Blockchain

1. Introduction

Stanisław Barański
stanislaw.baranski@pg.edu.pl
https://stan.bar

08.12.2022

Agenda

Motivation

History

Bitcoin and Blockchain

Ethereum

What is blockchain?

What do you think of when you hear the word ‘Bitcoin’?

visit menti.com and enter the code: 4105 8280
or directly: https://www.menti.com/alheo1yv89kk
or scan the QR Code

Motivation

- Transfer of value without trusted third party—electronic cash.
- Trust in cryptography, not in central authorities like banks. Bad people can change law, but not math nor cryptography.
- Privacy and/or anonimity.
- Censorship-resistant.
- Virtual-first currency, programmable money.

History

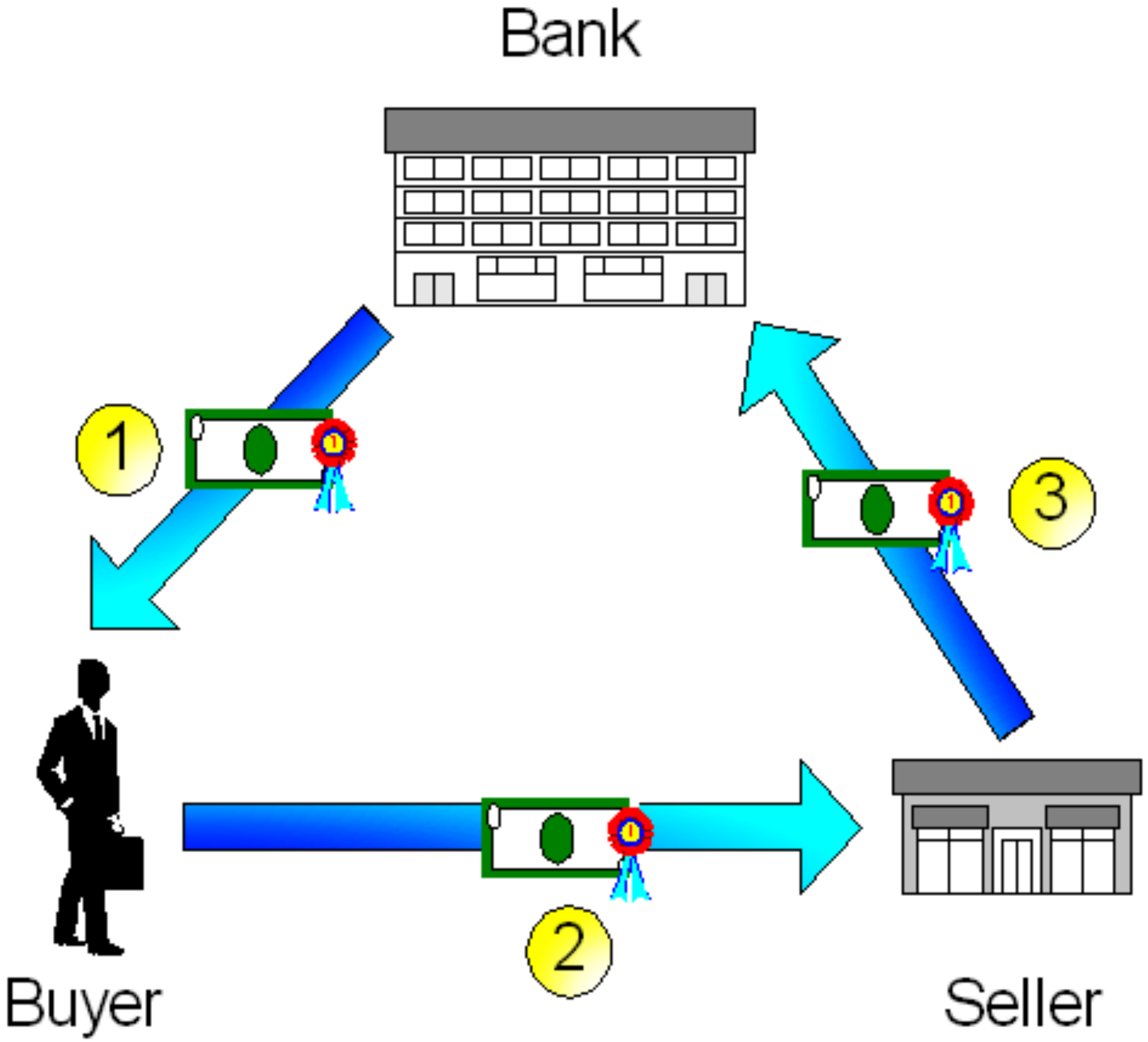
- eCash (David Chaum) - 1983

eCash

- Buyer buys eCash (certificate) from Bank.

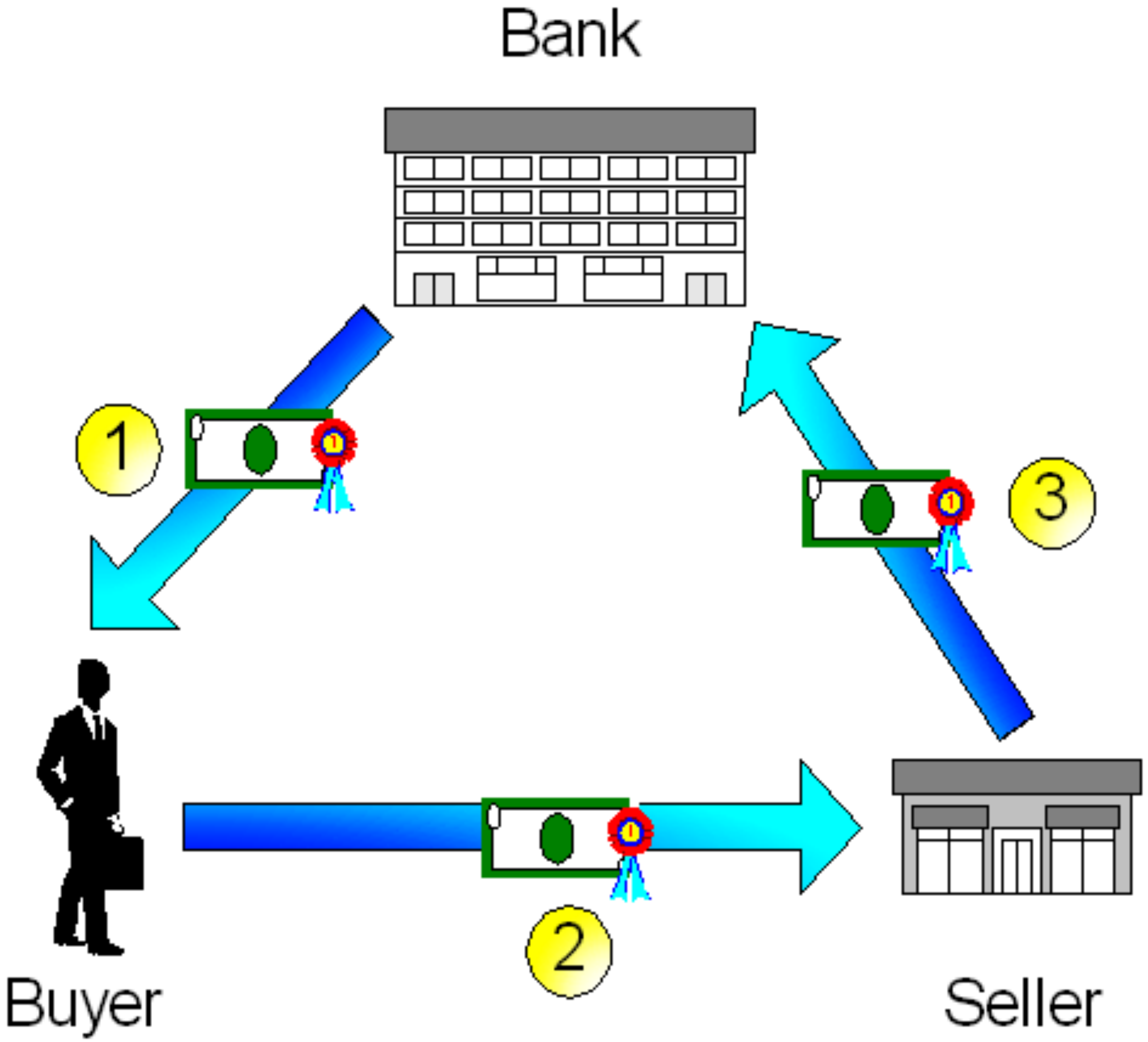
- Buyer sends eCash (certificate) to Seller.

- Seller sends eCash to Bank, which verifies if it hasn't been spend before and reedem eCash adding funds to Bank's account.



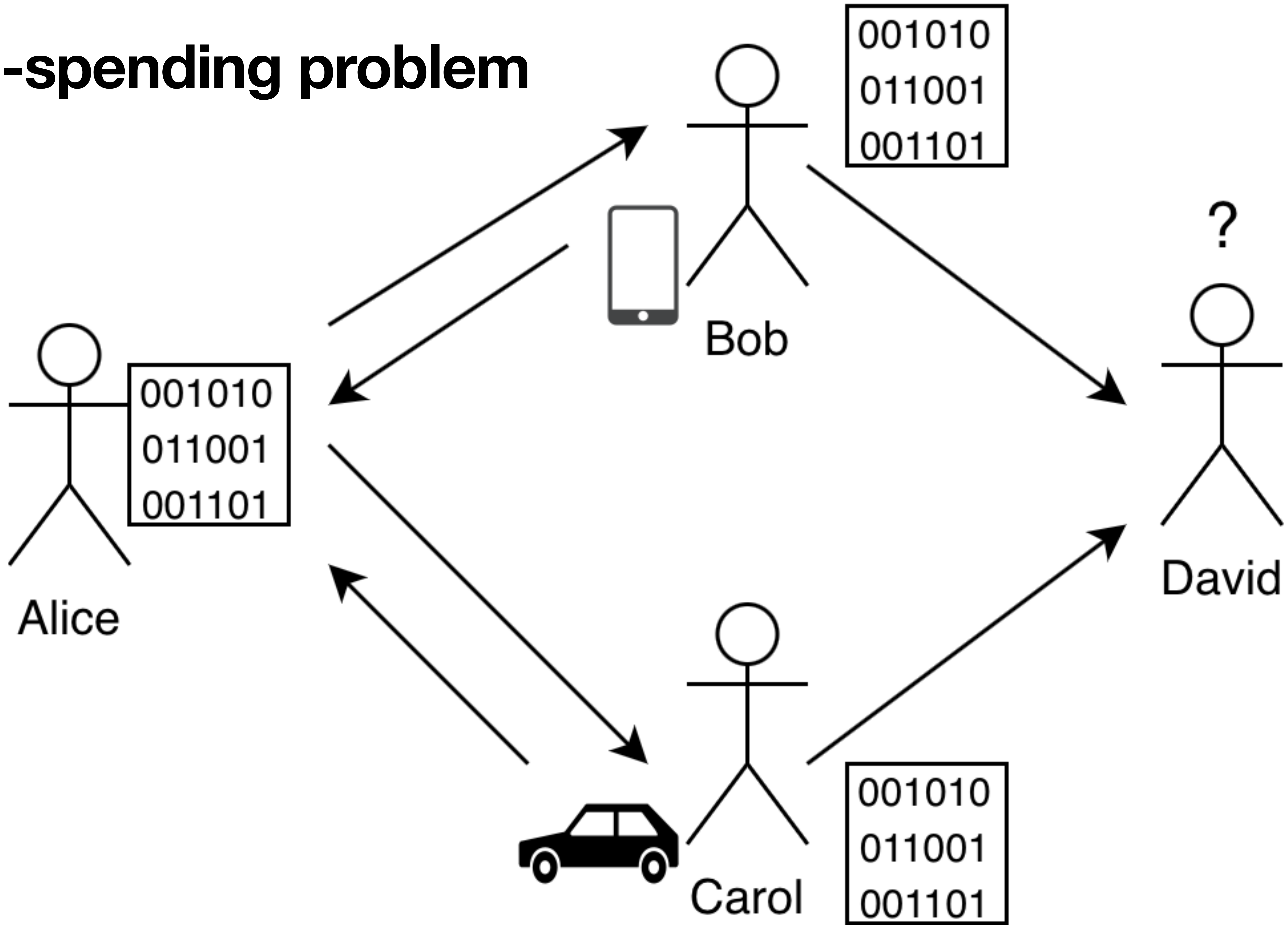
eCash

- Requires trusted third party—Bank.



