RSA AND REDACTABLE BLOCKCHAINS

DIMA GRIGORIEV AND VLADIMIR SHPILRAIN

ABSTRACT. A blockchain is redactable if a private key holder (e.g. a central authority) can change any single block without violating integrity of the whole blockchain, but no other party can do that. In this paper, we offer a simple method of constructing redactable blockchains inspired by the ideas underlying the well-known RSA encryption scheme. Notably, our method can be used in conjunction with any reasonable hash function that is used to build a blockchain. Public immutability of a blockchain in our construction is based on the computational hardness of the RSA problem and not on properties of the underlying hash function. Corruption resistance is based on the computational hardness of the discrete logarithm problem.

1. INTRODUCTION

A *blockchain* is a distributed database that is used to maintain a continuously growing list of records, called blocks. Each block contains a link to the previous (or the next) block. A blockchain is typically managed by a peer-to-peer network collectively adhering to a protocol for validating new blocks. By design, blockchains are inherently resistant to modification of the data. Once recorded, the data in any given block cannot be altered retroactively without the alteration of all preceding (or subsequent) blocks and a collusion of the network majority.

A blockchain can be either public or private. Usually, when people talk about public blockchains, they mean that anyone can write data. A typical example of a public blockchain is *bitcoin*. In contrast, a *private* blockchain network is where the participants are known and trusted: for example, an industry group, or a military unit, or in fact any private network, big or small. In particular, what is now called by a popular name *The Internet of things* is a good example of a private network where the blockchain technology could be very useful. The Internet of things (IoT) is the inter-networking of physical devices (also referred to as "smart devices"), e.g. vehicles, or buildings, or other items embedded with electronics, software, sensors, and network connectivity which enable these objects to collect and exchange data. The world of IoT is quickly evolving and growing at an exponential rate. Experts estimate that the IoT will consist of about 30 billion objects by 2020.

Concerns have been raised that the Internet of things is being developed rapidly without appropriate consideration of the profound security challenges involved. Most of the technical security issues are similar to those of conventional servers, workstations and smartphones, but the firewall, security update and anti-malware systems used for those are generally unsuitable for the much smaller, less capable, IoT devices. In particular, computer-controlled devices in vehicles such as brakes, engine, locks, etc., have been shown to be vulnerable to attackers who have access to the on-board network. In some cases, vehicle computer systems are Internet-connected, allowing them to be exploited remotely.

Blockchain technology would provide at least a partial solution to these security problems.

1.1. Immutable and redactable blockchains.

The role of hash functions in a blockchain is similar to that of page numbering in a book. There is, however, an important difference. With books, predictable page numbers make it easy to know the order of the pages. If you ripped out all the pages and shuffled them, it would be easy to put them back into the correct order where the story makes sense. With blockchains, each block references the previous (or the next) block, not by the block number, but by the block's hash function (the "fingerprint"), which is smarter than a page number because the fingerprint itself is determined by the contents of the block.

By using a fingerprint instead of a timestamp or a numerical sequence reflecting the block number, one also gets a nice way of validating the data. In any blockchain, you can generate the block fingerprints yourself by using the corresponding hashing algorithm. If the fingerprints are consistent with the data, and the fingerprints join up in a chain, then you can be sure that the blockchain is internally consistent. There are several ways to securely join blocks in a chain. One of the ways is, informally, as follows. Every block B_i has a prefix, which is the hash (or, more generally, a one-way function) of the fingerprint $H(B_{i-1})$ of the previous block B_{i-1}. If anyone wants to meddle with any of the data, they would have to regenerate all the fingerprints from that point forwards and the blockchain will look different.

A blockchain is *immutable* if, once data has been written to a blockchain no one, not even a central authority (e.g. a system administrator), can change it. This provides benefits for audit. As a provider of data you can prove that your data has not been altered, and as a recipient of data you can be sure that the data has not been altered. These benefits are useful for databases of financial transactions, for example.

On the other hand, with a private blockchain, someone with higher privileged access, like a systems administrator, may be able to change the data. So how do we manage the risk of an intruder changing data to his advantage if changing is made easy? The answer to that is provided by *redactable blockchains*; these should involve hash functions with a trapdoor or, more generally, one-way functions with a trapdoor. Trapdoor hash functions are a highly useful cryptographic primitive; in particular, it allows an authorized party to compute a collision with a given hash value, even though the hash function is second pre-image resistant to those who do not know a trapdoor. This property is therefore very useful in application to private blockchains since it makes it possible for an authorized party but not for an intruder to make changes in a blockchain if needed. We give more details in Section 2.

