Design of a forgetting blockchain: A possible way to accomplish GDPR compatibility

Simon Farshid
Frankfurt School of Finance & Management
s.farshid@fs.de

Andreas Reitz
Frankfurt School of Finance & Management
a.reitz@fs.de

Peter Roßbach
Frankfurt School of Finance & Management
p.rossbach@fs.de

Abstract

Practitioners as well as academics expect that blockchain technology is a game changer for a variety of use cases [1], [2]*. This is due to transaction immutability enabled by keeping a history of all transactions. Nevertheless, this strength can become its biggest weakness. There already exists a lively discussion on scenarios where it is necessary to delete submitted data from the chain after it is no longer needed. This becomes even more crucial with the introduction of the European General Data Protection Regulation (GDPR). In this paper, we make use of a design science research (DSR) approach to design an IT artifact in the form of a prototype that maintains most of the key features of blockchain technology but deletes old data. We evaluate the prototype with the help from experts to investigate what to expect from blockchains that delete data and derive principles on how to design them.*

1. Introduction

When we speak of Blockchain, we generally mean the technology instead of a specific implementation. One key property of this technology is its ability to secure old data against modification. This makes blockchain an append-only structure, where new data can be added but never removed. While this is one its biggest strengths, it can also become its most crucial weakness.

For example, when information about users needs to be put on a blockchain, strict privacy laws such as European General Data Protection Regulation[1] (GDPR) and the right of European consumers to demand that their data is “forgotten” or deleted, pose a challenge for this technology.

Even when personal information is not directly put on the blockchain, historical data can be analyzed to reveal identities of pseudonyms. For example, modern analysis of Bitcoin transactions has shown that wallets can be linked together, compromising the owner’s identity [3].

Consequently, it needs to be recognized that anonymization techniques may not stand the test of time. To mitigate such risks, the unnecessary information needs to be deleted from blockchains as a preventive measure.

While doing this, it is still useful to store information such as current account balances and recent transactions to prevent double spending attacks. However, there is no evidence storing transactions ranging multiple years back is a prerequisite for a secure blockchain.

Existing solutions that allow removing transactions on a blockchain (like the one provided by Accenture [4]) involve giving a trusted party permission to arbitrarily edit the blockchain. Such trusted parties are only available in specific situations. Since blockchain is a general technology, a more general, trust-free solution appears to be needed. We therefore state our research question:

How can we design a decentralized blockchain that forgets, and what implications arise from it?

To answer this question, we follow a design science research (DSR) approach [5]. We develop an IT artifact in the form of a prototype and evaluate it to figure out how to design trust-free deleting blockchains.

This paper starts with a short overview of related works and theoretical background of blockchain technology, followed by a brief introduction to the used DSR approach. We then describe the developed artifact and evaluate it. Finally, we discuss its implications, give an outlook and investigate limitations of our work.

[1] See https://www.eugdpr.org/ for details on GDPR

URI: https://hdl.handle.net/10125/60145
ISBN: 978-0-9981331-2-6

2. Related Works

Attempts to protect users' privacy on blockchain are as old as Bitcoin itself, since the Bitcoin network employs a few techniques to enable anonymous transactions. We identified three types of approaches to solve the problem: anonymization techniques, altering techniques, and decentralized deletion techniques.

Anonymization techniques store the identifying information of a user outside the blockchain. Bitcoin for example, uses anonymous addresses to handle transactions, and users are free to create and use as many addresses as they want [6].

Nevertheless, studies have shown that it is possible to link multiple Bitcoin addresses of the same person together by analyzing their transaction behavior, with the success rate increasing as more transactions are made [7].

To combat this, some anonymization techniques not only conceal the identity, but also the actions of each user. As an example, Monero [8] was built to support untraceable transactions.

It should be noted that the methods mentioned so far store the anonymized information indefinitely on the blockchain and are subject to scrutiny for an indefinite time [9]. As research progresses and computing power increases, systems currently believed to be secure may be found vulnerable in the future. As an example, the earlier versions of the aforementioned Monero algorithm were cracked, retroactively leaking identities of old transactions [10].

