BraidChains: a legitimate anonimity through pseudonym braiding

Janis Erdmanis

November 9, 2020

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

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

Braiding

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 pseuodnym 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 $S_T \to S_{T+1}$

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

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 protocol overview

Assumptions and Properties

- The clocks of honest participants are synchronized and the correct time can be btained with $time()$.
- Secure cryptographic primitives $hash$, $HMAC$, $sign$, $verify$, $randint$.
- 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;
- The mixes uses the same cryptographic parameters as the systems using their services. This assumptions is neceessary to make the protocol as coincise as possible.
- The pseudonym identifier is G^p where G is a generator and p is a secret key. This assumption is necessary for the paricular mix protocol we shall be using.
- 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

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

would allow adversary who spies on the network to do a corealation attacks and thus uncover the owners of the pseudonyms.

- The publically made information is scrutinized, the guardian is made accountable and if neceesary replaced by members on some previously agreed social contract.

In a remote electronic voting system the difference between a personalized hardware which does cryptographic operations and the voter starts to get blured. I sci-fi future we may experience cryptographic implants accelrating our brain power on such mundane operations or citizens who are no longer considered humans. In this repect I propose to consider voter as a cryptographicaly capable being who owns pseudonyms and can protect secrets. A way to ensure that the voter does protect his secrets and owns the pseuodnyms is an orthognal issue to the remote electronic voting system, which I shall address later.

The goal of the braiding protocol is to produce pseudnyms with high anonimity which can be used to sign votes which would be delivered to voting box over anonymous channel. Considering both braiding and actual voting as a single electronic voting ssytem I claim that it does offer a follwoing properties:

- Acountability. The evidence of a ballot conclusevly proves legitimacy, accuracy and fairness of the ballot. Only contoling the voter either in person or through malware adversary can change or vote in place of voter producing a valid evidence. A small adversary coalition can not paralize 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 certain vote. The voting protocol does not allow adverasary to record additional privacy revealing data without producing a suspicious evidence.

For the system we have atwo set of gaols. One with respect to the mebers themselves and the other set is with repsect to the external observers who want to validate that the members do indeed achieve their goals during the braiding protocol and that the anonymized pseudonyms are indeed legitimate without trusitng any agent except the memebers themselves.

Particularly for the memebrs there are two goals:

- Membership can't be lost to adversary during the execution of the braiding protocol even if adversary controls, writes code of every other agent in the system (except world). This property can be even achieved if the cleint hardware has malware as long as a certified device for pseuodonym gneration and braiding then by API it can be designed that keys never leaves the device. If that happens a proof against certifier is the device itself.

- The anonymity set of newly generated pseudonyms are the union of anonumity sets of coresponding particpnats int the braid if either braider or mix is honest. In case adversary controls both the anonimity set before and after braiding with respect to adversary remains the same.

In addition even an external observer from the publically available data can infer:

- That the current memeber pseudonyms are a product of the individuals in the rooster who had authorized the pseudonym changes by their previous memeber pseuodnyms.
- The memebership is not stolen by the adversary as long as a particular memeber is malware free, a fake pseudonyms added or an existing memeber removed without being explicit about it.
- The anonimity set of each memeber pseudonym and its change uppon if adversary had controlled a particular mix or a set of members. Also evaluate the types of adversary which could have succesfully participated in the system to uncover the privacy of the members.

Thus to sumarize theese goals the system we shall consider posseses a software independnace while provinding a strong privacy guarantees. It is fully accountable, transparent and privacy preserving.

Agents

The protocol has a different type of agents which we shall describe here and their role to the protocol:

- Guardian - an agent which sets up the system, anounces its parameters and maintains the services, for eample, checks that they are accessable from the interent.
- BraidChain - a guardian delegated agent which is responsable 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 (see the step 7 under authorization). The data on the braidchain is used to calculate a state of the current memebers in the system $S_T \subset P$.
- World - anyone who is interested in observing the integrity of the braidchain by auditing the transactions and receivieng and dealing with complaints in the case of disputes.
- Member - a person who owns the pseudonym whose identifier is listed in the current braidchain state S_T. The first pseudonym linked to the owners real identitiy is appoved by the guardian and submited to braidchain as registration transaction. In the braiding protocol the member participates in it with its keys of the pseudonyms which are created during execution of braiding protocol and stored locally in a *keychain* a datastructure with a fixed API for signing messages with its stored keys.

- Braider - a guardian delegated agent which is repsonsible for sucesful execution of the braiding protocol. For sucesful operation the braider operates with multiple states
  - S_T a set of current memebers 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 eligiable authetificate another time.
  - $WaitingList$ a waiting list with a subset of memebers and their established symmetric keys for the session.
  - *Penalty* a set of members and their penalties expressed in time for thoose which had caused the braiding protocol to fail.
- Mix - a generally trusted agent which provides honest mixing services without leaking mixing permuations 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 succesful if it is executed on the system which owns the key.

