BraidChains: a legitimate anonimity through pseudonym braiding

Janis Erdmanis

November 16, 2020

Introduction

In a remote electronic voting system, the difference between personalised hardware which does cryptographic operations and the voter starts to get blurred. I sci-fi future we may experience cryptographic implants accelerating our brainpower on such mundane operations or citizens who are no longer considered humans. In this respect, I propose to consider voter as a cryptographically capable being who owns pseudonyms and can protect secrets. A way to ensure that the voter is responsible for protecting his secrets and pseudonyms is an orthogonal issue to the remote electronic voting system, which I shall address later.

Having defined the voter that way allows us to differentiate between software and it's users. It is particularly important to acknowledge that an adversary can write software in any step. Furthermore, it is exceedingly more difficult to audit the software which runs in a complex environment defined with the operating system and its hardware. Instead, as Ron Rivest proposes we shall focus on auditing the evidence which proves that any modification in the software had not been able to make an undetected change in the election result which is called software independence.

Software independence is a necessary property for trustworthy elections but is not sufficient for a case when election outcome or its integrity evaluation falls in hands in a few auditors. In a paper ballot, the number of auditors is limited by physical factors, like counting speed, area of the place and geographic location. As they become so remote from society, how one would ensure that they are trustworthy and are not taking part in sabotaging elections with election officials? It would sound absurd to put a law in law in place which punishes dishonest auditors and thus makes them fearful of their integrity as a potentially corrupt state has the power defining honesty where the vital property is representativeness. The consequences of one corrupt auditor who announces that elections are rigged could be enough to paralyse the democracy.

It is thus of great importance to make the evidence public so anyone could audit the integrity, accuracy and fairness of the elections. Additionally, a small adversary coalition shall not be able to produce valid evidence either during or after the election process; otherwise, they become de facto auditors. Therefore we shall require that each voter takes part in making valid evidence and so the third party does not need to be trusted.

At the same time, the published evidence can't reveal or allow to infer the choice of each voters identity which would allow a mass scale bribing, coercion and shaming making voters fearful of expressing their true choice. Besides, it is important to ensure that it is hard for an adversary to record additional information, for example, from mixes, which can compromise privacy. Thus we require that the evidence also allows estimating how hard it could have been for an adversary to do that in particular elections. To summarise for a trustworthy remote electronic voting, we shall require the following properties:

- Accountability. The evidence of a ballot conclusively proves legitimacy, accuracy and fairness of the ballot. Only controlling the voter either in person or through malware adversary can change or vote in place of voter producing valid evidence. A small adversary coalition can not paralyse the voting system without being made accountable and isolated.
- Transparency. The evidence of the ballot is public for anyone to audit without requiring to trust a third party. In other words, the evidence can't be produced by a small adversary coalition either during the voting process or after that.
- Privacy. The published evidence does not reveal or allow to infer identities of voters making a particular vote. The voting protocol does not allow an adversary to record additional privacy revealing data without producing suspicious evidence.

Most of the voting systems on the literature consider a rather peculiar adversary model where certain parts of the protocol are trusted for verifiability. For instance, a voter could get a casting assurance (individual verifiability), and ability to validate that only eligible voters had participated, they had voted once, and that the votes had been counted accurately (universal verifiability). Nevertheless, for assurances, the voter would still need to trust the verifying system as all data for validation can't be published to protect voters privacy which is thus not transparent.

A similar situation is present for ensuring the anonymity of the vote. 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, although it is more disperse it, is still centralised. 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. The anonymity of such systems is guaranteed from a realistic point of view of the adversary who would not be able to control every ISP of the world to deanonymise the traffic or control every node providing mix services. Thus with the delocalisation and companionship, we can already today use quite strong means to exchange messages anonymously. However, anonymous message channels yet do not allow to avoid tracking. Unless the electronic voting is an anonymous pool where anyone can vote multiple times (thus not accountable), the election voting system is going to have authentification.

Two approaches exist to solve the problem of anonymous authentication. Blind signatures allow the creation of signatures of messages unliked to the person who asks them to the authority, with which one can have usual authentication. In contrast, ring signatures allow the creation of untraceable signature locally on behalf of the group without involvement of the authority. Both schemes, unfortunately, are not directly applicable to electronic voting where the message is the vote, and it is delivered with an anonymous channel to the ballot box.

Particularly for a blind signature scheme, there are two profound issues. In the process of signature generation, the user may abort the protocol. Is it because a network connection failed or because the user has malicious intentions of obtaining two votes? The second issue is that the blind signature issuer is not accountable. It can maliciously issue additional signed votes as long as some voters abstain from elections, which often is quite a large population, without leaving any traces. A straightforward extension here would be to involve multiple parties forming a blind contract (each party makes a blind signature) if absolute consent is reached.

For a ring, signature scheme with an extension of linkability is not practical. Substantial computational resources are needed to make ring signatures, which are large and scales poorly with the size of the group. Thus unless anonymity set size is relaxed, grouping randomised in advance this scheme currently is not practical. Although amazingly it has found a place in cryptocurrencies such as Monero, ZCash, ZCoin (in the form of zero-knowledge proofs) for guaranteeing transaction anonymity multiple currency inflation events for ZCoin makes us doubt the cryptographic protocols/primitives used.

