Yobicash

A cryptocurrency for data

Christian Nyumbayire

December, 20, 2016

Abstract

Yobicash is a cryptocurrency for storing, managing and sending data. Data ownership and privacy are secured by cryptography, and a memory-strict proof-of-work mining process assures data availability and consistency in the presence of untrustable nodes.

1. Introduction

Yobicash is a cryptocurrency that let users store, transfer and delete data in a verified, confidential way thanks to the use of a Bitcoin-like blockchain.

Through encryption, only data signers can read it, ensuring privacy. Data can be co-owned and managed by groups of users with different rights. Some may be just allowed to read, others may have the right to write, delete and transfer, others may have to first win in weighted elections with their peers to do any action.

To ensure its use value, Yobicash prioritize throughput and scalability, processing 10 thousand transactions per second for megabyte per block.

1.1. Historical background

It started in 1988, when Mark S. Miller and K. Eric Drexler published “The Ecology of Computation” [1]. Mark and Eric envisioned trustless computer networks sharing data and computation for economical returns. The security which any common distributed system could not achieve, would have been granted by decentralization and the market process. They called it the agoric system, from the ancient Greek for market, *agorà.*

In 2008 Satoshi Nakamoto, a pseudonymous user or group, published the paper of a cryptographically enforced digital currency, Bitcoin [2]. The currency tokens were produced through a shared computation effort called mining and transferred through a shared ledger of transactions, the blockchain. Market forces, cryptography and transparency were the only

means needed to achieve efficiently online what was quite impossible to achieve offline: a single source of truth from the cooperative work of untrustable actors. It was the first successful embodiment of an agoric system. Since its development Bitcoin has been the most successful cryptocurrency, reaching a market capitalization of 12 billion dollars

After Bitcoin, new cryptocurrencies have arisen, each one with a different market proposition. This ecosystem of agoric systems based on the blockchain (from here also as just "blockchains") is maturing and giving more returns, day by day. Hopes are that the blockchain as a general purpose technology will change the face of every market and government function by reducing verification and network costs [3].

Although the high hopes real world blockchains are still too specific in use, centralized and cannot scale to reach global usage.

1.2. Generality

Attempts of generality has been done by blockchains as Factom [4] and Ethereum [5]. They exemplify the way this problem has been addressed so far: generalizing data or generalizing computation.

Factom incentivizes nodes to store files which checksum – a cryptographic proof of their existence – is shared on a common blockchain. To address scalability Factom does not compel nodes to host all the files which are declared in its blockchain, which means that data access and ownership may not be ensured. Its blockchain guarantees only checksum existence, ownership and transfer.

Besides, data confidentiality is not ensured, which means that most users would not use it if not for very limited uses or that they would prefer to store personal data only in stores they trust.

By consequence, Factom and similar systems are not partition tolerant, which means that any node loss by the network may means a data loss for the blockchain.

Although this drawback, the approach of generalizing data – blockchains for data – let computation be done offline and verified through a proof of process. In a proof of process for code execution the blockchain takes care only of recording code, inputs, midsteps and outputs, so to make everyone interested and allowed capable to verify all the steps – the process – of the computation.

Ethereum instead attempts to generalize computation, and so its a blockchain for computation. Data on Ethereum is only the data needed to define its computation – "contracts" – and so has specific semantics and its limited in scope and size. Contracts are written in a specific programming language, Serpent, and each node in the network execs the contracts when they are called and checks other nodes' results by comparing them with its owns. The burden of

computation is shared among all the nodes. Data which exceeds the size and semantics of Ethereum has to be put offline, and referenced in some way by the system.

Many have attempted to use Ethereum as a generic programming environment, building on top of it token systems and complex contracts. Because of its limitations and the lack of safety of its programming language, Ethereum is more famous for its hacks [6] and missed chances [7, 8].

Yobicash is a blockchain for data, but differently from Factom and similar systems, it lets users put raw data directly in the blockchain, ensuring partition tolerance. Data is treated as converted value and can be converted back to value at any time, so it is not for nodes just a cost but also economical value. Confidentiality is ensured through encryption while different scalability measures, the market for fees and the incentive of every node owner to invest in storage make its usage sustainable over time.

