PeaceVote

Janis Erdmanis

June 1, 2020

1 Abstract

The electronic voting problem had been studied extensively for decades. However, no technology had achieved the same of trustworthiness as a regular paper ballot which is the only robust software-independent voting system capable of preserving participant privacy. A software independent, voting system designs which preserve privacy are possible if shuffling of votes are decentralized, but such a system would be extremely fragile and unpractical. In this paper, I propose identity braiding as a method for eliminating fragility of mixnet cascades for electronic voting applications and discuss a specific implementation of electronic voting for the PeaceVote electronic voting system.

2 Introduction

In a regular paper ballot, everyone can see that officials are acting according to agreed and fair rules as are also able to become the part of them without particular expert knowledge. On the contrary electronic voting, schemes suffer from openness and transparency, leaving doubt on whether the election officials are running the software that they claim to do and are not producing fake proofs to support their claims. That made Ron Rivest introduce a term software independence - a software is said to be software independent if election outcome is independent of undetected modifications to the software which runs the elections [Rivest].

It is rather easy to design an electronic voting system which is software-independent but does not preserve privacy. In a simple design, each citizen would own state-certified public/private key pair which forms a digital identity of the person. To run the elections, the state would publish possible option messages. It would collect option certificates (signed documents) from the voters and publish them to for everyone accessible bulletin board which everyone could use for verifying and counting the votes.

Preserving voters anonymity in an independent software system is hard, and all known systems make compromises. There are two issues with combining software independence and privacy. To achieve software independence, the data making a conclusive proof for accounting components of the system must be collected and published. This data shall contain votes for everyone with cryptographical links that it was produced during the elections by a legitimate voter. At the same time, the identity of the voter casting a particular vote shall remain anonymous. We shall call this transparency-anonymity issue.

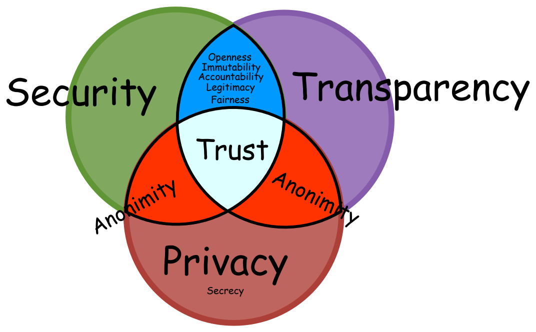

Figure 1: Fundamentals of building trust in election results.

The second issue comes from the execution of the protocol itself. Although the published data preserves anonymity and proves that the system had been software-independent, it is still possible that adversary had recorded additional information which is useful for revealing the identity of the voter. We shall denote that as the security-anonymity issue, as it is possible because the voting machine (or person) has malicious code on it.

This reasoning allows us to split software-independence into two logical components. First one is security with which we shall understand machines (including humans) which execute the election software, collects votes and produces data. With transparency, we shall understand all data forming a conclusive proof that the system had been software-independent by producing election outcome and applies software independence recursively to the proof itself. So the trust is a combination of security, transparency and privacy, which we can visualize in a Venn diagram as shown figure...

The main difference between a paper ballot and mainstream electronic voting solutions is by a number of machines (including humans) adversary needs to hack to compromise the proofs security-transparency (some policing officials in the paper ballot system) and to compromise the anonymity-security (an official collaborating with briber). A private company can be isolated from society (NDA agreements, individual selection, etc.), and thus, it makes it easier to hack. On the other hand, hacking software all election officials who with their social circles cover all society is impossible without being obvious. This leads us to an idea that a secure election system must be open to participation so it could run synchronically on multiple machines at once tied to social circles which can cover all society.

Most of the voting systems on the literature consider a rather peculiar adversary model. When the voting protocol is modelled as a process, certain parts of the protocol are trusted for verifiability. So voter could get a casting assurance (individual verifiability), and able to validate that only eligible voters had participated, they had voted once, and that the votes had been counted accurately (universal verifiability) and making it end to end verifiable, he/she still would need to trust the verifying system.

A similar situation is present for ensuring the anonymity of the vote, when the anonymity preserving system, for example, a mixnet cascade is often assumed to be held private to avoid malicious activities which can result in the absence of election results. Nevertheless, if we need to trust private mixes, the system nevertheless of the protocol centralised although a more disperse. Additional data revealing anonymity can be recorded (as long as authentification is part of a private system) and the organisation can generate necessary proofs of fair elections. Can we avoid those issues in the electronic voting system design?

The success of TOR shows that decentralised anonymous network of mixnet cascades can exist allowing to resist government censorship and hiding identity as Edward Snowden from the USA. Other approaches based on mixnets, crowds, onion routing, dc nets are actively developed and deployed [...]. When evaluating the anonymity of such systems, one also should take a realistic point of view of the adversary who would not be able to control every ISP of the world anonymize the traffic. Thus with the delocalisation and companionship, we can already today use quite strong anonymous sockets.