What happens when we remove the central authority?

📌 Double-spending problem



History

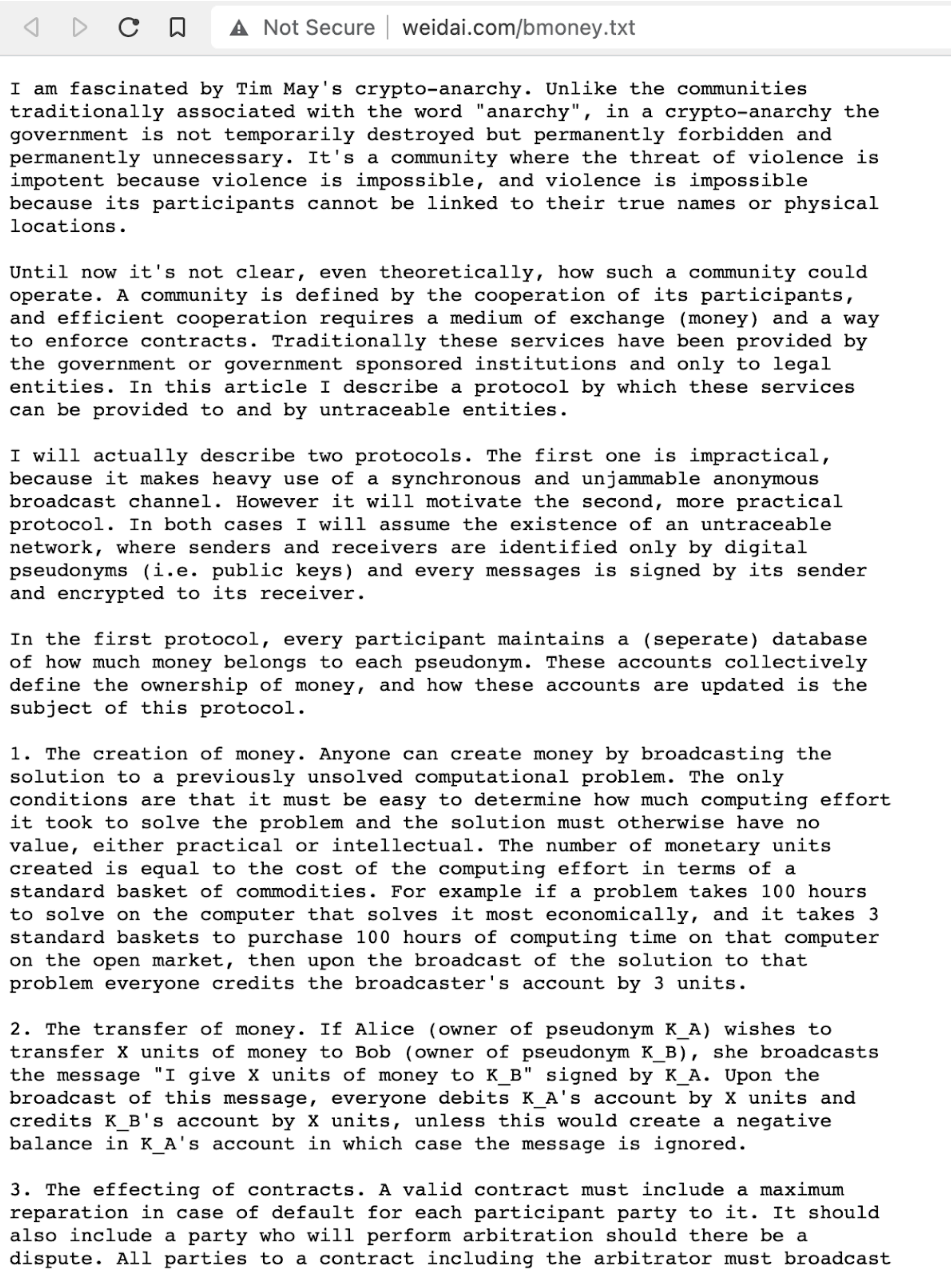
- eCash (David Chaum) - 1983
- b-money (Wei Dai) - 1998

b-money

an anonymous (pseudonymous), distributed electronic cash system.

Every participant maintains a (separate) database of how much money belongs to each pseudonym.

http://www.weidai.com/bmoney.txt



I am fascinated by Tim May's crypto-anarchy. Unlike the communities traditionally associated with the word "anarchy", in a crypto-anarchy the government is not temporarily destroyed but permanently forbidden and permanently unnecessary. It's a community where the threat of violence is impotent because violence is impossible, and violence is impossible because its participants cannot be linked to their true names or physical locations.

Until now it's not clear, even theoretically, how such a community could operate. A community is defined by the cooperation of its participants, and efficient cooperation requires a medium of exchange (money) and a way to enforce contracts. Traditionally these services have been provided by the government or government sponsored institutions and only to legal entities. In this article I describe a protocol by which these services can be provided to and by untraceable entities.

I will actually describe two protocols. The first one is impractical, because it makes heavy use of a synchronous and unjammable anonymous broadcast channel. However it will motivate the second, more practical protocol. In both cases I will assume the existence of an untraceable network, where senders and receivers are identified only by digital pseudonyms (i.e. public keys) and every messages is signed by its sender and encrypted to its receiver.

In the first protocol, every participant maintains a (seperate) database of how much money belongs to each pseudonym. These accounts collectively define the ownership of money, and how these accounts are updated is the subject of this protocol.

1. The creation of money. Anyone can create money by broadcasting the solution to a previously unsolved computational problem. The only conditions are that it must be easy to determine how much computing effort it took to solve the problem and the solution must otherwise have no value, either practical or intellectual. The number of monetary units created is equal to the cost of the computing effort in terms of a standard basket of commodities. For example if a problem takes 100 hours to solve on the computer that solves it most economically, and it takes 3 standard baskets to purchase 100 hours of computing time on that computer on the open market, then upon the broadcast of the solution to that problem everyone credits the broadcaster's account by 3 units.

2. The transfer of money. If Alice (owner of pseudonym K_A) wishes to transfer X units of money to Bob (owner of pseudonym K_B), she broadcasts the message "I give X units of money to K_B" signed by K_A. Upon the broadcast of this message, everyone debits K_A's account by X units and credits K_B's account by X units, unless this would create a negative balance in K_A's account in which case the message is ignored.

3. The effecting of contracts. A valid contract must include a maximum reparation in case of default for each participant party to it. It should also include a party who will perform arbitration should there be a dispute. All parties to a contract including the arbitrator must broadcast

b-money

How to create value

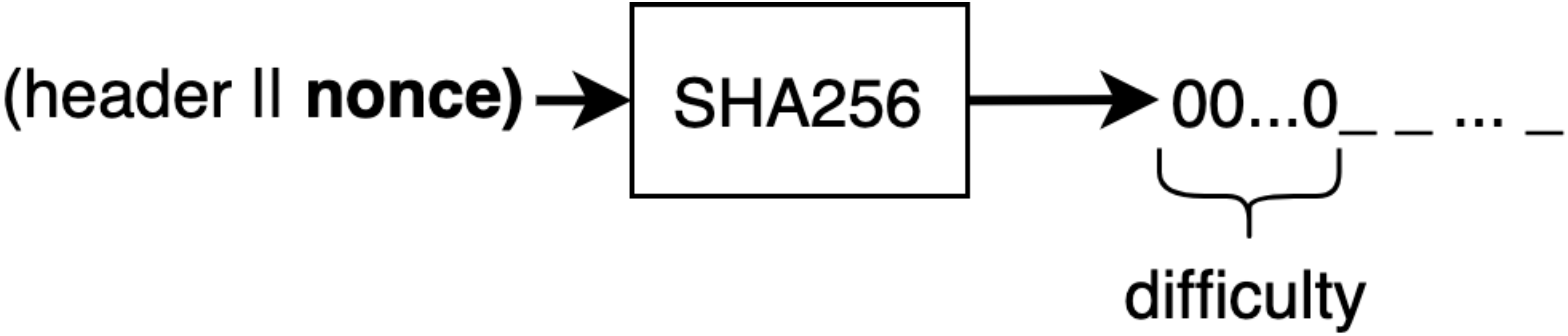
- We value only what is a scarce resource like time, electricity, trust, or gold.
- How to create artificial scarcity?

b-money

How to create value

- We value only what is a scarce resource like time, electricity, trust, or gold.
- How to create artificial scarcity?
- Cryptographic puzzles (problems in NP).
- Example: find $x \text{ s.t. } \mathrm{H}(x) \leq d$, where *x* is a number to find, *H* is a hash function, and *d* is a difficulty level.