Data can be co-owned by different signers having different voting weights. Signers with no voting weight will have no voice in the matter of data management, but will be able to see the decrypted data. Signers with the majority of the vote weights will have a major weight on management issues, if not the last word.

1.3 Decentralization

Decentralization assures resilience to the network. In the presence of geographically localized malicious actors interested of taking over the network or particular mistrusts among actors of different regions, it is a guarantee of fault tolerance.

Although cryptocurrencies should be decentralized by nature, the mining algorithm increasing difficulty can favor geographically localized nodes and disfavor others. This kind of concerns have been moved against Bitcoin, which has ended by being prevalently mined in China. This centralization has increased the mistrust of many actors on the system.

Many cryptocurrencies have addressed this problem by using mining algorithms which do not slightly favor anyone. Memory-strict hashing algorithms are the most famous class from this group of alternative mining algorithms. Yobicash uses a modified version of randmemohash, by Damian Lerner [9].

1.4 Scalability

Scalability is paramount for a blockchain to reach global usage. In the context of blockchains it means the capacity of the network to increase the number of transactions per second when needed. VISANet can process 56,000 tps (transactions per second) [10], while Bitcoin a theoretical maximum of 7 tps [11].

This feature is very important even for some applications to be viable, e.g. blockchain embedded financial markets and IOT applications.

Scalability was not a priority in the times of Satoshi. He expressed the need to increase the size of transactions and blocks in future times [12], not to change the protocol nor its data structures. Bitcoin, stacked to a limit of 1Mb for blocks and transactions, by this time confirms 3 tps on average.

Many papers has dealt with the issue [13, 14, 15, 16], but currently the shipped solutions are to reduce the times for confirming transactions [17, 18, 19] or to outsource the verification process [20].

Yobicash achieves 10 thousand transactions per megabyte of a block, which means that for a 4 MB block it can confirm 40 thousand transactions. Blocks have a size limit but it changes according to usage. The real limit on the real number of transactions per second is given by the actual average time of propagation of single transactions.

This feature could open up to new applications needing fast responses times.

2. Use cases

Yobicash is a means for securing data hosting, ownership and transfer in an unsafe channel – the cloud. Thanks to its generality and focus on scalability, privacy and decentralization, Yobicash allows for many use cases.

2.1. Safe hosting

Yobicash allows users to host data and co-own it with different voting weights. Some users may be allowed to be only readers, some may be considered editors while others may be defined administrators holding the majority of voting weights.

Data can be copied to other users by sending copies to them, while deletion are just conversions back to value.

2.2. Safe messaging

Messaging is just transferring data. Thanks to encryption, messaging is confidential by default.

2.3. Proof of existence

A proof of existence is an hosted checksum of a digital artifact or the same artifact. When putting the entire data may be expensive, resorting to its checksum may be a viable alternative when the actual host of the raw data is trusted. The timestamp of the transaction containing the proof may be considered as the timestamp of the digital artifact itself.

2.4. Proof of ownership

A proof of ownership or smart property is a signed proof of existence. The signer or signers declare by signing the proof to own the underlined digital artifact and, if present, its offline referee. Trusted signers can co-sign the proof to reinforce the claim. The transfer of the proof from a user or group of users to an other represent a transfer of ownership.

2.5. Proof of authorship

A proof of authorship is a signed proof of existence. Differently from the proof of ownership its not transferable. This difference is not enforced nor enforceable by the Yobicash protocol, but it is a responsibility of the parties to make it clear, e.g. by using particular semantics.

Proofs of authorship can be used to enforce intellectual property of digital products.

2.6. Proof of process

A proof of process is a series of chained proofs of existence or authorship witnessing different phases of a continuous or discrete process. The process may range from the execution of an offline task by some human beings or robots to the execution of some code by some remote servers to the time series of a financial ticket.

The signer or signers of a step of a process are considered witnesses, verifiers or executors of that process.

A proof of process can be used for distributed crowd-sourcing, crowd-funding and task management. Multi-party computations can store their steps on the Yobicash blockchain using it as a safe channel.

