Notes on SBT weighting

William George

In these notes we will analyze various methods of weighting Kleros juror selection by soulbound tokens (SBTs) that jurors may possess. SBTs were introduced in [3] as a tool to capture identity characters of web3 users and to enable a greater space for non-financial applications. In [3] and [2], mechanisms are considered to weight participants' contributions to quadratic funding schemes based on their SBTs. Motivation for such mechanism comes from having more plural involvement from a community, giving extra weight to diverse perspectives. Doing so requires have weighting mechanisms that are resistant to the forms of implicit collusion that come from having disproportionate involvement from a few users with similar backgrounds. These goals are also relevant to Kleros juror selection as social science research has shown that diverse panels can, in some circumstances, produce better decisions that homogeneous panels of high-performing individuals. Moreover, robust defenses against implicit collusion can also make a mechanism more resistant to the (explicit) collusion in many attack vectors. For a broad overview of these ideas, see [1].

1 Notation

We denote the set of participants by N and we denote by S the set of SBTs. For each participant $i \in N$ we denote by x_i the stake of i and T_i the set of SBTs held by i. Hence $|T_i|$ is the number of SBTs held by i.

2 Clusterweighting

This formula is inspired by the Clusterweighting formula in [2]. The basic idea is that the each SBT corresponds to a set to which users can belong. Then the participant i is seen as belonging to each of the $|T_i|$ sets for which she has SBTs with equal weight $1/|T_i|$. For a given SBT $s \in S$, we write that $i \in s$ to indicate that i possesses the SBT s to underliine this point of view of considering the SBTs as sets.

The contribution of each group is put under a square root so that groups with large contributions have diminishing returns. So one can think of the total contribution as

$$\sum_{s\in S}\sqrt{\sum_{j\in s}\frac{x_j}{|T_j|}}$$

.

However, this quantity partitions naturally by user as follows:

$$\sum_{s\in S}\sqrt{\sum_{j\in s}\frac{x_j}{|T_j|}} = \sum_{s\in S}\sum_{i\in s}\frac{\frac{x_i}{|T_i|}}{\sqrt{\sum_{j\in s}\frac{x_j}{|T_j|}}} = \sum_{i}\sum_{s\in S:i\in s}\frac{\frac{x_i}{|T_i|}}{\sqrt{\sum_{j\in s}\frac{x_j}{|T_j|}}}$$

So one can think of the i's portion of the total weight as being

$$\sum_{s\in S:i\in s}\frac{\frac{x_i}{|T_i|}}{\sqrt{\sum_{j\in s}\frac{x_j}{|T_j|}}}$$

.

Then we give the chances of i being drawn in a given draw under the Clusterweight formula as:

$$\frac{\sum_{s\in S:i\in s}\frac{\frac{x_i}{|T_i|}}{\sqrt{\sum_{j\in s}\frac{x_j}{|T_j|}}}}{\sum_i\sum_{s\in S:i\in s}\frac{\frac{x_i}{|T_i|}}{\sqrt{\sum_{j\in s}\frac{x_j}{|T_j|}}}} = \frac{\sum_{s\in S:i\in s}\frac{\frac{x_i}{|T_i|}}{\sqrt{\sum_{j\in s}\frac{x_j}{|T_j|}}}}{\sum_{s\in S}\sqrt{\sum_{j\in s}\frac{x_j}{|T_j|}}} \tag{1}$$

3 Eigenweighting

In Section 2, users are broken into sets based on the SBTs they have, with each SBT as essentially being seen as a set, allowing for the possibility that a user will belong to multiple sets if she has mulitiple SBTs. In a way, this is already an example of a simple clustering algorithm. In this section, we explore the possibility of using modern clustering algorithms to partition the users based on their SBTs. Particularly, we will discuss using spectral clustering which is based on the eigenvectors of the graph Laplacian of the adjacency matrix of the graph of users and their SBTs, see [4]. Hence we call this approach "Eigenweighting". A similar idea for quadratic funding was evoked as a possible direction of future work in [2].

Generally, the approach here is to consider the union of the sets of users and SBTs as the vertices of an undirected graph G. An edge is drawn between a user i and an SBT s if i holds s, i.e. if $i \in S$. Then the clustering algorithm produces a set of clusters, which we denote by C. For a given cluster $c \in C$, we denote the complement of that cluster, namely all vertices corresponding to users and SBTs that are not included in c, by $\bar{c}$. For two subsets $A, B \subseteq G$, we denote

$$W(A,B) = \sum_{i\in A}\sum_{j\in B}\mathbf{1}_{\text{edge between } i,j},$$

which gives a sense of how densely A and B are connected.

