We consider the following framework:

An asker pays c to pose a question.

There is a vault that accumulates excess amounts as will be seem below that contributes up to v to subsidize this question (this amount will vary such as by being a percentage of the value of the vault or by adapting to the number of responses over a given period, etc).

Respondents must do w work to find the (honest) answer to the question. When submitting this answer, they must pay a deposit of d which is lost if their answer is judged incorrect (and redistributed to honest respondents) and f in fees.

These fees consist of two essentially different kinds:

- Fees f_1 that cannot be calibrated by (and are in fact not paid to) the smart contract, such as Ethereum gas costs and lost time-utility of having one's capital locked up in deposits
- Fees f_2 that are paid to and can be calibrated by the smart contract

Then for each response to a question (regardless of whether this response is a first honest answer to a question, the respondent has already responded with the same answer, or the answer is false), the respondent must pay $f = f_1 + f_2$.

After the honest answer is determined (by using Kleros if necessary), each of the n honest respondents receives $\frac{c+v}{2^{n-1}}$ (in addition to redistribution of deposits if applicable). The remaining

$$\left(mf_2 + c - n\frac{c+v}{2^{n-1}}\right)$$

(where m is the total number of respondents including dishonest ones and irrecoverable costs includes such things as Ethereum fees and the time value of the deposits) goes in to the vault.

Assumptions

- How many people respond to a given question, and their answers are not visible until after the period to respond is closed.
  - This can be done by having the contract that manages these questions divide up time into response periods where users can respond to a selection of questions, then these questions are resolved and closed and in the next response period there are new questions. Responders send a message to the contract that includes d and f and commits both to a question and an answer to be revealed after the response period. On-lookers can see how many total answers there are but not how many there are for any given question.
  - This way you avoid having to analyze situations where an attacker floods a given question with a lot of (false) answers to discourage other respondents who see that the payout will be low if these answers are honest.

- The work to answer a question w is constant over a response period.
  - As the price of ETH varies, the value of work to answer something will change with respect to ETH even if that work remains constant, so we need to be able to readjust periodically the fees to compensate for this.
  - In reality, there will be some questions that are harder to answer than others, so w should vary even within a response period - maybe try to return to this assumption later.

We would expect that the number of (honest) respondents to increase until each question, on average, is paying out to an honest respondent their marginal cost to participate. By symmetry of the questions, this will lead to each question receiving the same average number of (honest) responses n. Suppose there are a total of N honest respondents. As we assumed that respondents cannot distinguish which questions are getting more answers than others, their best strategy is to randomly choose $\frac{n}{N}$ questions to respond to. Then the number of responses that each question actually receives is given by a binomial distribution Binomial(N, $\frac{n}{N}$).

We consider a given question from the perspective of a single respondent, conditional on him having selected to respond to that question. For each question that he selects, his marginal return is equal to marginal cost to participate (as he adjusts the percentage of questions to which he responds $\frac{n}{N}$ so that this is true for the marginal question he responds to, but by the symmetry of the questions it is true for all questions that he responds to). Then, if there are k other (honest) respondents to this question, there are a total of $k+1$ total honest responses, so we must have:

$$E\left[\frac{c+v}{2^{k+1-1}}\right] = E\left[\frac{c+v}{2^k}\right] = f + w,$$

where k follows Binomial($N-1$, $\frac{n}{N}$). (Note that this equation cannot hold if $f + w > c + v$; in this case it will not be in the economic interest of any honest respondent to submit a solution.) As $N \to \infty$, this binomial distribution approaches the Poisson distribution Poisson(n).

Then,

$$E\left[\frac{c+v}{2^k}\right] = (c+v)E\left[\frac{1}{2^k}\right] = (c+v)e^{-n/2}.$$

So

$$e^{n/2} = \frac{c+v}{f+w} \Rightarrow$$

$$n = 2\ln\left(\frac{c+v}{f+w}\right). \tag{1}$$

By the definition of n, this gives the average number of honest responses to a given question.

The probability that a given question receives no (honest) answers z is

$$z = \left(1 - \frac{n}{N}\right)^N.$$

As the total number of honest participants grows, $N \to \infty$, this approaches e^{-n}.