2.7. Smart oracles

Smart oracles are proofs of existence or authorship by trusted sources on raw data or checksums dealing with particular events or general facts.

Financial and IOT applications can read time series data from smart oracles writing on the Yobicash blockchain.

2.8. Smart contracts

Smart contracts are proof of processes where the process is the execution of some code involving changes of rights, properties or balances of the parties involved. They are the coded version of offline contracts, run by servers online.

This contracts can reference as inputs data already present in the blockchain, or even future steps of some proven process. That data may have been registered by generally trusted parties like in smart oracles or be just trusted by the parties involved.

The executed code can be in any language the parties agree on, from JavaScript to Julia to Haskell. It is the parties responsibility to choose the language that fits more their needs.

3. Design

Yobicash design is similar to Bitcoin's one, with some key differences to ensure simplicity, easy and confidential data management and scalability.

3.1 Overview

Yobicash is a Bitcoin-like peer-to-peer network. The network is composed by nodes, clients and wallets sharing transactions and blocks.

Nodes store and broadcast to each others transactions and blocks. The data stored is used to build the blockchain, a verified chain of confirmed blocks. The blockchain has a predefined first block, the genesys block. Every block after the genesys block is selected according to a block selection algorithm prioritizing blocks with more transactions and more data.

Nodes can choose to consume CPU and memory in the mining process, in return of a given amount of Yobicash, the coinbase. The coinbase can be consumed only after an epoch, 100

confirmed blocks. The coinbase amount decreases over time until the total amount of Yobicash produced reaches 2.1 * 2^80, 2.1 yobis.

The mining algorithm used is an hardened version of randmemohash [9], a memory strict hashing algorithm. The consumed memory helps decentralization allowing common devices to be competitive in the mining process.

The algorithm has a varying difficulty to assure that the average block confirmation time is 17 seconds.

Clients transfer value and data between themselves through transactions. They store transactions in personal wallets and send them to nodes to validate and include them in the blockchain.

Data can be converted back and forth from value following a dynamic conversion rate set by the market.

3.2. Cryptography

Yobicash relies on 4 cryptographic primitives: cryptographic hashes, public key signature, symmetric encryption and public key encryption. What is following does not constitute a complete specification of Yobicash, but delines its main features.

3.2.1. Cryptographic hashes

Cryptographic hashes are functions that maps binary data of any size to equally sized collision-resistant (unique) short binary strings. Digests, the outputs of this functions, are used in Yobicash as checksums and ids.

The cryptographic hashes used in Yobicash are SHA512 (or just SHA3 or Keccak) and Ripemd160.

From now on they will be referenced as SHA512(•) and RIPEMD160(•).

3.2.2 Public Key Signatures

A public key signature is a scheme for cryptographically signing a binary data message. The scheme generates a couple of keys, the private and public keys, used accordingly for signing

and verifying a signature. The private key (sk) is kept secret by its user, while the public key (pk) is publicly used to reference the user and verify her signatures (sig).

The phases of a public key signature are the keys generation (keygen) through a security parameter 1^{λ}, the actual signing (sign) and signature verification (verify).

$$keygen(1^{\lambda}) = (sk, pk)$$

$$sign(sk, msg) = sig$$

$$verify(msg, sig, pk) = 0/1$$

This scheme is used to sign and verify Yobicash wire messages and as a base for transaction weighted multisignatures.

Weighted multisignatures allow more users to sign a single message, and in the Yobicash this is used to sign transactions. In this scheme, when the sum of the weights of the actual signers is greater than a user-defined threshold, the signed message is considered valid.

The scheme used by Yobicash is ed25519 [21], which is based on Schnorr signatures and the curve25519. This scheme is more efficient and secure than secp256k1 [22], which is used by Bitcoin and Ethereum, and produces smaller signatures. The Bitcoin community is discussing the opportunity to move to Schnorr signatures [23]. By large is the signature scheme more used in security critical applications [24].

