On the Similarities between Blockchains and Merkle-Damgård Hash Functions

Conference Paper · July 2018

DOI: 10.1109/QRS-C.2018.00035

3 authors:

Kimmo Halunen
VTT Technical Research Centre of Finland
36 PUBLICATIONS **64** CITATIONS
SEE PROFILE

Visa Vallivaara
VTT Technical Research Centre of Finland
13 PUBLICATIONS **13** CITATIONS
SEE PROFILE

Anni Karinsalo
VTT Technical Research Centre of Finland
12 PUBLICATIONS **22** CITATIONS
SEE PROFILE

On the Similarities Between Blockchains and Merkle-Damgård Hash Functions

Kimmo Halunen, Visa Vallivaara and Anni Karinsalo
VTT Technical Research Centre of Finland Ltd
Kaitoväylä 1, Oulu, Finland
Email: firstname.lastname@vtt.fi

***Abstract*—Blockchain as a new technology has created a great amount of hype and hope for different applications. There is a promise of a better, decentralised trust based on strict guarantees from cryptography. However, there is a great similarity in the structure of blockchains and classical iterated hash functions of the Merkle-Damgård (M-D) type. As the structure of M-D type hash functions has been extensively studied and many different structural weaknesses have been exposed, it is plausible to think that blockchains also share these structural problems. In this paper, we present the most relevant problems of M-D type hash functions and their relation to blockchains. We also examine how these might affect currently established blockchains. Our results can help in avoiding some problems in the design of new blockchain systems and also provide some (theoretical) limits on the trustworthiness of current blockchains.**

***Index Terms*—blockchain; consensus mechanism; iterated hash functions**

I. Introduction

Blockchains have emerged as a new and revolutionary idea and are claimed to provide new opportunities in many digital domains. There is a promise of a better, decentralised trust based on strict guarantees from cryptography. This has created a large amount of hype and a multitude of different domains, where this technology is seen as a method to achieve revolutionary improvements.

The main initiator for this development has been Bitcoin [17], which is at the moment the most popular and most valued digital cryptocurrency. The blockchain method behind Bitcoin's consensus mechanism has since been taken into use in many other platforms and for different purposes. It is also now under scrutiny and research from many directions and new insights and results are needed to understand the power and limitations of this technology.

Blockchains as a structure share a striking commonality with classical Merkle-Damgård (M-D) hash functions. Namely, a blockchain forms an iterative structure, where the value of the previous block's hash is used as an input for the next block, for which a hash value is (usually) computed to gain a valid block. Thus, it is reasonable to ask the question, to what extent (if any) these similarities affect the security and trustworthiness of blockchains.

The structure of M-D hash functions has been studied extensively over the years and there are many different weaknesses and attacks that can be performed on such as SHA-1 [5], RIPEMD-160 [4] and SHA-256. For example [14] presents many such attacks that generate second preimages for given hash values. These attacks can also pose threats to blockchain systems, if these are not adequately taken into account.

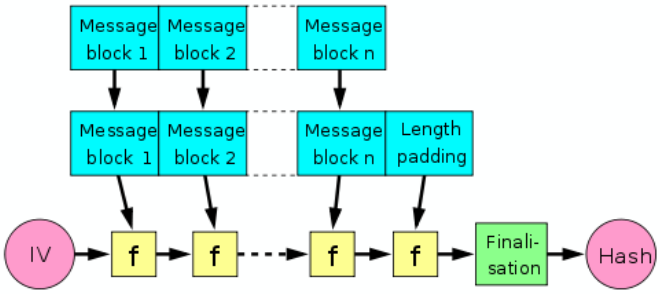

Fig. 1. Merkle-Damgård hash function construction

Our contributions are the following. We show how blockchain as a structure is similar to traditional M-D hash functions. We review existing results on the security of M-D hash functions especially from the structural point of view and how these relate to blockchains. We do not cover all specific manifestations of these such as MD5 [22] and SHA-1, except when they provide information on the security of actual blockchains. We also look into three different blockchain platforms (Bitcoin [17], Ethereum [24] and IOTA [20]) and their consensus mechanisms and study their security properties in the light of the weaknesses of an iterative structure.

II. Previous Work

Previous work related to this paper can be divided into two parts. The work done on the security of blockchain systems and the work done on analysing the the weaknesses of the iterative M-D hash function structure. Both are briefly examined in this section.