Blockchain altering techniques give access to a trusted party to alter or delete transactions. One such approach was presented by Accenture in 2016 [4]. Using a special mathematical function, it becomes possible to retroactively replace the content of old blocks. This solution and other methods we inspected give access to a trusted party to alter or delete transactions. In a situation where such trusted parties can be found, this technique can fix privacy issues by combining many transactions into one summary transaction which lacks historical information.

Lastly, we look at decentralized deletion techniques. Research in this area has focused on the scaling issues that blockchains face. Pruning data that is no longer required is advantageous when each new transaction increases the size of the blockchain. The amount of research considering its privacy benefits is sparse. As the algorithms share a main goal, namely to increase efficiency, they do not always delete information in a timely manner.

Once again, the Bitcoin paper [6] provides one of the earliest methods of pruning, with a variation being implemented in the software in mid-2015 [11]. Unfortunately, Bitcoin's pruning algorithm keeps information about the last transaction of each coin.

Another Blockchain technology, Ethereum, seems more hopeful in this regard.

In an article, the creator of Ethereum, Vitalik Buterin, has described pruning strategies viable for Ethereum and suggests a method that removes all old blocks [12]. As costs of storing transactions are low, no Ethereum-based software has so far implemented the suggested algorithm.

The method described in Buterin's article is particularly relevant for our research question, as it leaves no historical information on the disk. Only account balances and other necessary information is retained in the long term.

A final point to make is that the pruning algorithms presented are meant to be run by a few nodes in the network. Little is known about the side effects of running networks where every single node prunes information, therefore globally deleting it.

3. Theoretical Background

A blockchain is, as its name points out, a concatenation of blocks. Its consistency is ensured by cryptographic protocols. How this is done in detail is up to the specific implementation.

While the contents of blocks can be set arbitrarily, most implementations store transfers of an asset from participants (inputs) to other participants (outputs) in a transaction. The transaction is only valid if it is signed using the private key of the owner of the input. Therefore, anyone possessing the public key can verify the transaction, but not modify it. A set of those is then bundled together in a block. Next to these transactions, the block includes a checksum of the previous block, creating a chain structure. Now, if the content of the previous block is modified, subsequent blocks become invalid. Every participant holds a complete copy of all the blocks. By iterating through all transactions, it is therefore possible to decide if a transaction is allowed or not.

If new transactions are to be added, they are grouped together into a new block, for which a consensus needs to be found. Most approaches do this using the proof of work algorithm. A checksum for the new block needs to be calculated. This checksum needs to fulfill certain predefined requirements (e.g. starting with 5 zeros). To accomplish this, a nonce is added to the block. The nodes now compete to find a nonce that in combination with the transactions and other block data, yields a checksum that fulfills those requirements. The winning node (e.g. the first) is granted some reward, mostly in the monetary form of tokens. This process is called ‘mining’.

These blocks are distributed to every participant in the network and therefore everyone has the same information.

Smart Contracts and the State

Some state-of-the-art blockchains (e.g. Ethereum or Hyperledger fabric) provide a powerful additional feature: the ability to execute code [13]. This extends the blockchain from simply being a distributed database to a distributed computer.

Ethereum, for example, can be defined as a Turing-complete multi-purpose shared computer named the Ethereum Virtual Machine (EVM). The machine is a singleton, so there exists only one global instance. Using blockchain technology, the state of this machine (also referred to as the “*state*”) is agreed upon. The state acts as a persistent storage medium to store all account balances, smart contract code (or “*EVM code*”) and internal storage.

Each smart contract is given its own internal storage on the state to store arbitrary information. Unlike a traditional computer, the EVM limits what parts of the state may be altered. Participants cannot spend tokens of an account unless they possess the associated private key; once deployed, smart contract code can never be updated, and code only has write-access to its own storage space.

The state is modified with *transactions*, which can send tokens, upload code or call a smart contract function. The result of a transaction execution is an updated state. In addition to this updated state, a *transaction receipt* is created. These receipts contain log messages and errors during the execution. The state itself does not store any historical information [14].

Ethereum follows a “state-centric model” [15], meaning that the EVM only requires current state data to process a transaction. Transaction history is not available inside the virtual machine.

