Initial Ideas

1 Subsenate based RSA class-group

The original Subsenate proposal employs Merkle trees which require heavy cryptographic machinery such as zk-SNARK. A possible improvement of such a scheme is to use RSA accumulators based on RSA class-group [?]. This allows a more light weight traditional cryptography proof systems such as Sigma protocol.

2 blockchain for voting (conclusion from walk)

If the blockchain is used only for the voting system then monitary transactions are not needed, only consensus. In particular, the order (among the voters) does not change the results, and "double spending" is not an issue. The things that matter are that actions should be authenticated and signed. In addition "consensus" should be created from the union of actions (and not the intersetion), that is, if one "miner" sees a transaction, but the other miners don't see this, then this transaction should be added to the log of the "blockchain"

3 Creating infrastructure for dispute resolution (conclusion from walk)

For dispute resolution to work, each user should have a unique public key pk_i that can be generate from a shared (by anyone) public key pk. The unique private key sk_i however should not be derived from the shared private key sk

How to create a hierarchical voting system (i.e., hierarchical mix, hierarchical count, etc.)

4 Coersion resistance - walking conclusions

Any coersion resistance scheme has to include a secret before voting takes place. In my view this secret has to be shares across multiple players or using a trusted hardware or generate by itself in an untappable location, and prove to other actors that it possess the correct secret.

4.1 A proposal for CR

. Submit the tuple $(E_1(id, r_1^1), E_2(vote, r_2^1), t_0))$, where t_0 is the exact time that the tuple was casted (which must be verified in the BB). After casting this vote, the voter sends this tuple to the coercer. But then it cast another vote $(E_1(id, r_1^2), E_2(vote, r_2^2), t_0))$.

4.2 walking conclusions

1. The mix network would benefit by a proof of CR (i.e., that it would be impossible to convince someone else what has been submitted) Proposal for how to achieve CR - following the previous

idea (of two encryptions of the same ID), at the validation phase (which should occur after a mix) the voter invalidate the later ID by showing that there are two encryptionsq1 of the same ID and Invalidate the first one (note that the time itself) . Let E_1, E_2, E_3 be an encryption schemes for which the decryption can be reconstructed by collaboration of the servers (i.e., the decryption key is distributed) t be time of casting the vote, $\hat{t}$ be the time the coerced vote has been casted. The voter cast two ballots $c, \hat{c}$ to the bulleting boards, where c is the valid ballot and $\hat{c}$ is the invalid (coerced) ballot. Let L be a list of the eligible id's that are allowed to vote. Let ID be the id of the voter. The protocol is described as follows:

   (a) The voter prepares the real vote $c = (E_1(v), E_2(E_3(ID, r_1), proof_1, r_2), t)$ where $proof_1$ proves that ID is in L. fake vote $c =$

2. Perfect self mixing, that is a distributed computation that result in a perfect permutation (each voter selects at random a different row) is not always necessary. Voter can add a unique tag to her vote and then can recognize her vote even if there are multiple voter that choose the same row. This will prevent coercion resistance (CE), but perhaps the entire method is not CE.

3. Two ways to instantiate CE voting.

   (a) First is to use private credentials (i.e., secret key) which is issued before casting the vote. The private credential is used for encrypting the vote. This need to have the property that it can be faked (including the public credential which has to match public credential of another voter, as otherwise it can be exposed as fake) in the presence of coercer. Then after the vote is casted and a mixnet is applied, the public credential should open it to the real vote (where the coercer cannot trace even if it open).

   (b) Second option is using revoting. In this case one vote is used for showing the coercer. Then the voter use a second vote and a reference to the first vote, in order to nullify the first. The second vote used another public key. In this case there are two options: 1. each voter has unknown number of public keys recognized by the authority (think of a senarion where each voter get a unknown random number of public key (i.e., at the range 1-20)). The second option is to use a demi vote, encrypted with a fake credential, and the revote is encrypted with a real credential. The revote also refers to the first cast in order to nullify it (question, can it use an arbitrary fake public credential?)

