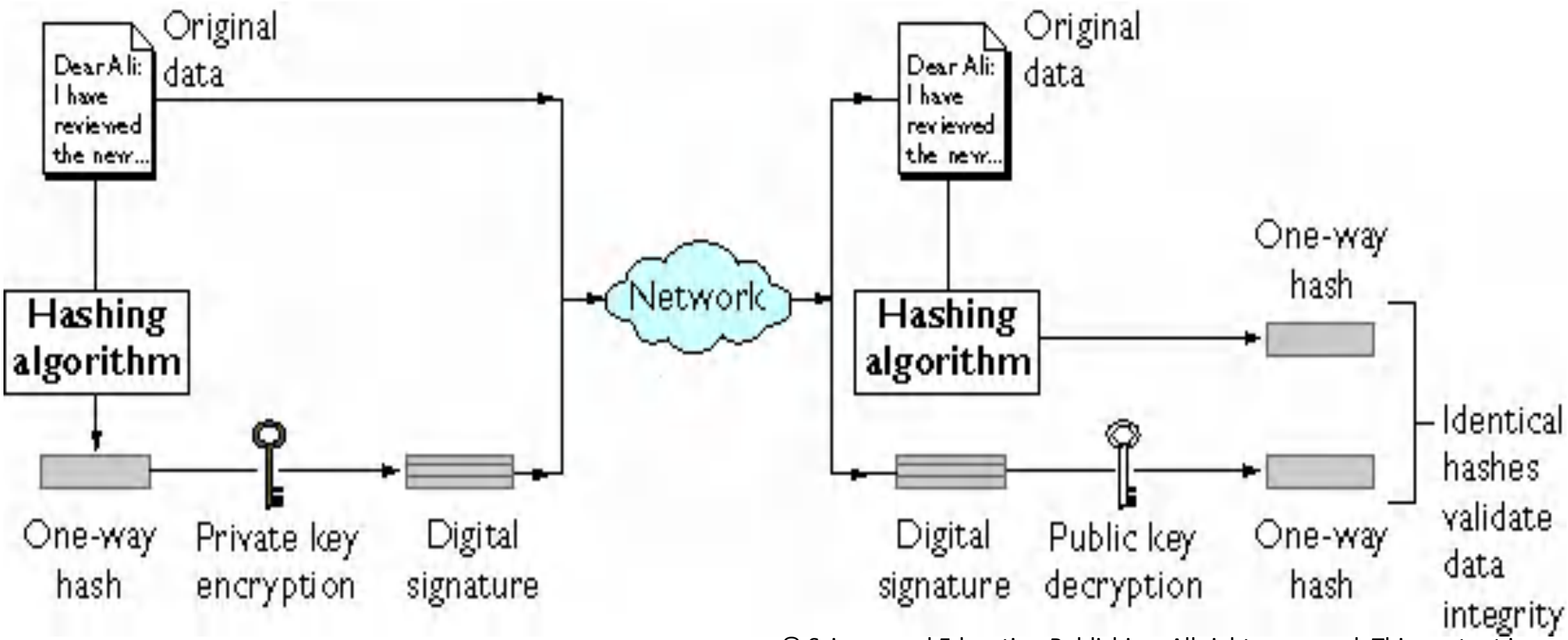
Merkle Tree – Binary Data Tree with Hashes