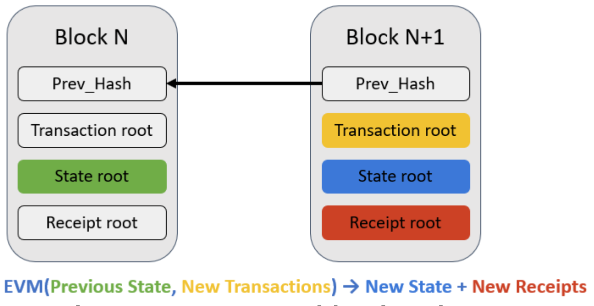


Figure 1: State transition in Ethereum

Hashes of transactions, receipts, and the state are included in block headers which make up the blockchain (see Figure 1). The Ethereum blockchain uses a proof of work algorithm to generate consensus on the order of transactions. Ethereum nodes need, just like their Bitcoin pendants, to download the full blockchain and verify all transactions. Ethereum nodes download and verify all transactions that were executed on the Ethereum computer.

Support for smart contracts in Ethereum has opened the door to a wide variety of use cases [16]. These provide new opportunities for companies to work together and share data. Using blockchain offers data integrity, security, fail-safety, and can be a cost-effective, decentralized alternative to using a service provider.

4. Research Methodology

In this paper, we make use of a design science research approach (DSR) [5]. In DSR the goal is utility, which means that instead of studying an already existing IT artifact, it involves the identification of an highly relevant problem [17] — the research cycles for the creation of a solution and the evaluation of it [18]. This artifact can be of various nature, for example a construct, model, method, or instantiation [19].

As there existed no solution that fulfilled all our requirements, we chose a DSR approach.

We followed the DSR methodology by executing the following steps: (1) identify the relevant problem; (2) define solution objectives; (3) design and develop an innovative IT artifact, (4) demonstrate it; and finally, (5) evaluate it.

4.1 Problem identification

We were made aware of the problem that immutability of the blockchain technology, and therefore the restriction that nothing can be deleted, poses an issue for companies that must follow the GDPR rules during a conference in 2017. Two researchers of the group talked to a practitioner from the financial service industry, who explained that the fact that companies cannot delete selected or all information from blocks, rules out using blockchain for a variety of relevant use cases.

Since the team already experimented with Ethereum and discussed possibilities to manipulate data in a blockchain, there already existed an initial idea to implement that feature. The team therefore proposed to create a proof-of-concept prototype of an Ethereum-based blockchain.

4.2 Definition of objectives

The primary objective is to build a working prototype to demonstrate that it is possible to delete transactions from a blockchain while maintaining functionality. Since this involves tinkering with some of the fundamental concepts, we define requirements that the prototype needs to fulfill: (1) it needs to be tamper-resistant; and (2) all information needs to be distributed to all nodes in the network. (3) It should be possible for new nodes to join the network afterwards; (4) all nodes can add transactions. In addition, the prototype (5) must be decentralized and may not rely on trustees and finally (6) we delete every transaction after a predefined amount of time.

4.3 Design and development

To resolve the privacy issues mentioned, we first develop a new pruning algorithm that locally deletes as much information as it can without breaking the software. We then build a private Blockchain network where every node runs our created pruning algorithm to conduct tests and our evaluations.

We build the prototype based on the Ethereum blockchain using the Parity client. Ethereum is one of the most common and mature protocols that provides smart contracts. Parity was chosen because the source code is well-documented, and the members of the team were already familiar with it.

Designing a pruning algorithm

To develop our pruning algorithm, we first identify which information stored by clients is not relevant for the operation of the blockchain. By doing so, we can assess which information we can safely remove without impairing the blockchain functionality.

We analyze which features of Ethereum require access to historical data and validate our conclusions by looking into the behavior of two Ethereum implementations: pyethereum [15] and Parity [20].

	Current block	Recent blocks	All blocks
First sync	X	X	X
Staying in sync	X	X	
Sending transactions	X		
Mining	X		
Transaction history	X	X	X

Table 1. Summary of the dependencies of each action

As smart contracts only have access to the state, their functionality cannot be impaired if we delete old blocks. Similarly, mining only requires the current block.

Sending transactions (and simulating their effects locally to see the consequences) only requires the current block (and its state) as well. During mining, multiple solutions can be found for a block's successor, creating two or more branches in the blockchain (see Figure 2). This means that a branch of blocks in the chain may be exchanged for a longer branch found. Therefore, clients need access to recent blocks to be able to stay in sync over extended periods.