3.2.3. Symmetric Encryption

A symmetric encryption scheme is a cryptographic scheme for encrypting binary data. The scheme generates a secret key which is used to encrypt and decrypt data. The schemes has 3 phases: generation of a secret key sk (keygen) through a security parameter 1^{λ}, encryption (encrypt) of a plaintext message msg and decryption (decrypt) of the resulting cyphertext cyph.

$$keygen(1^{\lambda}) = sk$$

$$encrypt(msg, sk) = cyph$$

$$decrypt(cyph, sk) = msg$$

Yobicash uses XSalsa20 as symmetric encryption scheme.

3.2.4. Public Key Encryption

A public key encryption scheme is a cryptographic scheme for encrypting binary data. The schemes generates a pair of keys, a secret key and a public key, used to encrypt and decrypt data. Data is encrypted by the encryptor by using the encryptor secret key and the decryptor public key, while it is decrypted by the decryptor by using the decryptor secret key and the encryptor public key. In this way communication confidentiality is ensured. In some cases, a public label may be used by both the parties to authenticate the data.

From a single user point of view the scheme is composed by four phases: key generation (keygen) through a security parameter 1^{λ}, encryption (encrypt) of the plaintext message msg and decryption (decrypt) of the resulted cyphertext cyph.

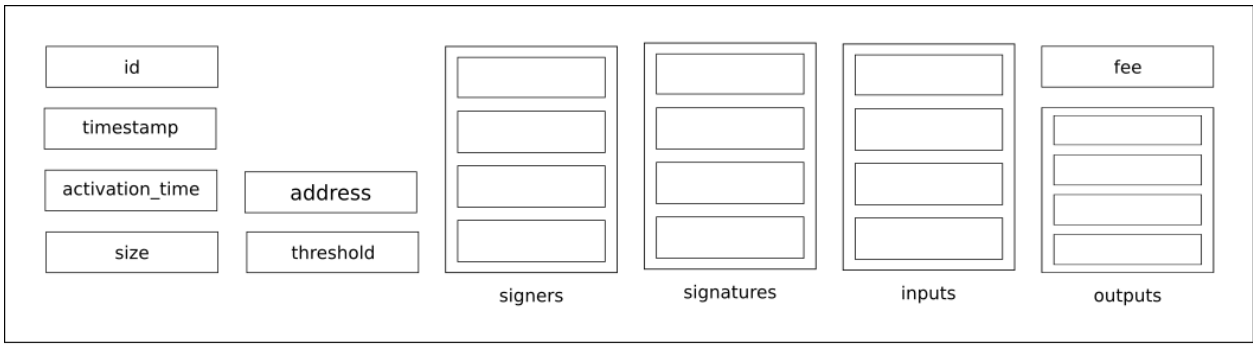
$$keygen(1^{\lambda}) = (sk, pk)$$

$$encrypt(msg, [label,] sk, pk') = cyph$$

$$decrypt(cyph, [label,] sk', pk) = msg$$

Yobicash uses curve25519XSalsaPoly1305 [25] as public key encryption scheme.

3.3. Transactions



transaction

Transactions are the cornerstone of any blockchain protocol. Through transactions value and data are signed, spent and transferred. We review the main fields of a transaction:

id: an id is a unique reference to the transaction. The id is a checksum of all the data contained in the transaction. It is computed as a **SHA512(timestamp || address || signature || threshold || cosigners || cosignatures || inputs || fee | outputs)**.

timestamp: a timestamp is a tuple of integers representing the seconds and nanoseconds passed from the unix epoch time 1/1/1970 0:00:00 UTC to the transaction creation.

activation_time: an activation_time is a timestamp indicating the time when the transaction can be included in the blockchain. Transactions with differing timestamp and activation_time have to be valid both at their arrival and at their activation.