However, anonymous sockets yet do not allow to avoid tracking. Unless the electronic voting is an anonymous pool where a voter can vote multiple times, the election voting system is going to have authentification. An attractive solution would be to use linkable ring signature scheme, sign the votes with that and deliver to the vote recorder with an anonymous socket. Unfortunately, linkable ring signatures suffer from their length and computational requirements and thus are not practical. Therefore for a sound system, we shall focus anonymisation of pseudonyms which can be used for authentification of voting messages. Anonymisation of pseudonyms has been studied lately with the context of blockchain and smart vehicle networks VANET. For blockchain coin shuffle had implemented as additional service which shuffles the transactions making them untraceable. In vehicular networks, the authority would be a benefit to know information on infrastructure quality for further improvement, and vehicular networks seem like a perfect solution where to protect privacy mix zones for changing pseudonyms is currently actively studied. However, the proposed schemes do suffer from the necessity to trust the mixing authority and sometimes also a necessity to trust other participants in the mix. Also, the anonymity of those schemes is of a concern since network anonymisation is often disregarded.

In this paper, I will propose a new pseudonym anonymisation scheme using only public-key cryptography which is suitable as substitutes for linkable ring signatures. Then I will show how to use them for electronic voting when simultaneously multiple proposals are voted on with PeaceVote system. Furthermore, I will touch on the issue of testing of the system for small communities, distributing software in a trusted manner while not being dependent on usual SSL certificates as well as how members can be added easily without making the system look complex for the voters. I would also argue that the resulting voting system is strongly software-independent due to the fact the anonymisation of pseudonyms is done in small batches and is strongly anonymous because of openness for participation.

3 Anonimous pseudonyms

Imagine an ordinary election. You go to pooling station; greet the gatekeeper who checks your passport; go to a voting booth but then instead of putting your pick for officials you put in the envelope your just generated pseudonym, close the envelope un deliver that to the ballot box. After the elections officials publish all pseudonyms in bulletin board (roster). The generated pseudonyms in this fashion would have two excellent properties; namely, they would be anonymous and at the same time would be legitimate means to be used for signing votes. We shall call such process braiding as it does have a natural hair-like structure with knots making hairs indistinguishable and also corresponds terminology used for operations on a quantum computer forming a superposition of qubits.

Braiding, in essence, solves the anonymous authentification issue the problem is, however, how can we substitute paper ballot with an electronic one. It may sound like a chicken and egg problem since we need to solve electronic voting to do the pseudonym braiding, but it is not quite so. One of the excellent properties of signatures made by pseudonyms is that they are linkable. The linkability allows doing multiple small braids allowing to use multiple small but absolute consensus ballots as shown in the figure... Those small ballots then can be created by multiple different braiding stations which do not need to be trusted since a lack of coordination preserves anonymity.

To avoid a situation where participants pseudonym gets stolen in a braiding process, it is necessary to make individual verifiability as part of the protocol unless other cryptographic proof is provided. That can be achieved if by the end of the ballot it is distributed for every participant to validate and sign thus forming a braid contract. That, however, introduces fragility in the system since any participant can refuse to follow the protocol and thus terminate the protocol while remaining anonymous. This limits the number of participants within the braid.

The fragility of the system can be resolved with two instruments. The first one is that the number of participants in braiding shall be kept small. The second one is using a binary search technique to try to isolate malicious pseudonyms from the honest ones and do the braiding with different participant sets until the braiding succeeds. Thus no participant can succeed in paralysing the system, and the protocol can be cascaded to acquire large anonymity set robustly.

Nevertheless, who should do the braiding? Multiple systems could fit in as voting protocols based on dc nets and p2p networks with which one could do braiding without any authority. For mixnets, it is necessary to trust the authority that no data is being recorded, which can be used to reveal the anonymity. If only a single mixnet participates which is maintained by the same organization/person, then the anonymity of the vote, in the end, is equivalent to the anonymity which would be provided by a conventional mixnet electronic voting system.

The fix for that comes from the fact that we no longer need to trust the mix to do its job since, in the end, participants control with absolute consent whether to accept and sign or discard the braid. That allows forming a system where mixes are distributed over all internet and hosts mixing service irrelevant of who may use it. That allows to make a braid cascade from multiple mixes and thus resolves any concerns of possible anonymity revealing data as long there is not an initiative to coordinate. In the end coordination of mixnet is a threat to anonymity, thus any necessity of that should be eliminated.

Openness can be considered as opposite to coordination. The voting system must be developed from the effort of the community, and as the open-source world had shown monolithic designs fail to attract interest. Thus, on the one hand, it is essential from the one hand to make the system such that it is easy to start whereas on the other it must not lose the ability customized without breaking off from the existing ecosystem. Thus, in the end, every party would find its place within the ecosystem and so make a pool of available mixers bigger.

4 PeaceVote

We shall separate the discussion of the proposed voting system in the following categories:

- setup;
- Certification. How to know that one uses the correct software?
- Registration. How to become a member of the organization?
- Braiding;

- proposing and voting;
- counting and verifying;
- defence.

As any voting system, it needs a maintainer, administrator and protector which we shall name as guardian which can be a single person or whole organization. The guardian in the PeaceVote system is responsible for the following actions:

- Start the services necessary for the voting system;
- Making choices on cryptography used (public key system, hashing, encryption, randomness);
- If applicable, selects mixers for braiding. Sets up a number of participants for the braid;
- Approves new members for the voting system;
- Defends the system against DDOS attacks and malicious pseudonyms;

All those choices must be transparent and available for public scrutiny to change the guardian if no confidence had been expressed.