As described in Section 5 of [4], spectral clustering algorithms approximate discrete graph cut algorithms that break the graph into the required number of clusters in such as a way that

- the number of edges between the clusters is small and
- the "size" (see below) of the clusters is not too small.

The second condition is here to make sure that the algorithm does not select a "trivial" clustering where most of the clusters only have a few vertices. The two ways to measure the size of the cluster that are discussed in [4] are the number of vertices in the cluster and the number of edges in that cluster. These give rise to the RatioCut and the NCut respectively:

$$RatioCut(c_1, \dots, c_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(c_i, \bar{c_i})}{|c_i|} \tag{2}$$

$$NCut(c_1, \dots, c_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(c_i, \bar{c_i})}{\sum_{j \in c_i} \sum_l \mathbf{1}_{\text{edge between } j,l}} \tag{3}$$

Then ideally, one choose the clusters that minimize these quantities. Unfortunately, this problem winds up being NP-hard; hence standard spectral clustering algorithms only approximate this problem. See [4] for further information on how these approximations are done.

Once one has assigned the users $i \in N$ into clusters, one can imitate the drawing formula of Section 2, giving each participant i of being drawn in a given round a chance of:

$$\frac{\sum_{c \in C : i \in c} \frac{x_i}{\sqrt{\sum_{j \in c} x_j}}}{\sum_i \sum_{c \in C : i \in c} \frac{x_i}{\sqrt{\sum_{j \in c} x_j}}} = \frac{\sum_{c \in C : i \in c} \frac{x_i}{\sqrt{\sum_{j \in c} x_j}}}{\sum_{c \in C} \sqrt{\sum_{j \in c} x_j}} \tag{4}$$

A formula that would capture even more subtleties of the structure of SBT membership of the participants might be to use soft clustering techniques - namely, to output a probability that a given user is in each cluster and then

$$\frac{\sum_{c \in C} \frac{x_i \text{prob}(i \in c)}{\sqrt{\sum_{j \in c} x_j \text{prob}(j \in c)}}}{\sum_{c \in C} \sqrt{\sum_{j \in c} x_j \text{prob}(i \in c)}} \tag{5}$$

In these notes, we will mostly work with the drawing formula of equation 4 to avoid the details of dealing with an explicit implementation of soft clustering techniques. However, some of the heuristics we observe might still be expected to apply in the soft clustering case.

4 Initial comparison of Clusterweighting and Eigenweighting

We consider the following attack models:

- Attack model 1: For attacks that seek to manipulate the partition of profiles into groups, we generally imagine an attacker who is capable of attributing fake SBTs to her own profile(s) or potentially also to other (honest) profiles.
- Attack model 2: We assume that an attacker is capable of the actions of Attack model 1 and is also capable of creating Sybil profiles.

In both cases, we imagine that there is an economic cost for an attacker to create a Sybil profile or to attach a fake SBT to a profile. Depending on the application considered, relative costs of different attacks may vary. For example, one might have a situation where creating a Sybil profile has a more or less fixed cost regardless of how many Sybil accounts have already been created (e.g. the cost of hiring someone on the street to record a PoH video for you). Alternatively, there may be situations where if the attacker can break the system and produce Sybil profiles, there is an important upfront cost to produce any fake profiles but then additional profiles have negligible costs. The creation of fake SBTs can have similar dynamics.

Heuritistically, it seem that the ability for an attacker with the capacities of Attack model 1 to increase her weight under 4 by manipulating the partition of profles under Eigenweighting is limited. Indeed, recall that Spectral Clustering approximates the partitions that optimize graph cuts that penalize a candidate group with few vertices. Hence, even if an attacker can assign SBTs at will, if she only has one profile it will be in only one group and she would have to attribute many fake SBTs to create enough weight between the other profiles to overcome the penalty for small groups for her profile to be the only vertix in that group.

Notice that an attacker can potentially separate profiles who in the "honest" partition, absent her attack, would be in the same group into different groups by giving them fake SBTs.

In contrast, Clusterweighting cuts the stakes of participants into "groups" that correspond to each SBT. As long as the attacker is not capable of removing honest SBTs from other users, she is not capable of removing them from the groups they are in by default. On the other hand, by giving a single malicious profile many fake SBTs, she can control all of the contributions in the corresponding square roots for those SBTs in 1.

