NIE-BLO – Blockchain

Introduction

by
Marek

Czech Technical University in Prague, Faculty of Information Technology
Department of information security

Hello Bitcoin

From a computer scientist’s perspective, Bitcoin can be seen as a peer-to-peer network of cooperating nodes. These nodes listen for transactions, order them into subsequent blocks and then publish these blocks on the network. Digital signatures of transactions are used to verify the ownership of funds, and a Proof-of-Work system based on hashing is used to rule out double-spending. These techniques bring absolute trust to the whole history of transactions, which, in turn, allows users to exchange value. Let us now unveil the fundamentals of this system.

Brief History

The initial era of Bitcoin is rather mysterious. The domain bitcoin.org was registered on 18 August 2008. On 31 October, Satoshi Nakamoto, whose identity is unknown, published a paper with the description of a peer-to-peer electronic version of cash named *Bitcoin* [8], and subsequently released an open-source implementation. On 3 January 2009, the *genesis block* was mined, which brought the Bitcoin network into existence. The first Bitcoin transaction was made on 12 January 2009. Since then, the code-base has been growing, and Satoshi Nakamoto completely disappeared from the Internet.

Brief History

The network has not experienced a single outage since the inception. At the time of writing this handout, there are a couple of thousands Bitcoin nodes operating worldwide, which makes the network reliable and robust. These nodes are run by volunteers since Bitcoin has no authority. Besides having no outage, there has been no successful attack on the Bitcoin blockchain that would deprive the owner of the funds they possess.

Bitcoin Protocol

The system is designed in a way such that there is no need for a trustworthy authority. This is accomplished by introducing a Proof-of-Work system alongside a data structure that was identified as the blockchain later on.

Bitcoin Protocol

The network protocol [13] operates by following these steps:

1. New transactions are broadcast in the network by clients.
2. Nodes gather incoming transactions into a block.
3. Each node starts finding a Proof-of-Work for the block being currently created.
4. Once any node finds the Proof-of-Work for the block, the node broadcasts the block in the network.
5. Other nodes accept this block only if all transactions in it are valid and not already spent.
6. Nodes express their acceptance of the block by starting to create another block on top of the received one, using the hash of the received block in the one currently being created.

Wallets and Addresses

Before discussing actual transactions, it is worth mentioning that a wallet in Bitcoin can be regarded in a simplified way as a public/private key pair. A user can generate an arbitrary amount of such key pairs; and so one user can handle any number of wallets.

Wallets and Addresses

A user generates a key pair on their own. This key pair is regarded as a wallet, and the public key of the key pair is regarded as the address of the wallet. The private key is kept secret and it can be used to manipulate the funds belonging to the wallet. The user then shares the address so other users that are in possession of some Bitcoin can send funds to this address by broadcasting a transaction that will either be accepted and written to the blockchain (only if it is a valid transaction) or simply ignored. Besides mining, there is no other way to obtain Bitcoin.

Wallets and Addresses

Note that there is no need to register anywhere. Also note that all the user needs in order to access the funds is the key pair. This means that the user is not bound to any device while using Bitcoin. The user can generate a new wallet on their phone (even with no internet connection), share the address of the wallet, retain the key pair and then destroy the phone. The user can also subsequently travel to the other side of the world just with the key pair and then use this key pair to access their funds on any device with an access to the internet and a Bitcoin client installed. This is possible due to the public nature of the blockchain described later on. However, if the user loses the private key, the funds are lost forever. Also, once the private key gets exposed to an adversary, this adversary instantly gains full control over the funds.

Wallets and Addresses

Another property of Bitcoin is that transactions are irreversible. This means that once the funds are sent to a wrong address (even an address that has no associated private key), and this transaction is written to the blockchain, there is no way to revert the transaction.