📌 Proof-of-work



https://andersbrownworth.com/blockchain/hash

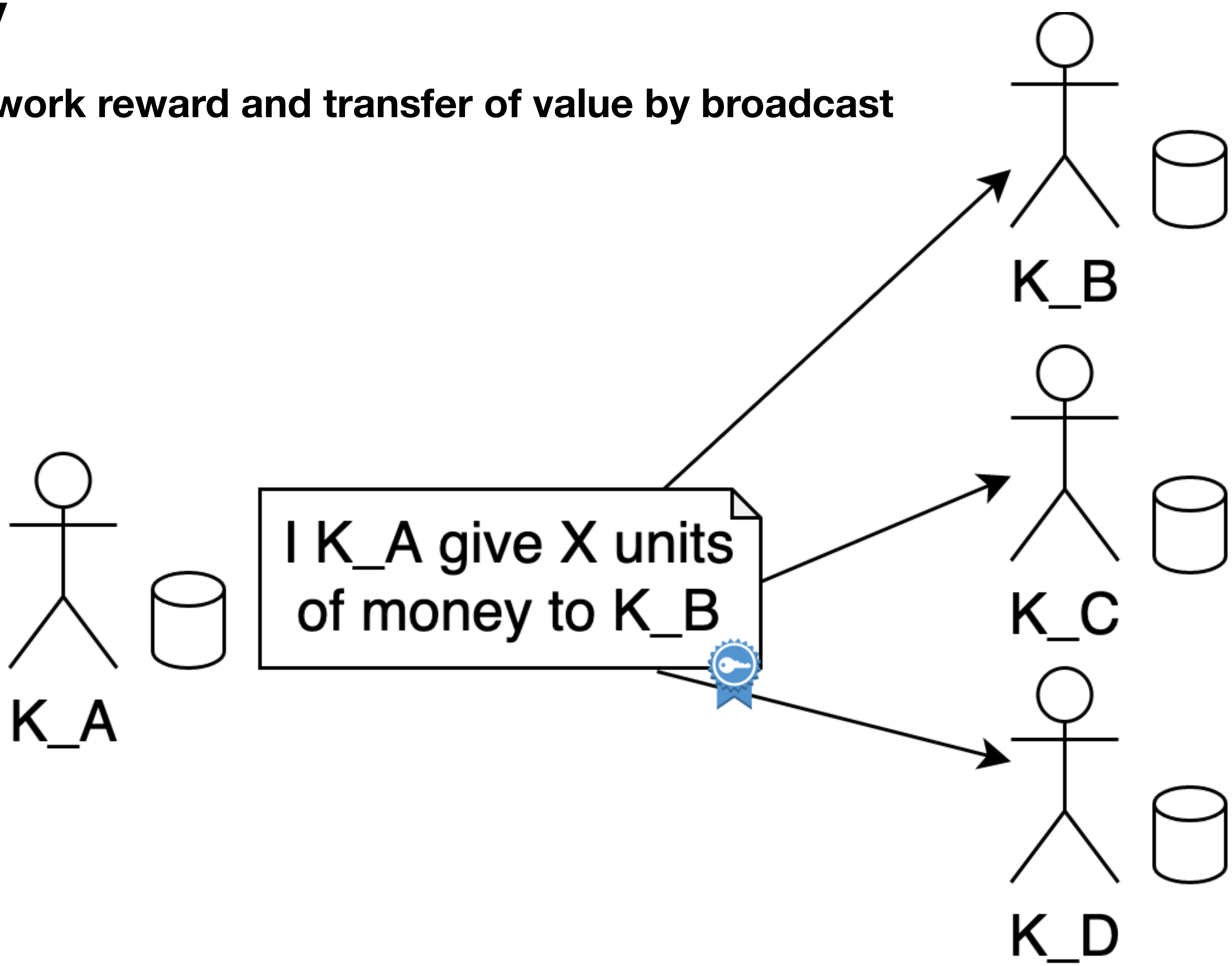
b-money

How to create value

- We value only what is a scarce resource like time, electricity, trust, or gold
- How to create artificial scarcity?
- Cryptographic puzzles (problems in NP)
- Example: find $x \text{ s.t. } \mathrm{H}(x) \leq d$, where *x* is a number to find, *H* is a hash function, and *d* is a difficulty level.
- Solving such a crypto puzzle consumes electricity, which is a scarce resource ✅
- The number of monetary units created is equal to the cost of the computing effort in terms of a standard basket of commodities.

b-money

Redeeming proof-of-work reward and transfer of value by broadcast



I K_A give X units of money to K_B

b-money

Solved:

- Money creation ✅
- Transfer of money ✅

(Impractical) assumptions:

- Assumed atomic broadcast
- 100% uptime peers

Not solved:

- Not fault tolerant

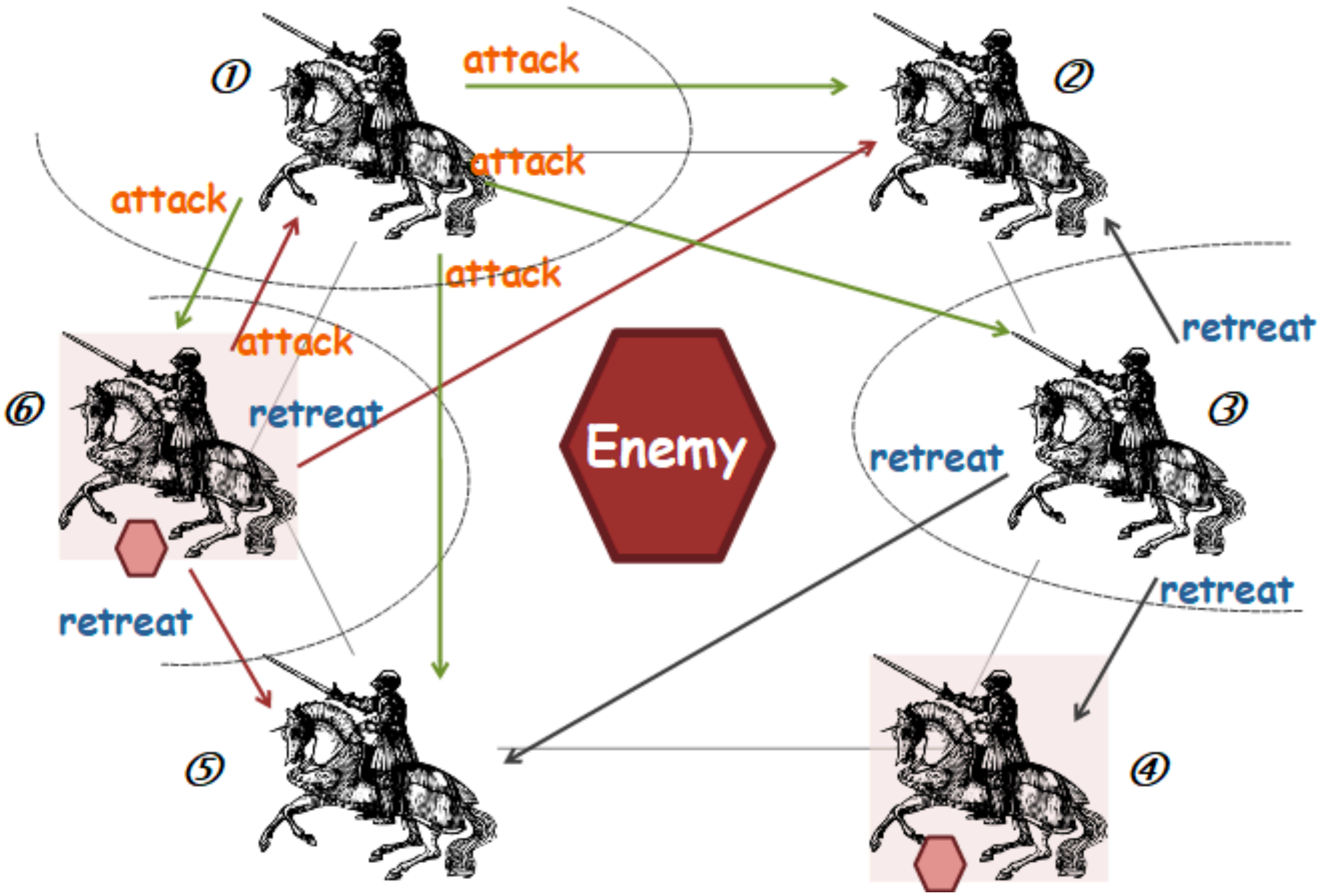
As a result, the idea was abandoned.

Fault tolerant systems

Distinguishing fault categories

- **Fail-stop**: nodes can crash, not return values, crash detectable by other nodes.
    - Paxos (Google Chubby)
    - ZooKeeper
    - <u>RAFT (Consul, etcd)</u>
- **Byzantine fault**: nodes can do all the above + send incorrect/corrupted messages.
    - PBFT
    - …

Byzantine fault



Fault tolerant systems

Distinguishing fault categories

- **Fail-stop**: nodes can crash, not return values, crash detectable by other nodes.
  - Paxos (Google Chubby)
  - ZooKeeper
  - RAFT (Consul, etcd)
- **Byzantine fault**: nodes can do all the above + send incorrect/corrupted messages.
  - PBFT
  - …
- They work in managed clusters and closed networks, but **not in p2p open-membership networks, because the total number of peers is unknown, and they are prone to Sybil attacks**.

History

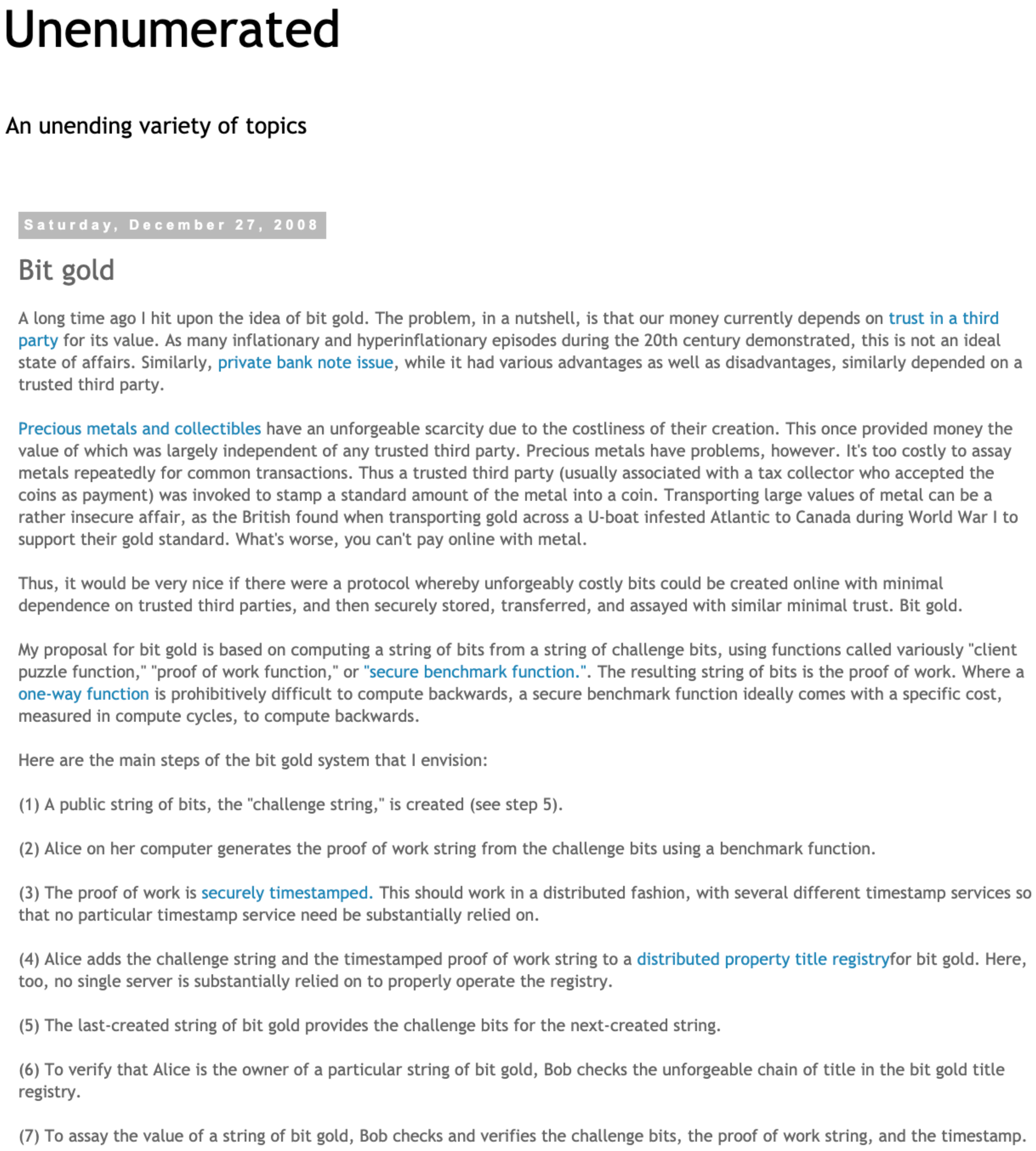
- eCash (David Chaum) - 1983
- b-money (Wei Dai) - 1998
- BitGold (Nick Szabo) - 1998/2005

BitGold

Assumes existence of distributed timestamp services and distributed Bitgold registry.