Clusterweighting	Eigenweighting
Cannot remove users from honest groups, only create new groups	Possibe to both create malicious groups and remove honest groups
No need for Sybil identities to gain disproportionate weight	Can only control as many groups/roots as have profiles

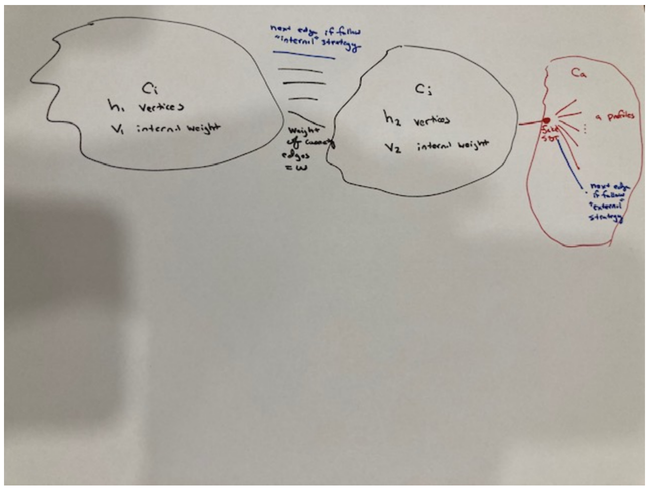
next edge if follow "internal" strategy
c_i
h_1 vertices
v_1 internal weight
weight of connecting edges $= w$
c_j
h_2 vertices
v_2 internal weight
c_a
Sybil SBT
a profiles
next edge if follow "external" strategy

Figure 1: The attacker considers two strategies to use her resources to place the next malicious edge in one of two positions. Her goal is to have the clustering algorithm separate c_a as its own cluster and combine c_i and c_j.

5 Effectiveness of different strategies to alter groups

Suppose that we use Eigenweighting to determine juror weighting. For a given draw, consider the distribution of SBTs that one has absent any attackers. This leads to some assignment of clusters $c_1, \ldots, c_k$.

Then, suppose that an attacker with the abilities of Attack model 2 wishes to break off a collection of her own Sybil profiles in a separate cluster. There are two natural strategies that we have identitified towards this end:

Attack Strategy 1. *The attacker can attribute a fake SBT to her Sybil profiles. The idea here is that this collection of Sybil profiles, which seem similar as they all have the fake SBT, can break off and form a new cluster.*

Attack Strategy 2. *The attacker can attribute a fake SBT to honest participants. The idea here is that this will make honest profiles, which are normally not grouped together as they do not share significant similarities, seem more similar so that the a collection of the attacker's profiles seems more divergent by comparison and is more likely to be selected as a separate cluster by the algorithm.*

See Figure 5 for a comparison of the locations where the attacker would put additional edges under each of these strategies.

Then the question is, under what circumstances should an attacker with bounded resources to attribute fake SBTs pursue one strategy or the other? An attacker that has the ability to place several fake edges in the graph may pursue both strategies; then the question might be how to dedicate the resources required for the next edge at a given moment.

We consider a few heuritistics to simplify this situation. We will analyze the effects of the two attacks above on the formulas for RatioCut and NCut in equations 2 and 3 respectively. Note that this analysis will only give us approximate information about the outputs produced by spectral clustering algorithms. Indeed, as we noted above, spectral clustering algorithms only give an approximation of the problem of finding clusters that minimize RatioCut and NCut as this discrete optimization is NP-hard.

We will consider just the scores of two possible sets of clusters under these formulas: the cluster where all vertices are in the same clusters as one has in the absence of attackers, $c_1, \dots, c_k$, or a clustering where two of these clusters are combined c_i and c_j and there is an additional c_a given by profiles of the attacker that all share a single SBT and have no other edges.

We suppose cluster c_i has h_1 vertices and internal weight

$$\sum_{t \in c_i} \sum_{s} \mathbf{1}_{\text{edge between } s,t} = v_1.$$

Similarly, c_j has h_2 vertices and internal weight

$$\sum_{t \in c_j} \sum_{s} \mathbf{1}_{\text{edge between } s,t} = v_2.$$

We denote the weight of edges between the clusters as

$$\sum_{t \in c_i} \sum_{s \in c_j} \mathbf{1}_{\text{edge between } s,t} = w.$$

Suppose the attacker has already attributed herself $a-1$ profiles with a given fake SBT that form the alternate candidate cluster c_a consisting of a vertices.

Then the two terms in the formula for the RatioCut corresponding to the two clusters under consideration, depending on whether the attacker places the internal or external edge are given as:

	c_i, c_j	c_i and c_j merged, a
internal	$\frac{w+1}{h_1} + \frac{w+1}{h_2+a}$	$\frac{1}{h_1+h_2+1} + \frac{1}{a}$
external	$\frac{w}{h_1} + \frac{w}{h_2+a+1}$	$\frac{1}{h_1+h_2} + \frac{1}{a+1}$

One can define

$$f(strategy) = RatioCut(c_i \text{ and } c_j \text{ merged, } a|strategy) - RatioCut(c_i, c_j|strategy).$$

Then the goal is to find which of the two strategies minimizes f as this means that the algorithm is that much closer to selecting the attacker profiles as a separate cluster.