HD Wallets

Modern Bitcoin clients (often called simply wallets) implement a *hierarchical deterministic* (*HD*) wallet for deriving the key pairs from a *seed* [16]. This approach makes Bitcoin wallets more user-friendly while having no impact on the Bitcoin network. The seed is a random 128 bit number that can be encoded into a human-readable format. The seed is then called a *seed phrase* which consists of 12 English words that the user has to retain. The wallet can then deterministically create an unlimited number of addresses. The same seed can also be used on multiple devices so the user can control the exact set of addresses on any of these devices. The user doesn't have to retain any of the private keys. It is crucial to generate the seed in a random fashion in order to prevent the possibility of a collision with another seed. In the case of a collision, the consequences are that both users end up with the exact same key pairs which is fatal.

Addresses

A Bitcoin address is a string of length between 26 and 35 of alphanumeric characters, beginning with the number `1`, `3` or the string `bc1` [10]. The latter type of addresses is case insensitive, while the two preceding ones are case sensitive. No address contains the uppercase letter `O`, uppercase letter `I`, lowercase letter `l` and the number `0` in order to prevent visual ambiguity. An example of a Bitcoin address is

`1BvBMSEYstWetqTFn5Au4m4GFg7xJaNVN2`

or

`bc1qar0srrr7xfkvy5l643lydnw9re59gtzzwf5mdq`.

Privacy

Bitcoin cannot be considered private nor untraceable since all of its transactions (containing public keys and Bitcoin amounts) are public, and it is possible to track them, not only backwards but also forwards. This means that if we focus on a certain address or transaction, we can backtrack all the previous transactions as well as observe the future transactions. It is also possible to see how much funds each address holds.
If we manage to associate someone's identity with a certain address, the privacy of that user is lost. The information about the complete transaction history is immutably and publicly stored in the blockchain forever.

Privacy

It is recommended to generate a new address for every transaction, and wallets usually offer such addresses for each payment request or invoice. Even when a user sends only a fraction o their funds they hold on a particular address, the source address is usually fully withdrawn, and the change is sent to a newly generated address that belongs to the sender. This is done automatically by the wallet. These methods make tracing more difficult.

Privacy Revisited

In order to make the transactions private and untraceable, we can utilize zero-knowledge cryptography [1, 2]. Bitcoin does not implement these techniques, but some other cryptocurrencies do.

Network Surveillance

It is possible to prevent any potential network surveillance or censorship by using the Tor anonymity network for connecting to the Bitcoin network. Some wallets have this functionality inbuilt.

Chain of transactions

Transactions [15] in Bitcoin have the following content (simplified):

hash of itself	Every transaction is identified by its hash. The implementation uses SHA-256.
input	Contains the following:
	previous transaction — A hash referencing the previous transaction.
	signature — The hash of the current transaction is signed with the private key of the owner.
output	Contains the following:
	value — The amount of bitcoins that the sender is willing to send.
	receiver's address — The public key of the receiver's Bitcoin address.

Chain of transactions

This transaction structure creates a chain of transactions since the input references the previous transaction. We can notice that the transaction also contains the receiver's address and that the whole transaction is signed (containing the hash of the previous transactions). This in fact creates a chain of trust — the receiver (and anyone else) can verify that the sender owned the funds that are being sent in the following way:

Chain of transactions

The receiver looks at the transaction referenced in the input, and gets the sender’s public key from it. Subsequently, they verify that the signature of the current transaction is valid. If this is true, we can be sure that the funds are transferred (to the owner of the public key specified in the output) only by the owner of the private key that corresponds to the public key stored in the output of the referenced previous transaction.

Chain of transactions

A simplified chain of transactions is depicted in Figure 1.