Currently, it is considered the best practice to issue SSL certificate on the internet service to make it available for secure access anywhere. That, however, is unsuitable for a voting system where we need to consider such authority to be malicious or even worse selectively malicious. Also, SSL certificates introduce complexity for setting up the system for the guardian, which we would like to make as seamless as possible.

This is where the DemeNet project comes in. In place of having a trusted authority, it is better to trust the decentralization of the communication channels.

The guardian is interested that the members of the voting system are real so that anyone should be able to verify that the whole community possesses human capacity which can be exercised as power and that unfounded participants do not suppress this power. Currently, governments are actively issuing ID cards with the ability to make digital signatures. Unfortunately, the digital signatures issued by them are only verifiable by a trust service provider which adds complexity for the guardian to verify them.

Ironically the business model of social networks which uses the collected data to individualize us is also making available data on the internet which can be used to assess the reality of the person and provides the means to establish relatively secure communication. Currently, it thus seems sensible to use these relatively secure channels to invite a person to the voting system and at the same time to deliver a trust anchor a demespec file.

The demespec file contains only essentials such as the pseudonym of the guardian, cryptographic protocols used, UUID of the self-governing community (deme) which thus is a trust anchor. Since demespec file contains little information, it small and thus it is not a hassle of considering it to be immutable. That allows it to be easily distributed; thus, a potential member could be sure for the genuineness of the invitation from the guardian looking through his trust sources.

4.1 Setting up the voting system

To set up the voting system, the guardian adds the DemeNet, PeaceVote packages and its dependencies to Julia from a trusted repository. The guardian also sets cryptographic primitives for the deme currently provided only by `PeaceCypher` package.

The first step is to create a trust anchor for the guardian, which is contained in the `demespec` file and saves that:

```
using DemeNet
using PeaceCypher
demespec = DemeSpec("PeaceDeme",:default,
    :PeaceCypher,:default,:PeaceCypher,:PeaceVote)
save(demespec)
```

which will create a deme with name `PeaceDeme`, use `PeaceCypher` for cryptographic primitives and say that `PeaceVote` is a peacefounder which for us to simplify discussion currently will not be necessary. Additionally by generating a `demespec` file a private key for the guardian is generated and placed under `.demenet/keys/uuid/maintainer`

The next step is to set up the server. First, a key needs to be generated, which we do as follows:

```
deme = Deme(demespec)
server = Signer(deme,"server")
```

where `deme` is a primitive which additionally with `demespec` also had loaded notary for performing digital signatures and cypher for establishing secure communications with authenticated Diffie-Hellman key exchange.

The next step is to set up the configuration for the braider of the mixnet type. The mixnet braider consists of two parts - it is a gatekeeper and the mixer. The gatekeeper is responsible for authentification with pseudonyms. It is responsible to redirect connection directly to the mixer where a new pseudonym is sent and later redelivered back to gatekeeper sorted and forwarded to participants for forming the contract.

The guardian needs to specify the message length and the number of participants for the braid, as well as authentification details for the mixer. Since the mixer is expected to be from a different deme, we define its id with `DemeID` which contains UUID of the deme, and thus cryptographic primitives can be loaded from the demespec file stored. The configuration then can be created as:

```
braiderconfig = BraiderConfig(BRAIDER_PORT,MIXER_PORT,
    UInt8(3),UInt8(64),SERVER_ID,MIXER_ID)
```

where `BRAIDER_PORT` and `MIXER_PORT` also contain information to establish a socket with server and mixer. `PeaceVote` only approves integers referring to local ports but with the power of multiple dispatch that can be extended with an external package with its own port type. That is particularly useful in a sense that `PeaceVote` does not need to specify whether the server is an ordinary server with IP or hidden service in the TOR. Also, it is useful to specify how the ports shall be contacted anonymously, whether it is over a TOR network or another way which must be specified by the guardian since anonymity loves company.

The next element needed is a configuration of the recorder. The recorder as name states is collecting certified applications, contracted braids, certified proposals and certified votes. The software running it is responsible for preventing redundant information to take part in the ledger. The configuration can be given as follows:

```
recorderconfig = RecorderConfig([MAINTAINER_ID,SERVER_ID],
    SERVER_ID,REGISTRATOR_PORT,VOTING_PORT,PROPOSAL_PORT)
```

For this service MAINTAINER_ID, SERVER_ID are trusted anchors who are permitted to add new members to the deme. The new members are collected in REGISTRATOR_PORT who are then able to participate in the braiding. PROPOSAL_PORT currently is open to all members to submit their proposals (in future it will be configurable for example to board members or etc.). VOTING_PORT is responsible for recording all certified votes for a particular proposal. Similarly, as BRAIDER_PORT the VOTING_PORT contains information on how the anonymous socket can be established.