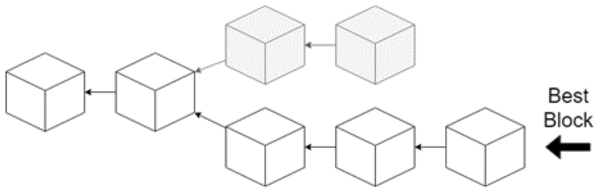


Figure 2: Not every block makes it into the final chain. Rollbacks are sometimes necessary.

At this point we can make storing a list of all blocks optional by making two specific assumptions: no new nodes are added to the network, and we don't need a transaction history. Special solutions for these cases are discussed in the Limitations section.

With these restrictions in place, we implement a feature that deletes blocks after a brief period. This follows GDRP's principle to store data for the shortest amount necessary.

Through inspection of the Parity code, we identify databases storing portions of each block, such as transaction data, receipt data, and state. Each is stored separately. Multiple indexes are also created to speed up database lookups.

To delete state data, we make use of an already implemented feature in Parity called state pruning. It was implemented to allow increased scalability of a blockchain, as each new transaction increases the size of the blockchain.

Unlike state pruning, deleting other data is not automatically done by the official Parity implementation. We add a function that deletes all data stored about a block across nine different databases (source code available in our GitHub repository[2]).

The finished software can connect to Ethereum networks and process new transactions while

[2] Available at: https://github.com/Yonom/parity-demo

automatically deleting old ones locally. We now proceed to the second development stage.

Building a network that forgets

So far, our software had no effect on a network because it was the only participant to delete information. For a network to forget, every participant must run our pruning algorithm.

We test this scenario using 5 hosts. running our modified Parity software; and another containing Etherchain Light [21], an open-source blockchain explorer.

For testing purposes, we have set the block deletion duration to 10 blocks (30 seconds). We deployed our Parity client over five hosts and additionally connected a block explorer to the first client (see Figure 3).

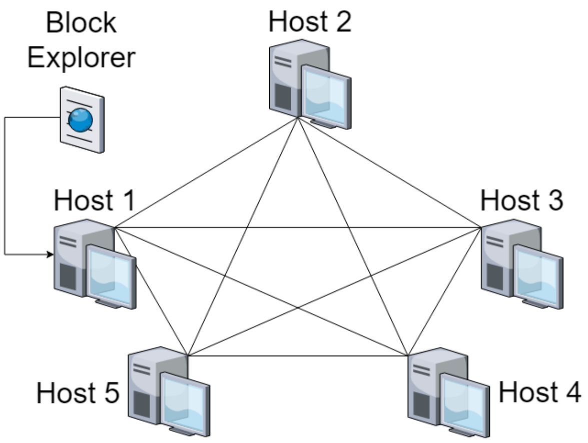
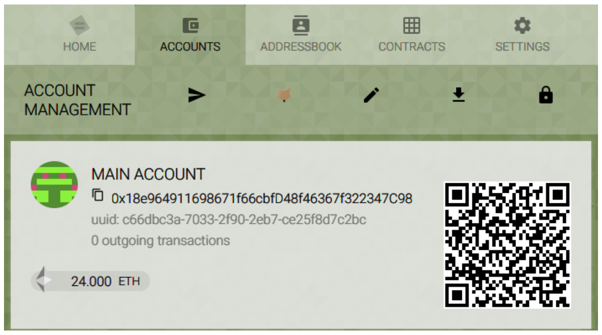


Figure 3: Our network constellation

.

Each host has a Parity user interface (see Figure 4) that allows interaction with the blockchain on behalf of that host.



MAIN ACCOUNT
0x18e964911698671f66cbfD48f46367f322347C98
uuid: c66dbc3a-7033-2f90-2eb7-ce25f8d7c2bc
0 outgoing transactions
24.000 ETH