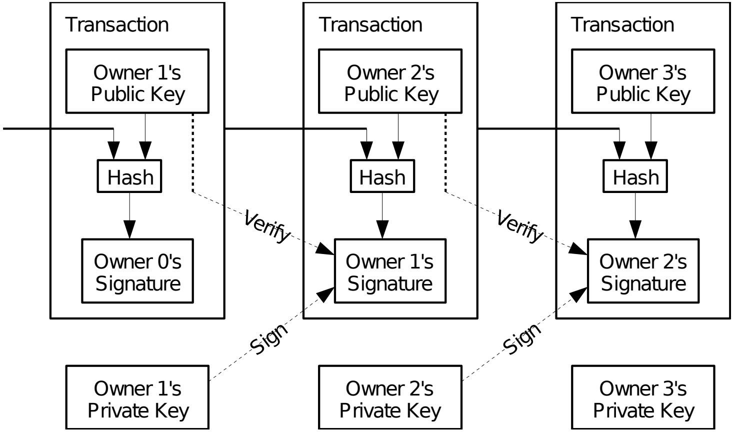

Figure: Outline of transactions and their ordering. [8]

Chain of transactions

Transactions can have multiple inputs and multiple outputs. This allows the value to be split and combined. All inputs are summed up, and this sum has to be spent in the outputs. [1] If we don't want to use all the funds specified in the input, we can simply send it back to the same wallet that we currently use for the transfer.

[1] Unused funds are used as an incentive for miners.

Chain of Transactions

The approach described above doesn't rule out double-spending. The owner could transfer their funds multiple times. To prevent this, we could introduce a global authority that would check every transaction. Bitcoin addresses this issue without the need of such an authority.

Cryptographic Primitives

The current implementation of Bitcoin utilizes the *Elliptic Curve Digital Signature Algorithm* (*ECDSA*); however, it would also be possible to use other algorithms such as the *Schnorr signature algorithm*. The description of ECDSA can be found in [3]. The algorithm employs the *secp256k1* elliptic curve defined in [4].

Cryptographic Primitives

ECDSA is based on the discrete logarithm problem for which there is no known algorithm that would solve the problem on a classical computer in polynomial time. In general, it is a function problem, and if formulated as a decision problem, it can be shown that it belongs to the $\mathcal{NP}$ complexity class. It can also be shown that the problem belongs to the *bounded-error quantum polynomial time* ($\mathcal{BQP}$) complexity class. This means that if humans ever manage to build a quantum computer, it might be possible to solve the problem in polynomial time by using Shor's order-finding algorithm; and therefore effectively break ECDSA.

Cryptographic Primitives

Let $\mathbb{F}_p$ be a finite field specified by the odd prime p and let $a, b \in \mathbb{F}_p$. The elliptic curve $E(\mathbb{F}_p)$ (named secp256k1 in the standard) over the finite field $\mathbb{F}_p$ is the set of solutions (which can be regarded as points) $P = (x, y)$ for $x, y \in \mathbb{F}_p$ that satisfy the equation:

$$E : y^2 \equiv x^3 + ax + b \pmod{p},$$

where $a = 0$, $b = 7$ and $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$. Since the coefficient a is zero, the equation becomes

$$y^2 \equiv x^3 + 7 \pmod{p}.$$

Cryptographic Primitives

In fact, the finite field $\mathbb{F}_p$ is the prime field $\mathbb{Z}_p$ with the characteristic p. It can be easily shown that the points on the curve form an aditive Abelian group under specifically defined addition. The standard also specifies the base point G which is a generator of a subgroup of the original group. Since the cofactor of the base point is 1, the base point actually generates the whole group. The order n of the base point; and in fact the whole group, is a 256 bit number. The private key in ECDSA is a randomly selected integer in the interval $[1, n-1]$ and the public key is a point on the curve.

Cryptographic Primitives

Note that a point on the curve consists of two coordinates, both of which are 256-bit numbers. However, for any x coordinate there will only ever be two possible values of y. We can therefore encode the y coordinate just by a single bit. This allows us to encode any point in a compressed form that is approximately 256 bits long.

