REDACTABLE BLOCKCHAIN – LEVERAGING CHAMELEON HASH FUNCTIONS FOR A GDPR COMPLIANT BLOCKCHAIN

Hauke Precht, Jorge Marx Gómez
Carl von Ossietzky Universität Oldenburg, Ammerländer Heerstraße 114-118, 26129 Oldenburg

With the increasing usage of blockchain technology, legal challenges such as GDPR compliance arise. Especially the right of erasure is considered challenging as blockchains are tamperproof by design. Several approaches investigated possibilities to weaken the tamperproof aspect of blockchains in favor of GDPR compliance. This paper presents several approaches, then focuses on chameleon hash functions by evaluating the possibility to use these specific functions in a private blockchain. The goal of the built system is to take a step towards the digitization of the bill of lading used in international trade. This paper describes the developed software as well as the core considerations around the system such as network design or block structure.

1. Introduction

As the blockchain technology is used in more and more domains, also the privacy aspect of the saved data on the blockchain draws attention. As one key feature of blockchain is the immutability of data, it clashes with General Data Protection Regulation (GDPR) requirements, i.e. the right to erasure. This must be granted if personal data is processed. As this right cannot be enforced on a blockchain, since the data is stored in immutable and linked blocks, it is currently not possible, from a legal point of view, to use blockchain technology in conjunction with personal data.

As it is not always clear if data is classified as personal data, often only hash values of actual data are stored on the blockchain. The drawback of this is that data still needs to be distributed in an old-fashioned way leading to (manual) processes of data handling around the blockchain.

The research project HAPTIK (haptik.io) makes an attempt to digitize the bill of lading via blockchain technology, which was already proposed in [1]. As the goal is to digitize the whole process around the handling of the bill of lading, an investigation was carried out whether personal data is part of a bill of lading. This led to a positive result, meaning that the GDPR must be applied [2]. Based on this result, several approaches as proof of concept were carried out, of which one is described in this paper in detail: leveraging chameleon hash functions to create a redactable blockchain, allowing the users to modify already accepted and verified blocks.

The design and prototype were developed with four students in a project group for over one year. This is part of every master students' curriculum to let them learn to work in a team and to contribute to ongoing research topics.

The following paper is structured as follows: First, an overview of related work is given, in which already existing approaches, towards a redactable blockchain, are presented, along with a short introduction of chameleon hash functions. Next, the implementation of the prototype is presented, where key design decisions are shown along with the key functions of the newly created blockchain. This paper concludes with a summary and outlook identifying further research aspects

2. Related Work

In 2017, Ateniese et. al proposed a framework enabling a redactable blockchain by using chameleon hash functions [3]. They describe a theoretical framework and a proof of concept implementation for the Bitcoin network [3]. The used Chameleon Hash functions were introduced in 2000 by Krawczyk and Tal [4]. These functions also generate a "trapdoor" key, which can be used to efficiently generate hash collisions. This is also used by Ateniese et. al with some enhancements [3]. The idea is to use this trapdoor key to calculate a collision. When redacting a block, it gets rehashed by using the trapdoor key which enables the function to calculate a collision. That way a block can be changed while maintaining its original hash. In comparison, classical hash functions aim to be collision-resistant [5]. As Ateniese et. al focuses mainly on the applicability of chameleon hash functions in bitcoin, while discussing private/consortium blockchains only shortly, we, in this paper, aim towards an evaluation of applicability in private/consortium blockchains. Especially the management of the trapdoor key is crucial as it is used to modify the block, meaning profound governance of this trapdoor key is required [3]. Next to this generally more abstract framework, Hylock and Zeng presented an application based on this framework in the specific domain of patient-centered health records [5] dealing with highly personal data. Next to using non-traditional hash functions, such as chameleon hash functions, a different approach towards a redactable blockchain is presented by Marsalek and Zefferer. These authors propose an architecture in which a second blockchain is used to track corrected blocks, named correction chain [6]. While these approaches act on a global scale, Florian et. al propose an erasing mechanism on the local scale, i.e. on node level by presenting the functionality-preserving local erasure [7]. They provided a prototype based on bitcoin as well, arguing that their solution is less invasive than, for example, the solution proposed by Ateniese et. al [7]. Another different approach is presented in [8] where the

authors propose a system leveraging multiple transaction versions representing different possible states while also encrypting non-active transactions. When a change is desired, a consensus about the new active transaction is triggered [8]. The idea is to plan possible (desired) ledger states, including no-operation, beforehand. All transaction versions are encrypted and only the key for the currently active version is distributed [8]. This way, no “real” deletion or changing of data is possible, while this approach also targets PoW-Blockchains, i.e. dealing with currency and coin amounts in wallets. Farshid et. al also proposed a “forgetting blockchain”, where they designed and implemented a pruning algorithm for the Ethereum blockchain, which deletes as much data as possible while maintaining consistency [9]. The drawback of this method is, that it is not possible to delete specific data but rather rely on the pruning algorithm to take care of it eventually.