Figure 4: Parity client

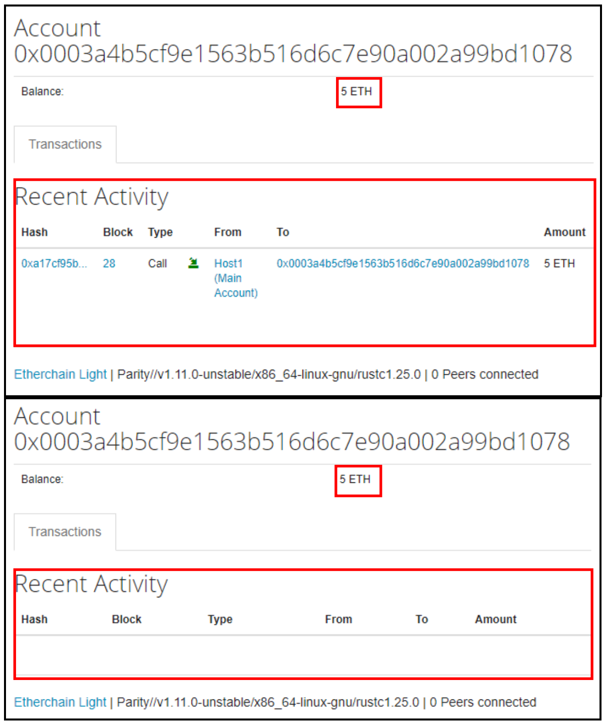
Additionally, we provide a block explorer showing the user a live view of how transactions vanish after a specific time is passed.

4.4 Demonstration

We demonstrated the built prototype to experts from the financial service industry. These experts were one blockchain engineer and two senior consultants. Since all our experts had a background in finance, our use case for the prototype were financial transactions. For every participant, we conducted a separate session to introduce the setup, the prototype, and finally, to get their feedback.

We first introduced the experts to the five hosts that built the foundation of our blockchain. We then gave a quick introduction to the user interfaces, the block explorer and an overview of the basic features (transfer ether; upload, execute, and delete smart contracts). We then asked the experts to transfer a few Ether across accounts. The experts could follow the effect using the block explorer, especially the format of the transaction and the resulting changes to the state.

After 30 seconds, the expert witnessed live using the block explorer how the "pruning" functionality, (which deletes old blocks,) caused the transactions to disappear from the block explorer (see Figure 5, for a screenshot of the situation before and after). The account, however, kept its balance but lost information about the token's origin. Since the experts saw that the transaction was verified before accepting it and before it got deleted, they were assured that it is legitimate.



Account
0x0003a4b5cf9e1563b516d6c7e90a002a99bd1078

Balance: 5 ETH

Transactions

Recent Activity

Hash	Block	Type		From	To	Amount
0xa17cf95b...	28	Call		Host1 (Main Account)	0x0003a4b5cf9e1563b516d6c7e90a002a99bd1078	5 ETH

Etherchain Light | Parity//v1.11.0-unstable/x86_64-linux-gnu/rustc1.25.0 | 0 Peers connected

Account
0x0003a4b5cf9e1563b516d6c7e90a002a99bd1078

Balance: 5 ETH

Transactions

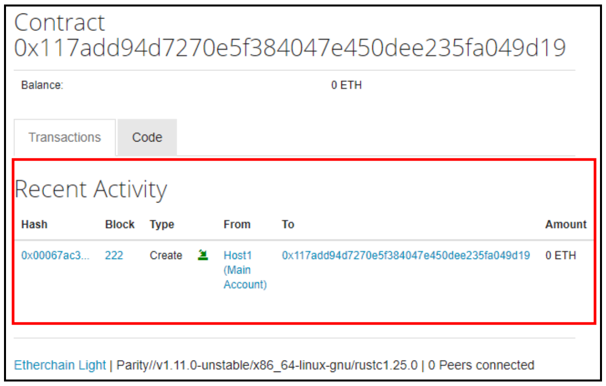
Recent Activity

Hash	Block	Type	From	To	Amount

Etherchain Light | Parity//v1.11.0-unstable/x86_64-linux-gnu/rustc1.25.0 | 0 Peers connected

Figure 5: An account with Ether balance and deleted transactions

We then proceed to the second part of evaluation, where we provide the experts with a simple contract that they can deploy to the blockchain.