A. Blockchain security

In [7] the authors show, that the Bitcoin [17] system is somewhat susceptible to some broken primitives such as hash functions and digital signature schemes. Their results pertain to Bitcoin and consider the breaking of hash functions as an oracle. They define a new oracle specifically targeting the special format of hash values used in Bitcoin mining and second preimages of these.

The results provide insight to the security of Bitcoin and also demonstrate the need to keep in mind the security of all the different primitives and their uses in such a system. The paper does not provide immediate attacks that could be utilised, because there has not been such weaknesses reported for the primitives used in Bitcoin.

Most of blockchain security related results consider only one single blockchain such as IOTA or Bitcoin [11] or present a high level overview of security issues [16]. Structural properties of blockchains have not been studied extensively, except when it comes to different consensus mechanisms. We relate our work to different consensus mechanisms in section IV and in our discussion of our findings.

In [6] the authors present a simulation based method to evaluate the security of proof-of-work (PoW) blockchains. Their analysis leads to some optimal strategies for adversaries against these types of blockchains. Their results are interesting, but they do not evaluate the inherent structure of blockchains, but base their evaluation on many other properties and incentive models. Furthermore, they are very much focused on Bitcoin, although many of their ideas generalise to other PoW systems.

B. Properties and Weaknesses of M-D Hash Functions

Many hash functions developed in the 1990s and 2000s are based on the Merkle-Damgård (M-D) construction, where a compression function (f) is iterated over the message m. The message is processed in blocks (from m_1 to m_n) and the result of the final iteration is defined as the hash value of that message. This design method has been under great scrutiny and several structural weaknesses in this mechanism have been identified. The process of M-D type hashing of a message is further illustrated in Figure 1.

Hash functions are usually required to satisfy three different high level security properties. These are called *preimage resistance, second preimage resistance* and *collision resistance*. Preimage resistance means that from a given hash function result (called the hash value) y, it should be hard to find any message x that yields $h(x) = y$ under a hash function h. Second preimage resistance means that given a message m it should be hard to find another message $m' \neq m$ with $h(m) = h(m')$. Finally, collision resistance means that it should be hard to find any two messages z and $z' \neq z$ with $h(z) = h(z')$.

The security of M-D hash functions is usually then derived from the security of the compression function and the chaining mechanism to the full hash function. In this way, many constructions have been analysed and can be shown to be secure or insecure. For example in [21] many ways to build hash functions from block ciphers (essentially an M-D type construction) are analysed.

There are many other analyses of the structural properties of M-D hash functions. A Joux multicollision [10] is a method for generating multicollisions i.e. sets of colliding messages for iterated hash functions for much lower amount of work than would be possible for an ideal hash function. The complexity

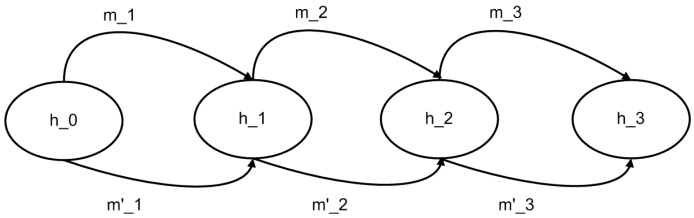

Fig. 2. Joux multicollision construction. There are eight different paths (messages) from h_0 to h_3 from only three collisions in the compression function.

of a k-collision for an ideal hash function is in $O(2^{\frac{n(k-1)}{k}})$[1], but for an iterated hash function (such as any M-D type hash function) such collision can be found in time $O(k2^{\frac{n}{2}})$. The idea of the method is shown in Figure 2.

Also for second preimages there are some methods, which can be used to find second preimages for iterated hash functions more effectively than for an ideal one. For extremely long messages, the likelihood of a collision in some intermediate values grows quickly, when the length of the message is over $2^{\frac{n}{2}}$ blocks [13]. This property can be exploited to generate a second preimage. In [13] it is stated that such messages would be artificially long and thus pose only a theoretical venue of attack against hash functions.

One of the first observations on iterated hash functions and generic second preimage attacks is from [3] by Dean. It exploits the fact that several M-D type hash functions use compression functions, which have easily found *fixed points*. A fixed point for a compression function f is a pair (h, m) for which $f(h, m) = h$.

