Overview of an article: *

Redactable Blockchain
— or —
Rewriting History in Bitcoin and Friends

Artem Fliunt

Supervisors:
- Michal Zajac
- Vitaly Skachek

University of Tartu
fliunt@tudeng.ut.ee

June 8, 2018

Abstract

In this overview of [AMVA16] paper we aim to provide a condensed summary of its contribution as well as elaborate on certain theoretical and practical aspects of it. Basic knowledge of hash functions, public key encryption schemes and blockchain is largely assumed but a brief introduction is given nevertheless. In the very beginning a motivation for having a redactable blockchain is given. Later a central notion of chameleon hash functions is explained as well as its suggested application in modifications of existing blockchain networks like Bitcoin [Nak09]. Finally we briefly cover practical implications of the solution and its conceptual controversy.

1 Introduction

Blockchain technology introduced by Satoshi Nakamoto in 2009 [Nak09] as a building block if Bitcoin cryptocurrency is a rapidly developed topic in several fields of computer science. In essence a blockchain is a decentralized distributed append-only database of records which was originally intended to be used for storing financial transactions.

To prevent double spending problem (the same coin being spent several times) that used to be a main issue with electronic currencies such a database *stores all records* and each consecutive chunk (block) depends on a previous one. This is covered by a notion of *append-only* system which effectively disallows to alter already committed records in any possible way. Blockchain is fully decentralized and distributed and provides strong guarantees that

*This overview was composed as a part of the requirements for a course *Research Seminar in Cryptography (MTAT.07.022)* held in University of Tartu during Sprint 2018.

no fraudulent transactions are stored.

Through an extensive research and development held by the community blockchain found its applications in non-cryptocurrency settings like identity federation [sho18], cloud storage [sto18], Internet of Things [hel18], hardened handheld devices [fin18] etc.

Wide adoption of a technology showed that append-only nature of a system might introduce several issues that may impact businesses involved in such networks. A need in redactable blockchain arises.

2 Motivation

Compliance with governmental regulations is an inevitable necessity for any business whether its a non-profit or a world-wide corporation. While blockchain provides necessary guarantees for some use cases it lacks any mechanism to edit or remove data from a network.

Bitcoin allows to store arbitrary data as part of a transaction via `OP_RETURN` opcode [opr18]. This mechanism was already exploited to insert improper content into a systems such as pornography [HC18], illegal content like child pornography [Hop18], malicious software [Pea18] etc.

Apart from compliance and ethical issues immutability of already inserted records of blockchain brings there are also business-specific needs. As any IT system may contain bugs there should be a way to alter records in blockchain. They may be a result of a software malfunction, human error or outdated information. Without these capabilities the technology struggles to provide necessary features.

3 Contribution

Main contribution of a reviewed paper is a proposed approach to make blockchain system redactable. This includes re-writing one or more blocks, compression of any number of blocks into smaller number of blocks or complete removal of certain blocks.

In addition authors provide a construction for a private-coin chameleon hash function that satisfies enhanced collision resistance definition in a standard model.

4 Prerequisites

4.1 Hash Functions

Hash function is a mapping of data of any size into a data of fixed size. In this overview we are primary interested in *cryptographic* hash functions which have to satisfy the properties below (formulated informally).

Definition 4.1 (**Pre-image resistance**)**.** A hash function H is pre-image resistant if for all messages $m \in M$ (where M is a message space) such that $h = H(m)$ it is hard to find m given only h and H.

Definition 4.2 (**Second pre-image resistance**)**.** A hash function H is second pre-image resistant if for all messages $m \in M$ (where M is a message space) given a message m_1 it is hard to find such a message m_2 that $H(m_1) = H(m_2)$ and $m_1 \neq m_2$.

Definition 4.3 (**Collision resistance**)**.** A hash function H is collision resistant if it is hard to find such two different messages $m_1, m_2 \in M$ (where M is a message space) such that $H(m_1) = H(m_2)$.