If one has $h_1 = h_2 = \frac{h}{2}$, we see that

$$f(internal) \leq f(external) \Leftrightarrow \frac{1}{h/2} + \frac{1}{h/2+a} + \frac{1}{h} - \frac{1}{h+1} \geq \frac{1}{a} - \frac{1}{a+1}.$$

For large h, we get equality here for $a \approx \frac{1}{2}\sqrt{h}$.

Thus an attacker should place roughly $\frac{1}{2}\sqrt{h}$ fake SBT edges among her own profiles before placing fake SBTs among other (honest) participants.

We can perform a similar analysis for NCut. Now the two terms in the formula corresponding to the two clusters under consideration, depending on whether the attacker places the internal or external edge are given as:

	c_i, c_j	c_i and c_j merged, a
internal	$\frac{w+1}{v_1+w_1+1} + \frac{w+1}{v_2+w+2a+1}$	$\frac{1}{v_1+v_2+2w+3} + \frac{1}{2a+1}$
external	$\frac{w}{v_1+w} + \frac{w}{v_2+w+2a+2}$	$\frac{1}{v_1+v_2+2w+1} + \frac{1}{2a+3}$

We now define

$$f(strategy) = NCut(c_i \text{ and } c_j \text{ merged}, a|strategy) - NCut(c_i, c_j|strategy).$$

Again, the goal is to find which of the two strategies minimizes f as this is the more effective strategy for the attacker.

Even more heuristically for the moment, if we imagine that $v_1 \approx v_2 \approx \frac{v}{2} >> a$, then we ignore $+1$ terms in the denominators that are accompanied by v_1 or v_2. Then,

$$f(internal) \leq f(external) \Leftrightarrow \frac{1}{v/2+2} + \frac{1}{v/2+w+2a+1} \geq \frac{1}{2a+1} - \frac{1}{2a+3}.$$

Then, we get equality here for $a \approx \frac{1}{2}\sqrt{\frac{v}{2}+w}$.

Note that when comparing using RatioCut and NCut for Clusterweighting, one would generally expect to have $v >> h$ unless the graph of SBTs is very sparse. Indeed, h can be thought of being decomposed into the sum of the number of SBTs and the number of users in the two clusters c_i and c_j:

$$h \approx \frac{2}{k}(|S| + |N|).$$

On the other hand, if a typical cluster has $|N|/k$ participants and $|S|/k$ SBTs, then

$$v_1 \approx v_1 + v_2 \approx 2\frac{|N|}{k} \cdot \frac{|S|}{k} = \frac{2|N| \cdot |S|}{k^2}.$$

Thus, if a typical user has many SBTs, NCut would require a higher number of profiles in the attacker cluster before it becomes worthwhile for the attacker to attribute internal edges. Depending on what the two different attacks are expected to cost an attacker this may or may not offer greater attack resistance. If a typical user only has a small number of SBTs that is more or less constant with respect to the growth of the total network of users and SBTs, then the two approaches do not necessarily offer significant differences.

6 Comparison of attack resistance for Quadratic Funding versus Kleros drawing

Recall that, when applying these ideas to quadratic funding, the goal of an attacker is to extract subsidies from the matching funds. Namely, the attacker wants to maximize

$$\left(\sum_{c\in C}\sqrt{\sum_{j\in C}x_j}\right)^2 - \sum_{c\in C}\sum_{j\in C}x_j.$$

See [3] and [2] for further detail. If we consider $\sum_{c\in C}\sum_{j\in C}x_j$ to be fixed, so the attacker can change how her contributions are allocated between groups, but the total contributions made by her and the honest participants is fixed, then this is equivalent to the attacke maximizing

$$\sum_{c\in C}\sqrt{\sum_{j\in C}x_j}.$$

In Section 5, we considered various ways that an attacker might try to influence the groupings produced by the clustering algorithm by attributing malicious SBTs. These attacks on the clustering algorithm are present whether one intends to use the produced clusters for applications of SBTs to quadratic funding or to juror drawing Kleros. However, once an attacker has obtained the ability to stake on different profiles that are clustered into different groups, there is also the question of how much contribution/stake she should attribute to each profile. In this section, we will observe that the optimal attack strategies for attacks on quadratic funding diverge from those for attacks on Kleros juror drawing.