4. Self-mixing: I would like to have the following property:

   (a) Each player encrypts a secret s_i so that a location the lies in a pre-specified range, i.e., $s_i \in [0, L]$. The encryption is homomorphic so that when the mixers combine the encryptions and decrypt, they get whether it was chosen by a single voter or more. If it was chosen by more than one (or more then some privacy threshold) then the mixers can apply some function f on the public encrytion (and known public key) so that $f(E_{pk}(s_i)) = E(s_{i+1})$ and $s_{i+1} \in [0, L]$.

   (b) Let $B = \{0,1\}^n$ be a vector of bits, where $B^j \in \{0,1\}$ denote the j index of the b's vector. I need to find some transformation f that allows a field element/group element to be mapped to encyption of bits, so that given field/group element $s_i = \sum_j 2^j B_i^j$ then $E(s_i) = \sum_j 2^j E'(B_i^j)$ and for every j, $f(E'(B_i^j)) = E(B_{i+1}^j)$ where $s_{i+1} = 2^j \sum B_{i+1}^j$

   (c) Continue: as s_i previous idea. Suppose that s_i is prime in the range $[1, L]$ Each player sends an encryption $c_i = E_{pk}(r_i, s_i)$ with a proof that r was used in c_i, e.g., use El-Gamal (g^r, h^{r*s_i}) and some proof that r is used there. Again, the reshuffle can be by

applying public transformation function f. Transformation f might be simply multiplying by $E_{pk}(r_i)$, that is you the mixers can obtain the next encrypted location at time t+1 by multiplying the cipher text with $E(r_i)$, i.e., $c_i^{t+1} = f(E_{pk}(r_i, s_i^t)) = E_{pk}(r_i, s_i^t) * E_{pk}(r_i)$. Computing the product of all players $C = \prod_i c_i$ and if C is not the product of all primes (or some of them as can be check by a computer). Such an encryption might be a combination of El-Gamal, RSA, or Pallier (need to verify that by calculations) and taking mod L (need to see how to combne it with the vote). To do it deterministically, again we can Encrypt the bits $E(b_i, r_i)$ and shuffle by $E(b_i, r_i) * E(r_i) = E(0, r_i) or E(1, r_i)$ and a proof that that is a single i such that $E(b_i, r_i) * E(r_i) = E(1, r_i)$ by connecting it to s_i

5. Public shuffling. The task: for each voted decide over public permutation. Naively there are many permutation possible $(n!^n)$.Point to consider:
   - Show that the expected number of shuffles in log(n). You can do this by calculating the expected number of 1 in each level. To show a log number it is enough to show that at least constant fraction is 1.
   - Note that a row of 0 (and clearly a row of n) exposes that all entires are 0. In this case probabliy the best thing to do for provacy is map each entry in the row to a unique row
   - Each iteration there is a privacy loss - for any row with sum i you know that i entries are 0 and n-i are 1, e.g., you know that from x11,x21,x31 one is 1 and two are 0. Since such correlations are created every iteration, the smaller privacy you get
   - The servers need to compute the post probability after every iteration, given all the past correlations (an not only the previous correlation. This is not Markovian!)
   - it make to do this method only in the begninig for few iteration, when the privacy is not completely eliminated
   - a privacy for each voter should be caculated separately
   - deciding the permutaion. This is not an easy task. It should love over all previous permutations (and perhaps computiongs also condional prob,). A heuristic can be to find the permutation that maximize the entropy, which is defined by $k \in [n] \sum p \log(p)$ where $p \in \{0, 1\}$

5 ZKproofOfNelements

How to prove that you know n elements in a commtiment to a set? The prover may either send a commitment to the n element ot an accumulator, along with some proof. Two possible avenues:

1. Possible answer: commit to a polynomial of degree n that repesent the set with n elements. This polynomial perhaps need the property that any polynomial of degree m ¡ n that divides the origianl polynomail is a commtiment to the subset of elements it the original set.
2. Another possible way is to enumerate the sets from 1 to n. And then to encode each element in the set as an element bit as encyption of 0 or 1 and then to provide zk proof that the encryption is correct and that the sum of encyptions is an ecnryption of n.

6 open question I heard

- How to prevent a election creator for giving exuse that the setup was wrong (I guess a threshold of trust may suffice but perhaps I should check that again)

7 open questions form myself

- PET with sets and not single elements
- A public mpc for "private set intersection" which proves to the public the intersection of the set