This recap of existing approaches considering the ability to redact in blockchain shows, that this is an emerging and ongoing research topic. As it is a considerably new approach, only one paper was identified, applying a redactable blockchain to an actual real-world use case. Further, the proposed solution focuses on public blockchain such as bitcoin.

In this paper, the authors investigate, if a redactable blockchain, based on chameleon hash functions, is suitable for private/consortium blockchains for possible digitization of a bill of lading.

3. Implementation of a Chameleon Hash Function Based Blockchain

The general applicability of chameleon hash functions in private/consortium blockchains is already, shortly, discussed by Ateniese et. al, considering the management of the trapdoor key [3]. As mentioned, the usage of a chameleon hash function is invasive [7] in regards to the used blockchain, meaning, when using an existing blockchain system, the existing hash function of this blockchain must be replaced. As this requires extensive changes in the underlying code and therefore profound knowledge of this code, it can be considered challenging. The group of students first analyzed such exchange of hash functions in the consortium blockchain Corda, but after one month without significant progress decided to refrain from trying to adjust the codebase. The main reason was insufficient code documentation and time constraints. Therefore, it was decided to create a own implementation of a (simplified) blockchain system, incorporating chameleon hash functions from the beginning and building the system tailored to alternating block in the easiest way towards the use case example of a digital bill of lading.

As the most knowledge regarding programming languages were at C#, this language was chosen for the implementation. To provide the user with a user interface, the React framework was used to build a web app. The actual data is stored in a LiteDB, which is an integrated database. This way, a three-layer model is built:

- First Layer: Webapp, which servers the user for input of data
- Second Layer: Service, which processes data from user input and serves data from the blockchain to the user
- Third Layer: Blockchain, providing the necessary hash functions, assets and verifying of new blocks (consensus).

The following paper will mainly focus on the blockchain layer by shortly describing the implementation of the chameleon hash function followed by the network design, consensus algorithm as well as the defined block structure. Note, that the focus lies solely on the chameleon hash functions and its possible applicability. Therefore, no mechanism for smart contracts, living on the blockchain, was implemented yet, to further minimize complexity in the first step.

3.1. Chameleon Hash Function

Due to the complex theoretical and mathematical nature of a chameleon hash function, no own implementation was used. Instead, an existing C library [10] was used and translated into C# due to compatibility issues. It is notable, that this was, at the time of development, the only library found, explicitly focusing on an implementation of chameleon hash functions. Note, that three different approaches towards a chameleon hash function were initially implemented in the library which was used for performance measurements, comparing block creation time and block redaction time. In their implementation, they evaluated three approaches of implementation: Simple Factorization (SF), Discrete Logarithm (DL) and Advanced Factorization (AF) [11]. The third approach, AF, showed similarities in block creation as well as in calculating a collision, which is why this algorithm was chosen in this prototype as well.

3.2. Network Design

Within the process of the bill of lading, the involved participants are generally known, especially parties like ports, customs, port agents and so on and so forth. As a private blockchain is built, each of the identified participants will be represented by an own node. Since not every party is included in every bill of lading process, they only receive the data required for processes they are part of (privacy by design). To achieve this, the system supports a set of blockchains instead of a single, large blockchain. A similar concept is known from Hyperledger Fabric where the system allows to create distinct channels, each holding an own blockchain [12]. This way, shared data among the parties are limited to a minimum, following privacy by design principles. The general idea of the network infrastructure is shown in figure 1 below.