The last part of the protocol is to decide upon the mixer service, which can be used for other demes and on the synchronization port where collected data from the recorder would be published. The full configuration for the PeaceVote thus can be formed:

```
braidchainconfig = BraidChainConfig(SERVER_ID,MIXER_PORT,
    SYNC_PORT,braiderconfig,recorderconfig)
```

Currently, only a single braider is provided. It is assumed that the guardian periodically changes the braiderconfig, thus preventing adversary to (even guardian) collect necessary data from mixers and guardians for revealing the identity of the pseudonym. It is possible to run multiple braiders at the same time, but one then would need to deal with scheduling strategies, etc which are not important for the content of this paper of understanding the PeaceVote system.

To start the server, the guardian initializes a BraidChain which contains ledger, configuration and deme. BraidChain contains all necessary elements analyses the data collected by the ledger like to count votes and to participate in the system.

```
braidchain = BraidChain(braidchainconfig,deme)
```

It was a conscious choice that the server and the user use the same configuration data, which shows that the system does not have a hidden state. A new guardian can take the place of the old ones with the data available in the braidchain and configure further evolution of the community at his wish. Though since demespec is immutable, he would need to form a new UUID for the deme.

The last step is to start the server:

```
system = BraidChainServer(braidchain,server)
```

which would start all service necessary for the voting system. Since it does not contain any hidden data which one should keep good care of not corrupting (except the server key) it although complex from the point of the number of asynchronous services the system contains it is at the same time pure of side effects. That greatly reduces the maintenance costs as in case of problems one can just reinstall the server, put in back the same key (or a new one) and put in the data of the ledger which is stored by every participant everywhere preventing any corruption.

4.2 Certification

The PeaceVote protocol works because the user uses the same software as the guardian. Nevertheless, how could users be sure that during the installation of new software, a spyware/malware had not also been added?

An ultimate answer to this question for the guardian would be to distribute smartcards which shall be used to ensure that keys are safe and that all signatures made had been logged for accountability of additional device which is necessary to interact with the card.

Another way is to trust the software distributor and after the fact test whether the system is self with the trusted sources selected by the user. For example, a user might receive hashes of the packages certified by guardians and the members of the demes and could compare that with hashes generated locally. That would shrink the area of attack and allow early detection of infected participants since they would certify packages themselves and distribute the certificates which at any time could be inspected more closely.

At the moment, however, such certification of software is not implemented since demes are assumed to be small and incapable of perturbing trustworthiness of certified authorities which allow online and mobile banking to function in the practical absence of fraud.

Another part of the certification is the configuration of the PeaceVote system. If the guardian would be of curios kind, he could make different settings for different participants who would allow revealing the identity of the pseudonyms. Thus it is crucial that all participants would see the same configuration and could confirm that themselves; thus, it must also be certified by each participant. We will return to this issue later in the section about braiding.

4.3 Registration

To take part in the deme the guardian or it's delegated authority needs to sign users pseudonym. From the guardian perspective, he is troubled with distinguishing genuine application over boots. In an ideal world, the pseudonym before sent to the guardian would be signed by a personal smartcard or by a state authority confirming users identity through authentification still would reduce us to the need of smartcard or another personal token. However, the ideal world is not always available, and if smartcards are available, those require developer power and some cheesy subscription model which together kills accessibility.

A solution to such problem could be by using our online presence in different social networks, let it be Facebook, twitter, google, LinkedIn, OpenID, etc. which generally do provide enough information for a person to assess whether the person is real or a bot. However, it is close to impossible to make a computer program which would do that automatically.

The guardian does have three options. The first one is upon application the guardian checks the identity online and tries to establish online communication over one of the communication channels to check whether the person applying is not a bot which had borrowed a random identity and on such assessment would sign the certificate. The second option is that the guardian upon a request delivers a token to the phone, email, etc. which can be used to register to deme. The last option is to allow everyone in and then kick off those who are not real.

The first option is tedious but might be necessary in case high assurance of the identity needs to ensure (investor board meeting). The last option of kicking off unreal members are possible by the guardian issuing order that all pseudonym braiding should restart from their anchors. The second option of delivering a token upon a request is most practical one which is what in one form or another we use daily, and this is the one we shall rely on the daily basis for demenet.

The registration for the DemeNet with tokens is currently implemented with Recruiters package. The algorithm of it is as follows:

- The guardian starts the server certifier service and the token receiver service.
- A potential member is sending the application to the guardian or his delegated trusted people.
- After a review which could also be automatic if the phone number is provided or other trusted source of communication, the applicant receives a token, demespec, and the information to connect to the registration server. The token is sent to the certifier service confidentially.
- The applicant generates a key, establishes a secure connection to the guardian certified certifying service and sends his pseudonym and token.
- If a token is found, a certificate is created by the server key and is added to the ledger. Additionally, a log file is kept on the communication channels which were used and what information had been used as a basis to issue the registration certificate.

It is important to note that to use Recruiters; the guardian is responsible for doing the piping of issued certificates of the server to the braidchain. As this and other steps for configuration are so common, it is recommended to use PeaceFounder package which integrates PeaceVote with Recruiters. A great thing with tokens is that one can send an invitation to people without them making to do work and if they decide to apply, they can do that instantly. This is particularly great to form the demes spontaneously just by starting a server, sending invites to the email, braid and vote. However, users would still need to install an app if it is actively used for multiple demes in the long term such cost would be treated as the cost of installing a web browser.