Coins as signatures, handwriting, phone number, IP addresses and as well with private keys, are in general means to authenticate which we shall refer as pseudonyms. Thus astonishingly follows that the anonymisation of coins is equivalent to the anonymisation of the voters themselves. The anonymisation of pseudonyms had been explored previously with mix services such as VANET and CoinShuffle. In vehicular networks, the authority would benefit from knowing information on infrastructure quality for further improvement where mix zones for changing pseudonyms is currently actively studied to protect the privacy of the car owners. But as in CoinShuffle requires to trust the third party not to steal your pseudonym.

To avoid trusting the third party, we could extend any ballot protocol with contract formation between all involved parties who would each individually check the resulting ballot to contain their new pseudonyms and make a signature on that if they approve that. Alternatively, one can perform a Secret Santa protocol to obtain the same result in fewer steps and with more privacy. However, such protocols do suffer from a requirement that every participant is honestly following the protocol and fail miserably if one participant is an adversary and thus is extremely fragile. It is inconvenient that to perform pseudonym change, all participants must be online at the same time, which makes it unpractical in real-life situations. On the other hand, such pseudonym anonymisation can be done with basic cryptography and as aborts can be resolved at any step.

To solve the issue of fragility and inconvenience, I propose to do anonymisation it in multiple small trustless ballots which I shall refer as braids. A benefit of such additional complexity is that the pseudonym can be used to authorise a new pseudonym anonymously allowing to increase the anonymity set size incrementally and thus a number of participants in each run can be small. This process I call a pseudonym braiding which in the result forms a secure knot (braid). The braids are stored in a public transaction log, allowing to fix the state of the system for external applications such as voting.

In contrast to ZeroCoin and ZeroCash protocols where there is a shared pool where coins (pseudonyms) are mined and then claimed a pseudonym braiding solves crucial issues. At any time a ballot proposer can select a row counter in the transaction log to specify all eligible pseudonyms deterministically allowing concurrency in voting between different proposals. Anonymity set for each member is deterministic, and thus multiple braidings can be enforced before a member is allowed to vote. That prevents an adversary from using correlations between mining and claiming to infer properties of a subgroup of members which could allow an adversary to target them differently than others in the next elections. These issues can be easily solved in ZeroCoin and ZeroCash protocols by isolating mining and claiming pools which in addition would allow auditing the used cryptographic protocols/primitives. However, it is still computationally expensive and a public blockchain distributed between peers is controversial where both I show can be avoided with more interactivity.

The usual electronic voting protocols which focus on anonymising the votes over voters are arguably less complex over a voting system which requires braiding before voting. Nevertheless, I show that the braiding protocol needs to be executed on an average only logarithmic number of times for each member to get close to absolute anonymity of the generated pseudonyms. In addition, a full transaction log does not need to be stored locally, and no waste is generated, which thus scales well. Furthermore, local operations can be performed in a low cost certified hardware making the electronic voting protocol also malware resistant. At the same time, coercion/bribing resistance can be provided with an additional paper ballot to which members authenticate in secret with their braided pseudonym providing receipt freeness.

Mathematical model of Braiding

Anonimity Set

To mathematically model the anonimization of the voter we shall introduce a notion of a pseudonym as a general means for authentification:

Definition (pseudonym). A pseuodnym p is a means to authetification for a particular **owner** with an unique and public **identifier**.

Definition (pseudonym equality). Two pseudonyms are equal if equal are their identifiers.

Definition (pseudonym set). P is a set of all possible pseudonyms if there does not exist a pseudonym q which is equal to any pseudonym $p \in P$.

There is an uncountable number of pseudonym systems possible. Let's list some to get an intuitive feel for definitions

- A signature or handwriting. In this case, the identifier is the slopes, the curves which can be identified to be unique and the owner is the one person which can reproduce the writing.
- A phone SIM card. A public identifier is a phone number, and the owner is the one person who can make calls, text messages with a corresponding phone number. In some cases, we take calls only from the known numbers, which shows that sometimes it is a means to authentication.
- A computer connected to the internet. The public identifier, in this case, is IP address, and the owner is the user of the computer. If the computer is behind the router as is often the case, we may look that the pseudonym owner is the person who configures the router. Authentication we often experience in the way of tracking giving us personal suggestions or as a restriction to some content.
- Electronic signatures. In this case, the identifier is the public key (in some cases together with generator), and the owner is the private key holder.

From examples, we shall exclude systems which use a secret token for the authentification as there is no public identifier which can be made public. Similarly, we shall exclude HMAC systems as there is no separation between owner and identifier.

Some pseudonyms are safer than others. For example, an IP address can be somewhat easily forged by an adversary; a phone SIM card assumes a trustworthy phone service provider. On the contrary, electronic signatures similarly as their analogue counterparts do not need trustees and thus are unforgeable.

Definition (unforgeability). A pseudonym $p \in P$ is unforgeable if only the owner can authenticate with the pseudonym p identifier.

In the definition, for a pseudonym, the owner as a person using the pseudonym is not linked publically. That is done with a purpose as we want to use a pseudonym to define anonymity. To make them as digital identities, we need to link the pseudonyms to their owners which is often done on the roster.

Definition (roster). A roster $R \subset P$ is a subset of all pseudonyms where every pseudonym $r \in R$ owner is publically known.

Most of the time when talking about anonymity, the probability theory can be used. To discuss the braiding protocol, we need to operate on the sets of pseudonyms. Besides, we will need to distinguish between absolute anonymity and a lack of membership to a particular group. Thus anonymity set is defined with respect to information which proves that one pseudonym is linked to a set of other pseudonyms and more information, for example, gained by an adversary can unlink and reduce the anonymity set size. This motivates the following definition.