Asymmetric Cryptography & Digital Signatures

Guarding against Tampering & Impersonation

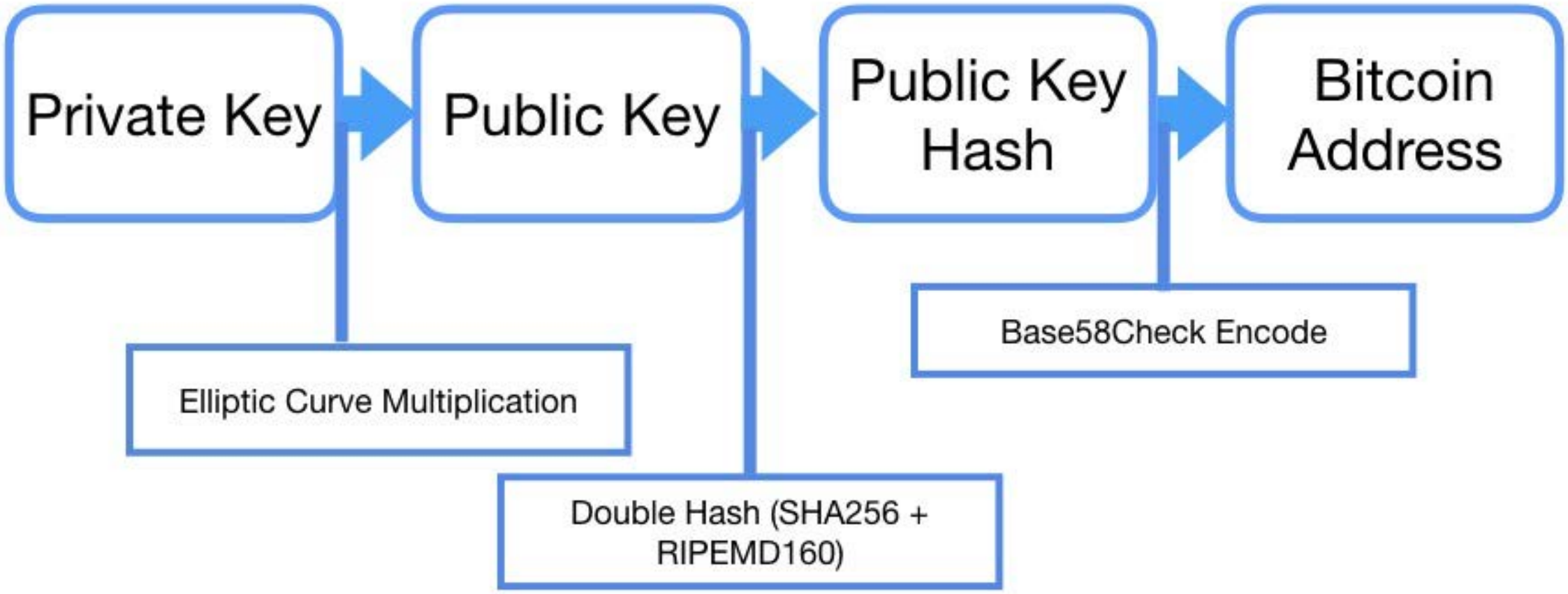
Digital Signature with Hash



Shyam Nandan Kumar et al. Review on Network Security and Cryptography.