We see that for quadratic funding, attackers have a relatively straightforward strategy for how much she should contribute from profiles that are clustered into different groups. She essentially wants each incremental contribution she makes to be in the group that has the smallest total contribution. Indeed, note that if we take $x, a, b \geq 0$ thinking of x as the attacker's incremental contribution that she might want to put in one group or another and a and b as being the existing contributions in those groups,

$$\sqrt{x+a}+\sqrt{b} \leq \sqrt{x+b}+\sqrt{a} \Leftrightarrow b \leq a.$$

On the other hand, it is not optimal for an actor attacking juror drawing in Kleros to put her incremental stake in the groups that have the smallest total stake. Essentially, the attacker has the tradeoff between:

- Trying to separate her stake from the stakes of other participants by putting it in groups that have as little other stake as possible. Then, when applying the square root of the total stakes in each group to determine the relative weights of the groups essentially calculating the denomoniator of equation 4, the attacker's stake in a group that benefits from taking the square root relative to the others.

- Putting her stake in a group with the stakes of other participants so that the other participant's group has more stake in it and then applying the square root gives those groups less weight relative to others.

Example 1. *Suppose that the clustering algorithm has partitioned the graph of users and SBTs into three partitions. Honest users have put contributions of 5, 1, and 0 into these partitions respectively, and the attacker can contribute 4 into either the second or third partition. Then the attacker manages to extract the maximal amount from a quadratic funding algorithm by putting her contribution in the third partition where there are no honest contributions as*

$$\left(\sqrt{5}+\sqrt{1+4}+\sqrt{0}\right)^2-(5+1+4) = 4.472 < \left(\sqrt{5}+\sqrt{1}+\sqrt{4}\right)^2-(5+1+4) = 17.416.$$

On the other hand, if one has the same clusters and the above values are the stakes into the different partitions, the attacker achieves a higher weight by staking into the second partition as

$$\frac{\frac{4}{5}\sqrt{5}}{\sqrt{5}+\sqrt{1+5}+\sqrt{0}} = .4 > \frac{\sqrt{4}}{\sqrt{5}+\sqrt{1}+\sqrt{4}} = .382.$$

To get a sense of the optimal distribution of stake between different groups, consider a situation where the SBTs and users are partitioned into two partitions where $B > 0$ honest stake is staked in the first partition and there is 0 honest stake in the second partition. Suppose the attacker can freely attribute a total of $A > 0$ stake in some combination between the two partitions. Then the attacker wants to choose $x \in [0, A]$ so that when she puts $A - x$ stake in first partition and x stake in the second partition, that maximizes

$$\frac{\frac{A-x}{\sqrt{B+A-x}}}{\sqrt{B+A-x}+\sqrt{x}} + \frac{\sqrt{x}}{\sqrt{B+A-x}+\sqrt{x}}.$$

This is achieved at

$$x = \frac{2-\sqrt{2}}{4}(A+B)$$

which gives a maximum of

$$\frac{A-2\sqrt{2}B+3B}{A+B}.$$

In general, one can produce complicated such formulas to give the optimal stake for an attacker depending on the existing honest stakes.

Note that, if an attacker can freely allocate her stake between all partitions, she can put a proportion of her stake in each partition that corresponds to the proportion of honest stake in that partition. Take s_c to be the honest stake in each cluster $c \in C$, so

$$\sum_{c \in C} s_c = H$$

is the total honest stake. Suppose the attacker has a total stake of A to allocate. Then, we are supposing that the attacker allocates $A \cdot \frac{s_c}{H}$ to each cluster c, so that her chances of being drawn given by Equation 4 are given by:

$$\frac{\sum_{c \in C} \frac{A \cdot \frac{s_c}{H}}{\sqrt{s_c + A \cdot \frac{s_c}{H}}}}{\sum_{c \in C} \sqrt{s_c + A \cdot \frac{s_c}{H}}} = \frac{\frac{\frac{A}{H}}{\sqrt{1+\frac{A}{H}}} \sum_{c \in C} \sqrt{s_c}}{\sqrt{1 + \frac{A}{H}} \sum_{c \in C} \sqrt{s_c}} = \frac{A}{H}.$$

Namely, if the attacker completely breaks the defenses around getting an arbitrary SBT, she can achieve a drawing weight that is at least the weight she would have in a version of Kleros that uses pure stake-weighting without SBTs.

7 Asymptotic effect of increased attacker stake

Suppose we take the case where there is an attacker i_a with stake x_a that only posses one SBT s_a which is possessed by no one else. Then the terms in the numerator of Equation 1 corresponding to the honest users are.

$$H = \sum_{s \in S, s \neq s_a} \sqrt{\sum_{j \in s} \frac{x_j}{|T_j|}}$$

Note that this is the side length of the "honest square" in the sense of [1].