Definition (anonymity set). A subset $A_S(p|I) \subset S \subset P$ of a pseudonym $p \in P$ in presence of **information** I is its anonymity set if the set $A_S(p|I)$ contains every pseudonym of $s \in S$ which is linked to a pseudonym p owner.

Corollary I. If a pseudonym $p \in P$ is not linked to a subset $S \subset P$ for a given information I the anonimity set is:

$$A_S(p|I) = \varnothing \tag{1}$$

Definition (fairness). A set $S \subset P$ is fair if for every pseuodnym $p \in S$ in presence of information I the anonimity set is the pseudonym itself

$$A_S(p|I) = \{p\} \tag{2}$$

Definition (absolute anonimity). The pseudonym $p \in P$ is absolutelly anonymous with resepect to a subset $S \subset P$ if its anonimity set is equal to the set S itself

$$A_S(p|I) = S \tag{3}$$

Estimation of effectiveness

Definition (member). A pseudonym $p \in P$ is a member if it has access and can authenticate to a particular service.

There are of course different kinds of services for which one can be considered a member. Here we shall always refer the member with respect to a braiding service which I will define shortly.

Definition (state). A state $S_T \subset P$ is a set of all members at a given moment $T \in N$.

Definition (braiding). A state transition $S_T \to S_{T+1}$ is braiding if a subset of memebers $M \subset S_T$ generate a new set of pseudonyms $Q = S_{T+1} \cap (S_T \setminus M)$ in such a way that pseudnym owners of M map one to one to the speudonym owners in Q while the anonimity set of every pseuodnym $q \in Q$

$$A_{S_T}(q|I) = M \tag{4}$$

Corollary II. The anonimity set of the new pseudonyms $q \in Q$ with respect to rooster $R \subset P$ after braiding is

$$A_R(q|I) = \bigcup_{m \in M} A_R(m|I) \tag{5}$$

This is an essential property for the braiding as it provides a possibility for exponential growth of anonymity set size with respect to a number of braidings a member participates.

Definition (software-indpendance). The braiding protocol is server-side software-independent for participant $m \in M \subset S_T$ if adversary controlling the braiding service and all other participants $M \setminus \{m\}$ can not exclude pseudonym m owner in the state S_{T+1}.

This property can be achieved, for example, if every participant after braiding $m \in M = S_T \setminus S_{T+1}$ must sign the resulting transaction $(S_T \setminus S_{T+1}, S_{T+1} \setminus S_T)$ before it is considered publically valid. In practice the signatures are used to obtain the pseudonym set M for the transaction.

Definition (braid). A braid is a proposed state transaction $B = (S_T \setminus S_{T+1},\, S_{T+1} \setminus S_T)$ formed during braiding at a given moment T.

Definition (optimal braid). An optimal braid is a braid whose participants $m \in M = S_T \setminus S_{T+1}$ anonimity sets are pairwise exclusive

$$A_R(m_i|I) \cap A_R(m_j|I) = \varnothing. \tag{6}$$

We can intuitively understand that if anonymity set size of every participant in the braid is small, then it is likely to be close to optimal. On the other limit, the anonymity sets get saturated, and its size for a participant before and after braiding would be about the same. This motivates us to introduce a gain associated with every member in a potential braid.

Definition (gain). A gain for a participant $m \in M = S_T \setminus S_{T+1}$ is an increase of anonimity set size making a state transition $S_T \to S_{T+1}$

$$\mathrm{gain}_T(m|I) = \left| \bigcup_{q \in M} A_R(q|I) \setminus A_R(m|I) \right| \tag{7}$$

Theorem (optimal gain). For a fixed size optimal braids with a number of participants N, the anonimity set size after L equal gain braidings for a single member $m \in S_T$ is

$$|A_R(m|I)| = N^L \tag{8}$$

Proof: Lets consider the first braid of a member $m_1 \in S_1 = R$. In this case the anonimity set for each memeber m consists of the member itself thus the anonimity size after first braiding is:

$$|A_R(m_1|I)| = N \tag{9}$$

In the second step $m_2 \in S_2$ as braids are optimal the anonimity set size is:

$$|A_R(m_2|I)| = N|A_R(m_1|I)| \tag{10}$$

the next step and so forth is the same $|A_R(m_{i+1}|I)| = N|A_R(m_i|I)|$ which proves the formula after substitution.

Using this theorem we can estimate the number of braidings each member ideally would need to perform until absolute anonymity $A_R(m|I) = R$ is reached:

$$L = \log_N |R| \tag{11}$$

This logarithmic dependence makes it feasible for gaining large anonymity set size in a reasonable number of interactions with external service while needing a small number of present participants to execute the protocol. That makes it feasible to run the braiding in the background which randomly selects time in which it interacts with the braiding service. A certified smart device would ensure that membership is not transferred to an adversary when malware is of concern.

Adversary

We already excluded membership transfer as braiding is made software independent. The braiding should be reasonably robust as the number of participants in the braid can be made relatively small. Thus adversary would not be able to sabotage the braiding process by causing a significant impact either externally controlling the stability of the network connection or internally by pretending to have a network issue or received an incorrect braid and thus refusing to sign it. That, however, does not exclude adversary from learning identities of owners by controlling the braiding service (gatekeeper + mix) or by exclusion method knowing some owners of particular pseudonyms who participated in braiding with other members.

