First post (and also new to the whole cryptocurrencies thing)! I've searched the forum for "equivocation" and didn't see anything obviously relevant; my apologies if I missed anything obvious or committed other types of errors (such as posting in wrong forum, formatting, etc.). I'll learn if you teach me. =)

Summary

The purpose of this post is to mathematically (but nothing fancy) analyze the freezing probability

(defined as the probability that 2/3

validation fails in a single epoch) when equivocation

(defined, as in https://github.com/ethereum/research/blob/master/papers/other_casper/casper_paper.md, as having 2 "main chains" that both appear long enough to be legitimate) is happening. Here's a summary of the main points:

- Assumptions:
- We assume a random voter model

where honest voters vote randomly between 2

"legitimate looking" checkpoints. This assumption seems very strong, but we think it is very important to study. We hopefully justify this assumption when we come to it.

- we assume the M

members of the validator set are either independently-acting honest validators or a colluding junta

Z

. We make the assumption that all validators have the same stake (1/M)

. We define the power

P = (|Z|/M)

of Z

as the sum of the total stake of Z

.

- We assume a random voter model

where honest voters vote randomly between 2

"legitimate looking" checkpoints. This assumption seems very strong, but we think it is very important to study. We hopefully justify this assumption when we come to it.

- we assume the M

members of the validator set are either independently-acting honest validators or a colluding junta

Z

. We make the assumption that all validators have the same stake (1/M)

. We define the power

P = (|Z|/M)

of Z

as the sum of the total stake of Z

.

- we analyze the freezing probability when all validators are honest. For a very small number of validators (say 5

, where the freezing probability is 31\%

), it seems it is possible for the situation to "take care of itself" in a couple of epochs. The freezing probability is already very high (87\%

) when we have 20

validators, and would go much higher with more validators. This estimate is directly relevant to where the online whitepaper says “…Suppose that malicious miners have an n:1

advantage against honest miners, and malicious miners need to create c

competing checkpoints to reliably prevent convergence (say, c = 3

, as if c = 2

then one of the two may easily get two thirds prepares by random chance})…,” as under our assumptions c=2

is bad enough with no adversary.

- we introduce dishonest adversaries with a total amount of computing power P

, which we call a junta

. A junta can significantly increase the freezing probability by simply splitting their own votes, which we call a Splitting Attack

. At 5

total validators, a 40\%

junta (2

people) can freeze the chain with 87.5\%

probability, much higher than 31\%

when we have all honest validators. At 20

validators, a 20\%

junta halves the probability that validation happens, with freezing probability going from 87\%

to 94\%

. This analysis shows that even if the junta has no information about what the honest validators are doing (e.g. if we use a commitment scheme for voting), the junta can do some significant additional damage, at seemingly very low risk of detection.

- Empowering a junta that is sophisticated enough to move after everyone else makes their votes, we greatly amplify the freezing probability if the junta performs a Contrarian Attack

by voting against the winner. At 5

validators, even a 20\%

junta (1

person!) can freeze the chain with 87.5\%

probability. At 20

validators, with just a 5\%

junta the freezing probability jumps all the way to 92.6\%

; a 20\%

junta effectively holds the chain hostage at 99.7\%

.

- The Splitting and Contrarian Attacks trade off between detectability vs freezing probability. Hybrid attacks that combine these desiderata are possible. We show one possible such hybridization as the Smoke Attack

. The freezing probabilites for the Splitting and Contrarian attacks actually “sandwich” that of the Smoke Attack (and all similar attacks), another indication that they are good toy objects to study.

To repeat, we are doing a “damage analysis” conditioned on equivocation happening.

We deliberately do not

do the following:

- We are not

estimating the probability of equivocation happening (or devising attacks to cause equivocation), a very interesting quantity with many unknown variables, such as the choice of the underlying block generation algorithm, latency, noisy networks, adversaries, exogenous events, or idiosyncracies of the implementation.

- We are not

making attacks to cause equivocation. (although there are some similar qualities to our problem; see conclusion.

- We are not

saying that these things are alarming and we should jump ship from PoS. The main obvious “out-of-protocol” solution to such attacks is to have social consensus (i.e. everyone agrees to choose b_1

over b_1'

). This catch-all solution reduces the cost to the cost of having the social consensus. The point of this is just to analyze what may happens in-protocol.

The Random Voter Model

As stated in the introduction, we assume a context where equivocation

is nontrivially possible. We define the term more generally than the result of an equivocation attack, but more generally as a situation where there are at least 2

competing checkpoints:

- the last finalized checkpoint block is b_0

at height h

,

- (at least) two potential checkpoint blocks, b_1

and b_1'

, have been proposed (probably, but not necessarily, due to adversarial intervention), at height at least h+1

.

- both checkpoint blocks are “legitimate looking.” One (strong!) interpretation of this assumption is that the honest validators (without social consensus or some other “out-of-protocol” action) do not know which checkpoint to support. In particular, honest validators would pick b_1

or b_1'

at random, if asked to choose a checkpoint.

This is a terribly unrealistic-sounding assumption. Since we use it everywhere, our work is useless unless it is justified! We now address it directly:

The most natural objection I can think of is that validators do not actually act randomly. We can introduce defaults

, loosely defined as “predictable behaviors” for validators, such as:

- timestamp defaults

: auto-vote for the checkpoint with the earlier timestamp. This immediately fixes many coordination issues, as all the validators would (usually) vote for the same block.