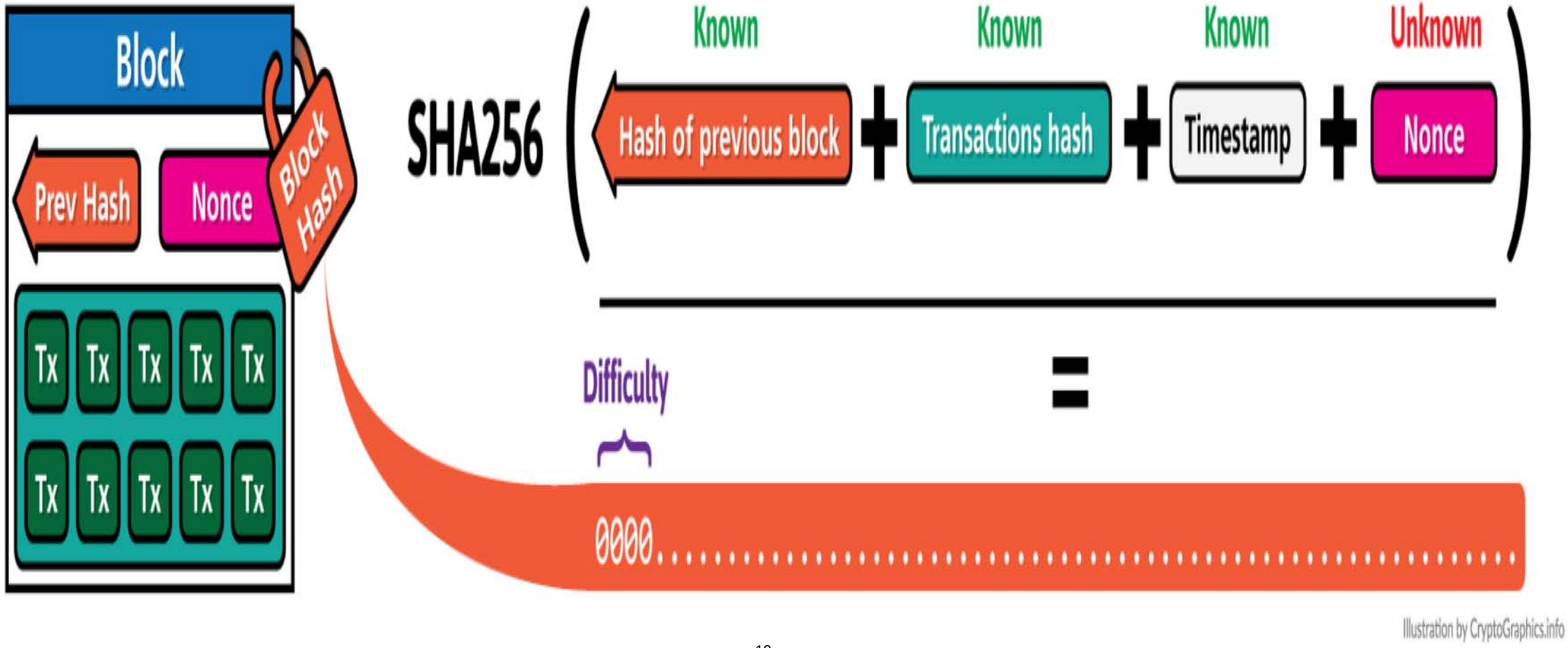
Bitcoin Address

Determined by – but not identical to - Public Key



Blockchain – Proof of Work

Chained Proof of Work for Distributed Network Consensus & Timestamping

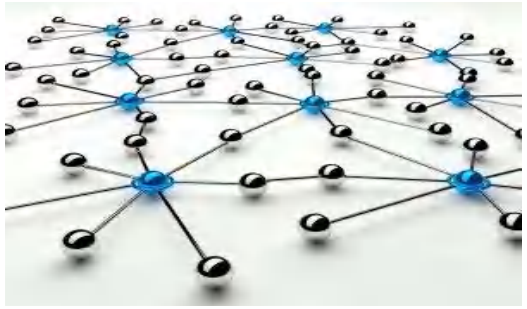


Native Currency

Economic Incentive System

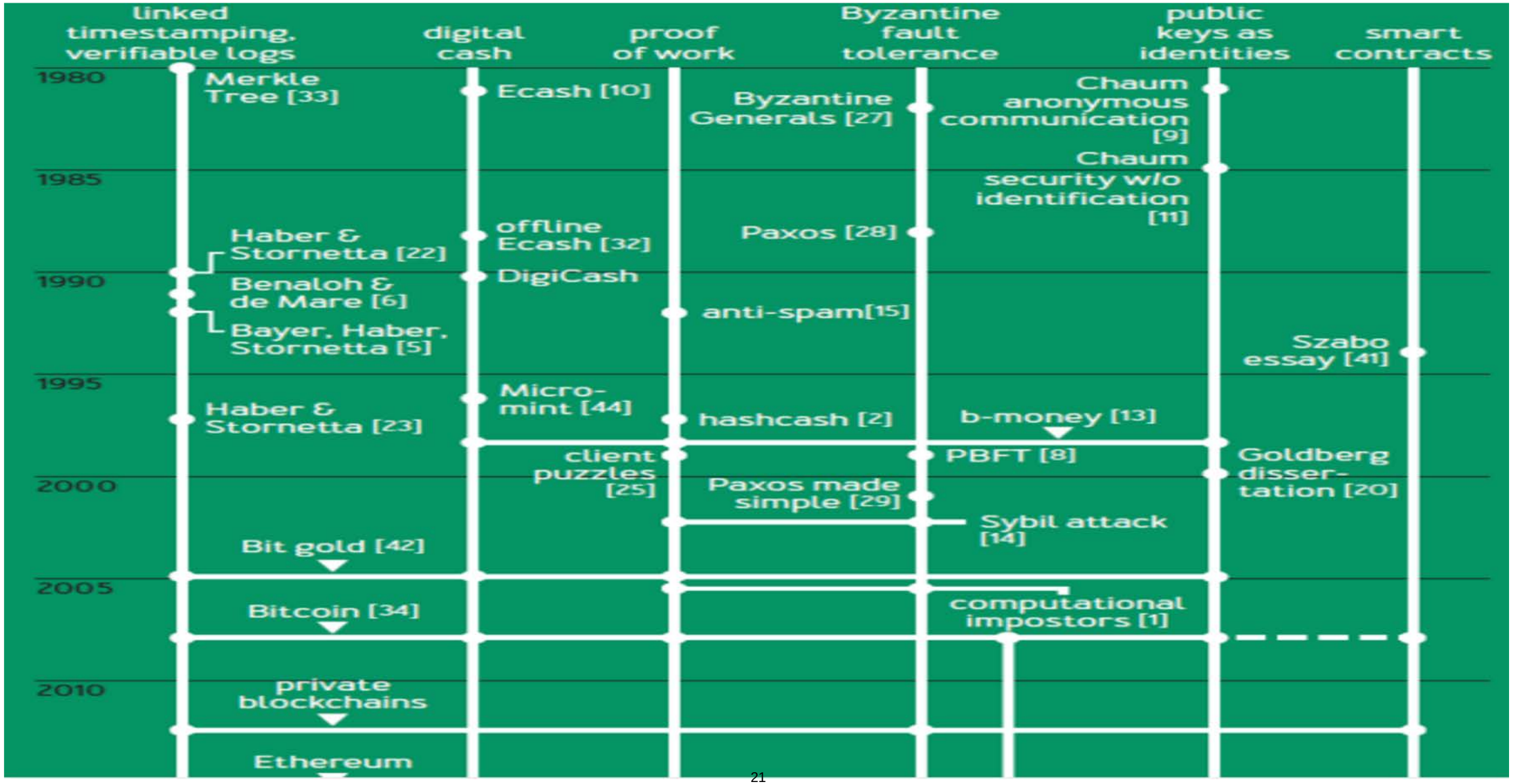
- Bitcoin – BTC
  - Reward Created through Coinbase Transaction in each block
  - Overall ‘Monetary Policy’ preset in Bitcoin Core
  - Reward halves (1/2s) every 210,000 blocks
  - Currently 17.3 million BTC; capping at 21 million BTC in 2140
  - Market based transaction fee mechanism also provided for in software

Network



- Full Nodes – Store full Blockchain & able to Validate all Transactions
- Pruning Nodes – Prune transactions after validation and aging
- Lightweight Nodes - Simplified Payment Verification (SPV) nodes – Store Blockchain Headers only
- Miners – Performs Proof of Work & Create new Blocks - Do not need to be a Full Node
- Mining Pool Operators
- Wallets – Store, View, Send and Receive Transactions & Create Key Pairs
- Mempool – Pool of unconfirmed (yet validated) Transactions