Cryptographic Primitives

We see that the curve offers 256 bits of entropy since we have approximately 2^{256} points; however, we have to consider it together with ECDSA which is based on the discrete logarithm problem. The algorithms that solve the problem, such as the baby-step giant-step algorithm, have time complexity $\mathcal{O}(\sqrt{n})$ where n is the order of the group. Since $n \approx 2^{256}$, the actual complexity is $\mathcal{O}(\sqrt{2^{256}}) = \mathcal{O}(2^{128})$. This gives us 128 bits of security which is equivalent to the 3072 bit RSA/DSA modulus (the size of a properly encoded public key is approximately 256 bits). We see that elliptic curve cryptography offers much shorter keys when compared to RSA/DSA.

Cryptographic Primitives

The actual Bitcoin address is a 160-bit hash of the public key. This means there are two ways a potential collision could occur. The first possibility is that two users generate the same keys that will result in the same hash. The second possibility is that two different keys will result in the same hash as well. Even though the keys are different, the signature will still be valid and the users will be able to manipulate each others funds. The chance for any of these collisions to occur is negligible provided we use a true random number generator.

Encryption in Bitcoin

Bitcoin uses encryption merely for the implementation of digital signatures.

Script

Each transaction in Bitcoin can be regarded as a short program or a trivial smart contract — each transaction contains executable data that are executed by the node. Bitcoin uses a scripting system called **Script** which is stack-oriented. It contains dozens of instructions and is intentionally not Turing-complete as it contains no instructions for jumps or loops. Such design is for security reasons — it is not possible to create a malicious transaction that would make the node stuck in an infinite loop. The following table demonstrates some instructions.
A zero value is interpreted as *false*.

Script Instructions

Instruction	**Input**	**Output**	**Description**
OP_DUP	x	x, x	Duplicates the top stack item.
OP_EQUAL	x_1, x_2	true / false	Returns 1 if the inputs are exactly equal, 0 otherwise.
OP_VERIFY	true / false	nothing / fail	Marks transaction as invalid if top stack value is not true. The top stack value is removed.
OP_EQUALVERIFY	x_1, x_2	nothing / fail	Same as OP_EQUAL, but runs OP_VERIFY afterward.
OP_HASH160	x_1	hash	The input is hashed twice: first with SHA-256 and then with RIPEMD-160.

Table: Common script instructions.

Script Instructions

Instruction	Input	Output	Description
OP_CODESEPARATOR	nothing	nothing	All of the signature checking words will only match signatures to the data after the most recently-executed OP_CODESEPARATOR.
OP_CHECKSIG	*sig*, *pubkey*	true / false	The entire transaction's outputs, inputs, and script (from the most recently-executed OP_CODESEPARATOR to the end) are hashed. The signature used by OP_CHECKSIG must be a valid signature for this hash and public key. If it is, 1 is returned, 0 otherwise.

Table: Common script instructions.

Script Example

Let us now demonstrate a Script example. Consider Figure 1 and say that the transaction containing "Owner 0's signature" is T_0 and the transaction containing "Owner 1's signature" is T_1. Let transaction T_0 contain in its output a script named *scriptPubKey* and let transaction T_1 contain in its input a script named *scriptSig*. The content of the scripts is the following:

scriptPubKey: `OP_DUP`, `OP_HASH160`, `<pubKeyHash>`, `OP_EQUALVERIFY`, `OP_CHECKSIG`.

scriptSig: `<sig>`, `<pubKey>`.

The data field `<pubKeyHash>` represents the address to which the funds are being sent and `<pubKey>` is "Owner 1's public key". The data field `<sig>` is "Owner 1's signature".

Script Example

We can now validate transaction T_1 by following the steps in table 4. Note that the data fields are implicitly pushed on top of the stack. The transaction is valid if the stack contains true at the top and the Script interpreter has finished the execution. We can see that the transaction is successfully validated. This example in fact shows one of the standard ways of validating transactions in Bitcoin.