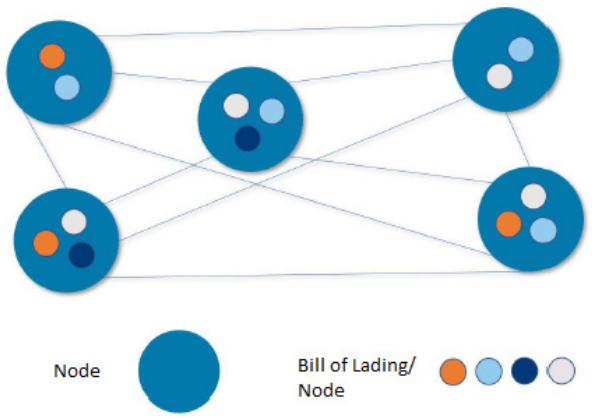

Fig. 1: Network Infrastructure

3.3. Consensus Algorithm

Since in private/consortium blockchains, the participants are generally known and access to the network is restricted, energy-consuming consensus algorithms like proof of work are not required. Instead, voting-based consensus algorithms are used [13], enabled by the identifiable parties which is a requirement, for example for the practical Byzantine Fault Tolerance (pBFT). This algorithm, which was already defined in 1999 by Castro and Liskov [14], proposes a solution to The Byzantine Generals Problem [15]. Within this prototype, a pBFT-like consensus algorithm was implemented as well requiring a 2/3 majority to finally accept actions. As the number of participants is considerably low, the possible slow performance can be neglected.

3.4. Block Structure

To this point, the general architecture of the developed blockchain is described, including used technology, network infrastructure and the used consensus algorithm. Next, the actual structure of a block is discussed.

For a starting point, the block structure of the most known blockchains, Bitcoin and Ethereum, is analyzed, starting with the Bitcoin block structure. It mainly consists of 5 objects: Magic number, block size, block header, transaction counter and a variable set of transactions [16]. These objects can be further divided into more detailed variables, for example, is the previous hash a variable in the block header [16]. A similar structure is used in the Ethereum blockchain, where the block header contains even more information in regard to Ethereum specific gas [17]. As a private blockchain is developed, also the Hyperledger Fabric block structure is taken as a reference into consideration as well. A Hyperledger fabric block consists of three sections: block header, block data and block metadata [18]. Like the block structures of bitcoin and Ethereum, the block header contains information such as the previous hash or the block number. The block data is also similar, containing transaction data. The block metadata, however, is used to store the certificate and signature of the creator of the block [18]. Block committer also adds valid/invalid indicators for each transaction into a bitmap. Note that the metadata is not taken into consideration for the block hash computation [18]. An abstraction of a block structure is given in [13] where the authors defined two major parts which every block structures have in common: the header, storing metadata such as the timestamp or hash of the previous block, and the block content, storing actual (transaction) data.

Based on the different existing block structures, a similar approach is developed in this prototype meaning that block metadata must be stored as well as the actual data. To simplify the structure in this prototype as much as possible, no distinct header object is modelled but typical header information is included in the block. As chameleon hashes are used, a block must also store a checksum as well as the public/trapdoor key pair. Note that the actual trapdoor key is only present on the node, which created the block. Therefore, also the creator of the block is modelled in the block structure for other parties to know whom to contact for a possible redact request. Furthermore, each involved node must sign the block. As a pBFT-like algorithm is implemented, a block requires 2/3 acceptance (by signing the new block) which also holds true when redacting a block. To be able to include the redacting of a block as part of the consensus, the mentioned checksum is included. In case the creator of a block changes the block without an existing request for his own good, it would be noticed by the other participants as the checksum changes. Further, the other nodes must manually accept any given redaction action, providing another layer of security preventing arbitrary changes by the creator. The newly created block structure in general block structure is shown in figure 2.

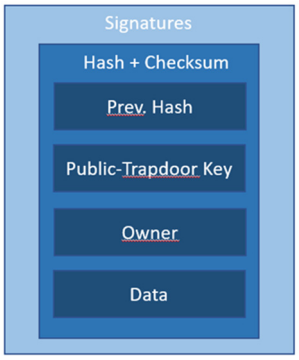

Fig. 2: Block Structure

In this given use case, the data of the block represents the parts of the bill of lading. Note that each block only contains newly added data, so the bill of lading will be constructed by reading and merging the data of each block since the calculation of a collision is time-consuming. Therefore, the goal is to minimize the number of blocks which require a change.

4. Block Creation and Redaction Walkthrough

With the described system, a private blockchain leveraging chameleon hash function is built. Next, the

process of block creation, as well as the process of redacting a block, is described, starting with the creation of a new block, which is shown in figure 3. In a first step, the initiating node (source node) sends the new block for verification to its known partners. The nodes evaluate the block and send either a verification signature or a rejection. The source node must collect 2/3 of verification signatures to consider the votes as an acceptance, then sending the newly accepted and signed block to the network. If this threshold is not reached, i.e. no majority voted for acceptance, no block will be created.