Then, we can simplify equation 1 to see that i_a's chance of being drawn is given by:

$$\frac{\sqrt{x_a}}{H + \sqrt{x_a}}.$$

We might try to define an analogous notion of collusion resistance to that given for quadratic funding in Appendix D of [2]. Of course, as x_a increases, asymptotically,

$$\frac{\sqrt{x_a}}{H + \sqrt{x_a}} \to 1$$

So, insofar as the object of interest for whether an attack is successful or not is the total subsidy extracted for quadratic funding and the attacker's chance of being drawn in our setting, it does not make sense to restrict attackers to only being able to manipulate this object by $O(\sqrt{x_a})$ in the way that is considered in [2].

However, one might now consider the rate at which

$$\frac{\sqrt{x_a}}{H + \sqrt{x_a}}$$

approaches 1. Namely, if one computes the error of this convergence

$$1 - \frac{\sqrt{x_a}}{H + \sqrt{x_a}} = \frac{H}{H + \sqrt{x_a}},$$

then a lower bound on this quantity as x_a grows translates into a guarantee that i_a cannot raise her odds of being drawn beyond some corresponding upper bound.

Then note that $H + \sqrt{x_a} \geq \sqrt{x_a}$ for all x_a and $H + \sqrt{x_a} \leq 2\sqrt{x_a}$ for sufficiently large x_a. Hence,

$$\frac{H}{H + \sqrt{x_a}} = \Theta(\frac{1}{\sqrt{x_a}}).$$

This seems to be the sense in which one can limit the attacker's manipulations to quadratic terms asymptotically.

However, as such asymptotics consider rates of attacker selection that are generally well over 50%, one might ask if this is the right notion to build into a definition of attack resistance. Instead one might be interested in the amounts at which one can cross the 50% threshold. For such a notion it will not make sense to look at asymptotics in x_a unless other values are also allowed to change.

8 Parameterization

We also have various questions related to parameterization. Particularly, one asks:

Question: how should the parameter k determining the number of clusters into which profiles are partitioned in Eigenweighting be determined?

References

[1] William George. Research topics in kleros juror selection: Blockchain identity, soulbound tokens, and models of attack resistancel. *Kleros Blog,* *https://blog.kleros.io/research-topics-in-kleros-juror-selection-blockchain-identity-soulbound-tokens-and-models-of-attack-resistance*, 2024.

[2] Joel Miller, E. Glenn Weyl, and Leon Erichsen. Beyond collusion resistance: Leveraging social information for plural funding and voting. *SSRN, https://ssrn.com/abstract=4311507*, 2022.

[3] Puja Ohlhaver, Eric Glen Weyl, and Vitalik Buterin. Decentralized society: Finding web3's soul. *SSRN, https://ssrn.com/abstract=4105763*, 2022.

[4] Ulrike von Luxburg. A tutorial on spectral clustering. *CoRR*, abs/0711.0189, 2007.

Appendix A: R Code For Simulations

We have the following R code in three slightly different versions where the attacker pursues different attack strategies to compare the effectiveness of these strategies.

Notation: M honest users each with one profile that can possess some combination of k honest SBTs, a profiles controlled by the attacker, stake per honest user is a random value between 1 and 100. Once total honest stake is determined as sum of these values, the attacker has t*(sum of honest stakes) staked on each of the a profiles. Then for values a=50, t=.004, the total stake of the attacker is 20% of the total stake of the honest participants.

The main output of a given run of the code is the variable honestweight. This is the sum of the drawing weight of all of the honest participants. Spectral clustering is performed using the unnormalized Laplacian, which corresponds to the RatioCut described in Section 5.