Using fixed points it is possible to generate *expandable messages*, which are messages that all have the same intermediate hash value before the length padding, but different lengths. In [13] several algorithms are given on how to construct such messages. These can then be used in conjunction with long messages to generate second preimages for this long message.

Another, more effective method is herding, where a message is "herded" on to a right track to form a message with a predefined hash value [12]. This method enables an attack called chosen target, forced prefix second preimage. This means that there is a chosen target hash value and a forced prefix of a message (what this could be in the context of blockchains is discussed in Section III). The end of the message can be "arbitrarily" selected as long as the whole message hashes to the chosen target value. In [12] there are several possible methods shown, that can be used to mount such attacks against iterated hash functions. It is also discussed that if there are more effective ways of finding collisions in the compression function, then the total complexity of the attack can be significantly decreased.

A further improvement on the attacks on second preimages is the diamond structure, where a diamond-like structure of precomputed values is generated in order to find second

[1]We use the O notation from complexity theory to classify the work needed for several different attacks on hash functions. Readers who are not familiar with this notation can check for example Wikipedia (https://en.wikipedia.org/wiki/Big_O_notation) for quick reference.

preimages. A good exposition on diamond structures can be found in [15]. All these method rely on finding collisions for the compression function, which underlies the hash function. In good hash functions, such as SHA-256 this is a hard problem, but in some other contexts this might not be the case.

In Figure 3 the basic diamond structure is presented. The process starts from some initial value (which does NOT need to be the initial value specified by the hash function). Then a large set of intermediate hash values is constructed. From these values pairs are constructed, which reach a common intermediate value ($h(2,1),\ldots,h(2,4)$). From these values the process is repeated again and again until the target value is reached from two different intermediate values. Each arrow in the figure represents a different message block that compresses to the destination value starting from the value in the node where the arrow begins.

There are also rigorous treatments on the complexity of these attacks with respect to the underlying compression function [2]. Thus, it is possible to make analysis and trade-off decisions based on these attacks.

III. Blockchain as an Iterative Structure

The structure of a blockchain is very much the same as an iterated hash function. The blockchain starts from the genesis block (an initial value) and then new blocks are added to the chain usually with a hash value of the previous block being an integral part of the new block. The new block consists of some number of transactions and other information and then the next chaining value (an intermediate value). This value usually needs to conform to certain restrictions, for example containing a number of leading zeros as in Bitcoin.

Some key differences between a blockchain and a M-D hash function are that in a blockchain all the chaining values are public and immediately available to anyone. In an iterated hash function, the interface usually only yields the final value and not the intermediate values. However, the adversarial models of hash functions usually assume that the intermediate values can be accessed by the adversary. Also in a blockchain the full hash function acts as the compression function in the iteration. A basic structure of a blockchain is described in Figure 4.

A. Why is this a problem in blockchains?

Because a blockchain is a similar structure as an iterated hash function, the problems related to iterated hash functions are easily transfered to blockchains. Depending on the nature of the blockchain and its consensus mechanism the structural weaknesses may pose a threat to the actual security and trust claims that are made. This can then lead into actual attacks on the system(s) that apply the blockchain.

First of all, we can see that second preimages are something that blockchains should not have. A second preimage means that there is an alternate (backdated) version of a blockchain that ends up with the same hash value as the current legitimate blockchain. Many of the methods presented in Section II-B allow for expanding the "message" or the alternate blockchain and thus showing a version of the blockchain, a fork, that has required more work and thus is deemed "more valid" by some consensus algorithms (the longest chain paradigm).

The fork is a byproduct of distributed consensus, it happens when a blockchain diverges into two potential paths forward with regard to a network's transaction history. This happens usually when two miners find a block at nearly the same time. The ambiguity is resolved usually with the longest chain rule. For example Ethereum uses the "Greedy Heaviest Observed Subtree" protocol, which picks the path that has had the most computation done upon it. [23]

Another thing that makes blockchains prime suspects for some of the attacks previously presented against M-D hash functions is the requirement for long messages. In many cases, the attacks for hash functions require (artificially) long messages to work. In essence, blockchains are already very long and they have the potential to grow even longer in the long term. Thus, some methods that would be heavily impractical for "regular" messages and hash functions, may become practical for blockchains.