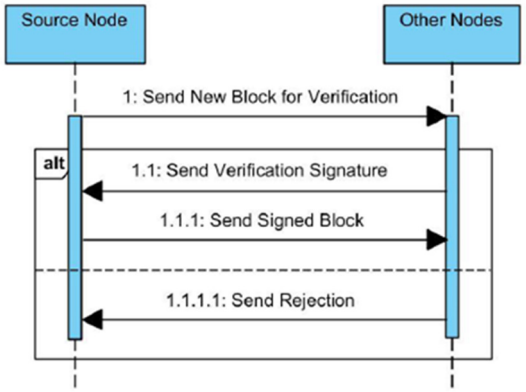

Fig. 3: Block Creation Process

Note that the accepted and signed block is sent to the other nodes, who then also verify that 2/3 majority signatures are present. This is possible as every node knows the other parties. The other nodes then add the new block to their own blockchain.

If the need to change data in an already accepted and appended block appears, the redaction function comes into play. The process is shown in figure 4 and is explained in the following. In the first step, the node who requests a change of data notifies the owner node (i.e. the node which created the respective block as this is the only one in possession of the trapdoor key) by sending a redact request. The node performs the requested change by modifying the block and rehashing the block via the chameleon hash function. The redacted block is sent to the other participants of the process for manual verification. At this point, it is up to the users to decide if a change is justified and allowed. Like the block creation process, 2/3 of the participants must accept the change by manually accepting the request via the provided web app. In case they decline the request, the changed block will be discarded, and a reject request message is sent to the owner node. The owner node verifies the signed verification/rejection signatures and in case a 2/3 majority of acceptance is reached, the newly redacted and signed block is sent to the network. In case no 2/3 majority is reached, the changes are discarded, and a rejection notification is sent to the requesting node. As this process involves the manual action of users, waiting time must be taken into consideration as well as possible timeouts. To deal with these issues, a request object is stored each time a block is sent, storing information of the state of the request. This serves as a basis to determine a possible timeout in which the nodes must answer. If this timeout is reached, the request is sent again. This is done until every node has sent either an accept or reject.

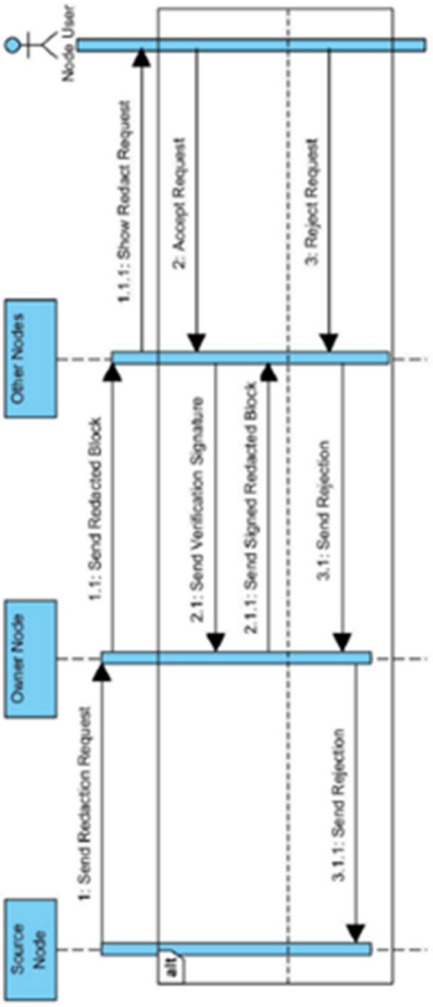

Fig. 4: Block Redaction Process

5. Conclusion and Outlook

This paper shows a newly implemented blockchain around the key feature of the redaction of existing blocks. It is shown that several approaches exist in terms of modifiability of blockchain. Some of these approaches use the chameleon hash function which is also used in this presented prototype. The prototype was developed by a group of students over the duration of one year as part of their master studies. It is shown that the block structure consists of additional fields in comparison to other block structures, e.g. a checksum which is specific for the chameleon hash function. Further, a walkthrough in terms of block creation and block redaction is given, showing how consensus is also applied when changing blocks. Note that the acceptance of change must be given by users and is not automated yet, so the last instance of decision making is a person. In the presented prototype, only the creator of the block can perform a change as only this creator is in possession of the required trapdoor key providing a very simple and basic solution of the problem of key

management. These challenges described by [3] are still valid and must be taken into consideration in further development.