4.4 Braiding

At his point the members have been registered, they have loaded the demespec file sent to them together with an invitation. They have synchronized their ledger with the guardian (one way) with the provided information in the invitation, for example, a port and IP address. The ledger contains BraidChainConfig certified by the guardian.

The user loads the configuration file, and its first step is to gain anonymity by creating a new pseudonym during the process of braiding. All keys for the pseudonyms are stored locally, which as we will see is essential for dynamism in the voting system - braiding shall not limit voting and voting braiding. The storage unit is called KeyChain and can be initialised as follows:

```
keychain = KeyChain(deme)
```

which will load keys from `.demenet/keys/uuid/*` and sort them in their chronological order.

When the keys are loaded with the braidchain the user can gain anonymity by braiding which can be simply done as:

```
braid!(braidchain,keychain)
```

This simple command in the context of `SynchronicBallot` is performing the following algorithm: - The user authentificates with the gatekeeper in both directions with the current pseudonym and encrypts the connection. - The gatekeeper forwards the connection to the mixer with which it has authetificated and encrypted. - The user using the same socket is now authenticating the mixer and establishing an encrypted connection. - The user generates a new key and sends the pseudonym to the mixer - The mixer collects all pseudonyms from the participants, sorts them and sends them back to the gatekeeper. - The gatekeeper forms a braid from all collected pseudonyms. Additionally, the guardian adds a hash of the current ledger. (Other relevant metadata can also be added such as time up to which the braid must be published to the ledger, etc.) - The participants validate the braid verifying whether their pseudonym is within the braid and also and also verifying that the hash of the local ledger is the same as hash put in the braid. If validation succeeds the participant signs the braid and delivers the signature to the gatekeeper. - If braiding went successful, the gatekeeper collects all signatures verifies them and publishes the braid to the braidchain. If unsuccessful, the procedure is repeated with variations of a set of participants, perhaps of the mixer, etc.

In this procedure, the user can be the guardian certified pseudonym or any anonymous pseudonym which follows this procedure. If the procedure is repeated, it becomes crucial to anonymize the IP address of the connection which can be done with TOR or others as the guardian wishes to configure that (it can not be up to users deciding how the socket is anonymized because anonymity loves company which leads to a single option).

Additionally, during the braiding participants does certify the ledger. This makes the information stored on the ledger immutable to any adversary. Additionally, that ensures that all members see the same ledger. This is convenient as a place where the guardian could distribute the configuration of the voting system, which prohibits deanonymization attacks by a vicious guardian. Additionally, the ledger is an excellent place to store other information necessary for certification such as used software hashes or other demespec files and thus forming a real deme network or simply DemeNet.

4.5 Proposing and Voting

After braiding each member owns a chain of anonymized pseudonyms (KeyChain) which can be used to sign messages in our situation votes. However, each pseudonym is itself anonymous and part of the braidchain which one should one use? Is the newest one enough?

Before the vote, an eligible member makes a proposal (by default all PeaceVote members can make a proposal but not with their pseudonyms). The proposal for the PeaceVote is one of the most abstract it can be for a voting system. In itself, it contains two important properties - how the votes for it are going to be counted and what is the input from the user - is it a single question with multiple options, or it is a whole survey a list of questions for which voter is forced to decide on the priorities of the organisation in an interconnected way having a constraint like in quadratic voting.

The proposal again is registered to a `BraidChain` to allow everyone who has access to votes counts them and to avoid a situation where a proposal gets changed to deliver an opposite outcome. Additionally, it defines a set of pseudonyms by the position in the BraidChain, which cuts it. This way, each anchor pseudonym would have only one vote.

The voter does the last step. He/she synchronises his braidchain with the guardian and checks its integrity (see next subsection on validation). Then he filters out the proposals from the BraidChain which are then parsed by user application of the PeaceVote and shown on the screen. To vote user would select a proposal and fill it as specified which he then would sign with keychain:

```
option = Vote(index,choice)
```

The `index` is the place in the braidchain of the proposal which determines on which proposal one votes on and at the same time defines an eligible set of pseudonyms. The `choice` is the result of filling out the proposal which has an arbitrary type as proposals can be of many kinds. Furthermore, the final part is created vote by digitally signing the option by a corresponding pseudonym in the keychain.

```
vote = Certificate(option,keychain)
```

Finally, voter delivers the vote anonymously by the options specified by the guardian:

```
record(braidchain,vote)
```

4.6 Validating and Counting

When the end of the voting period had happened, it is time to collect votes, validate their eligibility and count them. The votes themselves can be either published in the braidchain immediately can be delayed and published later or can be kept in secret which can be useful for implementing anti-bribery and anti-coercion mechanisms (see the next subsection). The second part when all votes had been collected is to validate them, which means that the vote had been signed by a pseudonym created in the process of braiding from root pseudonym and specified by the place of a proposal in the BraidChain. The last part is to count them as defined by the type of proposal.