B. Checkpoints and Second Preimage Attacks

An interesting point for attacking a blockchain through a second preimage attack are so called *checkpoints* which act as trusted anchors of the blockchain and enable easier validation of transactions etc. Thus, such a point is a valuable target for a second preimage attack, because it would give the attacker an easy way to propagate the attack through the network and also give stronger legitimacy to their claim.

Figure 5 shows a simplification of the attack on a checkpoint via diamond structure. The attacker can form the variable blocks from valid (historical) transactions by varying their order, number etc. in the block.

The main steps of the attack are the following:

1) The attacker chooses a checkpoint on the blockchain and some transactions that she wishes to alter that have happened before that checkpoint.
2) The attacker claims that some transactions happened differently before that checkpoint (a form of a forced prefix) than what was previously on the blockchain.
3) The attacker produces a chain, which contains the forced prefix and then a different chain than the current consensus chain, which ends in the same valid hash as the current consensus at the checkpoint.
4) The attacker's version of the chain can be made more valid by using expandable messages or some other technique to win the consensus.

As this would require the use of the attacks from [12], there would be a need for an effective collision finding algorithm. As such an attack algorithm requires varying the contents of the block, there should be enough degrees of freedom over which different values for the hash function can be tried. The possible variables are the nonce (if any is used), the number, order and content of transactions in each block and other possible variables included in the blocks of the blockchain.

IV. Evaluation of Consensus Mechanisms

Most often used hash functions for cryptocurrency consensus mechanisms are SHA-256 and Scrypt [19]. Both of them

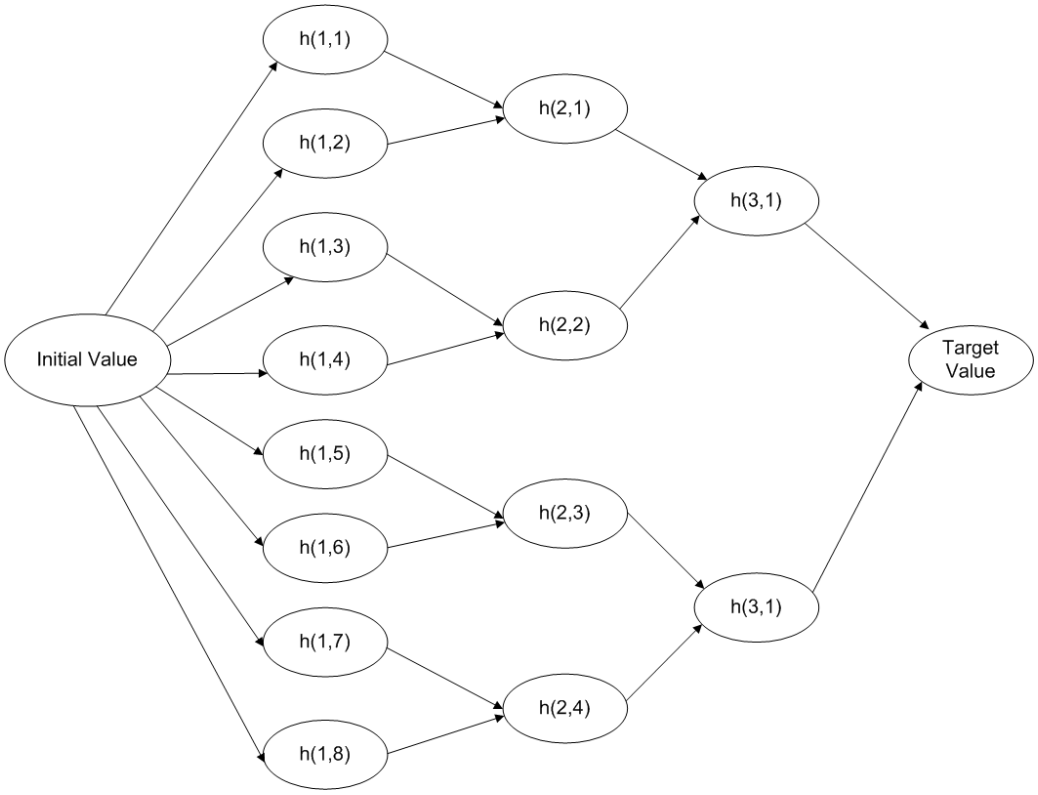

Fig. 3. Basic diamond structure. To create a second preimage the attacker needs to form a message with an extension that results in an intermediate hash value somewhere within the diamond. After that the attacker can follow the path from that node to the target value (each arrow represents some message block) and produce a second preimage.