Script Example

Stack	Script	Description
Empty.	`<sig>`, `<pubKey>`, `OP_DUP`, `OP_HASH160`, `<pubKeyHash>`, `OP_EQUALVERIFY`, `OP_CHECKSIG`	scriptSig and scriptPubKey are combined.
`<sig>`, `<pubKey>`	`OP_DUP`, `OP_HASH160`, `<pubKeyHash>`, `OP_EQUALVERIFY`, `OP_CHECKSIG`	Constants are added to the stack.
`<sig>`, `<pubKey>`, `<pubKey>`	`OP_HASH160`, `<pubKeyHash>`, `OP_EQUALVERIFY`, `OP_CHECKSIG`	Top stack item is duplicated.
`<sig>`, `<pubKey>`, `<pubHashA>`	`<pubKeyHash>`, `OP_EQUALVERIFY`, `OP_CHECKSIG`	Top stack item is hashed.

Table: Script example: validation of a standard transaction [14].

Script Example

Stack	Script	Description
`<sig>`, `<pubKey>`, `<pubHashA>`, `<pubKeyHash>`	`OP_EQUALVERIFY`, `OP_CHECKSIG`	Constant is added to the stack.
`<sig>`, `<pubKey>`	`OP_CHECKSIG`	Equality is checked between the top two stack items.
`true`	Empty.	Signature is checked for top two stack items.

Table: Script example: validation of a standard transaction [14].

Chain of Blocks

Transactions are collected in blocks [11], which contain (simplified):

transactions	A list of new transactions recently broadcasted by the network.
hash of itself	Each block is referenced by this hash. The implementation uses SHA-256.
previous block	A hash referencing the previous block.
time	A timestamp of the time when the block was created.
target	A number that determines the difficulty for the Proof-of-Work - explained below.
nonce	A 32-bit number that serves as the Proof-of-Work - explained below.

Proof-of-Work

After receiving some transactions (the amount is currently arbitrary), nodes start looking for a value of the nonce such that the hash of the whole block (containing the nonce) is less than the value specified in the current target. If a node happens to find such a nonce, it broadcasts the block and starts working on a new one. If the node exhausts the space defined by the size of the nonce (2^{32}), it can simply update the timestamp and reset the nonce.

Proof-of-Work

It is important to realize that the process of looking for the right nonce is not a long set problem. Each hash produces a random number between 0 and 2^{256}, so looking for the right nonce is a lottery. No matter how many nonces we have tried before, the probability of getting the right one remains the same. Thinking otherwise results into the gambler's fallacy belief. The value of the target is adjusted by the network every 2016 blocks so that on average it takes about 10 minutes for the whole network to generate a new block. This adjustment is made in order to keep the network stable while the hash power changes over time. Not every node in the network needs to search for the right nonce. Nodes that do so are called miners.

Valid Blocks

Other nodes will accept the newly broadcasted block only if all transactions in it are valid (by checking the preceding transactions), and only if the hash of the block (including the right nonce) is less than the specified target (to check this, only one hash is needed). Every block contains a reference to the previous block; and therefore, it is extremely difficult to change any data in the previously accepted blocks. This would require re-hashing all the blocks that follow after the altered one. As transactions in accepted blocks are buried under new blocks, it becomes impossible for anybody to change them. This mechanism brings complete trust to the whole history of all the blocks. It is also not anymore necessary to check every preceding transaction of a new block being currently verified. It is sufficient to check just a couple of preceding transactions since the older ones are immutable and they were checked already during their acceptance.

Double Spending Prevention

In order to prevent double-spending, when a merchant receives a new transaction in a new valid block, the merchant waits for a couple of next valid blocks (we can think of them as confirmations) being mined on top of the received one and then approves the transaction. At this point, the merchant can be sure that the funds were not double-spent.