Contract
0x117add94d7270e5f384047e450dee235fa049d19

Balance: 0 ETH

Transactions | Code

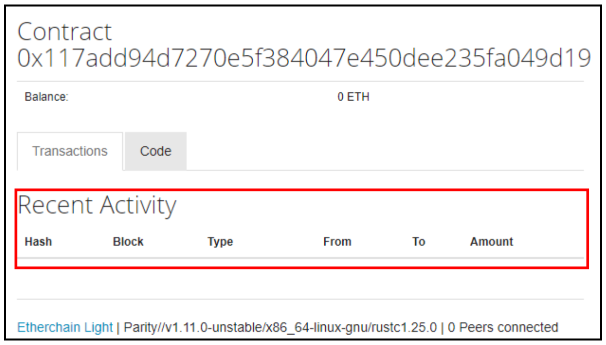
Recent Activity

Hash	Block	Type	From	To	Amount
0x00067ac3...	222	Create	Host1 (Main Account)	0x117add94d7270e5f384047e450dee235fa049d19	0 ETH

Etherchain Light | Parity//v1.11.0-unstable/x86_64-linux-gnu/rustc1.25.0 | 0 Peers connected

Figure 6: Creation of a contract

Figure 6 shows the transaction that creates the contract. Figure 7 shows how the transaction is deleted but the contract code remains in the state.



Contract
0x117add94d7270e5f384047e450dee235fa049d19

Balance: 0 ETH

Transactions | Code

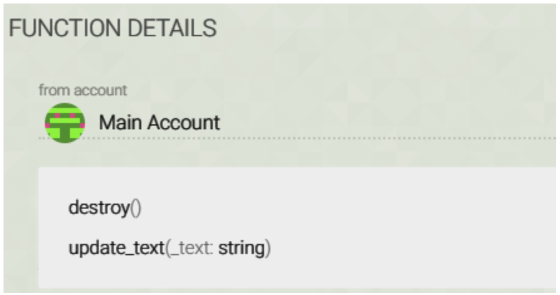
Recent Activity

Hash	Block	Type	From	To	Amount

Etherchain Light | Parity//v1.11.0-unstable/x86_64-linux-gnu/rustc1.25.0 | 0 Peers connected

Figure 7: Contract code still exists, but the creation transaction no longer does

A function in the contract allows the user to set the text of a variable to their liking (see Figure 8). The change of this string is sent in the form of a transaction. Like every other transaction, our experts observe that it is deleted after 30 seconds, meaning that only the most updated text is available.



FUNCTION DETAILS

from account
Main Account

destroy()
update_text(_text: string)

Figure 8: Available smart contract methods

Finally, after some experimentation, the expert calls the smart contract's "destroy" function (see Figure 8). This function internally executes an Ethereum feature called "self-destruct", which will remove the contract code from the state. 30 seconds after this method is called, no trace of the smart contract ever existing, including the code itself, remains on the blockchain.

As a last demonstration, we deactivate two random nodes. One of the nodes is restarted before 30 seconds elapse and another is restarted after a minute. The node that was restarted earlier can synchronize to the network without issue. The other node fails to synchronize and remains detached from the network.

Nevertheless, the blockchain continues to function regardless of which nodes are deactivated. It continuously mines new blocks and deletes old blocks without issue.

4.5 Evaluation

Based on the feedback we received from the experts and the issues identified, we assessed the accomplishment of our original objectives in Section 4.2.

We created a prototype blockchain that can remove historical data. Our demonstration shows that mining, transactions, and smart contracts still work despite the changes made to the software.

Additionally, since we did not alter the transaction verification protocol, we still possess a tamper-proof blockchain where only the owners of accounts can spend their tokens and the code of smart contracts cannot be altered or deactivated by third parties.

We remain decentralized, and no part of the network relies on a specific node. All nodes are configured the same way and we can turn off hosts at will.

Unfortunately, we hit a few limitations to our approach, such as the fact that adding new clients to the network is no longer straightforward.

We also compare our new method to other available approaches for implementing privacy on the blockchain based on two criteria: being decentralized and being immune to retroactive attacks (see Table 2). We considered a solution decentralized if there were no central actors with special powers. Therefore, both anonymization and block deletion techniques are decentralized. We could not find any decentralized chain editing techniques.

Immunity to retroactive attacks prevents old identities being revealed if today's algorithms are cracked tomorrow. When sensitive information is kept indefinitely, as is usually the case with anonymization techniques, this property is not given. Both chain editing and block deletion can permanently remove old information from blocks therefore, they are immune to retroactive attacks.