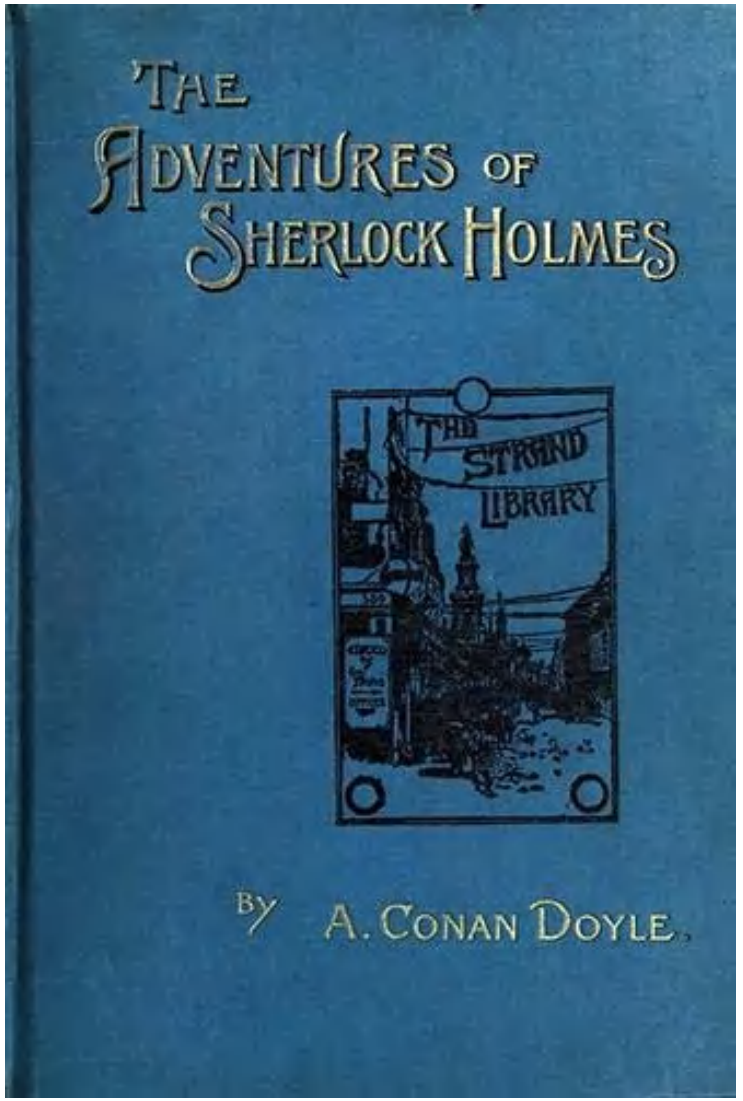
Narayanan and Clark's Chronology of Ideas in Bitcoin



Who is Satoshi Nakamoto?

Results of Ad Hoc Survey of Students

Group Led by Hal Finney
Nick Szabo
Craig Wright
Dorian Nakamoto
State Actor - US Government (NSA) or Otherwise



Class 6 (9/25): Study Questions

- What are smart contracts? How do they compare to traditional contracts? What are tokens?

- What are smart contract platforms such as Ethereum? What generally distinguishes them from Bitcoin?

- What are decentralized applications (DApps)? What has been the usage and why haven’t any DApps yet received wide consumer adoption?

Class 6 (9/25): Readings

Required

- *'Smart Contracts: 12 Use Cases for Business & Beyond'* Chamber of Digital Commerce
- *'State of the Dapps: 5 Observations from Usage Data'* McCann
- *'Ethereum Competitors: Guide to the Alternative Smart Contract Platforms'* Blockonomi

Optional

- *'Smart Contracts: Building Blocks for Digital Markets'* Szabo
- *'A Next-Generation Smart Contract and Decentralized Application Platform'* Ethereum
- *'Blockchain Technology as a Regulatory Technology'* De Filippi & Hassan

Guest Lecturer – Larry Lessig

- **Harvard Professor of Law and Leadership.**
- **Founder of Stanford Law's Center for Internet and Society.**
- **Clerked for Justice Antonin Scalia and for Appeals Court Judge Richard Posner.**
- **Awards include the Free Software Foundation's Freedom Award, Fastcase 50 Award and being named one of Scientific American's Top 50 Visionaries.**

'Code and Other Laws of Cyberspace'

- **Code/architecture** – physical or technical constraints
- **Market** – economic forces
- **Law** – explicit mandates by government
- **Norms** – social conventions

Conclusions

- Nakamoto's Bitcoin brought us Blockchain Technology
- Blockchain technology is within long history of Money & Ledgers
- Its Design Features also can be placed within history of technology
  - Timestamped Append-only Logs (Blocks)
  - Cryptographic Hash Functions & Digital Signatures
  - Network Consensus
- Key Innovation – Decentralized Chained Consensus Protocol
  - Addresses 'Costs of Trust'
  - Provides Peer-to-Peer alternative for Money, Ledgers & Computation

15.S12 Blockchain and Money
Fall 2018