To model such a situation, we can consider that adversary has three sources of information

$$I_A = I_{public} \cup I_{services} \cup I_{owners} \tag{12}$$

I_{public} is the information which is published for everyone to be able to audit the state transitions. As an example, those could be a participant signed braids. I_{server} is the information which can be gained by exploiting the electronic voting protocol, for example, recording additional information in the gatekeeper and the mix. As well as gaining and exploiting the knowledge of traffic which I assumed to be inaccessible to an adversary when anonymization service such as TOR is used. And lastly, there is information on the pseudonym owners which adversary could have gained by having malware on the user's devices.

In contrast to gaining absolute anonymity set in one step, for braided anonymity, the information on the identity of the particular member is scrambled only between members who jointly participated in forming braids. In case for optimal gain braiding the number of participants, W is:

$$W = (N-1) \log_N |R| \tag{13}$$

which is a fairly small number which adversary with the help of malware would need to control to learn who is the particular pseudonym owner. The difficulty for the adversary (unless the goal is to spy on one famous person) is the fact that each pseudonym has a different set of members between which the information is scrambled and thus major breaches of pseudonym identities are difficult.

Nevertheless, the adversary could attempt to game the system in a way that all participants except one are adversary controlled for every braid. On the other hand, the group manager does have the ability to enforce a policy on selecting who braids with whom. Thus a legitimate question rises on what approach the group manager needs to use to prevent the adversary from gaining useful information on other members.

One of the most effective policies seems to be not to allow to braid one member consecutively but require that between braids some K members had braided in between. In this way, if adversary controls less than K members, others remain anonymous for the adversary in spite of collected information. Another method could be to require that every participant from the braid makes about equal anonymity set size gains and thus excluding adversary as his controlled members would have a finite capacity to participate in a low gain braidings.

Braiding System

The goal of the braiding protocol is to produce new pseudonyms for members with larger anonymity sets than the one with which member authenticated. Repeating the protocol multiple times would give large enough anonymity which could then be used to sign votes and to send them over an anonymous channel to a collection site (voting box). At the same time, no member should lose his/her membership if any or all other agents in the protocol conspire against him/her. The braiding protocol should succeed on average if a following set of assumptions are satisfied:

- Client software is free from malware/spyware. Breaking this assumption could end up losing the keys of the affected[1] and would compromise the privacy of the individual;
- IP address anonymizer allows unlinkable sessions (this excludes VPNs). The adversary shall not be able to infer that a person who at a given moment, is using traffic anonymizer is using it to perform braiding for a given group. Breaking this assumption would allow an adversary who spies on the network to do correlation attacks and thus uncover the owners of the pseudonyms.
- The publically made information is scrutinized, election officials are made accountable and if necessary, replaced by members on some previously agreed social contract.

Agents

Definition (guardian). A guardian is a pseudonym who is on the top of the hierarchy to configure braiding and other services, checks their availability, announce their configuration on a public bulletin board and register new members to the braidchain.

The sate is an outcome of multiple member registrations and their braidings. To audit the protocol, we need to list the transactions in public and append-only bulletin board which contains member registrations issued by the guardian (or subordinate) and braids signed by involved participants.

Definition (braidchain). A braidchain is a guardian delegated agent which is responsible for adding valid member registration transactions from the guardian and adding valid transaction braids from the braider to the transaction log. Provides a service which hosts the transaction log and a service which provides certificates for a braid being added or not going to be added to the transaction log. The data on the braidchain defines a state of the current members in the system $S_T \subset P$.

The braidchain, with its record, defines the member pseudonyms who can participate in the braiding protocol. The first pseudonym linked to the owner's real identity is approved by the guardian and submitted to braidchain as registration transaction. In the braiding protocol, the member participates with its pseudonym in order to produce braid which transfers membership to a new pseudonym generated anonymously within the braid during the protocol execution, which it stores locally.

[1]Signing and key generation could be done on separate certified hardware which then would protect keys from being extracted.

Definition (member). A member is an agent who owns the pseudonym whose identifier is listed in the current braidchain state S_T.

In principle, it does not matter who writes braiding transactions in the braidchain as those upon added are audited. It is, however, much simpler if the guardian and its subordinates only have writing permissions to the braidchain whereas members and everyone else can only read it and audit which we shall refer to as world.

Definition (world). The world is anyone who is interested in observing the integrity of the braidchain by auditing the transactions and receiving and dealing with complaints in the case of disputes.

Now lastly, we are set up to define the agent who organizes the braiding between members and its chosen mix service.

Definition (braider). A guardian delegated agent which is responsible for the successful execution of the braiding protocol. For successful operation, the braider operates with multiple states

- S_T a set of current members according to the braidchain.
- $HOLD \subset S_T$ a set of members which at a given moment are already taking part of the protocol and thus are not eligible authenticate another time.
- $WaitingList$ a waiting list with a subset of members and their established symmetric session keys.
- *Penalty* a set of members and their penalties expressed in time for those which had caused the braiding protocol to fail.

Definition (mix). A mix is a generally trusted agent which provides honest mixing services without leaking permutations to the braider.

Each agent owns a secret key for the pseudonym identifier, which is part of the system. To represent that we shall assume that a method $key(agent)$ is successful if it is executed on the system which owns the key.