Assuming that all votes had been collected which by default in PeaceVote are stored on the BraidChain providing tamper resistance the next step is to validate the BraidChain which can be separated in following steps after the ledger had been properly synchronised:

- Parsing. At this step ledger which is just an array of byte vectors `Vector{Vector{UInt8}}` is parsed to a `Vector{Union{Certificate,Contract}}` where certificate/contracts contain the document and its signatures:

```
parsedledger = parse(braidchain.ledger)
```

- Validation of immutability. Move through each element in parsedledger and validate that hashes specified in the braid correspond to hashes of the local ledger:

```
m = validateimutability(parsedledger,deme.notary.hash)
```

returns integer `m` up to which hashes in the braids do agree with hashing of the local ledger.

- Attestation. At this stage, the certificates and contracts are cryptographically verified producing `Vector{Union{Intent,Consensus}}`. `Intent` and `Consensus` contain the documents and IDs of who issued the signatures. The operation for this procedure can be performed as follows:

```
verifiedledger = attest(parsedledger,deme.notary)
```

- The last step is to validate the ledger for its integrity. The result of validation is an index until which the ledger is considered valid. The validation procedure can be described as followsInitialisationon. Create an empty set for current pseudonyms, mutable set of parameters - current authorities who can issue member certificates, a current hash of the ledger.
- Loop. Move through each element in a verified ledger
- If the element is `Intent{Config}`. Validate that it comes from the guardian and if so, update the mutable set of parameters with approved authorities who can issue a new member certificate. Else break.
- If the element is `Intent{ID}` and is signed by current authorities, add the pseudonym to the current pseudonyms. Else break.
- If the element is `Consensus{Braid}` and is signed by N current pseudonyms where N is a number of new pseudonyms, then swap pseudonyms form the current pseudonym set as specified by the braid. Else break.

```
n = validateintegrity(verifiedledger,deme.demespec.guardian)
```

In case the `n` or `m` is smaller than the length of the ledger end of the ledger is being cut, the ledger again is synchronized with the guardian certified server and the procedure repeated for the end of the ledger.

After the validation of the ledger, we can operate on the data. First, we can extract the proposals from the verified ledger which one does as follows:

```
pindexes = proposals(verifiedledger)
```

which will return the indices of the proposals in the ledger. For each proposal index, first, we can get the corresponding proposal:

```
proposal = verifiedledger[pid].document
```

The votes can be obtained by:

```
votes = getvotes(pid,verifiedledger)
```

where each vote is filtered by the braidchain. The proposal may contain end date proposal.date when the voting on the proposal ends which can be incorporated by slicing the `verifiedledger`.

The final procedure is counting. To do so, the user does:

```
tally = count(proposal,votes)
```

where one can implement whatever method of counting one wishes. Whether votes can be changed after cast, whether it is a preferential ballot, or multiple question ballot with a constraint, for example, a budget, quadratic voting. All these are trivial to implement for the message based voting system.

5 Adversaries and defence

I identify three types of attacks adversary could exercise. The first one is an attack on the network infrastructure. In this category, we have DDOS attacks and attacks on the TOR network to reveal the real identity. The second type is infiltrator attack where the adversary is part of the system in this case, either guardian, mixer, or some coalition of members. The third kind is an attack on the secrecy of the voter. First, it could be malware/spyware on the device, or it could be coercer/briber who convinces the voter to collaborate.

5.1 Attacks on network infrastructure

Online systems are prone to attacks which can make them offline. Often that is achieved by buying botnet to make requests on the services until they do overload, which is the definition of a DDOS attack. No online system is protected from this kind of attack. Nonetheless, one can always reason about the costs.

The characteristics of PeaceVote system allows distributing the load over an extended period of time. The braiding procedure which requires a stable network and stable operations of gatekeeper and mixer are not required to happen on the election day. Thus DDOS attacks on braider become increasingly costly.

The other DDOS attack on election day can happen on the vote recorder. In the current design of PeaceVote, the remedy is to use shields which filter out valid votes from fake ones. Since all authentification data is in the BraidChain which is available to everyone, such shields would be easy to set up if necessary.

Another kind of attack we need to be aware of is the attack on the network anonymizer, for example, TOR network. It is known that if ISP is all-powerful and records all inputs and outputs the identity can be deanonymized in some given time period. Alternatively, there could be infiltrator attacks on the TOR network itself, which could make connections with inputs and outputs. The question is how plausible it is?

We can consider Edward Snowden's location as a target of the USA government. The information on his IP address could quickly lead where he is located; thus, he may just refuse to use it. On the other hand, he knows that TOR network makes his IP address untraceable which allows him to use the internet and for example participates in discussions in media, conferences and etc. where ends of TOR are assigned to his identity. To this point, Edward Snowden had not yet been found proving the strength of anonymity of the TOR network.

5.2 Infiltrator attacks

The other kind of attacks is when adversary had acquired control over some part of the infrastructure, including members. Realistically that could happen due to some security flaw which allows gaining control of the servers or the people themselves had become corrupt due to some circumstances. In any case, it is necessary that the adversary can be detected, isolated and its attack defended. We shall consider three kinds of adversaries and their colourations. The vicious guardian, mixer and coalition of a subset of members.

The isolated guardian as the adversary can block the accessibility of services and filter out which votes he would like to accept with the recorder and also accept fake identities. In the case when the guardian blocks accessibility it could turn out to be tricky to detect whether it is due to him being corrupt or due to receiving a DDOS attack, however for the later if shield or even multiple ones are used the attack could be defended by a wise guardian. Thus its impeachment due to DDOS attack should not be sympathized with.