4.2 Chameleon Hash Functions

Chameleon hash functions is similar to a *cryptographic* hash functions and have the same properties. Yet they have an important distinction: trapdoor key. Without this key it is hard to compute collisions (break collision resistance property) but with a key collisions can be generated efficiently.

Originally chameleon hashing was introduced by Krawczyk and Rabin [KR00] on a top of a notion of chameleon commitments [BCC88]. A generalized notion of chameleon hash functions now is referred to as *secret-coin chameleon hashing*.

Definition 4.4 (**Secret-coin chameleon hash**)**.** A *secret-coin* chameleon hash function is a tuple of efficient algorithms $\mathcal{CH} = (\text{HGen}, \text{Hash}, \text{HVer}, \text{HCol})$ specified as follows.

- $(hk, tk) \xleftarrow{\$} \text{HGen}(1^k)$: The probabilistic key generation algorithm HGen takes as input the security parameter $k \in \mathbb{N}$, and outputs a public hash key hk and a secret trapdoor key tk.
- $(h, \xi) \xleftarrow{\$} \text{Hash}(hk, m)$: The probabilistic hashing algorithm Hash takes as input the hash key hk, a message $m \in M$, and implicit random coins $r \in \mathcal{R}_{hash}$, and outputs a pair (h, ξ) that consists of the hash value h and a check string ξ.
- $d = \text{HVer}(hk, m, (h, \xi))$: The deterministic verification algorithm HVer takes as input a message $m \in M$, a candidate hash value h, and a check string ξ, and returns a bit d that equals 1 if (h, ξ) is a valid hash/check pair for the message m (otherwise d equals 0).
- $\xi' \xleftarrow{\$} \text{HCol}(tk, (h, m, \xi), m')$: The probabilistic collision finding algorithm HCol takes as input the trapdoor key tk, a valid tuple (h, m, ξ), and a new message $m' \in M$, and returns a new check string ξ' such that $\text{HVer}(hk, m', (h, \xi')) = 1$. If (h, ξ) is not a valid hash/check pair for message m then the algorithm returns $\perp$.

Originally introduced chameleon hashing is now referred to as *public-coin chameleon hashing* and is a special case of *secret-coin* where check string returned from Hash is equal to an implicit randomness used by this algorithm.

Additional notion that is important for usage of chameleon hashing in blockchain is *enhanced collision resistance* (informal definition):

Definition 4.5 (**Enhanced collision resistance**)**.** A hash function H is enhanced collision resistant if it is hard to find such two messages $m_1, m_2 \in M$ (where M is a message space) such that $H(m_1) = H(m_2)$ even after seeing polynomially many pairs $m_i, m_j \in M$ s.t. $H(m_i) = H(m_j)$.

4.3 Blockchain

As a name implies, blockchain is a distributed network of parties each of which store an [almost] identical chain of blocks. The below figure illustrates a structure of a typical blockchain (Bitcoin):

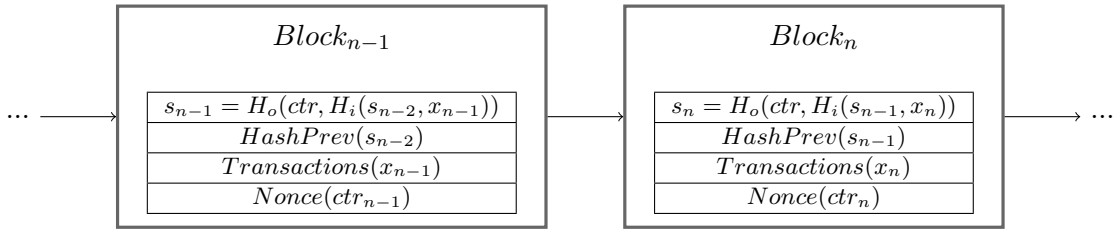

Figure 1: Bitcoin block structure

Each block i consists of the following data:

- Both H_i and H_o are cryptographically secure hash functions (usually the same function is used)

s_{i-1} – outer hash of a previous block

x_i – set of transactions included in this block. For Bitcoin a size of a block is limited to 10mb which places an upper bound on an amount of transactions each block can include.

ctr_i – nonce or counter. A value that is calculated as part of a Bitcon proof of work subroutine.