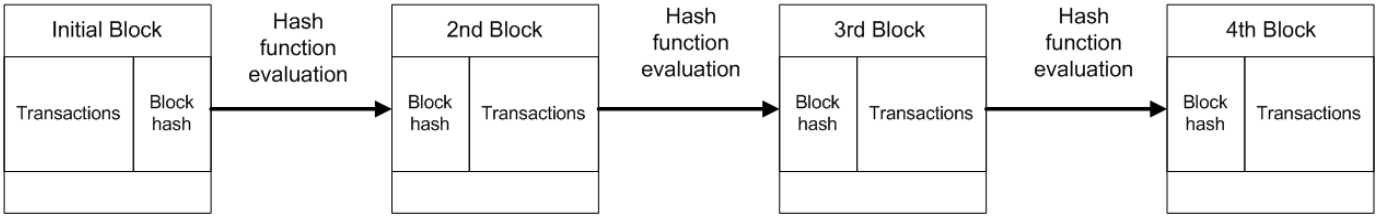

Fig. 4. Basic blockchain structure

can be mined with specialized hardware (ASIC). The creators of new blockchains sometimes seek to create hash algorithms that would be hard to translate into ASIC or other optimised platforms for different reasons. The new algorithms are not seriously tested and there may be vulnerabilities and unknown holes.

We examine the consensus mechanisms of three different blockchains namely, Bitcoin, Ethereum and IOTA. These all have different algorithms that they use for reaching consensus, but the main part is still the computation of a hash value of a certain type within some requirements. All these blockchains demonstrate the structural properties described earlier, but they are not equally vulnerable to the weaknesses.

A. Bitcoin

Bitcoin's consensus is based on the difficulty of finding a SHA-256 hash value with a predetermined number of zero bits in the beginning of the value. This difficulty is adjusted periodically to keep he block validation time at approximately 10 minutes per block. Thus, as the computing power used by the miners increases (or decreases) the diffculty is adjusted accordingly.

Because there are no known weaknesses on the SHA-256 hash function, the structural problems are not exploitable in the Bitcoin blockchain. Of course, any future attack on SHA-256 could bring the potential to use.

B. Ethereum

Ethash is based on Dagger Hashimoto algorithm and it works in the following way. At the start, data sets are generated and derived with hash function for verification which are updated every 30000 blocks. The mining function combines and hashes a random nonce, a hash of the block header and random slices from the data set. The function loops until a value, the proof of work, that satisfies the difficulty threshold is found. Finally, a block with the correct PoW is

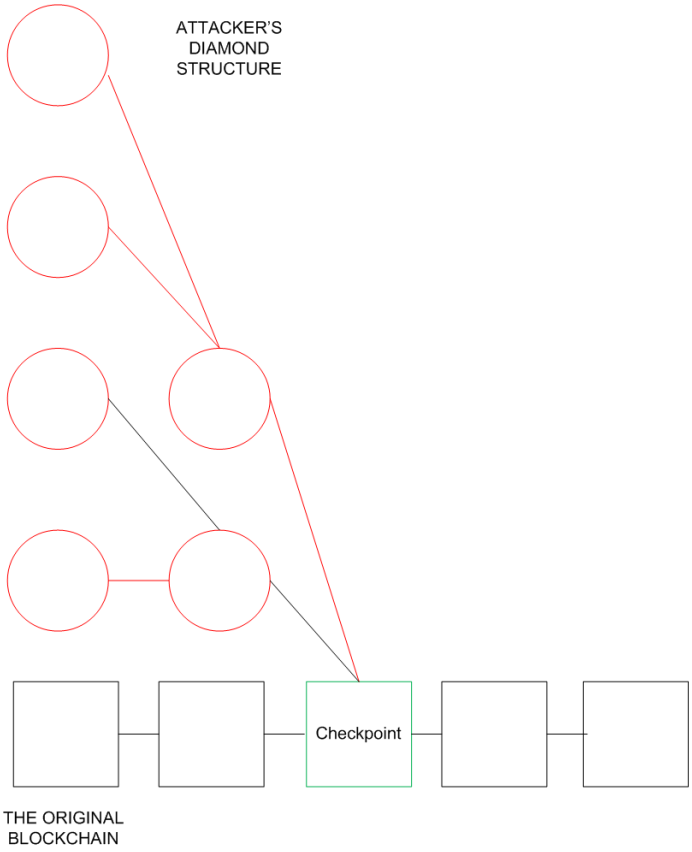

Fig. 5. Attacking a blockchain's checkpoint with a diamond structure. The black lines from the red circles demonstrate a valid second preimage to the checkpoint from the diamond structure.