So far, the application has only been tested under laboratory conditions. As a next step, this system should be evaluated by practitioners and business experts. Further, no performance tests could have been realized in the time being, so no findings in terms of performance of the chameleon hash function in this implementation can be made yet. As noted, several key aspects of blockchains were simplified in the development process, e.g. the designed block structure. Those simplified aspects must be revised to develop a more profound blockchain. This also affects possible transaction models or smart contract support. Nevertheless, this project helped students to understand blockchain technology while also enabling them to apply such technology in a real-world use case contributing to ongoing research.

Acknowledgements

The paper was written within the research project HAPTIK (www.haptik.io) at the University of Oldenburg. The research goal is the digitization of the bill of lading using blockchain technology.

References

[1] S. Wunderlich and D. Saive, “The Electronic Bill of Lading,” in Advances in Intelligent Systems and Computing, Blockchain and Applications, J. Prieto, A. K. Das, S. Ferretti, A. Pinto, and J. M. Corchado, Eds., Cham: Springer International Publishing, 2020, pp. 93–100.

[2] D. Saive and T. Janicki, “Datenschutz in elektronischen Frachtdokumenten,” RdTW - Recht der Transportwirtschaft Zeitschrift für Transportrecht und Schifffahrtsrecht mit dem Recht des Überseekaufs sowie Versicherungsrecht, Zollrecht und Außenwirtschaftsrecht, no. 6, pp. 201–207, 2019.

[3] G. Ateniese, B. Magri, D. Venturi, and E. Andrade, “Redactable Blockchain – or – Rewriting History in Bitcoin and Friends,” in 2017 IEEE European Symposium on Security and Privacy (EuroS&P), Paris, Apr. 2017 - Apr. 2017, pp. 111–126.

[4] H. Krawczyk and T. Rabin, “Chameleon Signatures,” in Proceedings of the Network and Distributed System Security Symposium, NDSS 2000, San Diego, California, USA, 2000. [Online]. Available: https://www.ndss-symposium.org/ndss2000/chameleon-signatures/

[5] R. H. Hylock and X. Zeng, "A Blockchain Framework for Patient-Centered Health Records and Exchange (HealthChain): Evaluation and Proof-of-Concept Study," Journal of medical Internet research, vol. 21, no. 8, e13592, 2019, doi: 10.2196/13592 .

[6] A. Marsalek and T. Zefferer, “A Correctable Public Blockchain,” in 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), Rotorua, New Zealand, Aug. 2019 - Aug. 2019, pp. 554–561.

[7] M. Florian, S. Henningsen, S. Beaucamp, and B. Scheuermann, “Erasing Data from Blockchain Nodes,” in 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS & PW), Stockholm, Sweden, Jun. 2019 - Jun. 2019, pp. 367–376.

[8] Ivan Puddu, Alexandra Dmitrienko, and Srdjan Capkun, “μchain: How to Forget without Hard Forks,” IACR Cryptol. ePrint Arch., vol. 2017, pp. 106–127, 2017.

[9] S. Farshid, A. Reitz, and P. Roßbach, “Design of a Forgetting Blockchain: A Possible Way to Accomplish GDPR Compatibility,” in Proceedings of the 52nd Hawaii International Conference on System Sciences, 2019.

[10] V. Patil, Chameleon hash function implementation: C Code. [Online]. Available: http://wwwusers.di.uniroma1.it/~patil/projects/cham/code.html (accessed: Aug. 19 2020).

[11] V. Patil, Chameleon Hash Function Implementations: Experimental setup and results. [Online]. Available: http://wwwusers.di.uniroma1.it/~patil/projects/cham/results.html (accessed: Aug. 19 2020).

[12] Hyperledger Fabric, Channels. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/channels.html (accessed: May 13 2020).

[13] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, “An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends,” in 2017 IEEE International Congress on Big Data - BigData Congress 2017: 25-30 June 2017, Honolulu, Hawaii, USA : proceedings, Honolulu, HI, USA, 2017, pp. 557–564. Accessed: Mar. 6 2019.

[14] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” in Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999, 1999, pp. 173–186. [Online]. Available: https://dl.acm.org/citation.cfm?id=296824

[15] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," ACM Trans. Program. Lang. Syst., vol. 4, no. 3, pp. 382–401, 1982, doi: 10.1145/357172.357176 .

[16] Bitcoin Wiki, Block. [Online]. Available: https://en.bitcoin.it/wiki/Block (accessed: Aug. 21 2020).

[17] Dr. Gavin Wood, “Ethereum Yellow Paper: a formal specification of Ethereum, a programmable blockchain,” Accessed: Mar. 6 2019.

[18] Hyperledger Fabric, Ledger (accessed: Aug. 21 2020).”