Public Buletin Board

For sucesful execution of the protocol every honest participant of the system needs to agree on a set of parameters and primitives used in the protocol. We shall assign guardian for this task which allows to narrow down to an immutable public buletin board which first contains:

- the cryptographic group $CryptoGroup$;
- the generator $G \in CryptoGroup$ from the cyryptographic group which is used for asymeteric cryptography - Diffie-Hellman key exchange and electronic signatures;
- sign and verify primitive specification;
- identifier of the guardian $ID_{guardian} \in CryptoGroup$.

This allows the guradian to issue the right security parameters by making a on them signature.

For instance we shall assume that the guardian hosts an entrance point to the system which provides its certified parameters for sucesful execution of the protocol which are:

- A timestamp of parameter distribution.
- Identifier of the braider $ID_{braider}$
- Identifier of the braidchain $ID_{braidchain}$
- A set of pseudonym idetifiers for the mixes $MIXES_{ID}$
- The IP address anonymizer, which is the same for a whoole group.
- HMAC primitive specification.

- Timeout T at which timestamped information is considered obselete (for simplicity I assume that it is the same for each step of the protocol).
- Netwoerk address of the braider, braidchain and mixes on the internet

All this specification is signed by guardian which thus can be made accountable, for any change made. In addition one may add a application specific parameters, for example, specifically for the voting and etc.

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 which can happen at any time of the braiding protocol — the person at the top hierarchy who does that we shall call guardian.

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

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 public and append-only bulletin board which lists all legitimate transactions which are signed by involved pseudonyms.

In principle, it does not matter who writes the 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.

The braiding protocol we will look into here is based on a simple mix based electronic voting system which is used to form a ballot from new member-generated pseudonym identifiers. To describe the protocol, we shall assume:

- The pseudonym of the mix and gatekeeper together with other configuration is made public bulletin board which is signed by the guardian;
- The guardian registers and keeps registering new members to the braidchain and thus updates the state S_T as braiding protocol proceeds or is repeated;
- Each member has the correct guardian identifier to obtain the most recent correct configuration from the 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 simply validated, for example, as part of a Diffie-Hellman key exchange protocol which allows establishing a secure connection.

Authetification

- The gatekeeper 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 gatekeeper, validates its pseudonym and established a secure connection.
- On the other end gatekeeper validates the pseudonym of the client to be in S_T, secures the connection and puts it in the waiting list.
- When waiting list $W \subset S_T$ contains elements $M \subset W$ 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 gatekeeper 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 gatekeeper.
- The gatekeeper 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 gatekeeper 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 gatekeeper 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 gatekeeper.
- If braiding is successful, each member $m \in M$ delivers signature to the gatekeeper 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 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.

Detailed description of the braiding protocol

- $id(p)$ where p is a pseudonym owner or a signature.
- $key(p)$ returns a key if the command is executed on a system which owns the key
- $||$ concatenites two objects into one
- $sign(msg, key(p))$ issues a signature of the message with a pseudonym $id(p)$
- $verify(msg, id(p))$ verifies a signature and returns $1, 0$
- $rngint()$ returns a random integer
- $HMAC(msg, key)$ returns authorization HMAC for a message for a key.
- $A \to B : Msg$ An ordinary BAN notation
- $A \xrightarrow{C} B : Msg$ a Msg which is transfered over an anonymous network with circuit C.
- $newcircuit()$ generates a new circuit for anonymous messaging.
- $sort(R)$ where $R \subset CryptoGroup$. Returns a sorted list of the gnerators in the set R.

- $\{MSG\}_K$. Sends 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.

- $braidset(W, B, P)$ where W is a subset of memerbers $W \subset state(B)$, P is a penalty list and B is the braidchain. Returns a subset of W or $\varnothing$ if the braidset is not sufficient.

- $threshold(W, B)$ where W is a subset of memerbers $W \subset state(B)$ and B is a braidchain. A guardian specified algorithm which returns 1 if the braiding prtocol can continue with a subset of memebers 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. Calcualtes and returns a hash of the ledger between the given braid and the next one inclusive.

-

Authetification

step 1 ($Member \rightarrow Braider$) Member $m \in P$ whoose 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:

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

step 2 ($Braider \rightarrow Member$) Checks that $|t_a - \text{time}()| < T$, $\text{id}(s_a) \in S_T \backslash H$, $\text{verify}(A||t_a, s_a) = 1$. 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:

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

step 3 ($Member \rightarrow Braider$) 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:

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

step 4 ($Braider$) Cheks that $h_{ab} = \text{HMAC}(A||B, K_{ab})$. Adds the connection to the waiting list $W_{T+1} \rightarrow W_T \cup \{(\text{id}(s_a), K_{ab})\}$ and pseudonym identifier to the holad $H_{T+1} \rightarrow H_T \cup \{\text{id}(s_a)\}$. Calculates a braid set from the members in the waiting list $B_{set} \leftarrow \text{braidset}(W, S_T, P)$. If $B_{set} \not\subset \varnothing$ the braider proceeds to the braiding part. Otherwise waits for another memeber 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 ($Braider \overset{C}{\to} \forall B_{set}$) Checks received hamc $h_q d = \text{HMAC}(Q||D, K_{dq})$. Forwards to every participant of B_{set} parameters received from the mix:

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

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 $p = \text{randint}()$ and calcualtes $P \leftarrow G^p$, $X \leftarrow P \cdot Q$. Sends the encryption X to the braider:

$$\forall B_{set} \overset{C}{\to} Braider : \{X\}_{K_{ab}} \tag{21}$$

step 6 ($Braider \to Mix$) Receives a set pseudonym identifier encryptions $\{X_k\}$ 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 $SecretBallot \leftarrow \text{sort}(\{X_k\})$ and sends that to the mix:

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

step 7 ($Mix \to Braider$) For each encryption $X_k \in SecretBallot$ calculates a coresponding pseuodnym $P_k \leftarrow X_k^{N-q}$ 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:

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

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 transaction is valid (see section on BraidChain) the transaction 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.

Synchronization

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 ($BraidChain \overset{C}{\to} \forall B^*_{set}$) Sets time $t_{query} \leftarrow \text{time}()$. Looks into the list of transactions and if $braid$ is part of the list sets a $row \leftarrow K$ else sets it $row \leftarrow 0$. Forms a signature $s_{bc} \leftarrow \text{sign}(t_{query}||row||braid, \text{key}(braidchain))$ and sends a following message to the participants from B^*_{set}.

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

step 3 ($\forall B^*_{set}$) Checks that $t_{br} + 2T < t_{query} < \text{time}()$, $\text{id}(s_{bc}) = \text{id}(braidchain)$, and $verify(t_{query}||row||braid, s_{bc}) = 1$. The final certificate which ends the protocol is:

$$p, P, braid, t_{query}, row, s_{braid}, s_{bc} \tag{29}$$

Note that in the synchronization step participants can ask to certify arbitrary braids to the braidchain at any time. Thus the protocol always ends up in a definite state.

Accountability and conflict resolution

World

If the braiding goes succesfull the braidchain records and anounces a transaction:

$$h_{BC}, t_{br}, ID_{mix}, ballot, contract \tag{30}$$

this transaction satisfies a certain properties:

- The hash h_{BC} is a valid hash of the braidchain $verify(braidchain, h_{BC}) = 1$.
- $length(ballot) = length(contract) = length(\{id(s)|s \in contract\})$
- For each signature $s \in contract$ the $id(s) \in S_T$, $verify(braid, s) = 1$ where $braid \leftarrow (h_{BC}, t_{br}, ID_{mix}, ballot)$.

Member

If member had sent a signature on the *braid* he/she obtains a follwoing certificate which decides whether braiding was succesfull and membership is transfered to a new pseudonym:

$$p, P, h_{BC}, t_{br}, ID_{mix}, ballot, t_{query}, row, s_{braid}, s_{bc} \tag{31}$$

To repeat, this certificate is valid if:

- $P = G^p$
- $P \in ballot$
- $t_{query} - t_{br} > 2T$
- $braid \leftarrow (h_{BC}, t_{br}, ID_{mix}, ballot)$, $id(s_{braid}) = id(m)$, $verify(braid, s_{braid}) = 1$
- $\text{id}(s_{bc}) = \text{id}(braidchain)$, $verify(t_{query}||row||braid, s_{bc}) = 1$

The value of *row* determines whether memebership transfer went succesful to a new pseudonym. In case $row = 0$ mebership transfer did not go succesful and the member can repeat the braiding protocol with the same pseudonym m whoose identifier is thus still in the state $id(m) \in S_T$. On the other hand if $row > 0$ the transaction is recorded to the braidchain to a particular row and thus the memebership is transfered $m \to (p, P)$ as $P \in S_T$ with which braiding protocol again can be repeated.

There may raise two conflict situations in a case the issued certificate does not reflect the state in the braidchain making authetifiaction to fail for the next braiding. In case the membership is transfered then member can publically anounce a certificate:

$$h_{BC}, t_{br}, ID_{mix}, ballot, t_{query}, row, s_{braid}, s_{bc} \tag{32}$$

which proves that a membership to a a psueodnym $P \in ballot$ is granted and must be reflected to braidchain.

In the opposite situation memebrship is not transfered to a new pseudonym according to issued certificate, but had changed the state in the braidchain. In this situation memeber can anouance all failed braiding certificates issued by the same pseudonym $id(m) = s^i_{braid}$:

$$h^i_{BC}, t^i_{br}, ID^i_{mix}, ballot^i, t^i_{query}, 0, s^i_{braid}, s^i_{bc} \tag{33}$$

where one of the ballots should contain a valid transaction recorded to the braidchain. In case none of them is the braid which made a valid transaction in the braidchain it could be either becauser the memeber is dishonest or the key had been stolen. To prevent both of thoose cases a certified hardware can be required to be used with a constrained API which does not allow to sign a a braid which does not contain an owned pseydonym.