submitted to the network. The amount of hashing rounds is adjusted dynamically so that the work amount and efficiency of Ethereum mining varies slightly through time.

As the hash function utilised in Ethereum is based on a cryptographic hash function, the SHA-3 competition winner Keccak [1], the structural weaknesses are not exploitable as such. It is interesting to note, that Keccak hash function is not of the M-D type, but still the structural weaknesses exist in the blockchain even if Keccak itself is not affected. Of course, if some efficient method for collisions for Keccak is found, then exploits are possible. However, then also many other system will be much affected.

C. IOTA

IOTA uses a directed acyclic graph called a tangle instead of simple chain structure as its underlying mechanism. One could be tempted to think that this might make attacks described earlier harder on IOTA, but in the case of iterated hash functions there are results that imply the opposite e.g. [8].

Furthermore there have been implications that the hash function used in IOTA is not as collision resistant as it should be [18], [9]. The attack shows that it is possible to generate fraudulent transactions in the IOTA blockchain.

Because many of the second preimage attacks for iterated hash functions presented in Section II-B of this paper rely on some form of collision finding as their basic tool, any such weakness could be exploitable in these attacks.

The main message of the above is that great care should be taken in choosing the right kind of hash function for the needs of the blockchain. Not all blockchain systems are alike, and thus not all have the same needs. However, when steering away from known cryptographic hash functions, the risk of a design flaw that will have serious implications on the security of the system increases.

V. Discussion

The theoretical implications of this research are clear. Blockchains share a common structure with M-D type iterated hash functions and as such suffer from similar structural problems. These problems are mostly related to the strength of the blockchains against second preimage attacks which mean conflicting versions of the "history" recorded in the blockchain.

The methods found to undermine the security of M-D type hash functions can be utilised also against blockchains. Thus, at least at a theroretical level, this should lead into re-evaluation of the level of security that blockchains provide. The amount of transactions in many blockchains is large enough to provide the necessary degrees of freedom to find the needed collisions etc.

Because of the ever growing nature of blockchains there are long chains of different hash values, which can be exploited in the attacks. This means that in very long term timescales these weaknesses should be taken into account.

On the other hand, the most popular blockchains such as Bitcoin and Ethereum have such strong hash functions in use that our findings do not have any implications on their security. Ethereum's blockchain generates new blocks very fast, around one block per 14 seconds, and it currently has over 5 million blocks. But it would take a little over 500000 years to generate 2^{40} blocks, a number that would have some meaningful impact for long message based attacks. As such, the long message attacks do not pose serious problems to these blockchains for the foreseeable future.

In many cases more practical attacks can be mounted against blockchains with less resources by acquiring some significant proportion of the hashrate or other computational resource that the blockchain uses to reach consensus. Conversely, if the consensus mechanism is not very strong cryptographically, the attacks demonstrated here might become viable and pose a real threat to the system. Because the development in the field of blockchains is fairly fast, it is possible that there will emerge new consensus mechanisms that do not take these possibilities into account and thus become vulnerable to these attacks. Also building upon self-made cryptographic primitives can also open a venue for these attacks. Traditional cryptographic hash functions and other primitives have been designed having these attacks as background information and are reasonably secure against these.

Another possibility is that future research on iterative structures yields new and more efficient methods to conduct attacks on blockchains. This similarity might work as a new incentive

to research these structural properties as cryptographic hash functions have steered away from this development. Such new results should be examined carefully in the context of blockchains and security.

A possible venue for future work is to research different types of blockchains and try to mount these attacks for example on checkpoints in systems that have weaknesses in their underlying hash functions or consensus mechanisms. Of course, any such findings should be disclosed responsibly in order to make it possible to mitigate the problems.