The need for a blockchain (even a public one!) to be redactable is well explained in [5]: "That permanence has been vital in building trust in the decentralized currencies, which are used by millions of people. But it could severely limit blockchain's usefulness in other areas of financial services relied on by billions of people. By clashing with new privacy laws like the "right to be forgotten" and by making it nearly impossible to resolve human error and mischief efficiently, the blockchain's immutability could end up being its own worst enemy."

2. REDACTABLE BLOCKCHAIN STRUCTURE

Recall that a blockchain is *immutable* if, once data has been written to a blockchain no one, not even a central authority, can change it. This is achieved by using a hash function H to "seal" each individual block, i.e., each block B_i has a fingerprint $H(B_i)$, and then connecting blocks in a chain by using another hash function (or just a one-way function) G, as described in our Section 1.1. That way, the blocks B_i become connected in an immutable blockchain because if somebody tampers with one of the blocks and changes it, he will have to change all blocks going forward (or backward), together with their fingerprints, to preserve consistency of the whole blockchain. This is considered logistically infeasible in most real-life scenarios.

Originally, blockchains were created to support the bitcoin network, which is public. Immutability for such a network is crucial. More recently, as we have pointed out in the Introduction, with the idea of the Internet of Things gaining momentum, private networks (small or large) have taken the center stage, and this creates new challenges. In particular, it is desirable, while preserving the tampering detection property, to allow someone with higher privileged access like a systems administrator or another authority to be able to change the data or erase ("forget") it [12]. A blockchain that can be changed like that is called *redactable*.

To make a blockchain redactable, *trapdoor hash functions* are useful. Trapdoor hash functions have been considered before (see e.g. [15]), but having just any trapdoor hash function is not enough to make a blockchain redactable since the authority who wants to change a block B usually wants to change it to a *particular* block B'. A way to make a blockchain redactable was first suggested in [1]. Recently, [4] claimed the first *efficient* redactable public blockchain construction. We also mention *chameleon hash functions* [9] that were recently used [3], [10] in redactable blockchain constructions.

Our approach is focused on private blockchains. It is quite different from [1], [4] and other methods and is simple and easily implementable. Notably, our method can be used in conjunction with any reasonable hash function that is used to build a blockchain. Public immutability of a blockchain (see Section 3.3) in our construction is based on the computational hardness of the RSA problem and not on properties of the underlying hash function. Corruption resistance (see Section 3.4) is based on the computational hardness of the discrete logarithm problem.

2.1. **A particular blockchain structure we use.** There are several possible structures of a redactable blockchain. Our general method should work with any known structure, but to make an exposition as clear as possible we choose a very simple structure as follows. Each block B_i will be in 3 parts: a permanent prefix P_i, the actual content C_i, and a redactable suffix X_i. There is also a hash $h_i = H(P_i,C_i)$, where H is a public hash function, and a public one-way function F such that $F(h_i,X_i) = P_{i+1}$. To make such a blockchain redactable, a central authority should have a private key that would allow for replacing C_i with an arbitrary C_i' of his/her choice, so that upon a suitable selection of the new suffix X_i', the equality $F(h_i',X_i') = P_{i+1}$ would still hold.

Instead of having the prefix of B_{i+1} depend on the block B_i, one can have the prefix of B_{i-1} depend on B_i, in which case the integrity check will have the form $F(h_i,X_i) = P_{i-1}$. Our method, with minor modification, works with this structure just as well.

We emphasize again that the hash function H and the one-way function F should be public since anyone should be able to create a new block in the chain as well as verify the integrity of the blockchain.

3. An RSA-based implementation

A particular implementation of the general redactable blockchain structure described above is inspired by the ideas underlying the well-known RSA encryption scheme [13] (see also [2], [7], [8] for later developments).

Public information:

– a large integer n, which is a product of two large primes

– a hash function H, e. g. SHA-256. (We emphasize again that our method can be used with any reasonable hash function, so the reader can replace SHA-256 here with his/her favorite hash function.)

Private information:

– prime factors of $n = pq$. These p and q should be safe primes (see e.g. [2]), as in modern implementations of RSA. A safe prime is of the form $2r+1$, where r is another prime.

Block structure. In each block B_i, there will be a prefix P_i, the actual content C_i (e. g. a transaction description), and a suffix X_i, which is a nonzero integer modulo n. We also want X_i *not* to have order 2, i.e., $X_i^2 \neq 1 \pmod{n}$. Thus, whoever builds a block B_i, selects X_i at random on integers between 1 and $n-1$ and then checks if $X_i^2 \neq 1 \pmod{n}$. If $X_i^2 = 1 \pmod{n}$, random selection of X_i is repeated. Once a proper X_i is selected, a public hash function H (e. g. SHA-256) is applied to concatenation of P_i and C_i to produce $h_i = H(P_i,C_i)$, and h_i is then converted to an integer d_i modulo n. The prefix P_{i+1} of the next block is then computed as $P_{i+1} = (X_i)^{d_i} \pmod{n}$.