Version 1: the attacker does not add any malicious SBTs. This version is used as a control for comparison to the other two versions. In the matrix SBTsattack, which in the other two versions includes rows for the edges added by the attacker for the two new SBTs, here there are two empty rows.

```
k=4
M=1000
a=50
t=.004
somme=0
stakes <-sample.int(100,M,replace=TRUE)
stakes[1]<-0
stakes[2]<-0
for (i in 1:M){
  somme=somme+stakes[i]
}
r=t*somme

m0 <- matrix(0, k, M)
m0side <- matrix(0,k,k)
m0bottom <- matrix(0,M,M)
mweights <- apply(m0,c(1,2), function(x) sample(c(0,1),1))
m0side <- matrix(0,k,k)
m0topleft <- matrix(0,M,M)
mweightst <- t(mweights)
mattackinsertbottom <- matrix(0,a,M)
mattackinsertside <- matrix(0,M,a)
mattackinsertcorner1 <- matrix(0,a,a)
mattackinsertcorner2 <- matrix(0,k,a)
mattackinsertcorner3 <- matrix(0,a,k)
mattackinsertcorner4 <- matrix(0,k,k)
mattackinsertcorner5 <- matrix(0,2,k)
```

```
}

mattackinsertcorner6 <- matrix(0,k,2)
mattackinsertcorner7 <- matrix(0,2,2)
mattackinsertline <- matrix(0,a,M+a+k)

ma <- matrix(0,1,M)
ma2 <- matrix(1,1,a)
ma3 <- matrix(0,1,k)

newrow1 <- matrix(0,1,M+a+k)
newrow2 <- matrix(0,1,M+a+k)

newrows <- rbind(newrow1,newrow2)
newrowst <- t(newrows)

existingwattackers <-
rbind(rbind(cbind(cbind(m0topleft,mattackinsertside),mweightst),mattackinsertline),cbind(cbind(mwei
ghts,mattackinsertcorner2),mattackinsertcorner4))

SBTsattack <- cbind(rbind(existingwattackers,newrows),rbind(newrowst,mattackinsertcorner7))


stakesatt <- matrix(r,1,M+a)
for (i in 1:M){
  stakesatt[i]=stakes[i]
}



eigen_resultatt <- eigen(SBTsattack)
# Extract the top-k eigenvectors
selected_eigenvectorsatt <- eigen_resultatt$vectors[, 1:5]
# Perform k-means clustering on the eigenvectors
cluster_assignmentsatt <- kmeans(selected_eigenvectorsatt, centers = 5)$cluster

cotecarreatt=0
honestweight=0
groupstakesatt <- matrix(0,1,5)
pourcentagesstakeatt <- matrix(0,1,M+a)
for (j in 1:5){
   for (i in 1:M+a){
      if (cluster_assignmentsatt[i]==j){
         groupstakesatt[j]=groupstakesatt[j]+stakesatt[i]
      }
   }
   cotecarreatt=cotecarreatt+sqrt(groupstakesatt[j])
}
for (i in 1:M+a){
   for (j in 1:5){
        if (cluster_assignmentsatt[i]==j){
           pourcentagesstakeatt[i]=(stakesatt[i]/sqrt(groupstakesatt[j]))/cotecarreatt
        }
    }
    if (i < M+1){
         honestweight=honestweight+pourcentagesstakeatt[i]
    }

}
```

Version 2: here the attacker places two additional SBTs. One of them corresponds to an "internal" attack in the sense of Section 5, while the other corresponds to an "external" attack. Namely, one of the SBTs is given to honest users to make them seem more similar to each other while the other is given to the attacker's own profiles to distinguish them.

```
k=4
M=1000
a=50
t=.004
somme=0
stakes <-sample.int(100,M,replace=TRUE)
stakes[1]<-0
stakes[2]<-0
for (i in 1:M){
  somme=somme+stakes[i]
}
r=t*somme
m0 <- matrix(0, k, M)
m0side <- matrix(0,k,k)
m0bottom <- matrix(0,M,M)
mweights <- apply(m0,c(1,2), function(x) sample(c(0,1),1))
m0side <- matrix(0,k,k)
m0topleft <- matrix(0,M,M)
mweightst <- t(mweights)
mattackinsertbottom <- matrix(0,a,M)
mattackinsertside <- matrix(0,M,a)
mattackinsertcorner1 <- matrix(0,a,a)
mattackinsertcorner2 <- matrix(0,k,a)
mattackinsertcorner3 <- matrix(0,a,k)
mattackinsertcorner4 <- matrix(0,k,k)
mattackinsertcorner5 <- matrix(0,2,k)
mattackinsertcorner6 <- matrix(0,k,2)
mattackinsertcorner7 <- matrix(0,2,2)
mattackinsertline <- matrix(0,a,M+a+k)

ma <- matrix(0,1,M)
ma2 <- matrix(1,1,a)
ma3 <- matrix(0,1,k)
ma4 <- matrix(1,1,M)
```

```
ma5 <- matrix(0,1,a)

newrow1 <- cbind(cbind(ma,ma2),ma3)
newrow2 <- cbind(cbind(ma4,ma5),ma3)

newrows <- rbind(newrow1,newrow2)
newrowst <- t(newrows)

existingwattackers <-
rbind(rbind(cbind(cbind(m0topleft,mattackinsertside),mweightst),mattackinsertline),cbind(cbind(mwei
ghts,mattackinsertcorner2),mattackinsertcorner4))

SBTsattack <- cbind(rbind(existingwattackers,newrows),rbind(newrowst,mattackinsertcorner7))


stakesatt <- matrix(r,1,M+a)
for (i in 1:M){
  stakesatt[i]=stakes[i]
}



eigen_resultatt <- eigen(SBTsattack)
# Extract the top-k eigenvectors
selected_eigenvectorsatt <- eigen_resultatt$vectors[, 1:5]
# Perform k-means clustering on the eigenvectors
cluster_assignmentsatt <- kmeans(selected_eigenvectorsatt, centers = 5)$cluster

cotecarreatt=0
honestweight=0
groupstakesatt <- matrix(0,1,5)
pourcentagesstakeatt <- matrix(0,1,M+a)
for (j in 1:5){
   for (i in 1:M+a){
      if (cluster_assignmentsatt[i]==j){
         groupstakesatt[j]=groupstakesatt[j]+stakesatt[i]
      }
   }
   cotecarreatt=cotecarreatt+sqrt(groupstakesatt[j])
}
for (i in 1:M+a){
   for (j in 1:5){
      if (cluster_assignmentsatt[i]==j){
         pourcentagesstakeatt[i]=(stakesatt[i]/sqrt(groupstakesatt[j]))/cotecarreatt
      }
   }
   if (i < M+1){
      honestweight=honestweight+pourcentagesstakeatt[i]
```

```
    }

}
```