VI. Conclusion

In this paper we have demonstrated and studied the similarities of blockchains and Merkle-Damgård (M-D) type iterated hash functions. The similarities are striking and lead to the conclusion that the blockchain structure itself suffers from similar weaknesses that have been reported against M-D type hash functions previously. The weaknesses do not have immediate security implications on the most widely used blockchains, becuase the chosen hash functions are secure and do not allow for exploitation of the strucutral weaknesses. In the case of new blockchains with possibly different consensus mechanisms the weaknesses may provide attackers with tools to compromise the integrity of the blockchain. It is also possible that this similarity will bring new interest in the study of iterative structures and the potential weaknesses found therein.

VII. Acknowledgments

This research has been supported by Business Finland through Towards Digital Paradise -project and the Naked Approach strategic research opening[2]. The authors wish to thank Juuso Takalainen for insightful comments on the paper. Also special thanks to David Göthberg for the illustration on Fig. 1 from Wikimedia Commons[3].

References

[1] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Keccak sponge function family main document," *Submission to NIST (Round 2)*, vol. 3, no. 30, 2009.

[2] S. R. Blackburn, D. R. Stinson, and J. Upadhyay, "On the complexity of the herding attack and some related attacks on hash functions," *Des. Codes Cryptography*, vol. 64, pp. 171–193, 2010.

[3] R. D. Dean and A. Appel, *Formal aspects of mobile code security*. Princeton University Princeton, 1999.

[4] H. Dobbertin, A. Bosselaers, and B. Preneel, "Ripemd-160: A strengthened version of ripemd," in *International Workshop on Fast Software Encryption*. Springer, 1996, pp. 71–82.

[5] D. Eastlake 3rd and P. Jones, "Us secure hash algorithm 1 (sha1)," Tech. Rep., 2001.

[6] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 3–16.

[7] I. Giechaskiel, C. Cremers, and K. B. Rasmussen, "On bitcoin security in the presence of broken cryptographic primitives," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 201–222.

[8] K. Halunen, "Multicollisions and graph-based hash functions," in *Trusted Systems*. Springer, 2012, pp. 156–167.

[9] E. Heilman, N. Narula, T. Dryja, and M. Virza, "IOTA vulnerability report: Cryptanalysis of the curl hash function enabling practical signature forgery attacks on the iota cryptocurrency," https://github.com/mit-dci/tangled-curl/blob/master/vuln-iota.md, 2017.

[10] A. Joux, "Multicollisions in iterated hash functions. application to cascaded constructions," in *Annual International Cryptology Conference*. Springer, 2004, pp. 306–316.

[11] G. Karame, "On the security and scalability of bitcoin's blockchain," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1861–1862.

[12] J. Kelsey and T. Kohno, "Herding hash functions and the nostradamus attack," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2006, pp. 183–200.

[13] J. Kelsey and B. Schneier, "Second preimages on n-bit hash functions for much less than 2n work." in *Eurocrypt*, vol. 3494. Springer, 2005, pp. 474–490.

[14] T. Kortelainen, "On iteration-based security flaws in modern hash functions," Ph.D. dissertation, University of Oulu, 2014.

[15] T. Kortelainen and J. Kortelainen, "On diamond structures and trojan message attacks," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2013, pp. 524–539.

[16] I.-C. Lin and T.-C. Liao, "A survey of blockchain security issues and challenges." *IJ Network Security*, vol. 19, no. 5, pp. 653–659, 2017.

[17] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, https://bitcointalk.org/bitcoin.pdf.

[18] N. Narula, "Cryptographic vulnerabilities in IOTA," https://medium.com/@neha/cryptographic-vulnerabilities-in-iota-9a6a9ddc4367, 2017.

[19] C. Percival and S. Josefsson, "The scrypt password-based key derivation function," Tech. Rep., 2016.

[20] S. Popov, "The tangle. 2014," *URL: https://iota. org/IOTA_Whitepaper. pdf*.

[21] B. Preneel, R. Govaerts, and J. Vandewalle, "Hash functions based on block ciphers: A synthetic approach," in *Annual International Cryptology Conference*. Springer, 1993, pp. 368–378.

[22] R. Rivest, "The md5 message-digest algorithm," 1992.

[23] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 507–527.

[24] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[2] http://www.nakedapproach.fi/

[3] https://commons.wikimedia.org/wiki/File:Merkle-Damgard_hash_big.svg