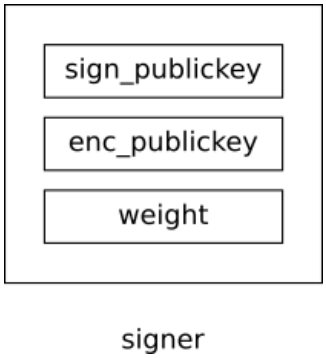
size: the size of the transaction. It has to be lower than the current size limit for Yobicash wire messages.

address: the address of the group of signers. It is referenced by transactions transferring value and data to the group. It is the result of **RIPEMD160(SHA512(signer || (SHA512(signer$_{-1}$ || SHA512(signer$_{-2}$ || SHA512(...))))** where n is the number of signers.

enc_publickey: the curve25519XSalsaPoly1305 publickey used by the group.

threshold: the threshold is an integer representing the cumulative weight required to consider the transaction signed.

signers: the signers signing the transactions. A signer is composed by the signer ed25519 public key (**sign_publickey**), her curve25519XSalsaPoly1305 public key (**enc_publickey**), and her weight in the signing process (**weight**). The signers field is a radix tree whit keys **(sign_publickey || enc_publickey)** and value **weight**.



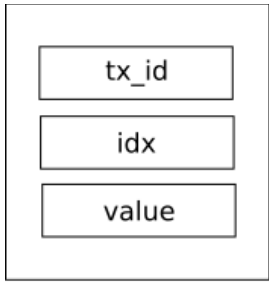
signer

signatures: the signatures of the signers which actually signed the transaction, organized in a radix tree. Signing is a voting process where every signature is a weighted vote in support of the transaction. A transaction is fully signed when the sum of the weights of the signers who signed it is greater than the transaction threshold.

inputs: the inputs of the transaction. Inputs are references to value and data sent to the address of the group. That value and data will be spent in this transaction and cannot have been already spent nor can be spent in the future. The value and data references has to already exist in the blockchain. Referenced data will be deleted by the blockchain after an epoch (100 blocks).

Referenced value and data is summed as Yobicash value following the conversion rate of bytes in Yobicash. The conversion rate is defined by the fees market, which sets the Yobicash value of a byte.

An input is composed by the id of a transaction in the blockchain (**tx_id**), the index of the spent output in that transaction (**idx**), the sum of the value and data to spend (**value**).



input

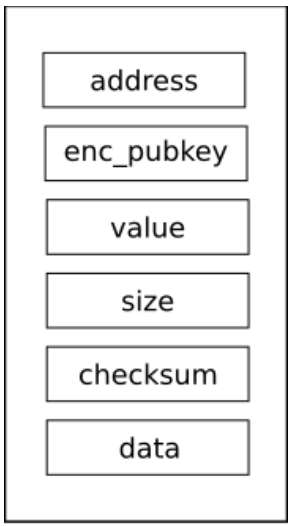
outputs: the outputs of the transaction. Outputs are transfers and conversions of value and data from the inputs.

Transferred data is authenticated by a checksum and encrypted with the curve25519XSalsaPoly1305, using the checksum as label.

Data is converted from value according to the conversion rate set by the last branch used in the previous epoch.

The sum of the outputted value and data cannot be greater than the sum of the inputed value and data.

An output is composed by the address of the recipient group (**address**) and its curve25519XSalsaPoly1305 public key (**enc_pubkey**), by the outputted value (**value**), the size of the encrypted data (**size**), the checksum of the decrypted data (**checksum**) and the curve25519XSalsaPoly1305 encrypted data (**data**). The nonce used to encrypt data is the first 24 bytes of the checksum.