Protocol overview

In this section, we shall look into an implementation of a software-independent braiding protocol which shall allow obtaining close to absolute anonymity for every member. The first part of the protocol is the configuration of services, announcement their configuration on a public bulletin board and registration of members where the later can happen at any time of the braiding protocol. We shall consider a specific braider with one separate mix agent, and we shall assume a following setup:

- Each member has the correct guardian identifier and configuration and corresponding digital signature configuration, allowing to validate messages signed by the guardian.

- The pseudonym of the braider, the agent location on the internet, anonymization service, together with other configuration is signed by the guardian and is published on a public bulletin board;
- Each member has an unlimited number of anonymous channels for sending and delivering messages.

The last part is significant to prevent linking of newly generated pseudonyms with old ones by keeping track of the IP address used. In practice, a TOR could be used where a new circuit is generated for every single braiding. To simplify the description, we shall not mention details of the key exchange protocol, but say that pseudonym is validated, for example, as part of a Diffie-Hellman key exchange protocol which allows establishing a secure connection.

Authetification

- The braider connects to the mix and validates that its pseudonym agrees with one specified in the bulletin board (signed by the guardian) and establishes a secure connection.
- A member $m \in S_T$ selects a random time and connects anonymously (through TOR) to the braider, validates its pseudonym and established a secure connection.
- On the other end braider validates the pseudonym of the client to be in S_T, secures the connection and puts it in the waiting list.
- When waiting list $WaitingList \subset S_T$ contains elements $M \subset WaitingList$ which satisfy a guardians specified **condition** the braiding starts with those M members.

The condition entails the number of participants required for a braid, a timeout period for the members who may have caused the previous braiding procedure to fail, and also a strategy which prevents for an adversary to use a small member coalition to unveil the gained anonymity as discussed in the previous section.

Braiding

- Each member connects through the braider to the mix, validates its pseudonym to be in the guardian maintained bulletin board and establishes a secure connection.
- Each member generates a pseudonym $q \in P$ in secret, stores its key (ownership) and sends the identifier to the mix.
- On the other end the mix collects $|M|$ messages, sorts them and sends the resulting list back to the braider.
- The braider receives the list, reinterprets that as a set of pseudonyms $Q \subset P$ and forms a braid.

As the braiding protocol did not produce any anonymity revealing information for external or internal parties as long as braider and mix do not collaborate, the anonymity set of each pseudonym in the list $q \in Q$ is $A_{S_T}(q|I) = M$. The last step is to confirm that the procedure was server-side software-independent to guarantee that no participant loses membership.

Authorization

- The braider sends the braid to every participant $m \in M$.
- Each participant $m \in M$ checks that his secretly generated pseudonym $q \subset P$ is listed in the braid $q \in Q$. If that is true makes the signature of the braid with m and sends it back to the braider.
- If braiding is successful, each member $m \in M$ delivers signature to the braider who forms a transaction and publishes that to braidchain. That officially announces the state change $S_T \to S_{T+1} = (S_T \setminus M) \cup Q$.
- As the members and world synchronize with braidchain, the new braids are audited for a membership transfer of his owned pseudonyms $O \subset P$. An index is built locally for the state index T_{grant} which grants membership for a pseudonym $o \in O$ and a state index T_{trans} which transfers the membership.

The result is that the member makes a secret map between the state number T and his owned pseudonyms $o \in O$. The last granted pseudonym can be used again repeating the braiding procedure which enlarges anonymity set in every braid according to eq. (5) which on average would require about a logarithmic number of braidings to get absolute anonymity with respect to a roster as evaluated in eq. (11). In practice, the member repeats the braiding protocol until a guardian specified anonymity set size threshold is reached, which then allows it to participate, for example, in voting.

Properties

To give the claimed properties of the protocol, we shall define an honest and responsible agent in the context of the protocol:

Definition (honest agent). An agent who does not deviate from the braiding protocol is free from spyware, and its clock is synchronized with a global time.

Definition (responsable agent). An agent who store safely their owned pseudonym keys and identifiers unaccessible to adversary.

With these definitions in place, I can state the claimed properties of the protocol in a finer precision:

- Honest and responsible member after participating in the braiding will always remain a member.

- If either braider or mix is honest the anonymity set after successful braiding is the union of participating member's anonymity sets.[2]
- If every participating agent in the braiding is honest, the protocol always succeeds.
- Dishonest agents can always be isolated and made accountable.
- Produced evidence as braidchain transaction log allows anyone to check that each honest and responsible member is legitimate to participate in further braiding.
- The braidchain allows to estimate potential adversary coalitions for each member which could have been capable of breaching his/her privacy.

Also, an attractive property is that eavesdroppers can be made as useful observers to audit agents of honest execution of the protocol, which is possible as messages between agents do not need to be encrypted. For example, that legitimate members can authentificate with the braider, that mixing is happening accurately at both ends and also allows to learn whether member or braider/mix was faulty for unsuccessful execution of the protocol.

Detailed description of the braiding protocol

Braiding Protocol

Secrets: Each participating member gnerates a random secret integer p_i
Private Inputs: Each participating member makes an input G^{p_i}
Common Input: blockhash of the braidchain, braid creation time, mix identifier
Common Output: braid transaction, braid record certificate

For the protocol, we shall employ the following set of assumptions:

- Random numbers generated locally are unpredictable.
- Each agent has secure means for synchronizing with a global time.
- Each member has the correct guardian identifier and corresponding digital signature configuration, allowing to validate messages signed by the guardian.
- IP address anonymizer allows unlinkable (this excludes VPNs) and anonymous sessions.
- The publically made information is scrutinized, the guardian is made accountable and if necessary, replaced by members on some previously agreed social contract.