Money creation (proof-of-work based + challenge string + timestamping)

https://unenumerated.blogspot.com/2005/12/bit-gold.html



Unenumerated

An unending variety of topics

Saturday, December 27, 2008

Bit gold

A long time ago I hit upon the idea of bit gold. The problem, in a nutshell, is that our money currently depends on trust in a third party for its value. As many inflationary and hyperinflationary episodes during the 20th century demonstrated, this is not an ideal state of affairs. Similarly, private bank note issue, while it had various advantages as well as disadvantages, similarly depended on a trusted third party.

Precious metals and collectibles have an unforgeable scarcity due to the costliness of their creation. This once provided money the value of which was largely independent of any trusted third party. Precious metals have problems, however. It's too costly to assay metals repeatedly for common transactions. Thus a trusted third party (usually associated with a tax collector who accepted the coins as payment) was invoked to stamp a standard amount of the metal into a coin. Transporting large values of metal can be a rather insecure affair, as the British found when transporting gold across a U-boat infested Atlantic to Canada during World War I to support their gold standard. What's worse, you can't pay online with metal.

Thus, it would be very nice if there were a protocol whereby unforgeably costly bits could be created online with minimal dependence on trusted third parties, and then securely stored, transferred, and assayed with similar minimal trust. Bit gold.

My proposal for bit gold is based on computing a string of bits from a string of challenge bits, using functions called variously "client puzzle function," "proof of work function," or "secure benchmark function.". The resulting string of bits is the proof of work. Where a one-way function is prohibitively difficult to compute backwards, a secure benchmark function ideally comes with a specific cost, measured in compute cycles, to compute backwards.

Here are the main steps of the bit gold system that I envision:

(1) A public string of bits, the "challenge string," is created (see step 5).

(2) Alice on her computer generates the proof of work string from the challenge bits using a benchmark function.

(3) The proof of work is securely timestamped. This should work in a distributed fashion, with several different timestamp services so that no particular timestamp service need be substantially relied on.

(4) Alice adds the challenge string and the timestamped proof of work string to a distributed property title registryfor bit gold. Here, too, no single server is substantially relied on to properly operate the registry.

(5) The last-created string of bit gold provides the challenge bits for the next-created string.

(6) To verify that Alice is the owner of a particular string of bit gold, Bob checks the unforgeable chain of title in the bit gold title registry.

(7) To assay the value of a string of bit gold, Bob checks and verifies the challenge bits, the proof of work string, and the timestamp.

BitGold

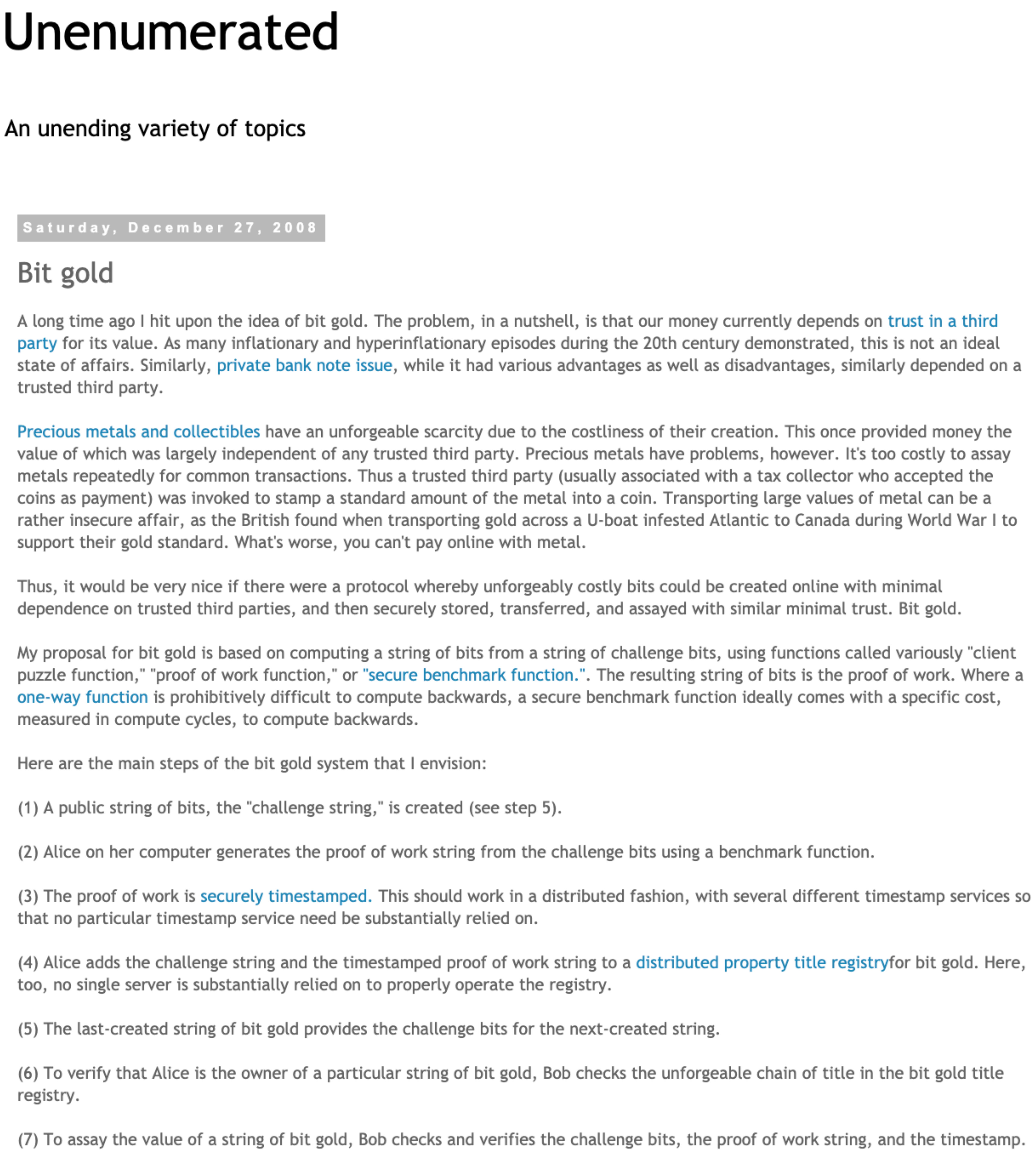
Solved (without b-money impractical assumptions):

- Money creation ✅
- Transfer of money ✅

Not solved:

- Distributed but not decentralised.
- Missing incentives to keep nodes honest.
- Not byzantine-fault tolerant.

As a result, the idea was abandoned.



Unenumerated

An unending variety of topics

Saturday, December 27, 2008

Bit gold

A long time ago I hit upon the idea of bit gold. The problem, in a nutshell, is that our money currently depends on trust in a third party for its value. As many inflationary and hyperinflationary episodes during the 20th century demonstrated, this is not an ideal state of affairs. Similarly, private bank note issue, while it had various advantages as well as disadvantages, similarly depended on a trusted third party.

Precious metals and collectibles have an unforgeable scarcity due to the costliness of their creation. This once provided money the value of which was largely independent of any trusted third party. Precious metals have problems, however. It's too costly to assay metals repeatedly for common transactions. Thus a trusted third party (usually associated with a tax collector who accepted the coins as payment) was invoked to stamp a standard amount of the metal into a coin. Transporting large values of metal can be a rather insecure affair, as the British found when transporting gold across a U-boat infested Atlantic to Canada during World War I to support their gold standard. What's worse, you can't pay online with metal.

Thus, it would be very nice if there were a protocol whereby unforgeably costly bits could be created online with minimal dependence on trusted third parties, and then securely stored, transferred, and assayed with similar minimal trust. Bit gold.

My proposal for bit gold is based on computing a string of bits from a string of challenge bits, using functions called variously "client puzzle function," "proof of work function," or "secure benchmark function.". The resulting string of bits is the proof of work. Where a one-way function is prohibitively difficult to compute backwards, a secure benchmark function ideally comes with a specific cost, measured in compute cycles, to compute backwards.

Here are the main steps of the bit gold system that I envision:

(1) A public string of bits, the "challenge string," is created (see step 5).

(2) Alice on her computer generates the proof of work string from the challenge bits using a benchmark function.

(3) The proof of work is securely timestamped. This should work in a distributed fashion, with several different timestamp services so that no particular timestamp service need be substantially relied on.

(4) Alice adds the challenge string and the timestamped proof of work string to a distributed property title registryfor bit gold. Here, too, no single server is substantially relied on to properly operate the registry.

(5) The last-created string of bit gold provides the challenge bits for the next-created string.

(6) To verify that Alice is the owner of a particular string of bit gold, Bob checks the unforgeable chain of title in the bit gold title registry.

(7) To assay the value of a string of bit gold, Bob checks and verifies the challenge bits, the proof of work string, and the timestamp.

History

- eCash (David Chaum) - 1983
- b-money (Wei Dai) - 1998
- BitGold (Nick Szabo) - 1998/2005
- Bitcoin (Satoshi Nakamoto) - 2008

Bitcoin

Bitcoin

First peer-to-peer electronic cash.