Double Spending Prevention

If there are multiple different blocks coming from different miners (some of them might contain double-spent funds, and at this point, the merchant cannot tell which block contains the primary transaction), the network will wait and accept the block that has the most blocks built on top of it — this block has the greatest Proof-of-Work effort invested in it. After this, the accepted block is considered valid. All other blocks are thrown away, and the network continues only with the one that is contained in the longest chain. As long as the honest nodes comprise the majority of the network, double-spending is ruled out.

51% Attack

It is possible to attack the Bitcoin network with an enormous computational power. This attack is known as the *51% attack*. If an attacker has at least 50% of the hash power of the network, they can generate forged blocks faster than honest nodes. This would not mean that they could create value out of nothing, but it would allow them to double-spend the funds they recently spent in the following way:

51% Attack

An attacker broadcasts a transaction which pays a merchant, while privately mining an alternative fork of the blockchain, in which the attacker includes a double-spent transaction, or excludes the transaction sent to the merchant. The merchant sends the goods after waiting for *n* confirmations (blocks). However, if the attacker has mined more than *n* blocks containing the fraudulent transaction, they release their private version of the blockchain, and the network will accept this version instead of the blocks mined by honest miners. If the attacker has less than *n* mined blocks, they simply wait until their private blockchain is longer than the one generated by honest miners, and then publish it. The attacker has always the ability to do so since they have the majority of the mining power of the network. The original transaction that was approved by the merchant will then be discarded by the network, and the merchant will not be able to use the received funds anymore.

The Feasibility of a 51% Attack

Such an attack is not infeasible, but rather economically demanding. The attacker would have to spend substantial resources in order to undermine the network; which would, in turn, decrease the value of Bitcoin, making the economic benefits of such an attack unprofitable. The attacker could instead use the resources for standard mining. The current hash rate of the Bitcoin network is approximately 100 exahash of SHA-256 per second which is an unprecedented number. The energetic demand of Bitcoin mining is a questionable topic. It is also worth noting that the mining power is currently centralized around a single company named Bitmain since this company is the current market leader in selling mining hardware.

51% Attack Revisited

Notice that a 51% attack is not a typical Sybil attack in which the attacker would create a large amount of fake identities which would be used to gain the mining influence. Mining power of the attacker is determined solely by their hash power which depends on computational resources and energy.

Coin Mints

One Bitcoin comprises of 10^8 satoshis. One satoshi is currently the smallest unit of the Bitcoin currency, so the network actually operates solely with satoshis and not bitcoins. When a node happens to find the right nonce, it is allowed to include a special transaction that introduces new coins. Such a transaction is called a *coinbase* transaction. These transactions do not reference any previous transactions. At the inception of Bitcoin, each coinbase transaction was worth 50 BTC. This number keeps halving every 210,000 blocks (approximately every 4 years). This process is known under the term *halving*.

Coin Mints

The total amount of bitcoins in circulation is given by the formula

$$\sum BTC = \sum_{i=0}^{32} \frac{210000 \lfloor \frac{50 \cdot 10^8}{2^i} \rfloor}{10^8}.$$

This is a geometrically decreasing sequence meaning that the final amount of bitcoins in circulation will not exceed 21 million. The coinbase transaction also motivates miners to support the network and spend their resources on hashing.

Transaction Fees

A node that creates a new block can also claim all inputs in transactions that do not have corresponding outputs. Such inputs are called transaction fees, and users can motivate miners to prefer their transactions to be processed with higher priority by providing a higher transaction fee. Fees also prevent spam transactions.

Transaction Fees

During the first couple of years, there were usually no fees associated with transactions. As the transaction volume was growing, fees became a norm, and transactions with no fees are simply ignored. This prevents an attacker from flooding the network by an enormous amount of dummy transactions that could slow down the network for regular users. It is also expected that fees will be the main motivation for miners once the predetermined number of coins have entered circulation.