The time is mainly introduced in this detailed protocol to allow members to synchronize the outcome of the protocol without revealing any previous state as that can be used for tracking. It also is useful to make guardian accountable on proving that it indeed had connected to a correct mix recently and also allows dismissing replay attacks quicker.

[2]Otherwise for an adversary who gains information it remains the same.

Symbol	Definition
$id(p)$	returns identifier of an agent or a signature p
$key(p)$	returns a key if the command is executed on a system which owns the key
$\|\|$	concatenites two objects into one
$A \to B : Msg$	sends messsage Msg from A to B
$A \overset{C}{\to} B : Msg$	a Msg which is transfered over an anonymous network with circuit C
$\{MSG\}_K$	A message which is HMAC authorized with a symmstric key K. In contrast to BAN usual notation there is no need for an encryption in the protocol.
G	A generator of a cyclic cryptographic group
$time()$	returns the current time on the agent on which the command is executed
$rngint()$	returns a secret random integer on the agent
$sort(R)$	returns a sorted list of elements in R which can wither be a list or a set

Figure 1: The notations used for describing the braiding protocol.

We shall use ordinary pseudonym identifier is G^s where s is a secret key or to parphrase in the notation above $G^{key(p)} = id(p)$. Also we shall assume that all agents uses the same cryptographic primitives. That allows to greatly simplify the protocol without affecting its security properties.

The setup assumes that the guardian publishes cryptographic parameters, specifications and configuration, which is necessary to check the integrity of the braidchain and participate in the braiding protocol. Theses parameters particularly include:

- G a generator for the cryptographic group
- T a timeout used for different parts of the protocol[3]
- The IP address anonymizer choice for the whole group.
- Network addresses of braider, braidchain and mix (optionally) as well as their pseudonym identifiers.
- The specification of cryptography primitives and methods which can be executed on any device. Those can be enlisted:
  - $sign(msg, key(p))$ issues a signature of the message with a pseudonym $id(p)$
  - $verify(msg, id(p))$ verifies a signature and returns $1, 0$
  - $hash(msg)$ calculates a hash of msg and returns it as an integer.

[3]For simplicity we shall assume that timeout is the same for all steps in the protocol. To break this assumption, one would add an index like T_i at every step where it is used.

- $HMAC(msg, key)$ returns authorization HMAC for a message for a key.
- $newcircuit()$ generates a new circuit for anonymous messaging.
- $braidset(W, B, P)$ where W is a subset of members $W \subset state(B)$, P is a penalty list and B is the braidchain. Returns a subset of W or $\varnothing$ if the set of members in W is not sufficient to form a braid.
- $threshold(W, B)$ where W is a subset of members $W \subset state(B)$ and B is a braidchain. A guardian specified algorithm which returns 1 if the braiding protocol can continue with a subset of members W.
- $lastbraid(bc)$ where $bc \in BraidChain$. Returns index of the last recorded braid in the braidchain.
- $blockhash(bc, N)$ where $bc \in BraidChain$ and N is an index of the braid on the braidchain. Calculates and returns a hash of the ledger between the given braid and the next one inclusive.

All this specification is signed by the guardian and published on public bulletin board which in practice could be hosted on the internet as an entrance point. The members would get to this entrance point anonymously at the beginning of the protocol which would prevent a malicious guardian from tracking individuals on their particular parameters used and also make it available for the world to check that the changes are not happening in the configuration too frequently so every honest member can get the same setup.

Authetication

step 1 ($Member \to Braider$) Member $m \in P$ whose identifier is in the present state of the braidchain $id(m) \in S_T$ generates a random integer $a \leftarrow \text{rngint}()$, calculates $A \leftarrow G^a$, sets timestamp $t_a \leftarrow \text{time}()$, isues a signature $s_a \leftarrow \text{sign}(A||t_a, \text{key}(m))$. Generates a circuit for anonymous messaging $C \leftarrow \text{newcircuit}()$ and sends a following message through the circuit to the braider:

$$Member \overset{C}{\to} Braider : A, t_a, s_a \tag{14}$$

step 2 ($Braider \to Member$) Checks that $|t_a - \text{time}()| < T$, $\text{id}(s_a) \in S_T \backslash Hold_T$, $\text{verify}(A||t_a, s_a) = 1$. Generates a random integer $b \leftarrow \text{rngint}()$, calculates $B \leftarrow G^b$, sets timestamp $t_b \leftarrow \text{time}()$, issues a signature $s_b \leftarrow \text{sign}(B||t_b, \text{key}(braider))$. Derives a symmetric key $K_{ab} \leftarrow A^b$, calculates a hmac $h_{ba} \leftarrow \text{HMAC}(B||A, K_{ab})$. Sends a following message to the member:

$$Braider \overset{C}{\to} Member : B, t_b, s_b, h_{ba} \tag{15}$$

step 3 ($Member \to Braider$) Checks that $|t_b - \text{time}()| < T$, $\text{id}(s_b) = \text{id}(braider)$, $\text{verify}(B||t_b, s_b) = 1$. Derives a symmetric key $K_{ab} \leftarrow B^a$ and checks the recieved hmac $h_{ba} = \text{HMAC}(B||A, K_{ab})$. Calculates a hmac $h_{ba} \leftarrow \text{HMAC}(A||B, K_{ab})$ and sends a following message to the braider:

$$Member \overset{C}{\to} Braider : h_{ab} \tag{16}$$

step 4 (*Braider*) Cheks that $h_{ab} = \text{HMAC}(A||B, K_{ab})$. Adds the connection to the waiting list $WaitingList_{T+1} \to WaitingList_T \cup \{(\text{id}(s_a), K_{ab})\}$ and pseudonym identifier to the hold $Hold_{T+1} \to Hold_T \cup \{\text{id}(s_a)\}$. Calculates a braid set from the members in the waiting list $B_{set} \leftarrow \text{braidset}(WaitingList_{T+1}, S_T, Penalty)$. If $B_{set} \not\subset \varnothing$ the braider proceeds to the braiding part. Otherwise waits for another member to join.

Braiding

step 1 (*Mix* $\to$ *Braider*) Generates a random integer $q \leftarrow \text{rngint}()$, calculates $Q \leftarrow G^q$, sets a timestamp $t_q \leftarrow \text{time}()$ and issues a signature $s_q \leftarrow \text{sign}(Q||t_q, \text{key}(mix))$. Sends a following message to the braider:

$$Mix \to Braider : Q, t_q, s_q \tag{17}$$

step 2 (*Braider* $\to$ *Mix*) Checks that $|t_q - \text{time}()| < T$, $\text{id}(s_q) = \text{id}(mix)$, $\text{verify}(Q||t_q, s_q) = 1$. Generates a random integer $d \leftarrow \text{rngint}()$, calculates $D \leftarrow G^d$ and derives a common key $K_{qd} = Q^d$. Calculates hmac $h_{dq} \leftarrow \text{HMAC}(D||Q, K_{dq})$ and sends a following message to the mix:

$$Braider \to Mix : D, h_{dq} \tag{18}$$

step 3 (*Mix* $\to$ *Braider*) Derives a key $K_{dq} = D^q$ and checks the received hmac $h_{dq} = \text{HMAC}(D||Q, K_{dq})$. Calculates hmac $h_{qd} \leftarrow \text{HMAC}(Q||D, K_{dq})$ and sends that to the braider:

$$Mix \to Braider : h_{qd} \tag{19}$$

step 4 (*Braider* $\xrightarrow{C} \forall B_{set}$) Checks received hmac $h_{qd} = \text{HMAC}(Q||D, K_{dq})$. Forwards to every participant of B_{set} parameters received from the mix:

$$Braider \xrightarrow{C} \forall B_{set} : \{Q, t_q, s_q\}_{K_{ab}} \tag{20}$$

step 5 ($\forall B_{set} \to$ *Braider*) Checks that $|t_q - \text{time}()| < T$, $\text{id}(s_q) = \text{id}(mix)$, $\text{verify}(Q||t_q, s_q) = 1$. Gnereates a new pseudonym $p \leftarrow \text{randint}()$ and calcualtes $P \leftarrow G^p$. Selects a random element from a group $K \leftarrow G^{\text{randint}()}$ and calculates $X \leftarrow K \cdot Q$, $Y \leftarrow P^{\text{hash}(K)}$. Sends the encrypted key X and encrypted pseuodnym identifier Y to the braider:

$$\forall B_{set} \xrightarrow{C} Braider : \{X, Y\}_{K_{ab}} \tag{21}$$

step 6 (*Braider* $\to$ *Mix*) Receives a set of pseudonym identifier encryptions $\{X_k, Y_k\}$ from $B^*_{set} \subset B_{set}$. If $\text{threshold}(B^*_{set}) = 1$ is satisfied the braiding protocol continues with B^*_{set}. Forms a secret ballot by sorting the pseuodnym encryptions $SecretBallot \leftarrow \text{sort}(\{X_k, Y_k\})$ and sends that to the mix:

$$Braider \to Mix : \{SecretBallot\}_{K_{dq}} \tag{22}$$

step 7 (*Mix* $\to$ *Braider*) For each encryption $(X_k, Y_k) \in SecretBallot$ calculates a coresponding key $K_k \leftarrow X_k^{N-q}$ and decrypts the pseudonym identifier $P_k \leftarrow Y^{N-\text{hash}(K_k)}$ where N is the order of the group G. Forms a ballot $Ballot \leftarrow \text{sort}(P_k)$ and sends it back to to the braider:

$$Mix \to Braider : \{Ballot\}_{K_{dq}} \tag{23}$$

Authorization

step 1 $(Braider \to \forall B^*_{set})$ Calculates the hash of the braidchain $H_{BC} \leftarrow \text{blockhash}(braidchain, \text{lastbraid}(braidchain))$, sets a timestamp $t_{br} \leftarrow \text{time}()$, and foms a $Braid \leftarrow (H_{BC}, t_{br}, ID_{mix}, Ballot)$. Sends that to each paticipant from the B^*_{set}:

$$Braider \overset{C}{\to} \forall B^*_{set} : \{H_{BC}, ID_{mix}, t_{br}, Ballot\}_{K_{ab}} \tag{24}$$

step 2 $(\forall B^*_{set} \to Braider)$ Checks that $\text{id}(mix) = ID_{mix}$, $|t_{br} - \text{time}()| < T$ and that $P \in Ballot$. Issues a signature of the braid $s_{Braid} \leftarrow \text{sign}(Braid, \text{key}(m))$. Sends the signature to the braider:

$$\forall B^*_{set} \overset{C}{\to} Braider : s_{Braid} \tag{25}$$

step 3 $(Braider \to BraidChain)$ Checks that $\text{id}(s_{Braid}) = \text{id}(s_A)$, $\text{verify}(Braid, s_{Braid}) = 1$. Recieves a valid s_{Braid} from every $B^{**}_{set} \subset B^*_{set}$. If succesful $B^{**}_{set} = B^*_{set}$ and a contract is formed from all received signatures $Contract \leftarrow \left\{s^k_{Braid}\right\}$ else $B^*_{set} \setminus B^{**}$ are added to the penalty list $Penalty$. Submits the braid transaction to the braidchain:

$$Braider \to BraidChain : Braid, Contract \tag{26}$$

step 4 $(BraidChain \to World)$ If braid transaction is valid (see below) it is appended to the braidchain which authorizes a state transition $S_{T+1} \to (S_T \setminus B^*_{set}) \cup Ballot$. Distributes new transactions to the world which audits them.

Braid Transaction
Is a result of verified multiparty computation where where subset of members $M \subset S_T$ produce an output $(Braid, Contract)$ such that:

- $Braid.h_{BC}$ is a valid hash of the braidchain $verify(braidchain, Braid.h_{BC}) = 1$;
- $length(Braid.Ballot) = length(Contract) = length(\{id(s) | s \in Contract\})$;
- For each signature $s \in Contract$ the $id(s) \in M$, $verify(Braid, s) = 1$;

which to be considered valid must be recorded in braidchain before $t_{br} + 2T$.

Synchronization

If the braiding protocol is aborted before participating member sends a signature on the braid s_{Braid} then the member can now for certain that membership won't be transferred. If the protocol ends for the member by sending a valid signature to the braider (either if it is accepted or not) the membership may or may not be transferred. To clarify this intermediate state member asks braidchain to issue a record certificate on whether particular braid is recorded or won't be recorded in the future.

step 1 $(\forall B^*_{set} \to BraidChain)$ Waits until $t_{br} + 2T < \text{time}()$. Asks for to the braidchain master whether $Braid$ is recorded:

$$\forall B^*_{set} \overset{C}{\to} BraidChain : Braid \tag{27}$$

step 2 ($BraidChain \overset{C}{\to} \forall B^*_{set}$) Checks that $Braid.t_{br} + 2T <$ time(). Looks into the list of transactions and if $Braid$ is part of it sets a $status \leftarrow 1$ else $status \leftarrow 0$. Forms a signature $s_{bc} \leftarrow \text{sign}(status||Braid, \text{key}(braidchain))$ and sends a following message to the participants from B^*_{set}.

$$BraidChain \overset{C}{\to} \forall B^*_{set} : status, s_{bc} \tag{28}$$

step 3 ($\forall B^*_{set}$) Checks that $\text{id}(s_{bc}) = \text{id}(braidchain)$, and $verify(status||Braid, s_{bc}) = 1$. The final result which ends the protocol is:

$$p, P, Braid, status, s_{Braid}, s_{bc} \tag{29}$$

Braid Record Certificate
Is a result of a query made to braidchain on whether a particular braid is recorded in the transaction log and thus giving information whether braiding between multiple involved parties was either succesfull or unsuseceful. The braid record certificate contains $(Braid, status, s_{bc})$ where

- $\text{id}(s_{bc}) = \text{id}(braidchain)$
- $\text{verify}(status||Braid, s_{bc}) = 1$

where $status = 1$ acknowledges that braid transaction is recorded in the braidchain permanently and $status = 0$ that a given braid will not be recorded in braidchain at any time in the future.

The braid record certificate determines whether membership transfer went successful to a new pseudonym. In case $status = 0$ membership transfer failed and the member can repeat the braiding protocol with the same pseudonym m whose identifier is thus still in the state $\text{id}(m) \in S_{T+1}$. On the other hand, if $status = 1$ the transaction is recorded to the braidchain permanently and thus the membership is transferred $m \to (p, P)$ and as $P \in S_{T+1}$ with which braiding protocol again can be repeated.

In case of a conflict situation where braid transaction is recorded, but braidchain had issued a braid record certificate for the transaction not to be recorded in the future and vice versa, the member can announce publically the last braid certificate produced. That gives a public proof of security breach in braidchain service for which guardian is accountable.

References

[1] I. Miers, C. Garman, M. Green and A. D. Rubin, "Zerocoin: Anonymous Distributed E-Cash from Bitcoin," 2013 IEEE Symposium on Security and Privacy, Berkeley, CA, 2013, pp. 397-411

[2] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, Madars Virza, Zerocash: Decentralized Anonymous Payments from Bitcoin, proceedings of the IEEE Symposium on Security & Privacy (Oakland) 2014, 459-474, IEEE, 2014

[3] Mauw S., Radomirović S., Ryan P.Y.A. (2014) Security Protocols for Secret Santa. In: Christianson B., Malcolm J. (eds) Security Protocols XVIII. Security Protocols 2010. Lecture Notes in Computer Science, vol 7061. Springer, Berlin, Heidelberg.

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

[5] 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

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

[7] 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.

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

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

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

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

[12] 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.

[13] Dimitris Gritzalis (2002) Secure Electronic Voting

[14] 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

[15] 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.

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

[17] 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.

[18] Juvonen, Atte. 2019. A framework for comparing the security of voting schemes.