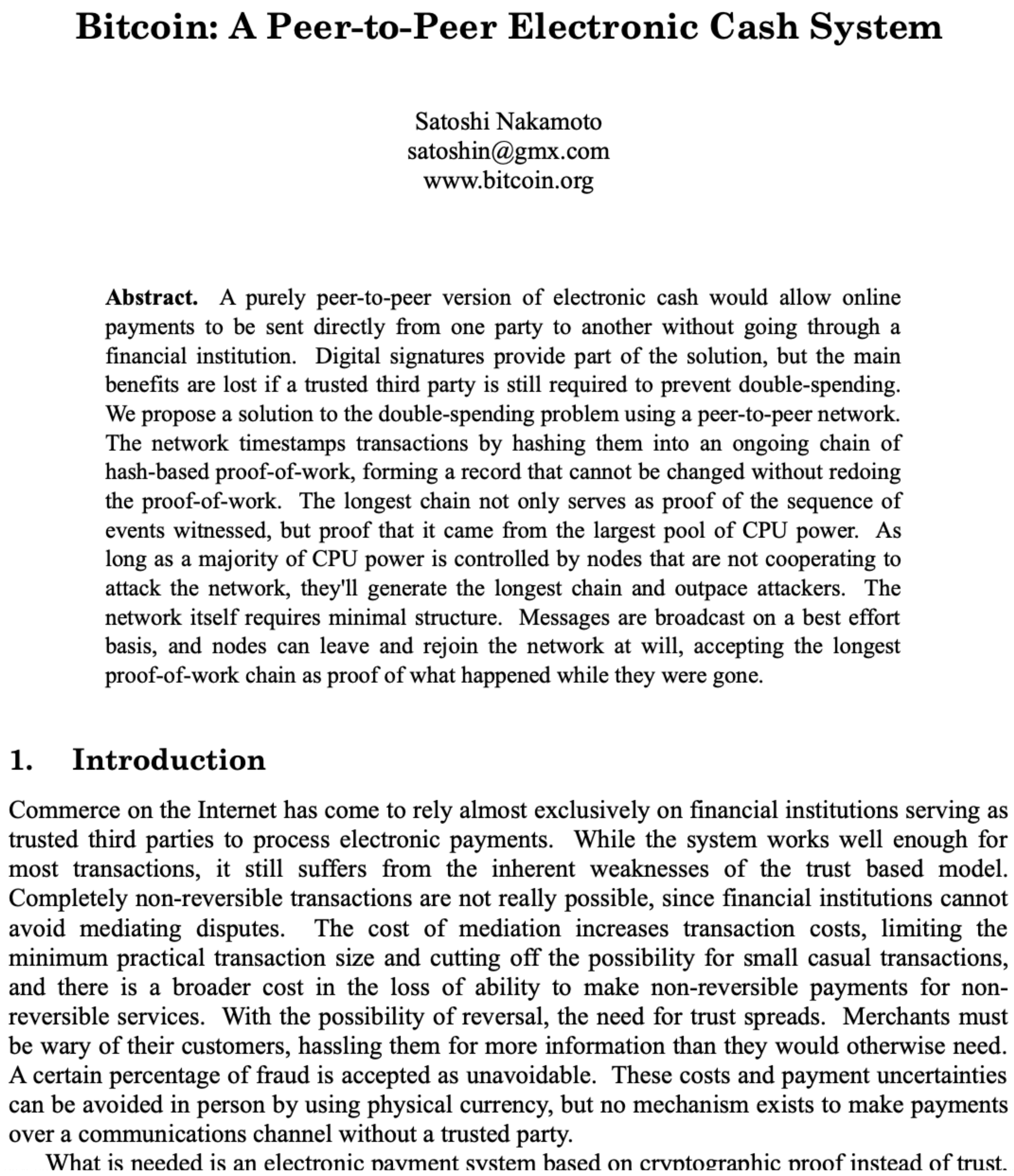
First to solve all ‘unsolvable’ problems, 25 years after the first attempt (eCash).

Published by anonymous author Satoshi Nakamoto.

- Most likely software developer, not a scientist (software first).

- Most likely one person, not a team (consistent in his opinions and writing style over years).

- Most likely lost access to his wallet (5% of total supply ~ 1mln BTC ~ $17bn @ 6.12.2022).



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.

https://bitcoin.org/bitcoin.pdf

Bitcoin
Innovations

Used proof-of-work to:

1. Achieve global consensus in open-membership network (leadership election by computing power)

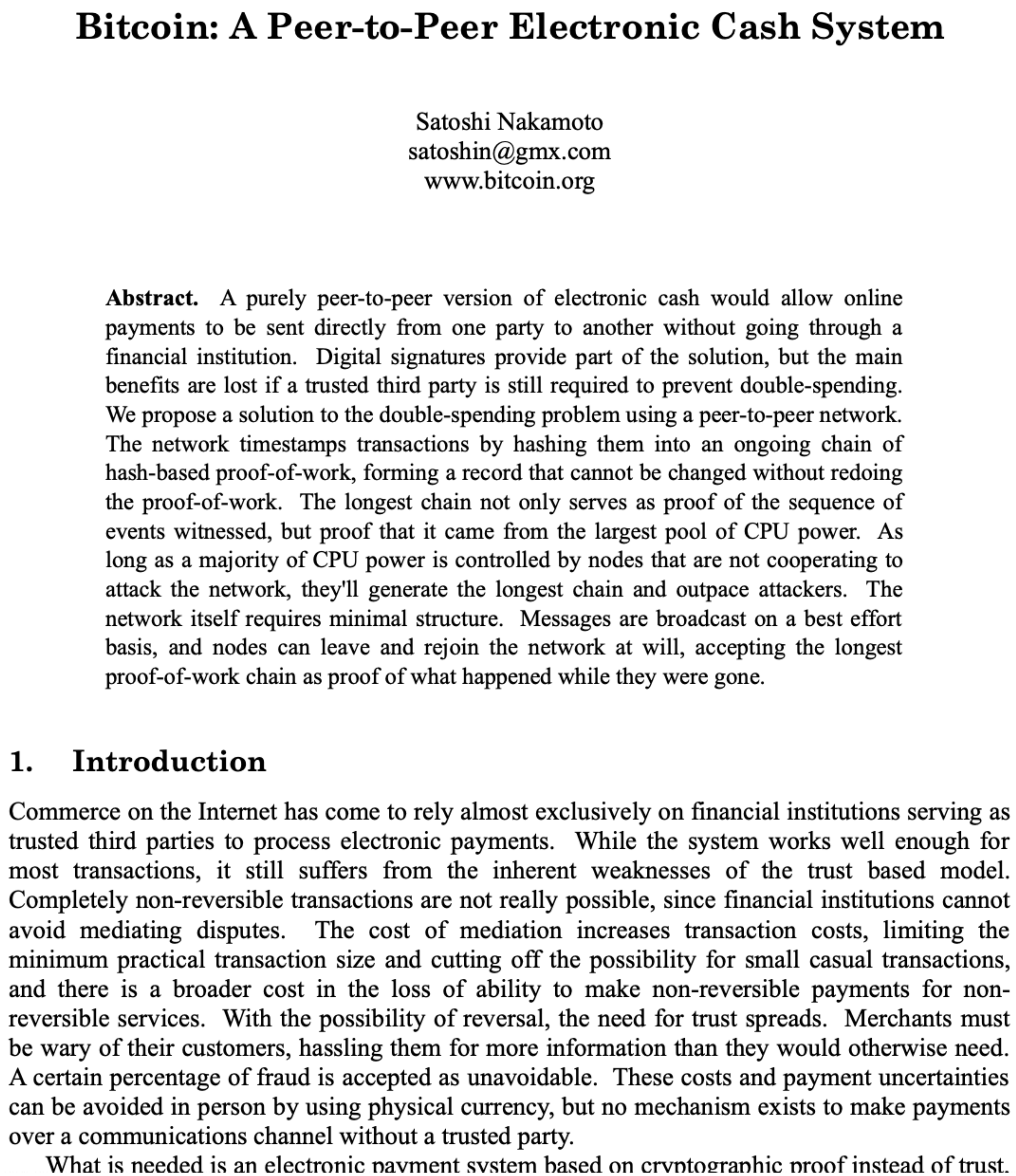
2. Create scare resource

3. Prevent Sybil attacks (vote = collective computing power)

4. Incentivisation to (honest) participation to the network.

5. Secure the network (reverting history requires 51% computing power)

Introduced blockchain — a data structure that timestamps transactions.



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.

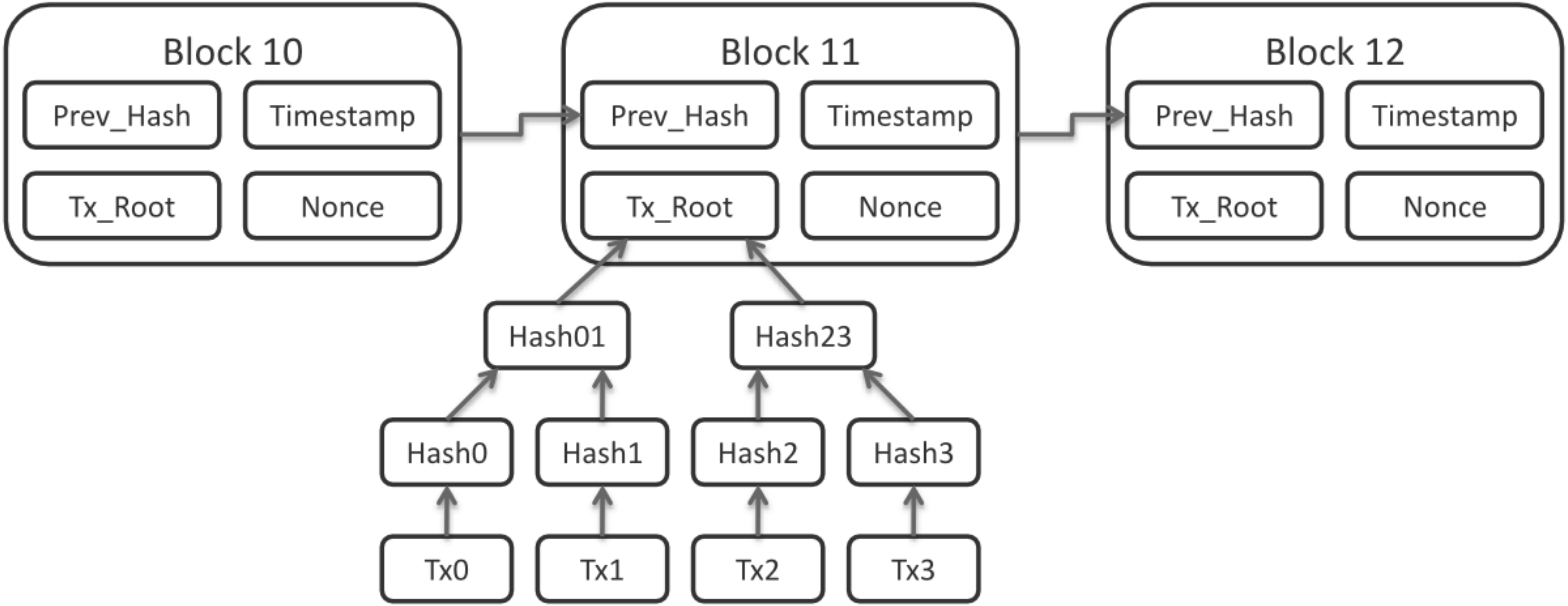
1. Introduction

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.