Version 3: here the attacker places two additional SBTs, both corresponding to “external” attacks in the sense of Section 5. Namely, the attacker gives her own profiles two new SBTs to distinguish them.

```
k=4
M=1000
a=50
t=.004
somme=0
stakes <-sample.int(100,M,replace=TRUE)
stakes[1]<-0
stakes[2]<-0
for (i in 1:M){
  somme=somme+stakes[i]
}
r=t*somme

m0 <- matrix(0, k, M)
m0side <- matrix(0,k,k)
m0bottom <- matrix(0,M,M)
mweights <- apply(m0,c(1,2), function(x) sample(c(0,1),1))
m0side <- matrix(0,k,k)
m0topleft <- matrix(0,M,M)
mweightst <- t(mweights)
mattackinsertbottom <- matrix(0,a,M)
mattackinsertside <- matrix(0,M,a)
mattackinsertcorner1 <- matrix(0,a,a)
mattackinsertcorner2 <- matrix(0,k,a)
mattackinsertcorner3 <- matrix(0,a,k)
mattackinsertcorner4 <- matrix(0,k,k)
mattackinsertcorner5 <- matrix(0,2,k)
mattackinsertcorner6 <- matrix(0,k,2)
mattackinsertcorner7 <- matrix(0,2,2)
mattackinsertline <- matrix(0,a,M+a+k)

ma <- matrix(0,1,M)
ma2 <- matrix(1,1,a)
ma3 <- matrix(0,1,k)


newrow1 <- cbind(cbind(ma,ma2),ma3)
newrow2 <-cbind(cbind(ma,ma2),ma3)

newrows <- rbind(newrow1,newrow2)
```

```
newrowst <- t(newrows)

existingwattackers <-
rbind(rbind(cbind(cbind(m0topleft,mattackinsertside),mweightst),mattackinsertline),cbind(cbind(mwei
ghts,mattackinsertcorner2),mattackinsertcorner4))

SBTsattack <- cbind(rbind(existingwattackers,newrows),rbind(newrowst,mattackinsertcorner7))


stakesatt <- matrix(r,1,M+a)
for (i in 1:M){
  stakesatt[i]=stakes[i]
}



eigen_resultatt <- eigen(SBTsattack)
# Extract the top-k eigenvectors
selected_eigenvectorsatt <- eigen_resultatt$vectors[, 1:5]
# Perform k-means clustering on the eigenvectors
cluster_assignmentsatt <- kmeans(selected_eigenvectorsatt, centers = 5)$cluster

cotecarreatt=0
honestweight=0
groupstakesatt <- matrix(0,1,5)
pourcentagesstakeatt <- matrix(0,1,M+a)
for (j in 1:5){
   for (i in 1:M+a){
      if (cluster_assignmentsatt[i]==j){
         groupstakesatt[j]=groupstakesatt[j]+stakesatt[i]
      }
   }
   cotecarreatt=cotecarreatt+sqrt(groupstakesatt[j])
}
for (i in 1:M+a){
   for (j in 1:5){
      if (cluster_assignmentsatt[i]==j){
         pourcentagesstakeatt[i]=(stakesatt[i]/sqrt(groupstakesatt[j]))/cotecarreatt
      }
   }
   if (i < M+1){
      honestweight=honestweight+pourcentagesstakeatt[i]
   }

}
```

Results of simulations

Below we summarize the results of a basic simulation: M=1000, a=50, t=.004, k=4, clustering into five clusters.

For each version, we ran the above code five times (each run corresponding to a random choice of the matrix of which SBTs belong to which honest user). Then we calculated the average value of honestweight for each version.

	Version 1 (control)	Version 2 (one external attack, one internal attack)	Version 3 (two external attacks)
Average honestweight	.872	.866	.859

It is unclear how to design statistical tests comparing the effectiveness of the attacks, e.g. how many simulations need to be performed in order to have a statistically significant comparison.