	Decentralized	Immune to retroactive attacks
Anonymization	X	
Chain editing		X
Block deletion	X	X

Table 2: Solutions to Blockchain privacy problems

We conducted evaluation sessions with experts from the financial service industry. None of the participants, who were all familiar with blockchain, expected a working result that could at least partially solve the problem. At the end of the session, all experts agreed individually that a GDPR-compliant financial transaction is possible with the prototype.

However, we received several questions which we included here.

'How can the blockchain be secure if we cannot verify history?'.
The blockchain is secure because we verify every change to the state. This verification cannot be repeated after we delete the transactions, but we can assume that our client verified older transactions.

'If the blocks are distributed to every participant, how can I prevent anyone from taking backups?'.
Although we delete the blocks from the chain by default, the technique does not allow us to prevent participants from taking backups. Still, this is an advantage as the standard behavior is to delete and not to keep data. Another note was that not all data is meant to be deleted:

'What if I want to store data longer than a few seconds?'.
Data in transactions is not available to smart contracts and transactions only stay around for a few days. Transactions are therefore not a suitable place to store information. We recommend storing the data in the state using smart contract logic.

'In case someone new joins the chain, who should they trust, if there are multiple nodes, as there is no history?'
Since we delete all history, the first block (called the "genesis" block) does not exist anymore and therefore cannot be used as an initial syncing point. A node therefore must ask multiple other nodes for a block in the middle of the chain and check if they all recognize this block.

5. Limitations

During our evaluation, we were able to identify several limitations. First, it seems impractical to run our network in a public setting. Our approach cannot enforce a network-wide deletion of old data, since it cannot control that participants keep backups. Having some participants make backups defeats the purpose of our blockchain and thus we believe that for now, it can only be used inside restricted environments where additional financial and legal incentives are established to prevent archiving (e.g. through auditing). Nevertheless, we recognize a need for future research in this area.

Second, one loses the built-in history of all actions performed on smart contracts, as old blocks and their transactions are removed from history. Smart contract code is still unalterable in our prototype and one could log all important information to the state, which is persistent. The logging is no longer mandatory and must be explicitly implemented in smart contracts that require history.

The third shortcoming lies in the process of adding new nodes. Since the information required to derive the current state from old transactions is no longer available on the network, the original "genesis" block cannot be used as an initial syncing point. Some other recent block must instead be taken as the starting point. This process is currently technical and tedious, and we believe it should be improved through software updates.

The choice of the correct initial block is subjective and requires trust. Previous work [22] on "weak subjectivity" shows that this process is less secure than using a well-known genesis block, and one must take great caution and ask multiple sources when looking for an initial block to trust. Thankfully, the process must only be executed once during the initial setup of the node.

Lastly, a node which is turned off for an extended period will fail to sync to the network as the blocks lying between its last block and current newest block may already be deleted. In this case, one must execute the tedious setup process again. For this reason, we recommend setting the deletion time to a reasonably long amount to account for possible system downtimes (e.g. seven days).

6. Discussion and conclusion

In this paper, we gave insights through the design science research approach that we used to develop and evaluate a novel IT artifact for the financial services industry. We accomplished this by first identifying a highly relevant problem. After evaluating the existing approaches and literature, we proposed a possible solution and defined the objectives that we wanted to reach with the prototype and the evaluation criteria. We provided an IT artifact in the form of a working proof-of-concept prototype of a blockchain that deletes predefined data after a predefined amount of time. The prototype was developed in an iterative manner and was evaluated with the help of domain experts. Finally, we derived helpful principles for designing data-protection compliant blockchains.

Our prototype uses a combination of an already implemented technique (state pruning) and a custom function to delete logs and other traces, as well as to enable the logging of predefined transactions in the state of the EVM.

This approach can be categorized as exaptation, as it is extending an at least partially known solution to a new problem, which should yield research opportunity and knowledge contribution [18]. The evaluation points out that our prototype solves the posed problem but introduces a set of limitations.

The limitations we face show us that there is potential for future research. We see a need for new smart contract best practices when it comes to blockchains that forget. Our team plans to investigate how smart contracts can be monitored and audited when historical information is not available.

The implications that arose from our results are twofold. For practitioners, we managed to identify a way that could allow blockchain technology to be used in an additional variety of scenarios that couldn't be done before. Additionally, while we have focused on the benefits of block deletion for permissioned blockchains, the techniques discussed here could be used to solve scaling issues on public blockchains, as the amount of data grows too big to be feasible to store forever.

