REDACTABLE BLOCKCHAINS AND POLYNOMIAL EQUATIONS

ALEXANDER DEMIN, ALEXEY OVCHINNIKOV, AND VLADIMIR SHPILRAIN

1. Introduction

A crucial part of most artificial intelligence systems is an adaptive network of electronic devices. Perhaps the most popular and quickly proliferating class of such networks is the *Internet of Things*.

The Internet of Things (IoT) represents one of the most significant technologies of this century. It is a natural evolution of the Internet (of computers) to embedded and cyber-physical systems, "things" that, while not obviously computers themselves, nevertheless have computers inside them.

IoT is experiencing exponential growth in research and industry, but it still suffers from privacy and security vulnerabilities. Conventional security and privacy approaches tend to be inapplicable for IoT, mainly due to its decentralized topology and the resource-constraints of the majority of its devices. Blockchains have been recently used to provide security and privacy in peer-to-peer networks and artificial intelligence systems such as, for example, smart cars, smart homes, smart cities, etc.

A blockchain can be either public or private. 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.

Originally, blockchains were created to support the bitcoin network, which is public. Immutability for such a network is crucial. More recently, with the ideas of artificial intelligence gaining momentum, private networks 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 system administrator or another authority to be able to change the data or erase ("forget") it. A blockchain that can be changed like this is called *redactable*. The need for a blockchain (even a public one!) to be redactable is well explained by the following quote from The New York Times [6]: "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." Eliminating redundancies and deleting outdated records is especially vital for networks (such as IoT, for instance) of computationally limited devices.

The present paper proposes a new secure, private, and lightweight architecture for redactable blockchains based on multivariable polynomial equations.

Earlier constructions (e.g. [1, 5, 11]) of redactable blockchains were either not practical, or not quantum-safe, or both. While preserving the simplicity and efficiency of the construction

in [11], we are going to develop new tools for constructing redactable authenticated data structures *with post-quantum security*. One possible way to avoid quantum attacks based on solving the hidden subgroup problem (including the attacks in [15]) is *not* to use one-way functions that utilize, in an essential way, one or another (semi)group structure. In our proposal, inverting the proposed one-way function means solving a polynomial equation (or a system of polynomial equations) in more than one variable. This is presently considered quantum-safe, i.e., there is no known quantum algorithm that could solve this problem efficiently if parameters are chosen wisely.

2. Background

Our approach in this proposal is focused on private blockchains. It is quite different from [1, 5, 14] and other methods and is transparent and easily implementable.

On a high level, our method bares some similarity with the construction in [11] that was based on the computational hardness of the RSA problem and therefore was not quantum-safe, whereas in this paper we aim at quantum-safe constructions. Still, here we first briefly describe the construction from [11] to give a background and put things in perspective.

There are several possible structures of a redactable blockchain. Our general method should work with any known structure, but to make the exposition as clear as possible we choose a very simple structure as follows. Each block B_i will have 3 parts: a permanent prefix P_i, the actual content C_i, and a redactable suffix X_i. There is also a function (e.g. a hash function) $h_i = H(P_i, C_i)$, 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.

An RSA-based construction. Now we briefly sketch the construction from [11].

Public information:

- a large integer n, which is a product of two large primes
- a hash function H, e.g. SHA-512. (We emphasize again that our method can be used with any reasonable hash function, so the reader can replace SHA-512 here with his/her favorite hash function.)

Private information:

- prime factors of $n = pq$. These p and q should be safe primes, as in modern implementations of RSA. A safe prime is of the form $2r + 1$, where r is another prime [9].

Block structure. In each block B_i, there is 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., $X_i^2 \neq 1 \pmod{n}$. This will ensure that X_i does not have a small order since p and q are safe primes, so if the order of an element of the multiplicative group $\mathbb{Z}_{pq}^*$ is not 2, then it is large. 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 $P_{i+1} = (X_i)^{d_i} \pmod n$.

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:

$$(X_i')^{d_i'} = \left(P_{i+1}^{e_i'}\right)^{d_i'} = P_{i+1} \pmod n$$

because $e_i'd_i' = 1 \pmod{\phi(n)}$ and $(P_{i+1})^{\phi(n)} = 1 \pmod n$.