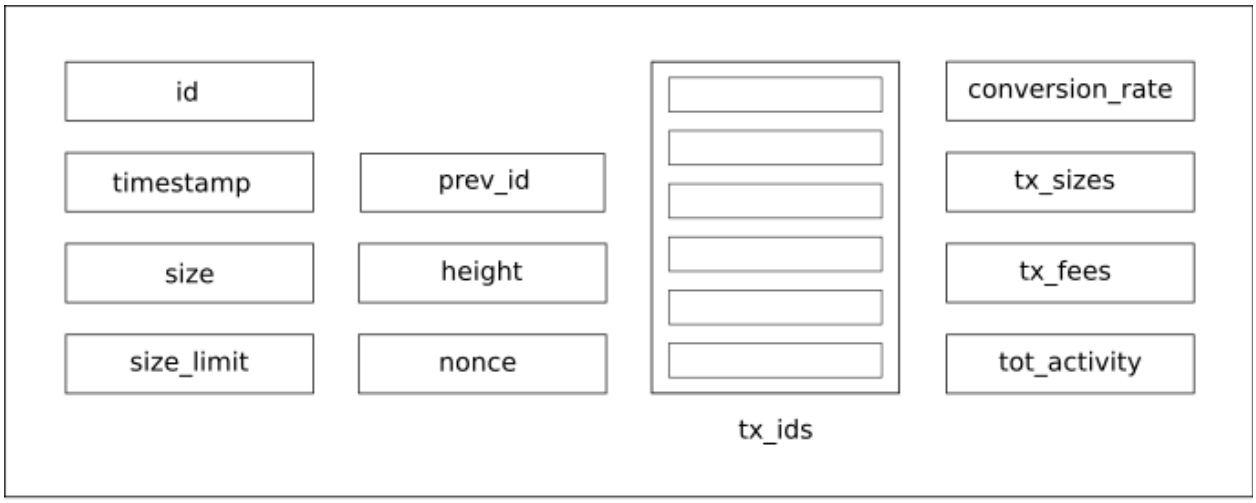


output

fee: the fee is used by the user to pay miners for their services and incentivize them to include the transaction in the next blocks. Generally fees are paid per transaction bytes. Yobicash by default set dynamic fees, but the user can overwrite this behavior. Dynamic fees are computed as the average fee per byte in the last two blocks.

The fee signals to the system the price in Yobicash of a byte of data. This is used by blocks to set the conversion rate.

3.4. Blocks



block

id: an id is a unique reference to the block. The id is a checksum of all the data contained in the block. Similarly to transactions, it is computed as a **SHA512(timestamp | size || size_limit || signer || signature || prev_id || height || nonce || tx_ids || tx_fees || tx_sizes)**.

timestamp: a timestamp is a tuple of integers representing the seconds and nanoseconds passed from the unix epoch time 1/1/1970 0:00:00 UTC to the block creation.

size: the size of the block. It has to be lower than the current size limit for Yobicash wire messages.

size_limit: the size_limit is the maximum allowed size for Yobicash wire messages (blocks and transactions). It is computed per block as:

```
size_limit = ceil(max(prev_block.size_limit,
                      includable_txs_size*1.05))
```

Where prev_block is the previous block and includable_txs_size is the average size of the valid active transactions which has not been included yet in the blockchain.

prev_id: the prev_id is the id of the previous block in the branch.

height: height is the height of the block in the branch. The genesys block – the first block of the blockchain – has height 0, and from it on all the heights increase by 1.

nonce: the nonce is the solution of the mining algorithm on the message **SHA512(prev_id || height || SHA512(tx_id_n || SHA512(tx_id_{n-1} || SHA512(tx_id_{n-2} || SHA512(...)))))** where n is the number of transaction ids in tx_ids.

tx_ids: tx_ids are the ids of the transactions registered by the block, organized in a radix tree. This transactions have to be unique and new in the branch and their size has to be lower or equal to the block size limit. The first id is of the coinbase transaction, the transaction containing the reward for the miner – the node which built the block.

tx_sizes: tx_sizes is the sum of all the sizes of the transactions referenced by tx_ids. It is used by the dynamic fees algorithm and by the dynamic size limits algorithm.

tx_fees: tx_fees is the sum of all the fees of the transactions referenced by tx_ids. It is used by the dynamic fees algorithm and by the block selection algorithm.

tot_activity: the activity is the sum of all the previous blocks in the block branch. The activity of a block is calculated as **`tx_sizes*count(tx_ids)`**.

conversion_rate: the conversion rate define the value of a byte in Yobicash. It is set as the average **`tx_fees/tx_sizes`** in the last epoch (100 blocks).

3.4.3 Block selection

Yobicash block selection gives precedence to the branches with the higher number of transactions, and so the most used by the users. This selection maximizes the activity registered on the blockchain to keep the highest average throughput across blocks and by consequence to guarantee scalability.

4. Future work

Yobicash is designed to be simple and to be used as a platform for more complex services. Security and scalability will remain the main areas of improvement to assure its value proposition.