3.1. **Private redactability.** Now suppose the central authority, Alice, who is in possession of the private key, wants to change the content of a block B_i from C_i to C_i' but does not want to change any other block. Then Alice computes the hash $h_i' = H(P_i, C_i')$ and converts it to an integer d_i' modulo n. The number d_i' should be relatively prime to $\phi(n)$, the Euler function of n. If it is not, then Alice should use a padding to have d_i' relatively prime to $\phi(n)$. Once it is, Alice finds the inverse e_i' of d_i' modulo $\phi(n)$. Then she computes $X_i' = P_{i+1}^{e_i'} \pmod n$. The integrity check now gives: $(X_i')^{d_i'} = (P_{i+1}^{e_i'})^{d_i'} = P_{i+1} \pmod n$ because $e_i'd_i' = 1 \pmod{\phi(n)}$ and $(P_{i+1})^{\phi(n)} = 1 \pmod n$.

3.2. **Padding.** Note that since $n = pq$, we have $\phi(n) = (p-1)(q-1)$. If d_i' is not relatively prime to $\phi(n)$, this means that either (1) d_i' is even, or (2) d_i' is odd but is divisible by a large prime (recall that p and q are safe primes, i.e., $\frac{p-1}{2}$ and $\frac{q-1}{2}$ are primes). Thus, our padding is going to be as follows. We will add a random number of, say, 0 bits to the block C_i'. Since the hash function H is assumed to pass all standard statistical tests [14], with probability $\frac{1}{2}$ the result of hashing will be a bit string that converts to an odd integer d_i', and with very high probability this d_i' is also going not to be divisible by $\frac{p-1}{2}$ or $\frac{q-1}{2}$. Thus, after just a few attempts at padding as described above, we will get a d_i' relatively prime to $\phi(n)$.

3.3. **Public immutability.** As can be seen from the "Block structure" paragraph above, a party who would like to change the content of a single block, would have to essentially solve the RSA problem: recover X from n, $X^d \pmod n$, and d, where d is relatively prime to $\phi(n)$. This is considered computationally infeasible for an appropriate choice of n and a random d, $0 < d < n$.

3.4. **Corruption resistance.** Another way of unauthorized modification of a block in a blockchain is *corruption*, i.e., changing the content of the block to something meaningless. To do that, the intruder can start with a random X_i and then look for a number d_i such that $(X_i)^{d_i} = P_{i+1} \pmod n$, for a given $P_{i+1}(\pmod n)$. The mathematical problem that the intruder would have to solve in this case is known as the *discrete logarithm problem* and is considered computationally infeasible if n is sufficiently large.

Two-step attack. If the hash function H used in our blockchain were not preimage-resistant enough, the following "two-step" corruption attack would be possible. Suppose the integrity condition is $P_{i+1} = (X_i)^{d_i} \pmod n$, and suppose the attacker was able to find out that the block B_i was changed so that $(X_i)^{d_i} = (X_i')^{d_i'} \pmod n$, and that the attacker got a hold of X_i, X_i', d_i, and d_i'. We will now omit the index i to make the following easier to read.

The attacker can corrupt this block (i.e., change it to something meaningless) as follows. Generically, $g.c.d.(d, d') = 1$, so we may assume that there are $a, b \in Z_n$ such that $da + d'b = 1$. Then $X = ((X')^a X^b)^{d'}$. Now if $X'' = (X')^a X^b$ and $d'' = d'd$, then $(X'')^{d''} = ((X')^a X^b)^{d'd} = (((X')^a X^b)^{d'})^d = X^d$. Thus, if the attacker can find another suffix, X'', and d'' such that $(X'')^{d''} = X^d$, they can therefore corrupt the block B_i.

The problem is, however, that d'' should be the hash of something, i.e., the attacker will face an additional problem of finding a preimage of d'' under the hash function H.

If preimage-resistance of the hash function H is a concern, the integrity condition is $P_{i+1} = (X_i)^{d_i} \pmod n$ can be replaced by $P_{i+1} = (X_i)^{d_i^2+1} \pmod n$ for an extra layer of security.