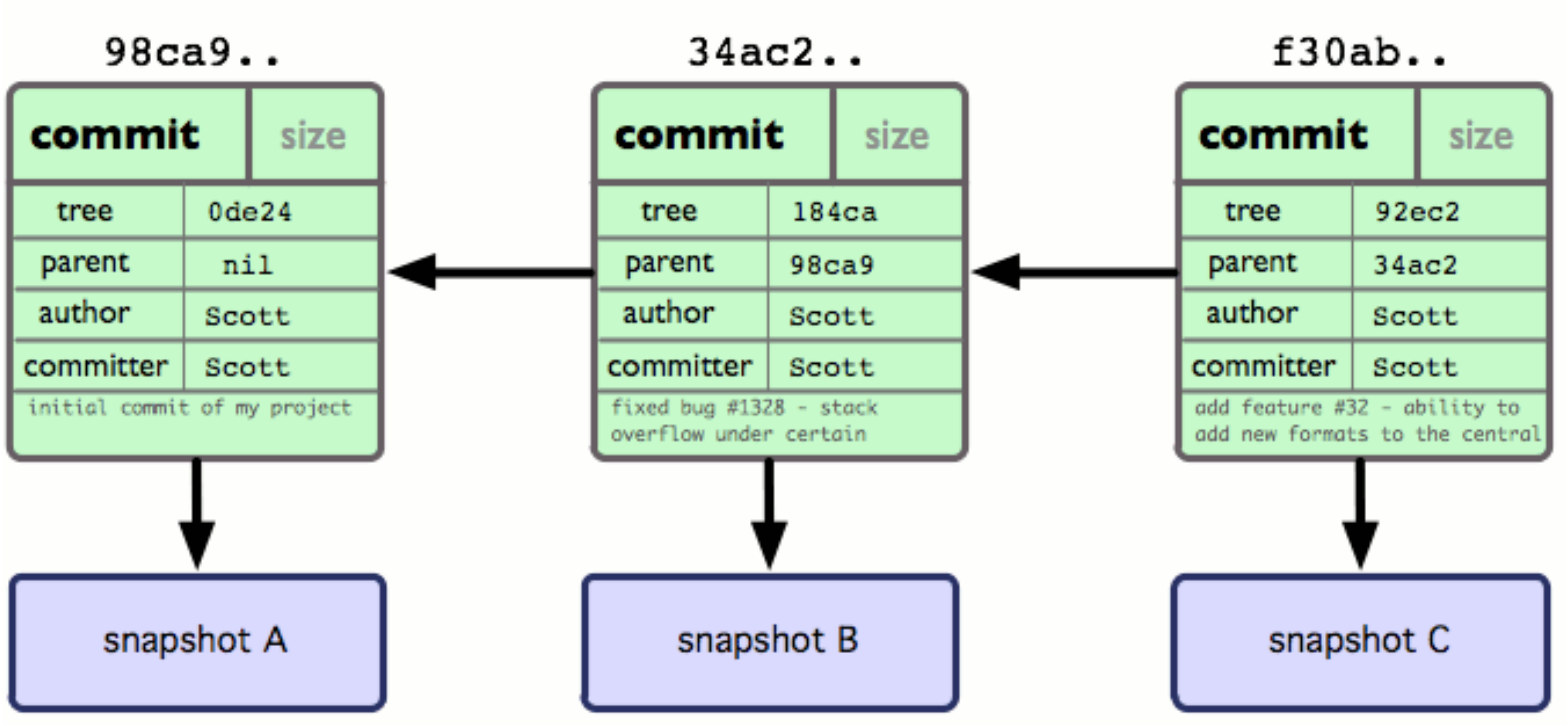
What is needed is an electronic payment system based on cryptographic proof instead of trust,

https://bitcoin.org/bitcoin.pdf

Blockchain


Block 10
Prev_Hash
Timestamp
Tx_Root
Nonce
Block 11
Prev_Hash
Timestamp
Tx_Root
Nonce
Block 12
Prev_Hash
Timestamp
Tx_Root
Nonce
Hash01
Hash23
Hash0
Hash1
Hash2
Hash3
Tx0
Tx1
Tx2
Tx3

Think of blockchain as a Git version control system
Think of proof of work as way of accepting pull requests



- commit <-> block
- commit hash <-> block hash
- previous commit <-> previous block
- file <-> address
- diff <-> transaction

https://andersbrownworth.com/blockchain/block

Bitcoin

Where does the value come from?

Utility — 1) easy and censorship resistant transfer of value; 2) secure store of value.

Deflationary — halves currency issuance every 4 years.

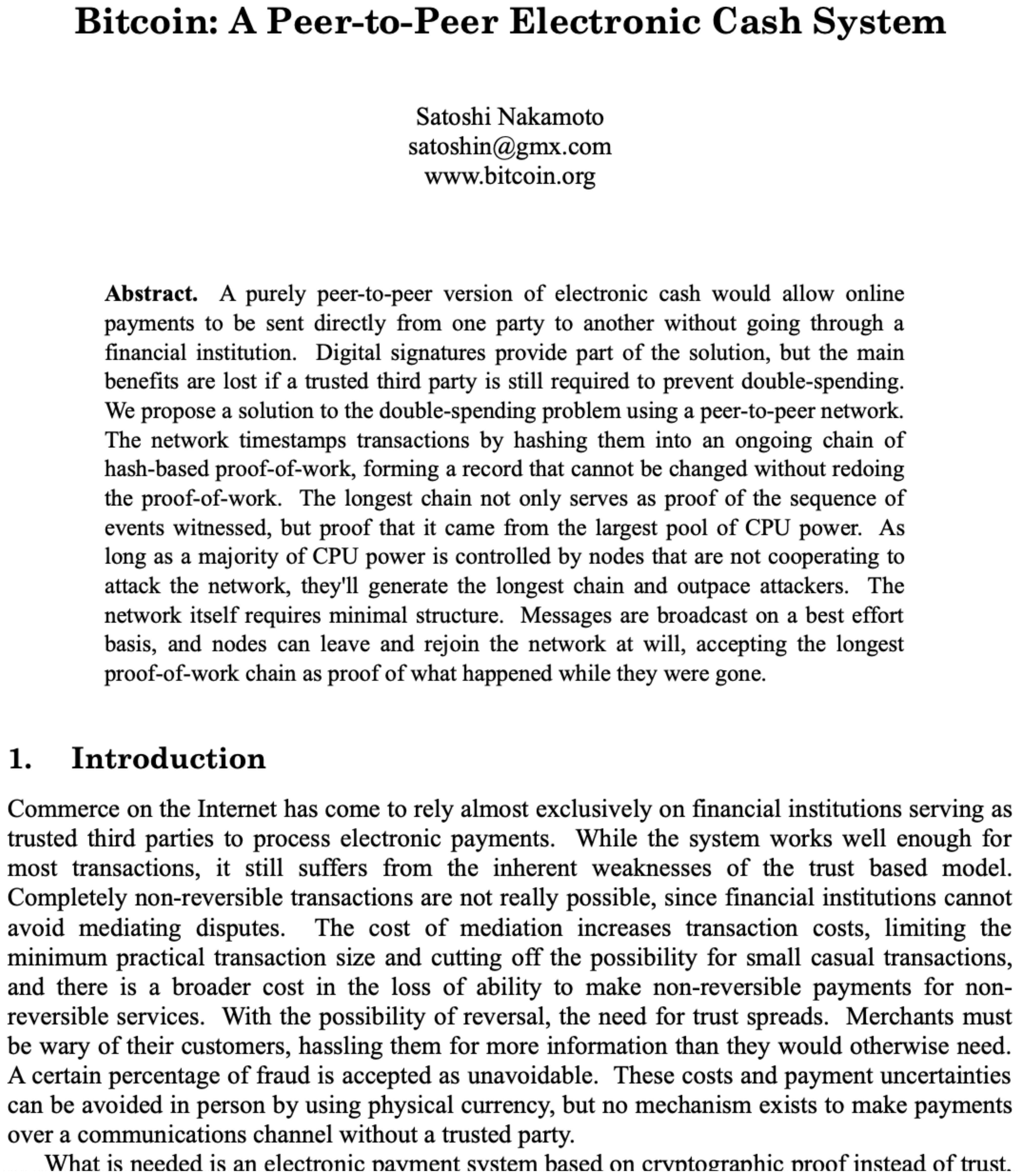
Cap at 21mln, currently 19.23mln (92%) in circulating supply.

The law of supply and demand.

Mining costs.

Speculation.

https://bitcoin.org/bitcoin.pdf



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,

Transition state machine

S − states

T − transactions

$\text{Apply} : S \times T \to S$ − state transition function

$S_{n+1} \leftarrow \text{Apply}(S_n, T_n)$

```
$\text{Apply}(s, t)$ = {
    $\text{ensure}(s[t_{from}] \geq t_{value})$
    $s[t_{from}] \leftarrow s[t_{from}] - t_{value}$
    $s[t_{to}] \leftarrow s[t_{to}] + t_{value}$
}
```

Transition state machine

S – states

T – transactions

$\text{Apply} : S \times T \rightarrow S$ – state transition function

$S_{n+1} \leftarrow \text{Apply}(S_n, T_n)$

```
$\text{Apply}(s, t)$ = {
    $\text{ensure}(s[t_{from}] \geq t_{value})$
    $s[t_{from}] \leftarrow s[t_{from}] - t_{value}$
    $s[t_{to}] \leftarrow s[t_{to}] + t_{value}$
}
```

Example:

$$s = \{\text{Alice} : 10,\ \text{Bob} : 0\}$$

t = “Send 8฿ from Alice to Bob”

$$\{\text{Alice} : 2,\ \text{Bob} : 8\} \leftarrow \text{Apply}(s, t)$$

Transition state machine

S — states

T — transactions

$\text{Apply} : S \times T \to S$ — state transition function

$S_{n+1} \leftarrow \text{Apply}(S_n, T_n)$

```
Apply$(s, t)$ = {
    ensure$(s[t_{from}] \geq t_{value})$
    $s[t_{from}] \leftarrow s[t_{from}] - t_{value}$
    $s[t_{to}] \leftarrow s[t_{to}] + t_{value}$
}
```

Example:

$s = \{\text{Alice} : 10,\ \text{Bob} : 0\}$

t = “Send 8₿ from Alice to Bob”

$\{\text{Alice} : 2,\ \text{Bob} : 8\} \leftarrow \text{Apply}(s, t)$

Each transaction is recorded on a public, immutable and decentralized data structure—Blockchain.