The guardian is also responsible for the acceptance of new members. A legit question for a participant would be whether those are actually real. This question can be tested statistically. To do so, a random set of participants are selected, and those identities then are matched with the guardian published contact details which should lead to some social network or etc. The last step is then to contact those people to see whether they know that they are part of the deme.

The other malicious action which guardian could do is to accept for recording only those votes which are of his liking. To detect such situation trustworthy the vote can be delivered first to a policing authority with TOR which then tries to publish it to the recorder. If it fails, it sends the vote to the next policing authority which again tries to publish that. In this way, a consensus of corrupt guardian would be created and thus could be impeached.

The guardian is also responsible for ensuring that mixer and members are not breaking down the service. In the case of the mixer, the most malicious thing would be to alter the ballots by inserting his own pseudonym or just slightly corrupting it. In such action, the braid would come to participants where one would find it corrupt and thus refuse to sign breaking the protocol. At this point, the guardian does have a hard time deciding on whom to blame.

At this point, the guardian has two things he can try. First one he can try to repeat the braiding procedure by excluding those who refused to sign the ballot until the braiding protocol succeeds. And later one letting those who terminate the protocol braid with themselves. If that fails a significant number of participants had entered the braiding prison, the guardian might suspect that the mixer is corrupt, which he can let publically known. The solution here is just to issue a new configuration which uses a new mixer which a trustworthy guardian should do regularly anyway, which we will now consider.

The next type of advisory is one who controls the mixer and the gatekeeper. Such a situation quickly rises when in the PeaceVote protocol mixer is self-hosted by the same guardian. In such a situation, advisory can silently record additional data which allows revealing identities with corresponding pseudonyms. To prevent the rise of such a situation, the participants may require that the mixer is changed regularly, which is served by trustworthy or unaccessible guardians on the internet.

5.3 Secrecy attacks

The last type of advisory is one who attacks the secrecy of the voter. In paper ballot system only the voter knows the secret on how he voted on and thus the briber/coercer is powerless on whether the voter is telling the truth unless he uses a lie detector or something. In electronic voting, there is no way to ensure that the voter is voting on the device without adversary behind his back observing. Additionally, the advisory could be operating anonymously behind the internet or the device the voter uses could have malware/spyware.

To avoid malware/spyware, a certification procedure is used. The goal is to avoid malware/spyware to come to form the PeaceVote protocol, which is open to extension with different cryptographic methods, network anonymizers and etc. As described in the certification section, the PeaceVote protocol could perform self-consistency checks with already trusted sources before using a new component. Initial trust source as a package repository would be provided by the distributor of PeaceVote.

The next type of malware/spyware could come from the device itself. First, it could come from another application on the device. To avoid such situations, Vendors are already putting applications in containers which do have restricted access. In the end, thus it is Vendors responsibility that device does not have a malware on it. That currently, for example, allows mobile banking to happen without significant issues of fraud. Thus why not make the same trustworthiness assumption on electronic voting?

The first most dangerous attack on secrecy is by anonymous briber/coercer. The briber can exchange a certain amount of money for an exchange of vote. Additionally, for the PeaceVote protocol, the briber could ask for the pseudonym key itself which they could use for unlimited voting while still having backing from a real identity.

To fight the selling of pseudonyms online, we need to prevent its extraction from the device. It could be made possible that the keys are stored and generated during braiding in the smartcard itself, thus preventing its users from selling their pseudonyms. The identity pseudonym then could be signed by a Vendor issued key on the card itself to prove that issued key and derived ones in the process of braiding are not possible to be extracted. The card itself then can be verified with biometrics (with which I mean one signs video chat with the key on the card, etc.) that it does belong to a real person.

Not able to buy pseudonyms the briber/coercer would turn to buy votes instead. In the case of anonymous advisory, the strategy would be that the vote would be collected by it before delivered to the recorder in that way, ensuring that an existing vote is not smuggled into him. Additionally, he may only give payment after the elections had ended and the bought vote had been accurately voted on.

To solve such issue, most voting systems implement some form of receipt freeness or as I like to simplify prooflessness. In the PeaceVote system prooflessness can be implemented if the votes are not published but only the tally at the end of the vote. Then one can implement the ability to silently change the vote after it had already been sold to briber/coercer. But in doing so, one sacrifices transparency, particularly, universal verifiability of the voters for rootlessness. The votes though themselves can be universally verified by the permitted personnel.

Perhaps a more successful extension to the PeaceVote system would be if the voter would have the ability to choose whether the vote is going to be public or hidden. By default, the voter would publish the vote for everyone to see, but in the case, there is an opportunity to sell it to the briber he/she does so. But after the fact submits a new vote which remains hidden from everyone except the guardian (or his derivative) who after elections issues correction to the public result as a tally.

As long as the election result, who happens to be elected and who not, is determined from the publically available ones there would be no point to question the integrity of the guardian who counts the hidden votes. Furthermore, to prevent maliciousness of the guardian, an independent auditor could be used to recount the correction.