Public immutability. As can be seen from the "Block structure" paragraph, a party who would like to change the content of a single block, must 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. Basic construction

Now we describe an integrity condition for blockchains that would involve quantum-safe one-way functions with trapdoor. In this section, to illustrate the idea, we first describe a basic version of the construction. The basic version is susceptible to several attacks. Later in the text (Section 5), we propose a construction that is resilient to those attacks.

Recall that P_i denotes the prefix of the block B_i. Prefixes are immutable, i.e., they cannot be changed by anybody. We do not go into details here on how this immutability can be arranged; literature on blockchains is awash with that. Then, C_i denotes the content of the block B_i; this is the crucial part of a block where meaningful information is stored. Finally, S_i denotes the suffix of the block B_i; this is an auxiliary part that makes it possible for a central authority to change the content C_i without violating the integrity condition.

We are assuming that all C_i, S_i, and P_i are univariate polynomials of degree d with coefficients in $\mathbb{Z}_n$ for a fixed large prime n. We assume that $d < n$. Needless to say, any content C_i can be encoded this way if n is large enough, say several hundred bits.

A crucial part of any blockchain (redactable or not) is the *integrity condition*. In our case, the integrity condition is of the following form:

$$F(P_i(t_j), C_i(t_j), S_i(t_j)) = P_{i+1}(t_j),\ j = 0, \dots, d,$$

where F is a public polynomial and $t_0, \dots, t_d$ are public distinct integers modulo n.

A trapdoor (i.e., a secret known only to a central authority) is a presentation of the polynomial F in the form

$$F(x, y, z) = G(x, H(x, y) - z),$$

where $G(x, y)$ and $H(x, y)$ are private two-variable polynomials.

To change the content C_i to C_i' while preserving the integrity condition, it is sufficient to find a new suffix S_i' such that $H(P_i, C_i) - S_i = H(P_i, C_i') - S_i'$. Solving this equation for S_i' gives $S_i' = H(P_i, C_i') - H(P_i, C_i) + S_i$.

Thus, whoever knows the function H can change C_i to any other content C_i' without violating the integrity condition. Note that any $\hat{G}, \hat{H}$ that satisfy the integrity condition can be used for this purpose (with $\hat{G}, \hat{H}$ not necessarily the same as the secret G, H).

4. Possible attacks on the basic construction

Now let us look at what an unauthorized party can try to do to modify C_i to some C_i'. There are several possibilities:

Attack 1. Solve the polynomial equation $F(P_i, C_i', X) = P_{i+1}$ for X where X is a polynomial of degree d with unknown coefficients from $\mathbb{Z}_n$. Then this X can be a new suffix of the block B_i. This gives rise to a square polynomial system in the coefficients of X. Solving such a system is a hard problem in general, but perhaps for some special polynomials it may turn out to be easy.

Attack 2. Recover the secret polynomial $H(x, y)$. The intruder can attempt to present $H(x, y)$ as a sum of monomials (with the degree bounded by the degree of the polynomial F) with unknown coefficients. Then the equation $F(P_i, C_i', S_i') = P_{i+1}$ would translate into a system of polynomial equations in these unknown coefficients. These equations are not linear, so solving them (even over $\mathbb{Z}_p$ for a prime p) requires Gröbner basis techniques as in [7, 4].

Attack 3. Find a functional decomposition of $F(x, y, z)$. The algorithm by Kozen and Landau [13] is efficient at finding compositions of univariate polynomials over commutative rings. The algorithm can be used in our setting by considering the multivariate polynomial $F(x, y, z)$ as a univariate polynomial in one of the variables x, y, z. An alternative approach [8], which has polynomial complexity in case the degrees are fixed, is to find a multivariate function decomposition of $F(x, y, z)$. In our case, one could construct the secret $G(x, y), H(x, y)$ of arbitrarily large degrees, hence this algorithm would likely be inefficient.

As the experiments in Section 7 show, the basic construction from Section 3 is susceptible to each of these attacks.

5. Advanced quantum-safe construction

Now we propose a construction (an integrity condition and a trapdoor) that is not vulnerable to previously mentioned attacks.

Let $F(x, y, z)$ be a three-variable polynomial with coefficients given by integers modulo n. Introduce new indeterminates $\bar{A} = (A_0, \ldots, A_d)$, and similarly for $\bar{B}, \bar{C}, \bar{D}$. Let $\hat{P}_i(t), \hat{C}_i(t), \hat{S}_i(t), \hat{P}_{i+1}(t)$ be univariate polynomials of degree d defined as follows:

$$\hat{P}_i(t) = A_0 + \ldots + t^d A_d, \quad \hat{C}_i(t) = B_0 + \ldots + t^d B_d,$$
$$\hat{S}_i(t) = C_0 + \ldots + t^d C_d, \quad \hat{P}_{i+1}(t) = D_0 + \ldots + t^d D_d.$$

Note that if one specializes $\bar{A}, \bar{B}, \bar{C}, \bar{D}$ to some particular numbers, then $\hat{P}_i(t), \hat{C}_i(t), \hat{S}_i(t), \hat{P}_{i+1}(t)$ become concrete prefix, content, and suffix, that is, P_i, C_i, S_i, P_{i+1}. Therefore, an integrity condition on P_i, C_i, S_i, P_{i+1} can be formulated as a condition (equation) on $\bar{A}, \bar{B}, \bar{C}, \bar{D}$.

The advanced integrity condition is a system of (fully expanded) polynomial equations in $\bar{A}, \bar{B}, \bar{C}, \bar{D}$:

$$F(\hat{P}_i(t_0), \hat{C}_i(t_0), \hat{S}_i(t_0)) = \hat{P}_{i+1}(t_0),$$
$$\dots$$
$$F(\hat{P}_i(t_d), \hat{C}_i(t_d), \hat{S}_i(t_d)) = \hat{P}_{i+1}(t_d),$$

where $t_0, \dots, t_d$ are private distinct integers modulo n. To verify the condition for some particular choice of P_i, C_i, S_i, P_{i+1}, one checks if the system obtained by substituting the coefficients in $\bar{A}, \bar{B}, \bar{C}, \bar{D}$ is consistent.

Let us show how the public can find P_{i+1} from P_i, C_i, S_i. Consider the system of equations that defines the integrity condition; the system depends on $\bar{A}, \bar{B}, \bar{C}, \bar{D}$. Recall that $\bar{D}$ are the coefficients of P_{i+1} and note that the system is linear in $\bar{D}$ by construction. Therefore, once $\bar{A}, \bar{B}, \bar{C}$ are specialized to particular values, the values of $\bar{D}$ can be obtained by solving a square linear system with $d+1$ equations.

A trapdoor is similar to that in the basic construction. Specifically, a trapdoor is a presentation of the polynomial $F(x, y, z)$ in the form

$$F(x, y, z) = G(x, H(x, y) - z),$$

where $G(x, y)$ and $H(x, y)$ are private two-variable polynomials. To change the content C_i to C_i' while preserving the integrity condition, it is sufficient to find a new suffix S_i' such that $H(P_i, C_i) - S_i = H(P_i, C_i') - S_i'$. Solving this equation for S_i' gives $S_i' = H(P_i, C_i') - H(P_i, C_i) + S_i$.

Compared to the basic construction in Section 3, in the construction in this section the polynomial $F(x, y, z)$ is private. This makes direct application of Attack 2 and Attack 3 impossible. Additionally, as we shall see, since $F(x, y, z)$ is private, it is not possible to use shortcuts in the computation of Gröbner bases in Attack 1.

Example. Calculations in this example are performed modulo $n = 11587$. Let the private polynomials $G(x, y), H(x, y)$ be

$$G(x, y) = xy^2 + x + y + 3,$$
$$H(x, y) = xy + 5.$$

Then the private $F(x, y, z)$ is

$$F(x, y, z) = x^3y^2 - 2x^2yz + 10x^2y + xy + xz^2 - 10xz + 26x - z + 8.$$

Let $d = 2$. Then $\bar{A} = (A_0, A_1, A_2), \bar{B} = (B_0, B_1, B_2), \bar{C} = (C_0, C_1, C_2), \bar{D} = (D_0, D_1, D_2)$. We therefore have

$$\hat{P}_i(t) = A_0 + A_1t + A_2t^2 \quad \hat{C}_i(t) = B_0 + B_1t + B_2t^2$$
$$\hat{S}_i(t) = C_0 + C_1t + C_2t^2 \quad \hat{P}_{i+1}(t) = D_0 + D_1t + D_2t^2$$

Let $t_0, t_1, t_2 = 1, 2, 3$, respectively. To obtain an integrity condition for the advanced construction, we consider the following system of equations:

$$F(\hat{P}_i(t_0), \hat{C}_i(t_0), \hat{S}_i(t_0)) = \hat{P}_{i+1}(t_0),$$
$$F(\hat{P}_i(t_1), \hat{C}_i(t_1), \hat{S}_i(t_1)) = \hat{P}_{i+1}(t_1),$$
$$F(\hat{P}_i(t_2), \hat{C}_i(t_2), \hat{S}_i(t_2)) = \hat{P}_{i+1}(t_2).$$

By expanding the terms, we obtain equations that depend only on $\bar{A}, \bar{B}, \bar{C}, \bar{D}$; these equations give us the public integrity condition:

$$\begin{aligned}
0 =& A_0^2 + 2A_0A_1 + A_1^2 + 2A_0A_2 + 2A_1A_2 + A_2^2 + 11585A_0C_0 + 11585A_1C_0 + 11585A_2C_0 + \\
& C_0^2 + 11585A_0C_1 + 11585A_1C_1 + 11585A_2C_1 + 2C_0C_1 + C_1^2 + 11585A_0C_2 + 11585A_1C_2 + \\
& 11585A_2C_2 + 2C_0C_2 + 2C_1C_2 + C_2^2 + 11A_0 + 11A_1 + 11A_2 + 11577C_0 + 11577C_1 + 11577C_2 + \\
& 11586D_0 + 11586D_1 + 11586D_2 + 28, \\
0 =& A_0^2 + 4A_0A_1 + 4A_1^2 + 8A_0A_2 + 16A_1A_2 + 16A_2^2 + 11585A_0C_0 + 11583A_1C_0 + 11579A_2C_0 + \\
& C_0^2 + 11583A_0C_1 + 11579A_1C_1 + 11571A_2C_1 + 4C_0C_1 + 4C_1^2 + 11579A_0C_2 + 11571A_1C_2 + \\
& 11555A_2C_2 + 8C_0C_2 + 16C_1C_2 + 16C_2^2 + 11A_0 + 22A_1 + 44A_2 + 11577C_0 + 11567C_1 + 11547C_2 + \\
& 11586D_0 + 11585D_1 + 11583D_2 + 28, \\
0 =& A_0^2 + 6A_0A_1 + 9A_1^2 + 18A_0A_2 + 54A_1A_2 + 81A_2^2 + 11585A_0C_0 + 11581A_1C_0 + 11569A_2C_0 + \\
& C_0^2 + 11581A_0C_1 + 11569A_1C_1 + 11533A_2C_1 + 6C_0C_1 + 9C_1^2 + 11569A_0C_2 + 11533A_1C_2 + \\
& 11425A_2C_2 + 18C_0C_2 + 54C_1C_2 + 81C_2^2 + 11A_0 + 33A_1 + 99A_2 + 11577C_0 + 11557C_1 + 11497C_2 + \\
& 11586D_0 + 11584D_1 + 11578D_2 + 28.
\end{aligned}$$

Now suppose the public would like to check if the integrity condition holds for the following block:

$$\begin{aligned}
P_i(t) &= 6533 + 9560t + 2512t^2 \\
C_i(t) &= 8147 + 6463t + 2498t^2 \\
S_i(t) &= 3507 + 5721t + 3086t^2 \\
P_{i+1}(t) &= 6240 + 10673t + 2184t^2
\end{aligned}$$

To perform the check, one makes the assignment

$$\begin{aligned}
(A_0, A_1, A_2) &\coloneqq (6533, 9560, 2512) \\
(B_0, B_1, B_2) &\coloneqq (8147, 6463, 2498) \\
(C_0, C_1, C_2) &\coloneqq (3507, 5721, 3086) \\
(D_0, D_1, D_2) &\coloneqq (6240, 10673, 2184)
\end{aligned}$$

and checks if the system of equations in the integrity condition is consistent. In this example, all three equations become $0 = 0$, and one concludes that the integrity condition is satisfied for this block.

Using this example, let us show how the public can construct $P_{i+1}(t)$ from the knowledge of $P_i(t), C_i(t), S_i(t)$. To do this, one makes the assignment

$$\begin{aligned}
(A_0, A_1, A_2) &\coloneqq (6533, 9560, 2512) \\
(B_0, B_1, B_2) &\coloneqq (8147, 6463, 2498) \\
(C_0, C_1, C_2) &\coloneqq (3507, 5721, 3086)
\end{aligned}$$

and the system of equations in the integrity condition becomes a linear system in D_2, D_1, D_0:

$$\begin{aligned}
0 &= 11586D_0 + 11586D_1 + 11586D_2 + 7510 \\
0 &= 11586D_0 + 11585D_1 + 11583D_2 + 2142 \\
0 &= 11586D_0 + 11584D_1 + 11578D_2 + 9254.
\end{aligned}$$

Solving this system, one obtains $(D_0, D_1, D_2) = (6240, 10673, 2184)$, as expected.

6. Suggested parameters and block size

Let $d_1, d_2, d_3, d_4 \in \mathbb{N}$. Let

$$G(x,y) = \sum_{i=0}^{d_1} \sum_{j=0}^{d_2} a_{i,j} x^i y^j \quad \text{and} \quad H(x,y) = \sum_{i=0}^{d_3} \sum_{j=0}^{d_4} b_{i,j} x^i y^j,$$

where $a_{i,j}$ are integers modulo n for $i = 0, \ldots, d_1$ and $j = 0, \ldots, d_2$, and $b_{i,j}$ are integers modulo n for $i = 0, \ldots, d_3$ and $j = 0, \ldots, d_4$.

The suggested values of parameters are: n is a prime of size about 20 bits, $d_1 = d_2 = d_3 = d_4 = 2$, $d = 50$. With these values of d_1, d_2, d_3, d_4, d, performing the proposed attacks seems impossible in practice, as we shall see in the experiments, Section 7.

The values of $a_{i,j}$ can be chosen as random nonzero integers modulo n for $i = 0, \ldots, d_1$ and $j = 0, \ldots, d_2$, and similarly for $b_{i,j}$ for $i = 0, \ldots, d_3$ and $j = 0, \ldots, d_4$.

With the suggested parameter values the size of the prefixes, contents, and suffixes, which is roughly equal to $(d+1)\log n$, is about 1000 bits.

7. Experimental results

In this section, we show how the attacks on the basic construction may be successful but turn out to be unsuccessful when used against the advanced quantum-safe construction.

Experiments in this project were performed on Intel© i9-13900 processor with 50 GB of memory using 24 cores. As the main platforms, we used Maple 2024 [2], and two Julia packages: Groebner.jl 0.8.3 [4] and AlgebraicSolving.jl 0.8.2 [3]. We measured memory consumption using the `time` command. The code of our experiments is available on GitHub [10].

7.1. **Attack 1.**

7.1.1. *Attack on the basic construction.* Fix a prime n. Let $d \in \mathbb{N}$, and let $P_i(t), C_i(t), S_i(t), P_{i+1}(t)$ be univariate polynomials of degree d with integer coefficients modulo n. Assume that $d < n$. Let $t_0, \ldots, t_d$ be distinct integers modulo n.

The integrity condition is:

$$F(P_i(t_j), C_i(t_j), S_i(t_j)) = P_{i+1}(t_j), \ j = 0, \ldots, d.$$

To change the content from $C_i(t)$ to $C'_i(t)$, one needs to find a suffix $S'_i(t)$ that satisfies

$$F(P_i(t_j), C'_i(t_j), S'_i(t_j)) = P_{i+1}(t_j), \ j = 0, \ldots, d.$$

One option is to search for a solution in the form of a polynomial with undetermined coefficients, $\hat{S}'_i(t) = \sum_{i=0}^{d} A_i t^i$ (the so-called ansatz polynomial), where $\bar{A} = (A_0, \ldots, A_d)$ are the new indeterminates:

$$F(P_i(t_j), C'_i(t_j), \hat{S}'_i(t_j)) = P_{i+1}(t_j), \ j = 0, \ldots, d.$$

This yields a system of $d+1$ polynomial equations in $d+1$ unknowns.

The computational hardness of solving such a system depends on the choice of secret polynomials. Let $d_1, d_2, d_3, d_4 \in \mathbb{N}$. Let the secret polynomials be

$$G(x,y) = \sum_{i=0}^{d_1} \sum_{j=0}^{d_2} a_{i,j} x^i y^j \quad \text{and} \quad H(x,y) = \sum_{i=0}^{d_3} \sum_{j=0}^{d_4} b_{i,j} x^i y^j,$$

TABLE 1. The memory consumption of a Gröbner basis solver for the square polynomial system for basic construction in Attack 1. OOM stands for out of memory (> 50 GB).

	d					
d_2	4	5	6	7	8	9
3	0.7 GB	0.7 GB	0.8 GB	2.0 GB	10.3 GB	OOM
4	0.7 GB	1.1 GB	6.4 GB	OOM	-	-
5	0.9 GB	5.0 GB	OOM	-	-	-
6	1.8 GB	28.5 GB	-	-	-	-
7	4.6 GB	OOM	-	-	-	-
8	12.3 GB	-	-	-	-	-
9	33.0 GB	-	-	-	-	-
10	OOM	-	-	-	-	-

where $a_{i,j}$ are integers modulo n for $i = 0, \ldots, d_1$ and $j = 0, \ldots, d_2$, and $b_{i,j}$ are integers modulo n for $i = 0, \ldots, d_3$ and $j = 0, \ldots, d_4$.

The degrees d_1, d_3, d_4 affect only the size of the coefficients that are taken modulo n, hence d_1, d_3, d_4 do not affect the computational hardness of solving the system. We fix $d_1 = d_3 = d_4 = 2$. The coefficients of $G(x, y)$ and $H(x, y)$ can be selected from a subset of integers modulo n, which typically produces a zero-dimensional system of polynomial equations.

It appears that the number of solutions of the system (in an algebraic closure of the ground field) is d_2^{d+1}, which coincides with the Bézout bound. The bit complexity of Gröbner bases for zero-dimensional systems is $(D^N)^{O(1)}$, where D is the degree of polynomials and N is the number of variables (see [12] and references therein). By the Bézout theorem, this upper bound is essentially sharp. Since the number of solutions equals the bound in the Bézout theorem, we expect that the above complexity is attained for our systems.

To assess the computational hardness of solving a system of polynomial equations experimentally, we use Gröbner bases to solve particular instances of our system. We fix $n = 11587$ and pick the coefficients of $G(x, y)$ and $H(x, y)$ as random integers in the range from 0 to $n - 1$. In Table 1, we list the memory consumption of Gröbner basis computation for different d_2, d. We do not report the running times since they are qualitatively similar to the memory consumption. We used the `groebner` command in the Julia package Groebner.jl. We observe that incrementing either d_2 or d by 1 roughly doubles the memory consumption.

To test a wider selection of algorithms and implementations, we repeated the experiments using two other software packages: Maple (that, as far as we know, uses the F4 algorithm), and AlgebraicSolving.jl, which has a top-of-the-line implementation of a signature-based algorithm. These experiments led to similar conclusions.

While experiments suggest that solving the above system is hard, a critical issue of the basic construction is the possibility to produce arbitrarily many more equations by specializing the integrity condition at different points. Let $\ell \in \mathbb{N}$ with $\ell < n$, such that $d < \ell$. Let $t_0, \ldots, t_\ell$ be distinct integers modulo n. Consider the following (overdetermined) system:

$$F(P_i(t_j), C_i'(t_j), \hat{S}_i'(t_j)) = P_{i+1}(t_j), \ j = 0, \ldots, \ell.$$

For $\ell = d+1$, in our experiments, such system has a single solution. Taking more equations ($\ell = 2d, 3d, \ldots$) significantly reduces the running time and memory consumption of the Gröbner basis solver, which is to be expected. Thus, in practice, it is possible to solve the system rather efficiently for the basic construction.

7.1.2. *Failure of Attack 1 on the advanced construction.* In case of the construction from Section 5, the polynomial $F(x, y, z)$ is not public. Therefore, it is impossible to produce arbitrarily many polynomial equations, and one must solve a square system, which is infeasible as we have seen before. The computational hardness of solving such a system is illustrated in Table 1. From this table, we expect the memory consumption to roughly double when incrementing d by 1, and, by extrapolation, solving for $d > 30$ would require at least 40 petabytes of memory.

We note that interpolating $F(\hat{P}_i(t), \hat{C}_i(t), \hat{S}_i(t)) - \hat{P}_{i+1}(t)$ as a polynomial in t of degree d is impossible whenever $G(x, y), H(x, y)$ are not linear.

7.2. **Attack 2.**

7.2.1. *Attack on the basic construction.* Fix a prime n. Let $d_1, d_2, d_3, d_4 \in \mathbb{N}$. Let the secret polynomials be

$$G(x, y) = \sum_{i=0}^{d_1} \sum_{j=0}^{d_2} a_{i,j} x^i y^j \quad \text{and} \quad H(x, y) = \sum_{i=0}^{d_3} \sum_{j=0}^{d_4} b_{i,j} x^i y^j,$$

where $a_{i,j}$ are integers modulo n for $i = 0, \ldots, d_1$ and $j = 0, \ldots, d_2$, and $b_{i,j}$ are integers modulo n for $i = 0, \ldots, d_3$ and $j = 0, \ldots, d_4$.

Introduce new indeterminates $A_{i,j}$ for $i = 0, \ldots, d_3$ and $j = 0, \ldots, d_4$ and $B_{i,j}$ for $i = 0, \ldots, d_1$ and $j = 0, \ldots, d_2$. In case d_1, d_2, d_3, d_4 are known, the polynomials $G(x, y)$ and $H(x, y)$ can be recovered by constructing the ansatz polynomials

$$\hat{G}(x, y) = \sum_{i=0}^{d_1} \sum_{j=0}^{d_2} B_{i,j} x^i y^j, \quad \hat{H}(x, y) = \sum_{i=0}^{d_3} \sum_{j=0}^{d_4} A_{i,j} x^i y^j$$

and solving the following system for the undetermined coefficients $A_{i,j}, B_{i,j}$:

$$F(P_i, C_i, S_i) = \hat{G}(P_i, \hat{H}(P_i, C_i) - S_i).$$

The dimension of such systems in all experiments we tried is 1. This can be explained by the following symmetry in the equations: if $G(x, y)$ and $H(x, y)$ satisfy the equality

$$F(x, y, z) = G(x, H(x, y) - z),$$

then $G(x, y - U)$ and $H(x, y) + U$ satisfy it as well for any U. By normalizing one of the secret polynomials (say, $H(x, y)$) to have the trailing coefficient equal to one, the system becomes zero-dimensional in all examples that we have tested.

For example, for $n = 11587$, for $d_1 = d_2 = d_3 = d_4 = 1$, for a particular choice of secret polynomials, one possible instance of the system of equations in the undetermined coefficients is:

$$\begin{aligned}
A_{0,1}B_{0,1} + 842 &= 0 & A_{1,1}B_{1,1} + 2023 &= 0 \\
11586A_{1,1} + 5373 &= 0 & A_{1,1}B_{0,0} + A_{0,1}B_{1,0} + A_{1,0} + 6818 &= 0 \\
11586A_{0,1} + 10712 &= 0 & A_{0,1}B_{0,0} + A_{0,0} + 8216 &= 0 \\
A_{1,1}B_{1,0} + 289 &= 0 & A_{1,1}B_{0,1} + A_{0,1}B_{1,1} + 986 &= 0
\end{aligned}$$

The system contains trivial linear equations that are readily solved, and substituting the solutions of linear equations into other equations would produce new linear equations, at which point the procedure can be repeated. This example system can be solved by such procedure. An issue with the basic construction is that it seems that this iterative procedure can be applied for any $d_1 \geqslant 1, d_2 \geqslant 1, d_3 \geqslant 1, d_4 \geqslant 1$, which makes it straightforward to solve the resulting systems.

7.2.2. *Failure of Attack 2 on the advanced construction.* The attack cannot be applied since an unauthorized party has no access to $F(x, y, z)$.

7.3. **Attack 3.**

7.3.1. *Attack on the basic construction.* Direct application of [13, Algorithm 4] to $F(x, y, z)$ considered as a polynomial in $\mathbb{Q}(x, z)[y]$ (that is, using y as the main variable) succeeds. The output of the algorithm are $G(x, y)$ and $H(x, y)$, possibly normalized to have trailing coefficients equal to zero. As we have seen, having this output is equivalent to having a trapdoor to the construction. The complexity of the algorithm is polynomial in the degree of $F(x, y, z)$ in y, which makes it efficient at breaking our basic construction.

7.3.2. *Failure of Attack 3 on the advanced construction.* In this scenario, the polynomial $F(x, y, z)$ is not available publicly. Therefore, direct application of [13, Algorithm 4] is impossible.

8. Conclusions

We proposed a new quantum-safe and lightweight construction for redactable blockchains based on multivariate polynomial equations.

We studied several possible attacks on our scheme using various Gröbner bases computation techniques and determined that none of them is feasible against our construction.

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. L. Bernardin, P. Chin, P. DeMarco, K. O. Geddes, D. E. G. Hare, K. M. Heal, G. Labahn, J. P. May, J. McCarron, M. B. Monagan, D. Ohashi, S. M. Vorkoetter, *Maple Programming Guide*, Maplesoft, a division of Waterloo Maple Inc., 1996-2023.
3. J. Berthomieu, C. Eder, M. Safey El Din, *msolve: A Library for Solving Polynomial Systems*, Proceedings of the 46th International Symposium on Symbolic and Algebraic Computation (2021), 51–58.
4. A. Demin, S. Gowda, *Groebner.jl: A package for Gröbner bases computations in Julia*, preprint (2024), `https://arxiv.org/abs/2304.06935`.
5. 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`)
6. *Downside of bitcoin: A ledger that can't be corrected*, The New York Times, 2016. `https://tinyurl.com/ydxjlf9e`
7. J.-C. Faugére, *A new efficient algorithm for computing Gröbner bases (F4)*, J. Pure Appl. Algebra, **139** (1999) 61–88.
8. J.-C. Faugére, *An efficient algorithm for decomposing multivariate polynomials and its applications to cryptography*, J. Symb. Comput. **44** 1676–1689
9. J. von zur Gathen, I. E. Shparlinski, *Generating safe primes*, J. Math. Cryptol. **7** (2013), 333–365.

10. GitHub repository, `https://github.com/sumiya11/Redactable_blockchains_and_polynomial_equations`.
11. D. Grigoriev and V. Shpilrain, *RSA and redactable blockchains*, Int. J. Computer Math.: Computer Systems Theory **6** (2021), 1–6.
12. A. Hashemi and D. Lazard, *Sharper Complexity Bounds for Zero-dimensional Gröbner Bases and Polynomial System Solving*, International Journal of Algebra and Computation **21** (2011) 703–713.
13. D. Kozen, S. Landau, *Polynomial decomposition algorithms*, J. Symb. Comput. **7** (1989), 445–456
14. I. Puddu, A. Dmitrienko, S. Capkun, *μchain: How to forget without hard forks*, preprint, https://eprint.iacr.org/2017/106.pdf
15. P. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput. **26** (1997), 1484–1509.

Max Planck Institute of Molecular Cell Biology and Genetics, Pfotenhauerstrasse 108, 01307 Dresden, Germany
Email address: `demin@mpi-cbg.de`

Department of Mathematics, Queens College, City University of New York, Queens, NY 11367
Email address: `alexey.ovchinnikov@qc.cuny.edu`

Department of Mathematics, The City College of New York, New York, NY 10031
Email address: `shpilrain@yahoo.com`