Mining Pools

Miners nowadays unite their hashing power under mining pools as the probability of hitting the right hash for a single miner is very low.
A mining pool then rewards each miner according to their hashing contribution to the pool. The pool sends candidate blocks with lowered target to the miners. These candidate blocks are called *shares*. The target is lowered so that the miner can solve it in a reasonable time. By measuring how long it takes to each miner, the pool can determine the miner's hash power, and subsequently the reward for the miner. Each share contains a coinbase transaction with the receiver's address set to the pool's address. Note that at some point, one of the miners generates a hash that is not only lower than the target required by the share but also by the whole Bitcoin network. Also note that once a miner finds such a hash, they cannot simply change the coinbase transaction and point it to their address since doing so would invalidate the hash.

Blockchain Size

The actual implementation hashes the transactions into a Merkle tree. This is an optimisation in order to allow verification of payments without storing the transactions. This avoids using the whole blockchain, the current size of which is about 350 GB. The growth is naturally linear at the rate of approximately 60 GB per year. Merkle trees are also useful for keeping the integrity (by providing hash checks) of the whole blockchain since it is distributed over the world.

Light Clients

For a Bitcoin light client, it is sufficient to keep only the root hash of the Merkle tree. If such a client needs to verify a transaction, it can link it to a place in the chain according to its timestamp and check that other full nodes accepted it. This method is called Simplified Payment Verification (SPV). However, light clients are vulnerable to second preimage attacks (an attacker could forge the interior of the Merkle tree). Also note that if a user does not wish to share which transactions are of an interest to them, they have to run a full node (and store the whole blockchain) since light clients request specific transactions from other nodes.

Scalability

A drawback that was not originally mentioned in the original paper is that the Bitcoin network is not able to process an arbitrary amount of transactions over a certain period of time. The difficulty is regularly (every 2016 blocks) adjusted so that it takes about 10 minutes for the whole network to generate a new block.

Scalability

The difficulty is given by the value of the target which is a 256 bit number. This value is stored in each block in the “Bits” field, the size of which is 4 bytes. The first byte is the exponent e and the remaining three bytes constitute the mantissa m. The value of the target is then calculated according to the formula

$$target = 256^{e-3} m.$$

The target is adjusted every 2016 blocks so that

$$target_{new} = target_{previous} \frac{\delta}{2 weeks},$$

where δ is the actual time difference between the 2016 blocks.

Scalability

Every block can contain only a limited amount of transactions. Based on this, it is possible to estimate that the network is able to process only about 7 transactions per second worldwide. This issue is being currently solved by introducing another layer of transactions over the original blockchain. This solution is called the Lightning Network [9].

Broader Outlook

The initial ideas and purposes of Bitcoin as described in [8] can be extended to various other fields. For example, since transactions can contain a certain amount of arbitrary data, we can store this data to the blockchain; and since we can think of the blockchain as an immutable distributed public database, this data will be immutably and publicly stored around the world for as long as the humanity does not lose its computers. If we store a hash of a photograph of a contract with a counter-party into the blockchain at a certain time, the counter-party will never be able to deny the existence of the document after this time. A demonstration of this possibility is the famous message that Satoshi Nakamoto left in the genesis block:

> *"The Times 03/Jan/2009 Chancellor on brink of second bailout for banks".*

The message proves that the block could not be created earlier than 3 January 2009. It is the headline of a British national newspaper The Times.

Extensions

Another popular extension of the blockchain is the use of *smart contracts* [12, 6].
The Proof-of-Work algorithm by which Bitcoin achieves distributed consensus can be changed to a different one called *Proof-of-Stake (PoS)* [5], which doesn't inherently require computational power.

Politics

Let us now briefly discuss Bitcoin's financial and sociological impact. It's natural that Bitcoin attracts the attention of not only computer and cryptography enthusiasts, but also politicians and economists, since it brings a new level of freedom and responsibility to the human society. Bitcoin has the potential to reduce the power of governments as it renders central banks powerless as regards its control. It also opens the question whether the energetic demands of the Proof-of-Work are worth it when confronted with the current environmental situation.