4. Other authenticated data structures

A "chain", or a path, is the simplest kind of a connected graph. This type is adequate and sufficient for public data structures such as cryptocurrencies, except that in those, occasional "forks" may exist, in which case the underlying graph is a *tree*. Authenticated data structures built on trees were considered before (see e.g. [11]), albeit not in the context of the present paper (i.e., not in terms of redactability). Here we explain how to make a data structure redactable if the underlying graph has a node of degree greater than 2. The following procedure easily generalizes to an arbitrary underlying graph.

Suppose three blocks B_1, B_2, and B_3 are connected in a chain $B_1 \to B_2 \to B_3$ as usual, but the block B_2 is also connected to another block B', which means that in the underlying graph the node corresponding to the block B_2 has degree 3. Suppose now a central authority wants to modify content of the block B_2. If she follows our procedure from Section 3, she would have to find a suffix X_2' for the block B_2 such that $(X_2')^{d_2'^2+1} = P_3$ and at the same time $(X_2')^{d_2'^2+1} = P'$, where P' is the prefix of the block B'. This system of equations will not have a solution if $P_3 \neq P'$. A way around this is introducing an "intermediate" block B_{in} between B_2 and B'. The prefix of B_{in} will be the same as that of B_3, i.e., equal to $(X_2')^{d_2'^2+1}$. The content of B_{in} can just indicate that this block is intermediate, i.e., does not have any other function. The suffix X_{in} will be selected following the procedure in Section 3, i.e., so that $X_{in}^{d_{in}^2+1} = P_3$.

Acknowledgement. Both authors are grateful to the Hausdorff Research Institute for Mathematics, Bonn for its hospitality during the work on this project.

REFERENCES

1. G. Ateniese, B. Magri, D. Venturi, E. Andrade, *Redactable blockchain – or – rewriting history in bitcoin and friends*, in: 2017 IEEE European Symposium on Security and Privacy, INSPEC Accession Number: 17011479. (See also `https://eprint.iacr.org/2016/757.pdf`)
2. J. Benaloh, *Dense probabilistic encryption*, First Ann. Workshop on Selected Areas in Cryptology, 1994, 120–128.
3. D. Derler, K. Samelin, D. Slamanig, and C. Striecks, *Fine-grained and controlled rewriting in blockchains: chameleon-hashing gone attribute-based*, in: Network and Distributed Systems Security (NDSS 2019).
4. D. Deuber, B. Magri, S. A. K. Thyagarajan, *Redactable blockchain in the permissionless setting*, in: 2019 IEEE Symposium on Security and Privacy, 124–138. (See also `https://arxiv.org/abs/1901.03206`)
5. *Downside of bitcoin: A ledger that can't be corrected*, The New York Times, 2016. `https://tinyurl.com/ydxjlf9e`
6. J. von zur Gathen, I. E. Shparlinski, *Generating safe primes*, J. Math. Cryptol. **7** (2013), 333–365.
7. S. Goldwasser. S. Micali, *Probabilistic encryption*, J. Comput. Syst. Sci. **28** (1984), 270–299.
8. D. Grigoriev, I. Ponomarenko, *Homomorphic public-key cryptosystems and encrypting Boolean circuits*, Appl. Algebra Engrg. Comm. Comput. **17** (2006), 239–255.
9. H. Krawczyk and T. Rabin, *Chameleon signatures*, in: Network and Distributed System Security Symposium (NDSS 2000).
10. S. Krenn, H. C. Pöhls, K. Samelin, D. Slamanig, *Chameleon-hashes with dual long-term trapdoors and their applications*, in: Progress in Cryptology – AFRICACRYPT 2018, Lecture Notes Comp. Sci. **10831**, 11–32.
11. R. C. Merkle, *A digital signature based on a conventional encryption function*, in: Advances in Cryptology – CRYPTO '87. Lecture Notes Comp. Sci. **293** (1987), 369–378.
12. I. Puddu, A. Dmitrienko, S. Capkun, *μchain: How to forget without hard forks*, preprint, https://eprint.iacr.org/2017/106.pdf
13. R. Rivest, A. Shamir, L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM **21** (1978), 120–126.
14. Secure Hash Standard (SHS), `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf`
15. F.-Y. Yang, *Improvement on a trapdoor hash function*, Int. J. Network Security **9** (2009), 17–21.

CNRS, MATHÉMATIQUES, UNIVERSITÉ DE LILLE, 59655, VILLENEUVE D'ASCQ, FRANCE
Email address: `Dmitry.Grigoryev@univ-lille.fr`

DEPARTMENT OF MATHEMATICS, THE CITY COLLEGE OF NEW YORK, NEW YORK, NY 10031
Email address: `shpil@groups.sci.ccny.cuny.edu`