Similarly, the same bribery/coercion mechanism could work in person as long as the briber is not looking behind the back of the voter and then collect the device after the vote until the end of elections. To prevent such scenario, the voters should have the ability to defend themselves, for example, by issuing a tagged vote in the presence of briber/coercer which would give a signal to authorities that a possible crime is happening and try to catch briber/coercer while the crime is happening.

6 Discussion

Electronic voting is a very complex problem which can have great consequences on democracy. Each system comes with its own need of expert knowledge, and thus discussion of the most trustworthy design publically could potentially be steered to insecure systems to the benefit of capital. It is thus vital to battle test voting systems and starts small. As the power of democracy built with them becomes more significant, they stand a chance to gain interest in hacking by corporations and capital who are interested that there would not be such thing as a society.

The PeaceVote voting system, while complex is the first system which achieves true transparency, security and privacy. The trust on it is derived from those components and not from trusted personnel. The system is strongly software-independent for voting. The system does not contain a hidden state and can be reset/restarted at any time with the publically available data. The guardians are powerless. Their wise actions and reputation are what matters for a new guardian to show up and impeach the old one. The design allows us to have a giant mixnet cascade, which is possible due to openness and introduced a process called braiding. The system scales in two directions. Though one may wish to fragment it to some regions, etc. to limit the processing power and storage needed for participants.

Although complex it is one of the easiest to set up. In contrast to blockchain ideology, it is fine to start small. The trust is built from wise actions and not from competitions of wasted work (PoW) or legitimatized inequality (PoS). A new guardian can start with a single raspberry pi and use automatic configuration which would generate a key for the deme and for the server which does not require a permanent IP address or SSL certificate to be trustworthy. Then to use it, voters only need to install an application on their smartphones which would allow participating in different demes.

The PeaceVote voting system also allows easy to be used for whistleblowing where in place of issuing votes, one can just sign a leaked document with a pseudonym. This could be particularly useful for corporations which, according to most common European laws require that whistleblowing should be possible and even encouraged to prevent ill effects on society.

Ensuring strong anonymity of whistleblowers is what PeaceVote can do.

The infrastructure around PeaceVote is more generic than the voting itself. Particularly DemeNet addresses the issue of requiring SSL certificate issued by an authority which is substituted with self-consistency and directly distributing the ids of the guardians. That allows easily to add other services like chatting, messaging service, a localised search engine. A news distributor where the voting could allow for subscribers to decide upon what content they would like to see and thus giving the initiative to pay subscriptions.

The internet had become a dystopia, full of tracking and suspicion of sincerity of information. It's time to come together and start to found some peace with the world around us. Its time to voice of expressing trust and breaking systematic media suppression of public opinion. [foo]

References

[1] Ben Adida. 2006. Advances in cryptographic voting systems. Ph.D. Dissertation. Massachusetts Institute of Technology, USA. Advisor(s) Ronald L. Rivest.

[2] Anderson R., Needham R. (1995) Programming Satan's computer. In: van Leeuwen J. (eds) Computer Science Today. Lecture Notes in Computer Science, vol 1000. Springer, Berlin, Heidelberg

[3] Chaum, D. L. (1981). Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, 24(2), 84-90.

[4] Ruffing, T., Moreno-Sanchez, P., & Kate, A. (2014, September). Coinshuffle: Practical decentralized coin mixing for bitcoin. In European Symposium on Research in Computer Security (pp. 345-364). Springer, Cham.

[5] Serjantov, Andrei & Danezis, George. (2003). Towards an Information Theoretic Metric for Anonymity. 2482. 10.1007/3-540-36467-6_4.

[6] Danezis, G., & Diaz, C. (2008). A survey of anonymous communication channels (Vol. 27, p. 30). Technical Report MSR-TR-2008-35, Microsoft Research.

[7] Dolev, D., & Yao, A. (1983). On the security of public key protocols. IEEE Transactions on information theory, 29(2), 198-208.

[8] Christian Franck (2008). New Directions for Dining Cryptographers. MSc thesis.

[9] Golle, P., & Juels, A. (2004, May). Dining cryptographers revisited. In International Conference on the Theory and Applications of Cryptographic Techniques (pp. 456-473). Springer, Berlin, Heidelberg.

[10] Dimitris Gritzalis (2002) Secure Electronic Voting

[11] Bernhard M. et al. (2017) Public Evidence from Secret Ballots. In: Krimmer R., Volkamer M., Braun Binder N., Kersting N., Pereira O., Schürmann C. (eds) Electronic Voting. E-Vote-ID 2017. Lecture Notes in Computer Science, vol 10615. Springer, Cham

[12] Rivest, R. L. (2008). On the notion of 'software independence'in voting systems. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881), 3759-3767.

[13] Sampigethaya, K., & Poovendran, R. (2006). A survey on mix networks and their secure applications. Proceedings of the IEEE, 94(12), 2142-2181.

[14] Dzieduszycka-Suinat, S., Ph., I.M., Kiniry, J., Zimmerman, D.M., Wagner, D., Robinson, P., & Adam (2015). The Future of Voting End-to-end Verifiable Internet Voting Specification and Feasibility Assessment Study Internet Voting Today No Guarantees End-to-end Verifiability E2e-viv.