A proof of work subroutine is what makes production of a block computationally expensive and prevents fraudulent blocks from flooding the system. If majority of participants in a system are honest an adversary won't be able to produce POWs quickly enough to extend a blockchain into a fraudulent fork. Block is valid if and only if

$$validblock_q^D(B) := (H_i(ctr, H_o(s, x)) < D) \wedge (ctr < q) = 1$$

Here $D, q \in \mathbb{N}$ are block's difficulty level and a maximum amount of hash queries a user is allowed to make to attempt to find a right nonce for this block.

The algorithm used is Hashcash [Bac02] and informally it works as follows: find a hash $H_o(ctr, H_i(s_{i-1}, x_i))$ such that it would have at least l zeroes in the beginning of a binary representation of it (which is equivalent to finding a hash smaller then a certain number D).

5 Construction

This overview will cover only centralized setting of a redactable blockchain construction. While it is controversial in terms of giving a power to mutate blockchain to a single entity (effectively making the whole system partially centralized) we argue that even decentralized setting presented in a paper doesn't solve centralization problem for existing blockchains with a relatively large amount of parties in a network.

The main idea of a paper is to *set the inner hash H_i to be an enhanced collision resistant chameleon hash function*. The enhanced property here plays a central role in making the resulting system secure: we want an adversary to be unable to produce new collisions on its own even seeing the previous one broadcasted in a network.

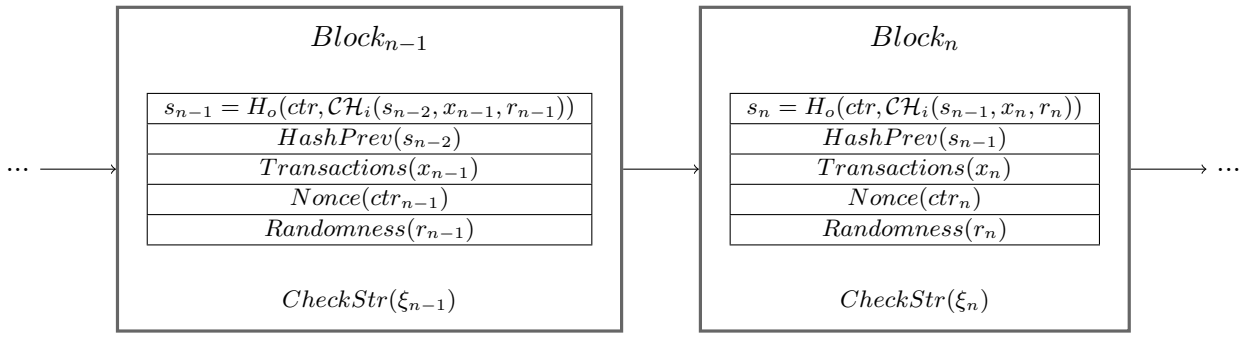

Figure 2: Redactable blockchain structure

Note that check string of each block is only formally a part of this block. It does not have to be included in a calculation of the block digest and is stored only for other participants to be able to verify block's validness.

Now it is possible to define two algorithms with former being the one that redacts any given block and latter the one that removes blocks completely (both require a correct trapdoor key to be passed in).

Algorithm 1: Chain Redact

input: The input chain $\mathcal{C}$ of length n, a set of block indices $\mathcal{I} \subseteq |n|$, a set of values $\{x'_i\}_{i \in \mathcal{I}}$, and the chameleon hash trapdoor key tk.

output: The redacted chain $\mathcal{C}'$ of length n.