Ethereum – generalization of Bitcoin

S — states

T — transactions

$\text{Apply} : S \times T \to S$ — state transition function

$S_{n+1} \leftarrow \text{Apply}(S_n, T_n)$

```
Apply$(s, t)$ = {
    ensure$(s[t_{from}] \geq t_{value})$
    $s[t_{from}] \leftarrow s[t_{from}] - t_{value}$
    $s[t_{to}] \leftarrow s[t_{to}] + t_{value}$
}
```

Ethereum

T - smart contract codes

$S_{n+1} = \text{Apply}(S_n, T_n)$

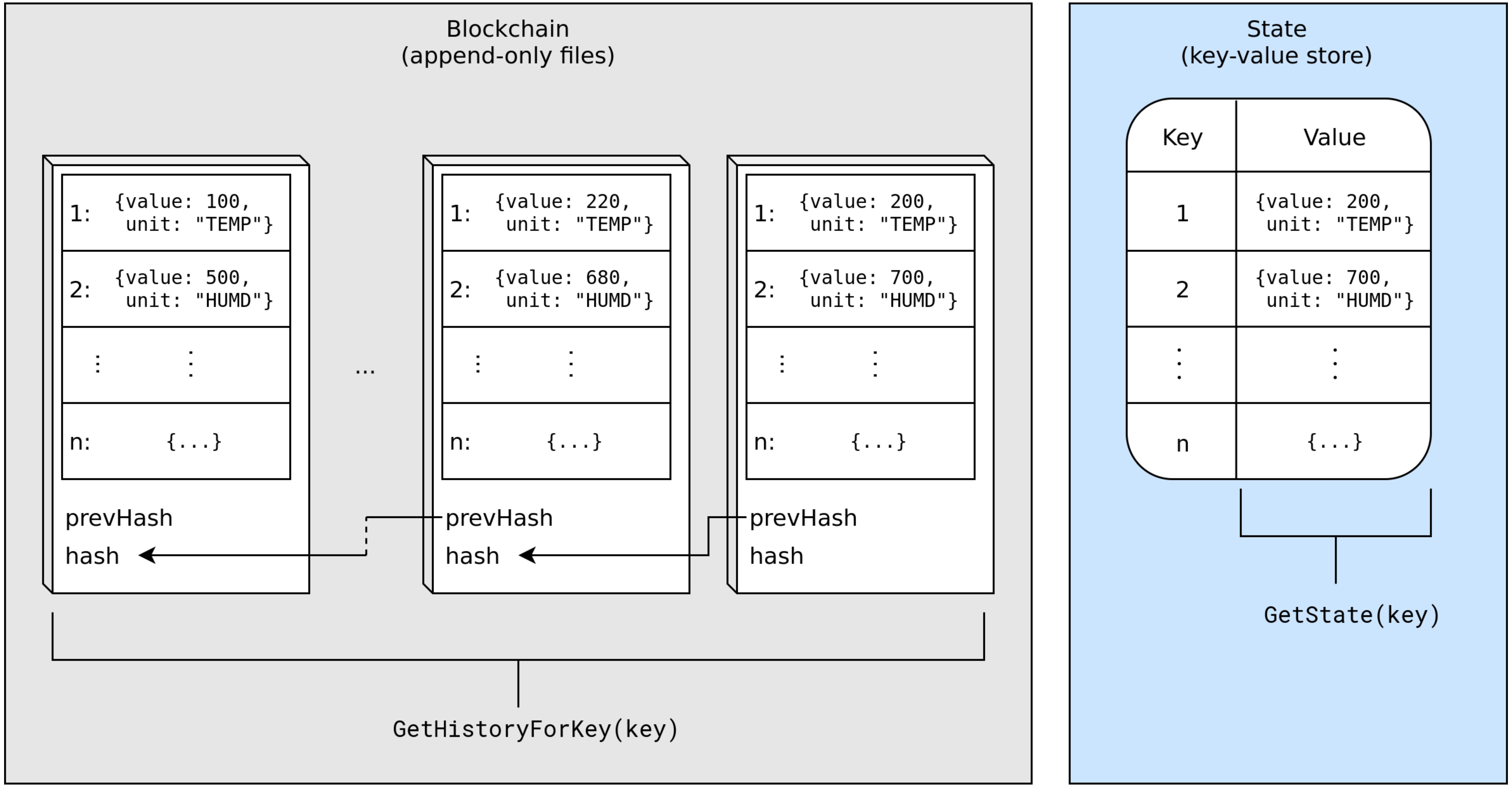
$\text{Apply}(S_n, T_n) = \text{EVM}(S_n, T_n)$

What is blockchain?

Blockchain is not a database

It’s an integral and secure history of changes to the database

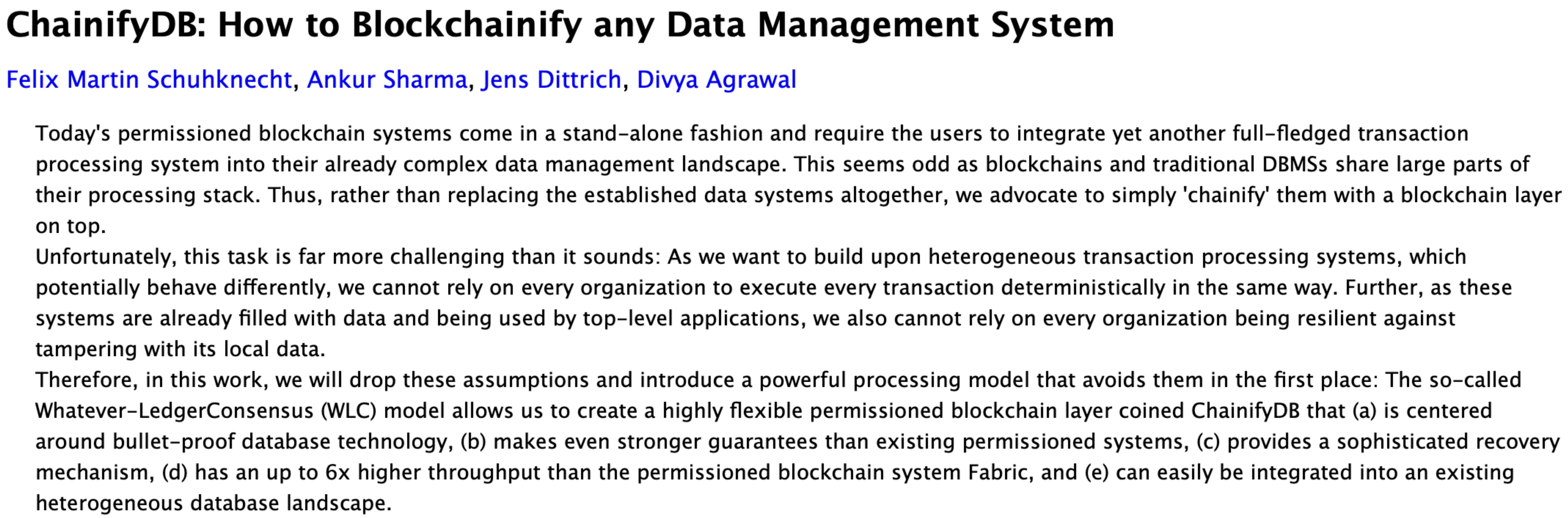
Files


Blockchain
(append-only files)
1: {value: 100, unit: "TEMP"}
2: {value: 500, unit: "HUMD"}
n: {...}
prevHash
hash
...
1: {value: 220, unit: "TEMP"}
2: {value: 680, unit: "HUMD"}
n: {...}
prevHash
hash
1: {value: 200, unit: "TEMP"}
2: {value: 700, unit: "HUMD"}
n: {...}
prevHash
hash
GetHistoryForKey(key)
State
(key-value store)
GetState(key)

Key	Value
1	{value: 200, unit: "TEMP"}
2	{value: 700, unit: "HUMD"}
⋮	⋮
n	{...}

LevelDB
CouchDB

Blockchainification



ChainifyDB: How to Blockchainify any Data Management System

Felix Martin Schuhknecht, Ankur Sharma, Jens Dittrich, Divya Agrawal

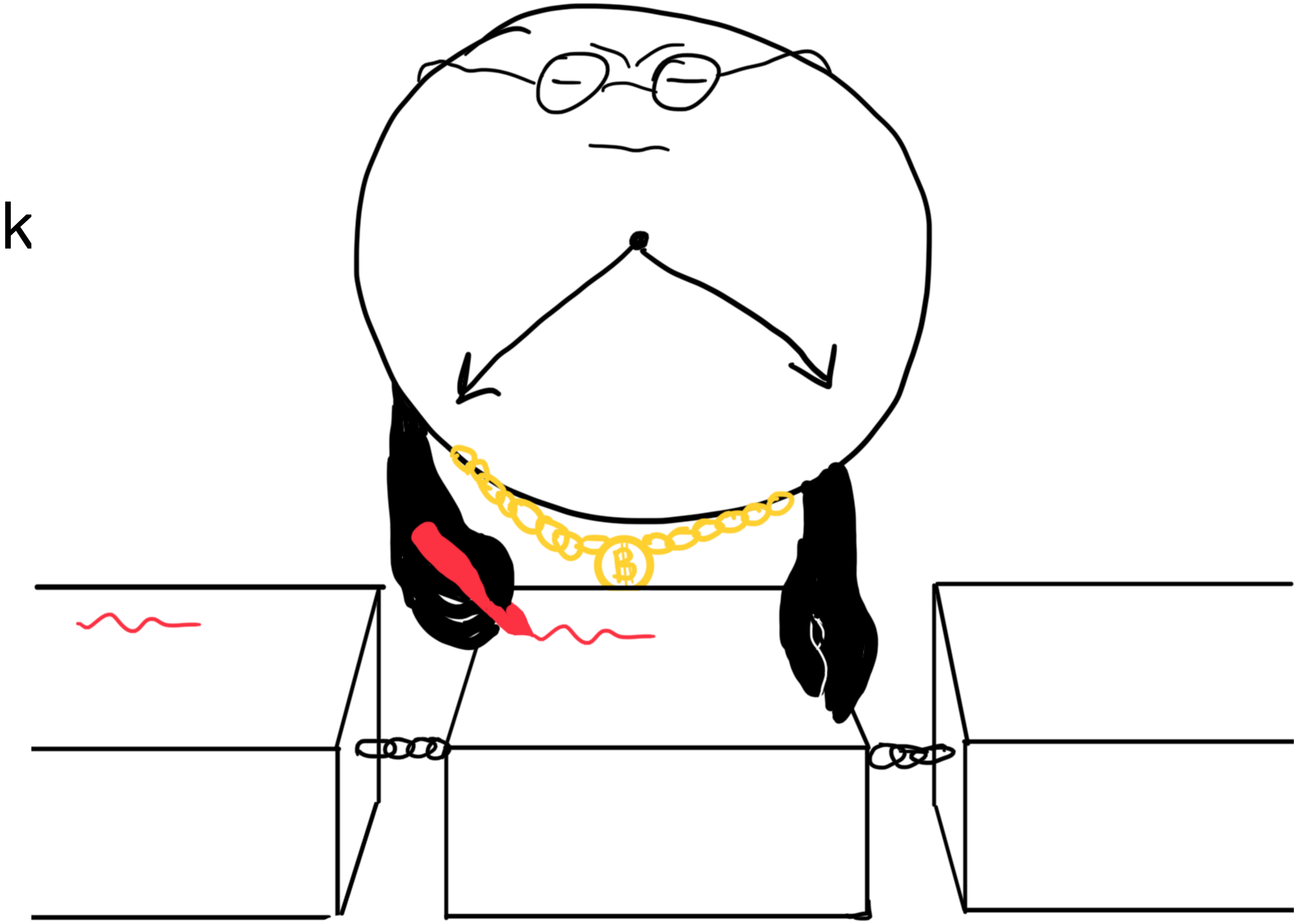
Today's permissioned blockchain systems come in a stand-alone fashion and require the users to integrate yet another full-fledged transaction processing system into their already complex data management landscape. This seems odd as blockchains and traditional DBMSs share large parts of their processing stack. Thus, rather than replacing the established data systems altogether, we advocate to simply 'chainify' them with a blockchain layer on top.