So, in equilibrium, this probability is

$$z = e^{-2\ln\left(\frac{c+v}{f+w}\right)} = \left(\frac{f+w}{c+v}\right)^2. \tag{2}$$

Ideally, we could calibrate the values in our system (c, v, f_2) in terms of z and the griefing factors for various attacks.

1 Grief to drive down rewards for other respondents

The additional cost for an attacker to respond a second time to a given question is f (as she has already done the required work to find the solution). The attacker receives no additional payment for this second submission due to the structure of the payments. On the other hand, this reduces the payout for all the other k respondents by half for a total grief of:

$$\frac{1}{2}k\frac{c+v}{2^k}.$$

As k is again distributed as Poisson(n), based on a simple calculation using properties of Poisson distributions, the expected value of this grief is

$$\frac{(c+v)n}{4}e^{-n/2} = \frac{(c+v)\ln\left(\frac{c+v}{f+w}\right)}{2}e^{-\ln\left(\frac{c+v}{f+w}\right)} = \frac{f+w}{2}\ln\left(\frac{c+v}{f+w}\right).$$

Rearranging equation 2 we have

$$\frac{w+f}{c+v} = z^{1/2},$$

so then in terms of z, the grief causes a damage of

$$-\frac{1}{4}\ln(z)\,(f+w).$$

Then the griefing factor for this attack is

$$g = \frac{\text{total damage}}{\text{cost of attack}} = \frac{-\ln(z)\,(f+w)}{4f}. \tag{3}$$

Note this is bounded below by

$$-\frac{\ln z}{4}.$$

So, for example, if we demand that no more than a proportion of $z = .05$ of questions do not receive an honest answer, this lower bound is approximately .74893, and grows as w grows. Thus, for most reasonable choices of parameters, an attacker will likely be able to do at least as much harm to the other respondents as she inflicts upon herself. If we expect even lower rates of questions not receiving honest answers, the damage that an attacker can cause on other respondents via this attack can become quite large.

On the other hand, if the attacker submits more than two responses to a given question, the cost of the attack grows super-linearly. For example, if she submits three answers she pays in $3f + w$ and receives $\frac{c+v}{2^{k+2}}$ (whereas if she behaved honestly she pays in $f + w$ and receives $\frac{c+v}{2^k}$) so the cost of the attack is

$$2f + \frac{c+v}{2^k} - \frac{c+v}{2^{k+2}} = 2f + 3\frac{c+v}{2^{k+2}},$$

which has expected value

$$2f + \frac{3}{4}(f + w).$$

In general, if the attacker answers $l + 1$ times, the cost of her attack (the additional l answers after the first) is

$$lf + \left[\frac{c+v}{2^k} - \frac{c+v}{2^{k+l}}\right] = lf + \frac{2^l - 1}{2^l}\frac{c+v}{2^k},$$

which has expected value

$$lf + \frac{2^l - 1}{2^l}(w + f). \tag{4}$$

Following the reasoning of above, the total grief is then

$$k\left[\frac{c+v}{2^k} - \frac{c+v}{2^{k+l}}\right] = k\frac{2^l - 1}{2^l}\frac{c+v}{2^k},$$

which has expected value

$$\frac{f+w}{2} \cdot \frac{2^l - 1}{2^l} \ln\left(\frac{c+v}{f+w}\right)$$

As l increases the griefing factor decreases; this corresponds to the fact that the total grief that an attacker can cause is bounded by the total reward for a given question. So to perform a large scale attack on the system, an attacker would have to perform this attack on many questions; the viability of doing so would depend on the size of w.

Remark 1 *Note that all of the value that has been griefed from the respondents goes into the vault. However, under our assumptions if the respondents are acting rationally, the subsidies from the vault on future questions will not compensate respondents in a way that repays them, on average, for the grief done to them.*

This is because we have seen that the number of respondents to each questions is distributed as Poisson(λ) where λ adjusts to the vault subsidies so that the expected payout for each honest respondent is $f + w$. Namely, the number of respondents will increase due to the vault subsidy, so then each of them including the previously griefed respondents will not be paid on average more than before.

However, by increasing the number of respondents, the vault serves to increase the probability that the system produces at least one honest answer for each question, improving the general health of the system.