$\mathcal{C}' \leftarrow \mathcal{C}$;

Parse the chain $\mathcal{C}'$ as $(B_1, \dots, B_n)$;

for $i := 1, \dots, n$ **do**

  if $i \in \mathcal{I}$ **then**

    Parse the i-th block of $\mathcal{C}'$ as $B_i := \langle s_i, x_i, ctr_i, (h_i, \xi_i) \rangle$

    $\xi'_i \leftarrow \mathrm{HCol}(tk, (h_i, s_i || x_i, \xi_i), (s_i || x'_i))$;

    $B'_i := \langle s_i, x'_i, ctr_i, (h_i, \xi'_i) \rangle$;

    $\mathcal{C}' \leftarrow \mathcal{C}'^{\lceil n-i+1 \rceil} || B'_i || ^{i \rceil} \mathcal{C}'$;

  endif

endfor

return $\mathcal{C}'$

Intuituion behind the above algorithm is quite simple: we take a set of indices of blocks to change and a set of corresponding new data to insert. Then for each block we find a collision if a correct trapdoor key was given. Then we insert new blocks into a chain. *Note* that

new blocks have the same outer hashes (digests) as previous ones despite having possibly different content. This allows to preserve chain integrity while redacting data inside some blocks.

Blocks can be removed (and as a result blockchain will be shrinked) using the following algorithm:

Algorithm 2: Chain Shrink

input: The input chain $\mathcal{C}$ of length n, a set of block indices $\mathcal{I} \subseteq |n|$, and the chameleon hash trapdoor key tk.

output: The new chain chain $\mathcal{C}'$ of length $n - |\mathcal{I}|$.

$\mathcal{C}' \leftarrow \mathcal{C}$;

Parse the chain $\mathcal{C}'$ as $(B_1, \dots, B_n)$;

for $i := 1, \dots, n$ **do**

  if $i \in \mathcal{I}$ **then**

    Parse the i-th block of $\mathcal{C}'$ as $B_i := \langle s_i, x_i, ctr_i, (h_i, \xi_i)\rangle$

    Parse the $i+1$-th block of $\mathcal{C}'$ as $B_{i+1} := \langle s_{i+1}, x_{i+1}, ctr_{i+1}, (h_{i+1}, \xi_{i+1})\rangle$

    $\xi'_{i+1} \leftarrow \mathrm{HCol}(tk, (h_{i+1}, s_{i+1}||x_{i+1}, \xi_{i+1}), (s_i||x_{i+1}))$;

    $B'_{i+1} := \langle s_i, x_{i+1}, ctr_{i+1}, (h_{i+1}, \xi'_{i+1})\rangle$;

    $\mathcal{C}' \leftarrow \mathcal{C}'^{\lceil n-i}||B'_{i+1}||^{i+1]}\mathcal{C}'$;

  endif

endfor

return $\mathcal{C}'$

Again, the intuition is quite simple: in order to remove block B_i we need to redact a block B_{i+1} to use not s_{i+1} but s_i (digest of the block before the one to remove). Apart from it the procedure is quite similar to the one in Algorithm 1.

6 Instantiations

While it is possible to use public-coin chameleon hash function introduced by Ateniese and de Medeiros [AdM04] which is proved to be enhanced collision resistant under generic group assumption the authors of the paper introduced a more robust secret-coin chameleon hashing that has the same property but in a standard model. Lesser assumptions of a latter model make the whole reductable blockchain construction presented in a paper more sound.

Details of a construction won't be covered in this overview due to their theoretical complexity and a necessity to cover much more prerequisites. The main idea is to encrypt randomness used in a chameleon hash function while providing a proof in zero knowledge that calculated values are correct.

A new construction introduces a one more infrastructure burden: non-interactive zero knowledge scheme used required a common reference string to be generated once for a lifetime of a blockchain by a central party (the one that is in a possession of a trapdoor key).

7 Practical Aspects

A construction presented in a paper uses a variety of available cryptographic primitives to solve the originally stated problem. Still there are some issues with this construction that make it harder to use in practice:

- Each block now is logically composed from two parts: the block and its check string (ξ produced by a chameleon function Hash algorithm). An implementation have to guarantee that check strings are stored intact with blocks otherwise participants won't be able to verify already existing blocks.