Unfortunately, this task is far more challenging than it sounds: As we want to build upon heterogeneous transaction processing systems, which potentially behave differently, we cannot rely on every organization to execute every transaction deterministically in the same way. Further, as these systems are already filled with data and being used by top-level applications, we also cannot rely on every organization being resilient against tampering with its local data.

Therefore, in this work, we will drop these assumptions and introduce a powerful processing model that avoids them in the first place: The so-called Whatever-LedgerConsensus (WLC) model allows us to create a highly flexible permissioned blockchain layer coined ChainifyDB that (a) is centered around bullet-proof database technology, (b) makes even stronger guarantees than existing permissioned systems, (c) provides a sophisticated recovery mechanism, (d) has an up to 6x higher throughput than the permissioned blockchain system Fabric, and (e) can easily be integrated into an existing heterogeneous database landscape.

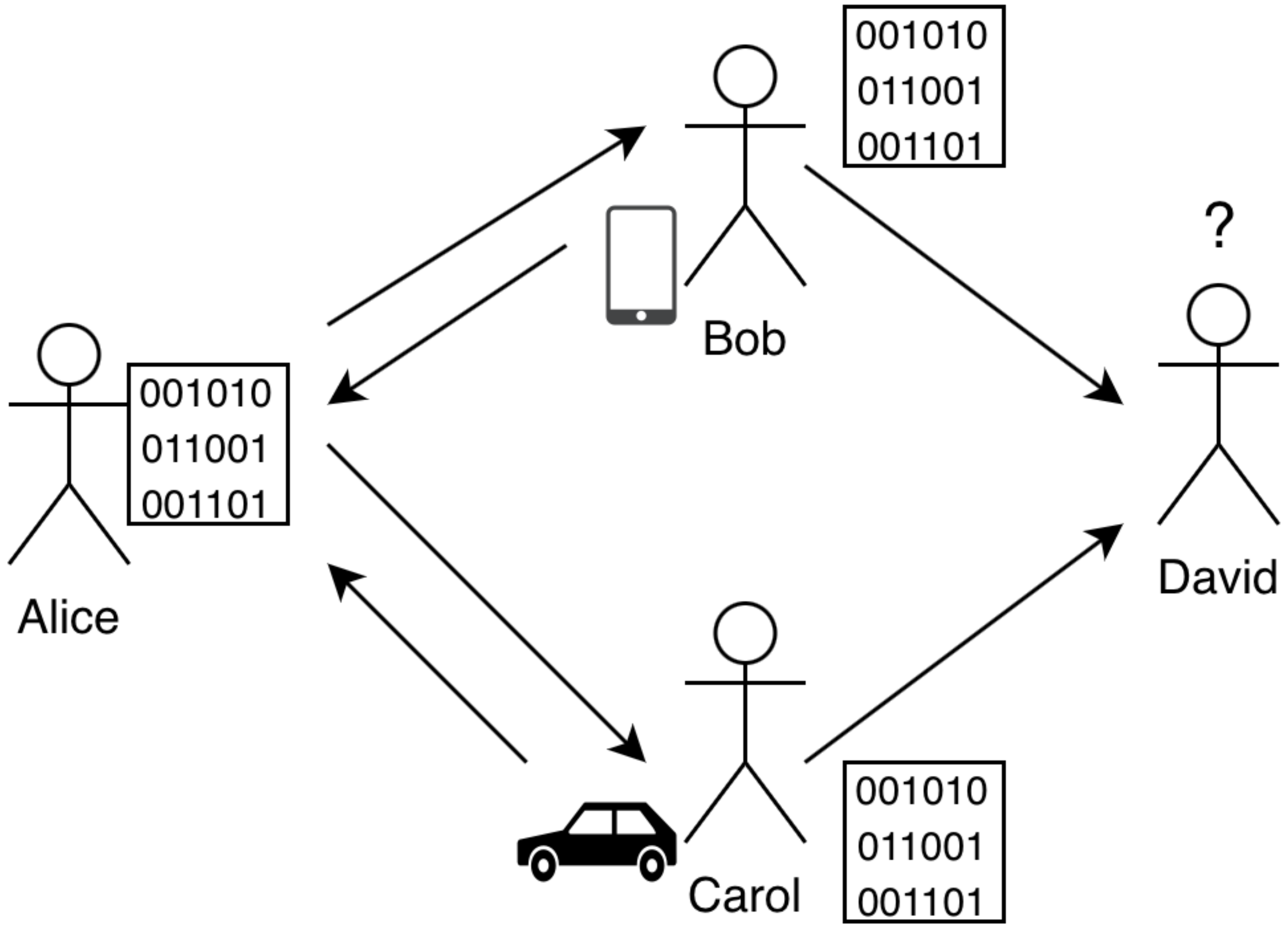
Blockchain

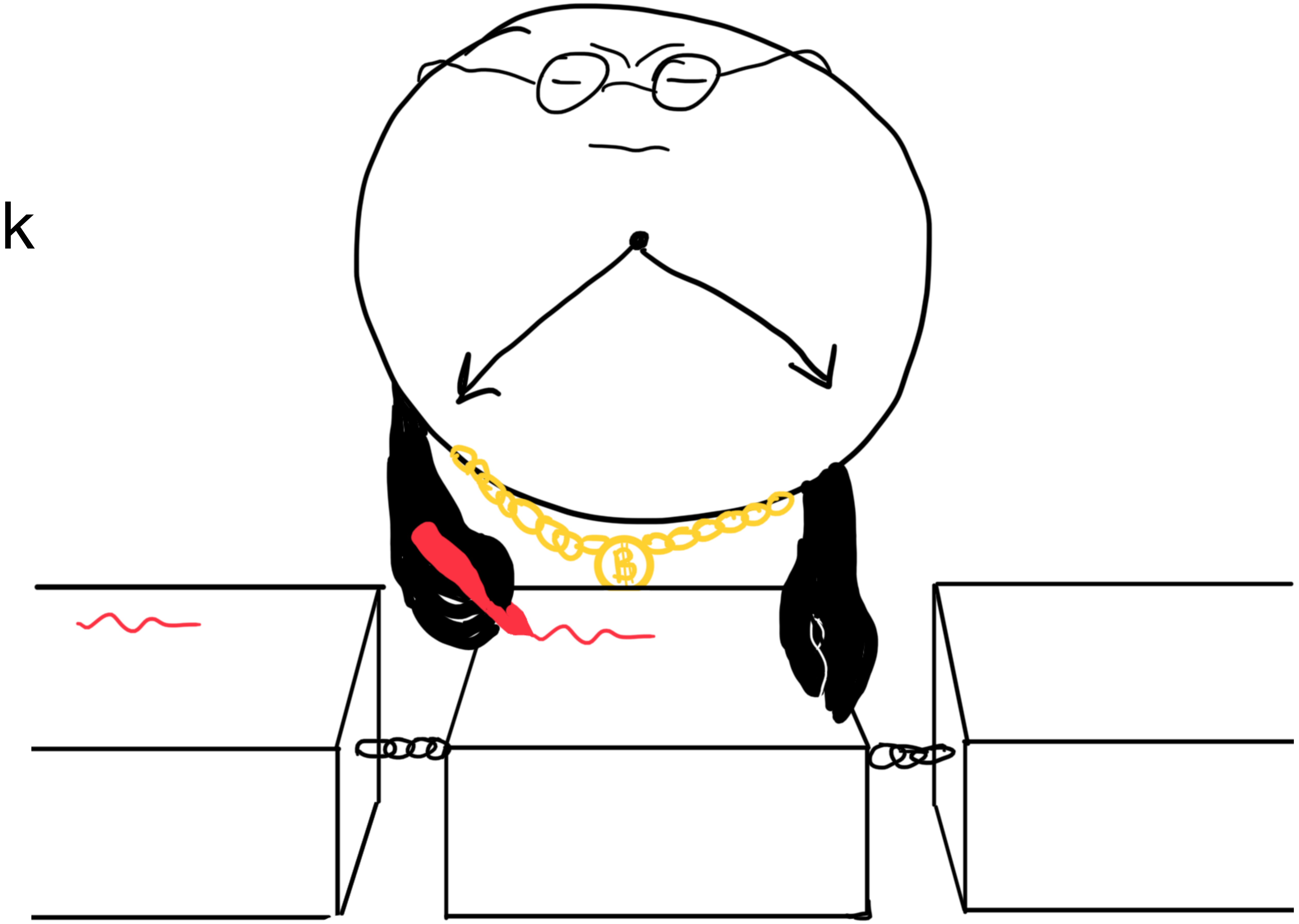
Blockchain is a decentralised clock securely validating and timestamping blocks of data.



Blockchain

Blockchain is a decentralised clock securely validating and timestamping blocks of data.





Blockchain is a

- **In the context of a databases,** blockchain is a secure and integral method of managing a distributed database.
- **economy,** blockchain is a technology behind digital cryptocurrencies.
- **distributed systems,** blockchain is an consensus algorithm for public and open-membership networks.
- **security,** blockchain is a technology that enforces rules of data correctness, agreed upon the majority of participants.
- **cryptography,** blockchain is decentralised trusted third party.
- **law,** blockchain is decentralised public notary.

Blockchain is a

- **In the context of a databases,** blockchain is a secure and integral method of managing a distributed database.
- **economy,** blockchain is a technology behind digital cryptocurrencies.
- **distributed systems,** blockchain is an consensus algorithm for public and open-membership networks.
- **security,** blockchain is a technology that enforces rules of data correctness, agreed upon the majority of participants.
- **cryptography,** blockchain is decentralised trusted third party.
- **law,** blockchain is decentralised public notary.

And in each area it solves some problems. That’s why its called disruptive technology.

What would you like to see in the next lecture?

visit menti.com and enter the code: 3400 0857
or directly: https://www.menti.com/al3z3283gq57
or scan the QR Code

Questions?

Stanisław Barański

stanislaw.baranski@pg.edu.pl

https://stan.bar