4.1. More scalability

The current scalability measures may do a lot for some time, but throughput is never too high. The more can be made to raise throughput maintaining the trustless and Bitcoin-like structure, the better the users will be served. A lot could be done just through network simulations, fastening development. Increasing scalability and sustainability will open to new applications at the benefit of users and the network.

4.2. Quantum Cryptography

Quantum computers threaten all the value stored by current cryptocurrencies. From public key signatures to currently used zero knowledge proofs to public key encryption, everything used so far except for Keccak hashes (sha3 hashes) will be easily broken [26].

Current research on post-quantum cryptography is still immature. There have been attempts to suggest already usable algorithms [27], but a little bit of research will make clear that many of the reported algorithms are far from being safe.

Nonetheless — when needed — will be important to switch to post-quantum cryptographic algorithms. So far supersingular isogeny elliptic curve cryptography and lattice based cryptography seem the more promising solutions.

5. Conclusions

Yobicash is currently the first attempt to make data a first class citizen in blockchains and to embody the idea of the blockchain as a general purpose technology. This is achieved by making data value, by ensuring its confidentiality, by letting users co-own it and by taking care of potential scalability issues.

This kind of blockchain – a blockchain for data – enable a number of applications that ranges from simple notaries to complex multi-party computations.

More will have to be done to track and increase scalability and to assure viable alternatives to current cryptography when quantum computers will be available. But so far that post-quantum cryptography is still immature.

References

[1] M. S. Miller, K. E. Drexler, *The Ecology of Computation*, 1988

[2] S. Namamoto, *Bitcoin: A peer-to-peer electronic cash system*, https://www.bitcoin.org/bitcoin.pdf, 2008

[3] C. Catalini, J. S. Gans, *Some Simple Economics of the Blockchain*, 2016

[4] P. Snow, B. Deery, J. Lu, D. Johnston, P. Kirby, *Factom: Business Processes Secured by Immutable Audit Trails on the Blockchain*, 2014

[5] https://github.com/ethereum/wiki/wiki/White-Paper

[6] N. Atzei, M. Bartoletti, and T. Cimoli, *A survey of attacks on Ethereum smart contracts*, 2016

[7] http://blog.velocity.technology/velocity-is-shutting-downs

[8] G. Greenspan, *Why many smart contract use cases are simply impossible*, http://www.coindesk.com/three-smart-contract-misconceptions, 2016

[9] S. D. Lerner, *Strict Memory Hard Hashing Functions*, http://www.hashcash.org/papers/memohash.pdf, 2014

[10] https://usa.visa.com/dam/VCOM/download/corporate/media/visa-fact-sheet-Jun2015.pdf

[11] https://en.bitcoin.it/wiki/Scalability

[12] https://bitcointalk.org/index.php?topic=1347.msg15139#msg15139

[13] Y. Lewenberg, Y. Sompolinsky, A. Zohar, *Inclusive Blockchain Protocols*, 2015

[14] ghost paper

[15] I. Eyal, A. E. Gencer, E. G. Sirer, R. van Renesse, *Bitcoin-NG: A Scalable Blockchain Protocol*, 2015

[16] Y. Sompolinsky, A. Zohar, *Accelerating Bitcoin's transaction processing: fast money grows on trees, not chains*, 2015

[17] https://litecoin.org

[18] http://bitcoinfibre.org/public-network.html, http://www.falcon-net.org

[19] https://bitcoincore.org/en/2016/06/07/compact-blocks-faq

[20] J. Poon, T. Dryja, *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*, 2016

[21] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, B. Yang, *High-speed high-security signatures*, 2011

[22] H. Mayer, *ECDSA Security in Bitcoin and Ethereum: a Research Survey*, 2016

[23] W. Pieter, *Tree Signatures*, https://blockstream.com/2015/08/24/treesignatures

[24] https://ianix.com/pub/ed25519-deployment.html

[25] D. J. Bernstein, *Cryptography in NaCl*, 2009

[26] M. S. Brown, *Classical cryptosystems in a quantum setting*, 2004

[27] PQCrypto, *Post-Quantum Cryptography for Long-Term Security*, 2016