Finally, there is a lack of understanding on the possibilities that smart contracts offer. Especially when it comes to use cases, there is almost no reference material on the potential abilities of smart contracts and the possible benefits that an extended use could bring.

We hope that advancements in the capabilities of technology, like the one presented in this paper, make the use of blockchain more practical and speed up progress in this field.

7. References

[1] F. Hawlitschek, B. Notheisen, and T. Teubner, "The limits of trust-free systems: A literature review on blockchain technology and trust in the sharing economy," *Electron. Commer. Res. Appl.*, vol. 29, pp. 50–63, 2018.

[2] R. Beck, C. Müller-Bloch, and J. L. King, "Governance in the Blockchain Economy: A Framework and Research Agenda," *J. Assoc. Inf. Syst.*, 2018.

[3] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *Security and privacy in social networks*, Springer, 2013, pp. 197–223.

[4] R. Lumb, D. Treat, and O. Jelf, "Why distributed ledger technology must adapt to an imperfect world," 2016. [Online]. Available: https://www.accenture.com/t20160927T033514Z__w__/ae-en/_acnmedia/PDF-33/Accenture-Editing-Uneditable-Blockchain.pdf. [Accessed: 23-Nov-2017].

[5] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2007.

[6] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: Www.Bitcoin.Org. [Accessed: 05-Mar-2017].

[7] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in bitcoin using p2p network traffic," in *International Conference on Financial Cryptography and Data Security*, 2014, pp. 469–485.

[8] N. Van Saberhagen, "CryptoNote v 2.0," 2013. [Online]. Available: https://cryptonote.org/%0Awhitepaper.pdf. [Accessed: 19-Nov-2017].

[9] P. Roßbach, "Security in Blockchain Applications," *2018-03-13*. [Online]. Available: https://blog.frankfurt-school.de/security-in-blockchain-applications/?lang=de. [Accessed: 01-May-2018].

[10] M. Möser, K. Soska, E. Heilman, K. Lee, H. Heffan, S. Srivastava, K. Hogan, J. Hennessey, A. Miller, and A. Narayanan, "An Empirical Analysis of Traceability in the Monero Blockchain," *Proc. Priv. Enhancing Technol.*, vol. 2018, no. 3, pp. 143–163, 2018.

[11] "Bitcoin Core version 0.11.0," 2015. [Online]. Available: https://bitcoin.org/en/release/v0.11.0. [Accessed: 07-Feb-2018].

[12] Vitalik Buterin, "State Tree Pruning - Ethereum Blog," 2015. [Online]. Available: https://blog.ethereum.org/2015/06/26/state-tree-pruning/. [Accessed: 24-Nov-2017].

[13] F. Tschorsch and B. Scheuermann, "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[14] Buterin Vitalik, "Design Rationale," *2014-12-05*, 2014. [Online]. Available: https://github.com/ethereum/wiki/wiki/Design-Rationale#accounts-and-not-utxos. [Accessed: 15-Oct-2017].

[15] V. Buterin, "GitHub repository: pyethereum," *2014-12-05*. [Online]. Available:

https://github.com/ethereum/pyethereum. [Accessed: 02-Jan-2018].

[16] K. Nærland, C. Müller-Bloch, R. Beck, and S. Palmund, "Blockchain to Rule the Waves-Nascent Design Principles for Reducing Risk and Uncertainty in Decentralized Environments," in *ICIS*, 2017.

[17] H. A. Simon, *The sciences of the artificial*. MIT press, 1996.

[18] S. Gregor and A. R. Hevner, "Positioning and presenting design science research for maximum impact.," *MIS Q.*, vol. 37, no. 2, pp. 337–355, 2013.

[19] S. T. March and G. F. Smith, "Design and natural science research on information technology," *Decis. Support Syst.*, vol. 15, no. 4, pp. 251–266, 1995.

[20] - Parity Authors, "Github repository: Parity." .

[21] - Etherchain light Authors, "GitHub repository: Etherchain light," 2018. .

[22] V. Buterin, "Proof of Stake: How I Learned to Love Weak Subjectivity - Ethereum Blog," 2014. [Online]. Available: https://blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-subjectivity/. [Accessed: 01-Mar-2018].