Economy

There are economists who say that Bitcoin is necessarily going to vanish since it has nothing to offer, and there is no real value behind it. There are also people from the same field that claim that Bitcoin is going to change the world completely. Some people see it as a great store of value while others don’t trust it whatsoever. There are many philosophical stances people adopt on Bitcoin, ranging through the whole spectrum of both positive and negative radical opinions.

Forks

Since the Bitcoin code base is an open platform, anyone can fork the source code in the software engineering sense, and start either a completely new chain of blocks from the very beginning or *fork* the existing Bitcoin blockchain at a specific point, and continue in its own way. In both cases, there is a new cryptocurrency created since the new blocks are not compatible with the Bitcoin blockchain anymore. In the latter case, users can keep and use all the funds they owned in the standard Bitcoin blockchain. There are currently thousands of such cryptocurrencies. This demonstrates the voluntary nature of Bitcoin since anyone can use (including operating a full-node or mining) any chain of blocks they prefer. Operators running Bitcoin nodes and miners decide by themselves what ‘version’ of the blockchain they use and nobody can force them not to do so. The same holds for ordinary Bitcoin users and programmers who propose and develop new features.

Bitcoin’s Price

This brings us to a dangling question that some people might ask: what determines the price of Bitcoin? The answer is probably simpler than some may expect — it’s the willingness of others to get it. Bitcoin is currently considered quite volatile. It is expected that potential future adoption by the masses will make it much more stable. Let us conclude our discussion with a quote by the mysterious creator of Bitcoin, Satoshi Nakamoto, from 14th February 2010 [7]:

> *“I’m sure that in 20 years there will either be very large transaction volume or no volume.”*

Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza.
Zerocash: Decentralized anonymous payments from bitcoin.
`http://zerocash-project.org/media/pdf/zerocash-extended-20140518.pdf`, May 2014.

Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza.
Succinct non-interactive zero knowledge for a von neumann architecture.
`https://eprint.iacr.org/2013/879.pdf`, May 2015.

Daniel RL Brown.
Sec 1: Elliptic curve cryptography.
Certicom Research, http://www. secg. org/download/aid-80/sec1-v2. pdf, 2009.

Daniel RL Brown.
Sec 2: Recommended elliptic curve domain parameters.
Standards for Efficient Cryptography, 2010.

Vitalik Buterin.
What proof of stake is and why it matters.
https://bitcoinmagazine.com/articles/
what-proof-of-stake-is-and-why-it-matters-13775314
August 2013.

Ethereum community.
A next-generation smart contract and decentralized application platform.
https:
//github.com/ethereum/wiki/wiki/White-Paper/
f18902f4e7fb21dc92b37e8a0963eec4b3f4793a.

Satoshi Nakamoto.
https://bitcointalk.org/index.php?topic=48.
msg311#msg311.

Satoshi Nakamoto.
Bitcoin: A peer-to-peer electronic cash system.
https://bitcoin.org/bitcoin.pdf, November 2008.

Joseph Poon and Thaddeus Dryja.
The bitcoin lightning network: Scalable off-chain instant payments.
https://lightning.network/lightning-network-paper.pdf, January 2016.

Bitcoin Wiki.
Address.
https://en.bitcoin.it/wiki/Address.

Bitcoin Wiki.
Block.
https://en.bitcoin.it/wiki/Block.

Bitcoin Wiki.
Contract.
https://en.bitcoin.it/wiki/Contract.

Bitcoin Wiki.
Protocol documentation.

https://en.bitcoin.it/wiki/Protocol_documentation.

Bitcoin Wiki.
Script.
https://en.bitcoin.it/wiki/Script.

Bitcoin Wiki.
Transaction.
https://en.bitcoin.it/wiki/Transaction.

Pieter Wuille.
Bip 0032.
https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki.