Ach dieses Wahldings?[*]

DaveJakeNic

May 26, 2022

Abstract

The bitcoin white paper on arXiv changed the world. There was no care to follow academic procedures because it was clear, from today's perspective, that such system would eventually emerge and gain wide acceptance and popularity. But actually, the technology used therein was entirely old and unoriginal. Hence, why did Bitcoin emerge so lately? The reason is that the right idea has to traverse all the way to the engineer who is capable of implementing it. And the idea has to gain attention. This is the same for the method presented here. This is the only reason why we even care to present it as a paper.

The method presented here is entirely trivial. Yet, once implemented, it will likely end all global unfairness forever over time.

Contents

[*]That's what she said

1 Introduction

1.1 Motivation

There can be no algorithm for truth.

Truth is a model of interpretation for information because there is no such thing as truth. We cannot even tell whether there are more than three spatial dimensions. Truth is what each individual considers true. Truth is what they believe.

Imagine a system that anonymously but verifiably reflects just what each person thinks. This could be used to end every problem of today: Misinformation, confirmation bias, ignorance of public opinion, dictatorships, and all forms of inequality, thus eventually also implying the end of existence for money as we know it today. In the author's opinion, it will result in a situation where every human has exactly the same power. Imagine what that means!

1.2 Practical Meaning

The result of bitcoin for the public is that you have an app on your phone and it shows you an account value. You can use the account value to acquire and distribute money. I don't want to digress on what money is, though.

The result of this paper's algorithm is that you will have an app on your phone where you can ask questions and anonymously answer other people's questions. The catch is, however, that every human can only submit one single answer per question –despite anonymity.[1] The purpose is then that everyone can see all questions and all answers that had ever been asked (in visually appealing statistical diagrams; or via an exportable format that they can individually analyze with their own favorite tool). In short: Everyone holds a mirror of accurate and unforgeable knowledge of what the people think. You no longer hold media and claims and news in your hand. You hold truth. It is difficult to grasp, like google was, before actually appreciating what it means.

1.3 How to Implement?

The way how the system works must be similar to bitcoin: A decentral cryto-graphic mechanism that prevents any fraud or outage unless a huge fraction of participants wishes to break it. For bitcoin, this fraction is 50% of all hashing power. For my voting algorithm presented, this is 100% of all users.

For bitcoin, there is now a wide public understanding how it works: combine blockchain and hashcash. For voting in contrast, the challenge of crafting a cryptosecure algorithm seems significantly less trivial at first sight.

Regardless, in this paper, we demonstrate how the combination of two simple concepts achieves the intended mechanism.

2 Problem Statement

With an algorithm for voting we mean that there is a specified pool of accounts, called *vote pool*. In a vote pool, each account belongs to precisely one distinct real person. The creation of a vote pool is already a challenge by itself. Solutions to generating a vote pool will be presented later in this paper.

Once a pool of respective accounts has been established, we consider the application of an ordinary question.

Definition 2.1 (Ordinary Question)**.** *An ordinary question is a scenario where one account in the vote pool (called host) will (non-anonymously) ask a question and select a subset of his/her/their vote pool to answer it. Once posted, every account from that subset (called voter) can submit exactly one anonymous answer (called vote) to the host's question.*

Example 2.2. *There is a vote pool of Grace, Anton, Bobby, and Carlo. Olive is outside. Grace will ask for a vote. She invites Anton, Bobby, and Carlo. They engage. She asks a question. There are at most three answers. Olive and everyone else can check that each answer belongs to one distinct voter between Anton, Bobby, and Carlo*[2]*.*

[1]Some crypto-experts claim that exactly this would be technically impossible.

[2]actually, each voter can anonymously pass their right to vote to Grace, but we come to that only at the very end.

We will later explain why the scenario of referendums (i.e., questions that everyone has a right to answer) can be reduced into ordinary questions.

3 Challenges

An algorithm for voting is an algorithm for conducting an ordinary question. It must achieve the following tasks:

1. Authenticity: Each vote belongs to one account.
2. Anonymity: No vote can be traced back to a voter. This must not be based on encryption but on random chance. Thereby, anonymity is immune to any hacking attempts.
3. Fairness: Each voter can submit only one vote.
4. Verifiability: Every human on earth can verify for eternity the authenticity and fairness of each vote in each voting.
5. Security: The integrity of the cryptographic system prevails regardless of whatever the participants do.
6. Unforgeability: Nobody can vote twice or change a vote or voting question in hindsight.
7. Decentrality: No-one can shut down the system or prevent a given question from being asked or prevent any vote from being submitted.
8. Independence: The system is being run (including all implied costs) from all those who want to participate in a voting. It is non-government, non-profit, non-commercial, and open-source.
9. Stability: There is no need for multiple systems of that kind because the system as crafted is sufficient to yield exactly and only and fully its intended purpose. There is no intended maintenance. There is also no money involvable, so there is no point to craft redundant systems (e.g., in contrast to bitcoin).

4 Overview

The two mechanisms at heart of our algorithm are as follows:

1. Solving the oracle problem: A method to create a pool of $m \in \mathbb{N}$ verified accounts. This method will be used to give each vote-eligible human precisely one account. Notice that such an account cannot be anonymous.

2. Solving the anonymous authentication problem: A method to launder[3] $m \in \mathbb{N}$ verified accounts into $m \in \mathbb{N}$ anonymous accounts, each of which can be accessed from precisely one of the aforementioned verified accounts. This method will be used to give each eligible voter exactly one anonymous voice.

We will first present the second method (i.e., the authentication anonymisation procedure) and then the first.

5 The Authentication Anonymisation Procedure

We dive straight into the algorithm. It consists of several phases: invite, engage, spread, answer, reveal, and verify. In each phase, people broadcast messages over the internet. A blockchain can be used to store all messages centrally but we explain later why this is not quite necessary.

5.1 Invite

The host (supposed female for the purpose of this example) will send a signed message where she names all those voters (supposed male for the purpose of this example) whom she asks to participate. In that message she also provides a temporary public RSA[4] key.

5.2 Engage

Each named voter, who wants to participate, will reply with a signed message where they confirm their interest to vote. Each voter's message also contains a temporary public RSA key.

5.3 Spread

The host will create a *recipe* at random, compute its hash, and publish the hash in a signed message.

Definition 5.1 (Recipe)**.** *A recipe has four parameters: $m \in \mathbb{N}$ number of voters, $n \in \mathbb{N}$ list-length, $k \in \mathbb{N}$ number of fakes, and $L \in \mathbb{N}$ bitsize per number.*

A recipe holds several lists. Each list has length n. One of the lists is called real list*. It contains random integers of L bits.*

For each of the m voters, there is a subrecipe.

Each subrecipe is created at random as well.

[3]One could use my algorithm for a bitcoin protocol with perfect money laundering built in.

[4]We suppose RSA for simplicity. It does not matter whether it is RSA or ECC.

Definition 5.2 (Subrecipe). *A subrecipe holds the respective voter's name and is personalized per voter.*

A subrecipe contains k fake lists. These are lists that have length n and are filled with random integers of L bits. Also, the subrecipe contains one index list *that defines a reordering of the real list. Finally, the subrecipe contains an index* $\ell \in [0, k] \cap \mathbb{N}$.

We also need the concept of a challenge.

Definition 5.3 (Challenge). *A challenge is a concealed number* $x \in \mathbb{N}$ *that takes some amount of time to reveal. The difficulty to reveal is controlled with a parameter* $Q \in \mathbb{N}$.

Example 5.4. *For example, take x, fill it with sufficiently many zero bits at the front, and AES-encrypt it with a key that is all zeros, except for the last* $Q \in \mathbb{N}$ *bits that are random. A challenge can be crafted (*$y =$`make_challenge(`x, Q`))` *in very little amount of time. But the time for decryption (*$x =$`solve_challenge(`y, Q`))` *can be controlled via the difficulty parameter Q.*

Using the recipe and a voter's temporary public RSA key, the host will send one personalized encrypted *package* with her signature to each voter who engaged.

Definition 5.5 (Package). *A package holds the respective voter's name and is personalized per voter.*

A package holds $k + 1$ *lists. The* ℓ^{th} *of these lists is the (voter-specific) permutation of the real list, as given per his subrecipe. The other lists are the fake lists.*

Finally, each number in each list in the package is concealed with a challenge. And any labels (e.g., "real list" or "fake list") are removed within the package.

5.4 Answer

Each voter will generate $k + 1$ RSA key pairs r_j, $j = 0, \dots, k$, at random. He (the voter) will then select one concealed random number from each of the $k+1$ lists in his package and reveal it by cracking the challenge. This way, they obtain $k + 1$ numbers x_j, $j = 0, \dots, k$. They will now compose $k + 1$ *pairings* from x_j, r_j for $j = 0, \dots, k$.

Definition 5.6 (Pairing). *A pairing is generated from a number x and a key pair r.*

The pairing consists of three things.

1. *An index number* $\hat{\imath}$*, to efficiently find a pairing from a large unsorted list of pairings.*
2. *A random salt* $s \in \mathbb{N}$

3. *An encryption of $x+s$ with the host's public RSA key.*

4. *the public key $\tilde{r}$ of r.*

The voter will anonymously post each pairing at a different random time in a random order via TOR browser, together with a hashcash proof-of-work of difficulty D, on the public message board. The hashcash merely acts to prevent DoS attacks against the message board. Basically, any message must be provided either with signature or hashcash.

5.5 Reveal

After some deadline, the host and each voter will reveal their temporary private keys. Thus, from this point, all the data that has been sent thus far can be read by everyone. Also, the host will now publish a signed message with her recipe.

The next step can be done by anyone. Nonetheless, the host is obliged to encrypt all the numbers x from all posted pairings. The host will then write a signed message with subject "these are the official vote keys". In this message, she lists all those public keys $\tilde{r}$ with indices $\hat{\imath}$, which stand in pairings with numbers x that live in the real list. For example: Suppose some anonymous voter published a pairing with $x=3$, $\hat{\imath}=1$, and $\tilde{r}=\{5,7\}$; and suppose the real list contains the number 3; then Grace writes $\hat{\imath}=1$, $\tilde{r}=\{5,7\}$ into her list of official vote keys.

Here ends the procedure. Now, the host can reveal to everyone what she actually wanted to ask. Each voter holds a secret RSA key in r_j that belongs to one of the public keys $\tilde{r}$ and identifiers $\hat{\imath}$. He (the voter) can thus now use his $\tilde{r}$ and $\hat{\imath}$ to anonymously authenticate any vote that he submits to this question. If the public notices that the same key has been used to sign two votes then the second vote is simply ignored.

5.6 Verify

Anyone who is interested in verifying the result of the procedure (e.g. Olive) should to the following:

1. Check all signatures of all signed messages. Ignore all messages with invalid signature. Anyone who failed to submit a mandatory message (invite, engage, spread, answer, reveal) is excluded. For instance, if Anton failed to engage then return *Anton banned.*

2. Compute the hash of the recipe and compare it to the signed hash in the host's signed spread message. If the hashes do not match then return *Grace resign.*

3. Decrypt all pairings to get all submitted numbers x. Then, for the real list and each fake list, count the occurances of pairing elements x in each list.

4. Assert that at most 1 element per fake list has been submitted in a pairing. Otherwise, the respective voter is banned. For instance, suppose Anton and Carlo have at least one fake list with more than one submitted number. Then return *Anton banned, Carlo banned*.

5. If no voter was to be banned but there are more than n (number of voters who engaged) pairing elements x of the real list, then return *Grace resign*.

We will explain in the security analysis sections below what each step implies.

Everytime there is a banning, the host will repeat the authentication anonymisation procedure with only the remaining voters until no-one is banned. Everytime the host resigns, she will not be able to host a question for a certain amount of time.

6 Analysis

6.1 Cost and Parameters

The method parameters are n, m, k, L, Q, D (list length, number of voters, number of fake lists, number of bits per random number, difficulty of challenge, difficulty of hashcash). We ignore D because of its isolated single purpose.

The algorithm has complexity $\mathcal{O}(n \cdot m \cdot k \cdot L \cdot Q + k \cdot D)$ or less; depending on whether you are the host or the voter and how naive certain algorithms are implemented on a respective user's system (searching an item in a list, etc).

6.2 Security: Role of number k of Fake Lists

The k fake lists shall ensure that each voter will indeed only submit one pairing per list. Otherwise, he (the voter) could submit two (or more) keys $\tilde{r}$, which would enable him to submit two (or more) votes. This is prevented by the fake lists. For each additional pairing, there is a random chance of $\frac{k}{k+1}$ that his pairing has a number x from his fake lists. Thereby, he would be detected and banned.

However, instead of the voter, it could also be that the host impersonated the voter and submitted pairings with fake x of a certain voter. We then say the host *bullied* the voter. While this can happen, it has no effect on the result, because recall that the host can decide who participates in an ordinary question anyways. So she can ban the voter upfront or only in the end (and then has to redo the entire procedure with the recipe and so on).

6.3 Security: Role of the Recipe

The recipe, and especially the signed hash of the recipe, ensure that the host cannot forge the packages. This means that any fraudulent attempt of the host would immediately be noticed whenever someone would check the recipe and its hash.

Less clear on first glance, the hash also prevents the host from conspiring: If the host shares secret data from the recipe with any voter or group of voters then these voters can expose the host by submitting many pairings with numbers x from the real list. This way, the fourth step in the verification procedure will trigger and force Grace to resign.

6.4 Security: Brute-Force Attacks

We now approach the subject of brute-force attacks. In these attacks, at least two voters must conspire (i.e., share their packages) and use computational power to identify the real list. Their brute-force attack would be in vain for double voting but could force that the host into resignation.

Thereby, a brute-force attack could prevent the voting and must thus the prevented. We explain how this prevention works.

6.4.1 Role of the Challenges of Difficulty Q

The challenges conceal each number differently. (I.e., Even the same number has a different seal in each package because it uses a different random AES key.) If each number was in plain sight then conspiring voters could compare all their lists to find the two lists that hold the same elements. They would have to crack sufficiently many challenges before they will be so lucky to find two identical numbers in any of their different lists.

A trade-off for Q can be found such that a normal voter has an insignificant amount of work but a group of conspiring voters have a significant amount of work. This can be achieved because each normal voter will only need to crack one number per list whereas the conspiring voters have to crack many numbers per list. (We get into the brute force complexity analysis below.)

6.4.2 Role of the Number of Bits L

A reasonable number of $L = 256$ bits already makes it entirely impossible to guess a random number correctly. It also makes it unconsiderably unlikely that any two fake lists (or the real list) contain the same number. We therefor do not consider the subject of L with rigor.

6.4.3 Role of List Length n

This is the least trivial aspect. As known, in a class with 23 students, there is a probablity of more than 40% (> 0.4) that none of them have birthday at the same day. In general, if there were m students in a class and the year had n days then the probability of no-one having birthday at the same day is at least

$$P_{m,n} := \exp\left(-\frac{m^2}{n}\right) .$$

If m grows (more students) then the probability decreases; but if the number n of days per year increases then the probability grows.

To see this, notice that the probability $\alpha_{k,n}$ of k people not having the same one in n birthdays is

$$\alpha_{1,n} = 1\,, \qquad \alpha_{2,n} = \alpha_{1,n} \cdot \frac{n-1}{n}\,, \ldots \quad \alpha_{k,n} = \alpha_{k-1,n} \cdot \frac{n-k+1}{n}\,.$$

Notice that we can bound $\alpha_{m,n} \geq \left(\frac{n-m}{n}\right)^m > P_{m,n}$ via explicit Euler method.

This shows the following: If $k = 1$ and two voters A and B conspire then they have to crack $\mathcal{O}(m) = \mathcal{O}(\sqrt{n})$ numbers in order to find two identical elements in a list of A and B with a probability that is bounded below by a constant. If instead multiple voters conspire then their best strategy is still to only crack lists from A and B. Further, if $k > 1$ then the problem gets only harder.

In conclusion, $\mathcal{O}(m)$ consipiring voters have at least $\Omega(\sqrt{n}/m \cdot Q)$ amount of work for solving challenges whereas any non-conspiring voter has only $\mathcal{O}(k \cdot Q)$. We hence see that by merely selecting $n = \text{const} \cdot m^2$ for sufficiently large constant we can adequately counter the chances of any brute-force attack.

Finally, consider that a brute-force attack will merely prevent one attempt of the authentication anonymisation procedure. And we could run this procedure a thousands times and only pick one result that worked. This would mean a devastating workload for any brute-force attempt. Thus, there is very little incentive and very huge cost for any brute-force attack.

6.5 Reductions of Referendums into Ordinary Questions

We have seen how a single individual in the pool can host a voting where they hold the domiciliary rights, so to say. That means the host can invite and ban voters at her own discretion. In a referendum, however, everyone must be able to participate in a voting. To realize this, a referendum will be hosted in turn by every pool member who wants to host it. The sums of all votes divided by the number of turns are then considered as the total number of answers to this referendum question.

Tow questions arise:

1. Will anonymity be compromised? Answer: Practically no, but to be sure, when m pool members host the same ordinary question redundantly then just a false answer every now and then. This will not significantly swing the outcome of the referendum but fully protects your vote's anonymity.

2. Will it be fair? Answer: Yes. You may guess that because hosts can systematically ban voters, they could use it to silence your vote. But consider that you can ban them as well. Hence, even if everyone knows what everyone else will vote and they ban each other systematically then it will still not change the outcome of the referendum.

6.6 Final Remarks

If an invited voter engages but does not submit a pairing with an x of the real number then the host can occupy this voter's vote. This is so because she knows the real list and can submit a pairing if this voter does not. And she can decrypt the pairings as they come in.

Also, we do not really need a blockchain unless for referendums. So see this, notice that the ordinary question is hosted by Grace along. So it suffices if everyone sends messages directly to her and she broadcasts to Anton, Bobby, and Carlo. Only eventually she has to provide a copy of all messages to anyone (like Olive) who wants to verify.

7 Solving the Oracle Problem

In the past sections we saw how members in a given vote pool can conduct ordinary questions and referendum questions. But we have not discussed yet how the members can form a vote pool in the first place.

In principle, the need for a vote pool is that every person can send a signed message. Their signature is to authenticate themselves as eligible voters. Otherwise, someone could make several accounts and thereby double-vote. Thus, the only purpose of a voting pool is to create an account system in which each real human has exactly one account.

7.1 Engineering Solution

We could use passports. That is, when you go to the passport office to refresh your passport, instead of your signature you write a public RSA key onto your passport's signature field. Then, that RSA key can be used for signing your digital messages (, supposing that the office maintains a public ledger of all public keys).

However, the issue with these passports is that they are issued by governments. A government is a centralized organ which might not be trustworthy: They may accidentally or maliciously refuse to issue a passport to someone or issue multiple identities to the same biological person. Thus, it could be advantageous to build a decentralized system to form the vote pool.

7.2 Decentralized Solution

The following solution is less user-friendly but more secure. Just as with passports, we can imagine that finger scan or iris scans can intimidate users or violate privacy rights. Nonetheless, the purpose of a photograph or anything on the passport is to describe the person to whom it belongs. And if we craft a voting pool, it really means we craft a passport system. Thus, our pool is as weak as the passport. A strong passport might necessitate a certain degree of privacy invasion.

Our passport, like any passport, consists of four things:

- Singleton: This refers to a unique measurement of your biophysical system. Governmental passports typically use biometric face images.
- Signature: This refers to something that only you can generate. Governmental passports typically use your hand-writing.
- Approval: This refers to something that authenticates the originality of your passport. Governmental passports typically use different sorts of watermarks.
- A contact info: This is something that can be called to get in touch with you.

If we are only one voting pool and we want to avoid the approval from a centralistic organ then obviously we must approve ourselves. This will add the burden onto us that we have to ensure that no-one gets hold of multiple identities (a.k.a., multiple passports for the same biophysical human). The way for us to achieve this with the least effort is by using a strong singleton.

7.2.1 Role of the Singleton

"Singleton" is a term in computer science that has been adapted from the notion of something that exists only once.

As is known, some people look identical. Thus, biometric photos are weak singletons. Fingerprints are unique but two print samples from the same finger may differ. This makes it tedious to use. A reasonable choice for singleton could be the iris scan encoding[5]. This encoding[6] will be exactly the same at every occasion where the same human is scanned. Also, no two humans have the same iris scan encoding[7].

Therefor, if we made a list of iris scan encodings for each person in the voting pool then we could quickly identify if anyone tried to register twice (because then there are two identical iris encodings in the list).

7.2.2 Role of the Signature

The purpose of a signature is to prove that something originated from you. There is knowledge how RSA key pairs can be used for digital signatures. Since we envision a digital system, it is clear that digital signatures are to be used.

[5]Actually, it could be sha256(iris_scan_code+salt), thus no-one is deprived of their iris code.

[6]of both eyes

[7]Of course some people do not have eyes and then we have to re-iterate on what could be a suitable singleton for everyone. So there have to be practical compromises.

7.2.3 How to do the Approval?

We use a web of trust. That is a network where each person (she for the purpose of this example) knows a few people around her. She will meet[8] with each of them and they measure each other's iris scan. If the encoding matches with their passports, they will both use their digital signature to approve each other's passport. If a scan encoding differs from the passport then she will publish a signed message of the iris scan that she measured.

She can then host a voting with all the people whom she wants to vote on whether this person shall be abandoned from the voting pool. This could be used within a consensus mechanism to consolidate a vote pool.

8 Alternative Uses

8.1 Money Laundry and Airplane Laundry

Money laundry is associated with corruption when it could also mean instead that malicious actors trace you via your money that they falsely believe belongs to them. Or imagine Elon Musk wants to swap his private jet in a pool of m billionaires who each own one private jet of equal value.

We briefly explain how this would work by using the example of bitcoin. We suppose an augmentation of the protocol that foresees a money laundering operation every once in a week. We will describe this matter only superficially.

Consider $\tilde{m} \leq m$ account holders who hold a total of m accounts, each filled with one bitcoin. Once per week, each account holder will host one authentication anonymisation procedure. Resigned hosts will be ignored. Among the remaining procedure results, those with the largest participation (i.e., least bans) will be considered as candidates. The winning candidate is agreed upon via consensus and receives some crypto reward. The winning host's authentication anonymisation procedure generates m new anonymous accounts, each with one bitcoin, to which the original $\tilde{m}$ account holders each hold a proportionate amount of log-in keys.

8.2 Vote Coin

Did you ever feel disadvantaged because you live in a country with expensive electricity or because you have no hashing power? Did you ever want to be a crypto millionaire? Here is your chance! Here comes *Vote Coin.*

Build a vote pool for real in the real world. Meet people. Learn iris fetishism. And finally, replace proof-of-work consensus via proof-of-amount-of-iris consensus.

Jokes aside; I hold my hopes up that individual's greed for money can help establish the equality-promoting system that is presented in this paper. This should hold as well for money laundering to implement the authentication

[8]They get in touch via the contact info.

anonymisation, as well as for the vote coin to implement a decentralized vote pool.

9 Complementary Material

There is a github website:

`https://github.com/davejakenic/Free-the-World-Algorithm`

On that website

- The readme.md on the website describes the algorithm once more (in a just slightly less up-to-date form).
- We give a video link to a presentation. You can use it for whatever purpose you have in mind.
- We give you the slides so you can use them in whatever way you like. For instance, we encourage you to hold the presentation at a conference.
- We give you a toy implementation of the authentication anonymisation procedure on the website. It is slow because I wanted a code without any dependencies. It is in C++ and internally runs scripts of Python 3.10.4 .
- On the website you also find the latex code of this paper. Feel free to submit it to any journal under your real of fake name in any deviation from the original form. Really, just get it out.

10 Final Remarks

Thanks for reading.

If you want to help making it big then you can use all of the material on the website to spread the message. I do not own some of the presentation images, though. So maybe there are legal implications unless it is clearly for scientific or educational purpose.