2 Sketch of attack for increasing response rate to depress competition

At this point we need to make some assumptions about how users adjust their estimation for N. One can calculate the theoretical n in the limit for large N from known parameters using equation 1, and one can see how many responses any given question got, but as the number of responses to any given question is a random variable, one can only attempt to infer N from this information.

To simplify, we imagine that the questions are sequential and that respondents base their guess of N only based on the previous question. They can compute n, the equilibrium level supported by the parameters c, f, and w, theoretically. Then, as they randomize their choices so that they choose each question with probability n/N, they can infer changes in N by changes in the number of responses. Note if the number of responses users see doubles, then (absent changes in c, f, and w that would justify a change of n), the best estimate that the users can make for N based off this single data point also doubles.

Imagine an attacker wants the other respondents to believe that N has doubled to benefit from a depressed participation on the following question. Then she should grief $l = n$ times. By equation 4, the average cost of this is

$$nf + \frac{2^n - 1}{2^n}(w + f)$$

(which is still valid even for non-large N as we still have $E\left[\frac{c+v}{2^k}\right] = f + w$). So if the respondents previously believed there were N respondents, now (on average) they will believe there are $N_1 = 2N$ respondents.

As noted above, this attack adds value to the vault, changing v to v_1. However, respondents will adjust their value of n to n_1 compensate this effect. Namely, they calibrate so that randomizing with k drawn from Binomial$(N_1, n_1/N_1)$ gives an expected payoff $E\left[\frac{c+v_1}{2^k}\right] = f + w$.

However, in fact k, the number of responses to the next question (excluding the attacker), is given by Binomial$(N_1/2, n_1/N_1)$.

Going back to approximating this distribution by a Poisson distribution [at least for now in this sketch], this is approximated by Poisson$(n_1/2)$. Then

$$E\left[\frac{c+v_1}{2^k}\right] = (c+v_1)e^{n_1/4}.$$

But we saw for k distributed as Poisson(n_1)

$$E\left[\frac{c+v_1}{2^k}\right] = (c+v_1)e^{n_1/2} = w+f.$$

So, rearranging, for k distributed as Poisson($n_1/2$)

$$E\left[\frac{c+v_1}{2^k}\right] = \sqrt{(w+f)(c+v_1)}.$$

Again, the cost of this attack is upper-bounded by $nf + (f+w)$, which in this approximation is

$$2f \ln\left(\frac{c+v}{f+w}\right) + (f+w).$$

So, this attack would often be viable under the assumptions described. In fact, we imagine that respondents are more sophisticated in their estimation of N, so that for an attacker to perform this attack she would need inflate the response rate of *many* questions. This would substantially increase the costs of the attack and, depending on how quickly the respondents react to seeing a few questions with very low response rates, may not have a substantially larger payoff. The viability of such an attack is something to consider, particularly in the context of experiments on quickly how users adapt their response rate as the average number of responses changes.

3 Potential attack - Asker's attack

Prior to asking a question, a potential Asker pays the f cost to respond a second time to one or more other questions. This increases the amount in the vault and then when the Asker poses his question, there will be a larger subsidy. As we imagine that $c >> f$ this increased subsidy may be sufficient to compensate for the costs of the attack.

The viability of this attack depends on exactly how we choose to structure the subsidies from the vault - for example, we could have many different versions of this contract each with a specific asker's price c hard-coded in and each with a separate vault, with subsidies going directly to the respondents and not influencing the asker's cost. Then this attack doesn't work, but we have money scattered across many different vaults now and it is more likely that this money gets trapped than before, etc.

4 Rough ideas on how to adjust fees/manage vault subsidy

The only piece of information that the contract can see to gage whether things are functioning well is the number of responses to each question that are (ruled) correct. If c, v, and f are fixed in ETH, we can use equation 1 to think of changes in n as measuring changes in w the required work to answer the question with respect to ETH. Then $c+v$, and f_2 can be set to change accordingly (noting that they should still satisfy equations 2 and 3 for given parameter choices of z and k - namely for a given level of security that we presumably set in advance and try to maintain as constant) to maintain a given value for n. Then the question is how to best balance adjusting c with adjusting the subsidy v to achieve a given level $c + v$ (in a way that doesn't open up too many attacks on the vault, doesn't trap too much money in the vault, etc).