- A controversy of a "central" party that is in a possession of a trapdoor key violates a seemingly decentralized by desing blockchain system. A proposed in a paper "decentralized" setting with a trapdoor key secret-shared among several parties slightly reduces a severity of this issues but still makes it impossible to efficiently share a trapdoor key among all the participants of a system.

- There is a need in a secure and authenticated way of distribution of a blockchain mutations across the network. A naive approach of broadcasting changes as is to all nodes might be exploited by an adversary to influence a network consensus (reduce chain consistency).

8 Appropriate Use Case

We propose an example use case which we find a fitting scenario for a redactable blockchain construction introduced in a paper. We argue that in this use case the benefits gained by using a construction from a paper outweights its issues (listed above).

- A certain relatively small number of banks (say, 10) are willing to simplify an interbank transactions for their customers

- They decide to maintain a private redactable blockchain

- A trapdoor key is secret-shared among all participants and all mutations broadcasted are signed by a private key of a respective bank

- Authenticity of all signing keys of each bank are established off-chain by any means that participants find secure

Such a system would allow to make interbank transfers much quicker. Typical SWIFT transaction takes about 1-3 working days while Bitcoin payment is considered to be safe after 5-6 blocks (which is an hour of time in total in 2018).

Due to a limited amount of participants secret-sharing of trapdoor key is relatively efficient and mutations can be authenticated.

9 Summary

The paper covered in this overview introduced an innovative way of making a blockchain redactable. While the idea itself is controversial to some extend the authors provided an excellent motivation proving a need in such a solution.

While the construction has its own issues there definitely exist use cases where it can be used to a benefit of all the participants.

References

[AdM04] Giuseppe Ateniese and Breno de Medeiros. On the key exposure problem in chameleon hashes. Cryptology ePrint Archive, Report 2004/243, 2004. https://eprint.iacr.org/2004/243. 6

[AMVA16] Giuseppe Ateniese, Bernardo Magri, Daniele Venturi, and Ewerton Andrade. Redactable blockchain – or – rewriting history in bitcoin and friends. Cryptology ePrint Archive, Report 2016/757, 2016. https://eprint.iacr.org/2016/757. 1

[Bac02] Adam Back. Hashcash - a denial of service counter-measure. 09 2002. 4

[BCC88] Gilles Brassard, David Chaum, and Claude Crpeau. Minimum disclosure proofs of knowledge. 37:156–189, 10 1988. 3

[fin18] Finney by Sirin Labs – Blockchain powered smartphone. https://sirinlabs.com, 2018. [Online; accessed 11-May-2018]. 2

[HC18] Steve Hargreaves and Stacy Cowley. How porn links and ben barnanke snuck into bitcoin's code. http://money.cnn.com/2013/05/02/technology/security/bitcoin-porn/index.html, 2018. [Online; accessed 11-May-2018]. 2

[hel18] Helium – Blockchain-based IoT network. https://helium.com, 2018. [Online; accessed 11-May-2018]. 2

[Hop18] Curt Hopkins. If you own Bitcoin, you also own links to child porn. https://www.dailydot.com/business/bitcoin-child-porn-transaction-code, 2018. [Online; accessed 11-May-2018]. 2

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures., 01 2000. 3

[Nak09] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 03 2009. 1

[opr18] OP_RETURN Bitcoin opcode. https://en.bitcoin.it/wiki/OP_RETURN, 2018. [Online; accessed 11-May-2018]. 2

[Pea18] Jordan Pearson. The bitcoin blockchain could be used to spread malware, interpol says. https://motherboard.vice.com/en_us/article/ezv8jn/the-bitcoin-blockchain-could-be-used-to-spread-malware-interpol-says, 2018. [Online; accessed 11-May-2018]. 2

[sho18] ShoCard – Blockchain-based identity federation. https://shocard.com, 2018. [Online; accessed 11-May-2018]. 2

[sto18] Storj – Blockchain-based cloud storage. https://storj.io, 2018. [Online; accessed 11-May-2018]. 2