- longest-chain defaults

: auto-vote for the checkpoint with a longer chain tied to it. This is very intuitive in context of e.g. a classic Proof-of-Work miner.

It is natural for us, wanting proof-of-stake to succeed, to think that “the validators will just XXX by default and solve the problem.” Sadly, this is exactly the mental blindspot that adversaries want to have!

First of all, these defaults, like social consensuses, are “out-of-protocol” with respect to Casper, and we should not make these assumptions in a cautious security mindset.

Second, we claim that having deterministic defaults may in fact introduce problems similar to a random voting model, even if there is no randomness in the voting

:

- Suppose we have timestamp defaults. If adversaries collude to create a block with a fake timestamp about 1/2 way temporally through the validation period, then this default behavior would actually cause the honest validators to randomly (depending on the timing of the block) to split their votes, which is very similar to having the honest validators voting randomly; if this attack is precise enough, it can actually be made strictly worse

than random validators.

- Suppose we have long-chain defaults. If the 2 checkpoints have tying longest chains, if there is no further tiebreaker then the random voting assumption becomes good. If there is an artifical tiebreaker that causes honest validators to flock to b_1

over b_1'

, then the adversaries can enter a new block about halfway through the validation period for b_1'

, thus splitting the votes in a similar to the previous example.

Finally, the most “academic” (but potentially useful, as academic things sometimes are) reason: random voting is in some sense “the worst case” allowed by the Casper protocol. It makes sense to analyze as something that is both easy to analyze and provides some bounds and benchmarks for us. Even if such contexts never happen in practice (ideally, this may happen as a result of someone doing these analyses, who is in turn motivated to make sure the random voting context never happens! =D), the calculations done in this analysis may be useful to have in mind for someone working on the security of Casper and is thinking adversarially and pessimistically. In this sense, they’re similar to the (very strong) assumptions cryptographers have for Eve or other adversaries when they do cryptography.

Honest and Random Voters

In this section, we analyze a toy situation of where there is no coordinated malicious junta, borrowing some of its results for later sections. All the damage comes from voters voting randomly:

- We model a vote for b_1

with stake s

as +s

and a vote for b_1'

as -s

. This means 67\%

vote for b_1

corresponds to a value of 2/3-1/3 = 1/3

, and similarly a 33\%

vote would correspond to -1/3

.

- All the validators (all honest!) vote for b_1

(and also b_1'

) uniformly at random. Suppose there are M

voters, each with power 1/M

. Then we basically get a Gaussian centered at 0

with a variance of M/M^2 = 1/M

.

In this situation, the freezing probability is just (approximately) the probability that a N(0, 1/M)

Gaussian ends in (-1/3, 1/3)

, or \frac{1}{\sqrt{2\pi/M}}\int_{-1/3}^{1/3} e^{-Mx^2/2} dx.

- Suppose we have 5

validators, the probability is roughly 31\%

(the integral above actually gives 54\%

, but we use the binomial distribution for a more accurate estimate since our M

is small).

- With 20

validators, we get to 87\%

.

- With 100

validators, 99.9\%

.

We can quickly check that with either the timestamp or longest-chain defaults, the problem goes away basically immediately, though recall that it is still theoretically possible (as with the examples given in the previous section) for an adversarial junta to change the environment to one which is well-approximated by random voters. These probabilities obviously get significantly worse with more legitimate-looking checkpoints, but 2

are clearly enough to be bad.

The Splitting Attack

In this situation, we introduce the aforementioned junta of malicious voters. Like the toy example in the previous section, we still assume that the honest voters are ignorant that they are being attacked and vote randomly for the two checkpoints.

- As before, we model a vote for b_1

with stake s

as +s

and a vote for b_1'

as -s

.

- The set of non-Z

validators vote for b_1

(and also b_1'

) uniformly at random. Suppose there are M

of them, each with stake (1-P)/M

. This time, we basically get a Gaussian centered at 0

with a variance of (1-P)^2/M

.

- The members of Z

split the vote exactly, thus contributing 0

to the sum.

We call this a Splitting Attack

, because the junta Z

is splitting votes to maximize freezing probability.

- The Splitting Attack is the natural attack when the junta has no information about what votes other validators are casting. This is not in the specs of Casper, but is certainly possible under some implementations / extensions of Casper. Obviously, if the junta can “act after” everyone else given the information of what other people have, the junta does not want to split the votes; it wants to vote against the “winning” coalition. We explore this next with the Contrarian Attack.
- In particular, the Splitting Attack ends up being the optimal strategy for the junta when we have a commitment scheme (one of our proposed “solutions” to the Smoke Attack in Section~\ref{sec:smoke}), when Z

has no information what the honest voters are doing.

- Analyzing the Splitting Attack gives us a lower bound of freezing probability compared to the adversarial Smoke Attack.

In this situation, the freezing probability is just (approximately) the probability that a N(0, (1-P)^2/M)

Gaussian ends in (-1/3, 1/3)

, or \frac{1}{\sqrt{2\pi(1-P)^2/M}}\int_{-1/3}^{1/3} e^{-Mx^2/2(1-P)^2} dx.

- Suppose we have 5

(total) validators. For P = 20\%

(or 1

junta member), it’s not well-defined what that actor is doing in this attack, so the probability is still 31\%

for pathological reasons. For P = 40\%

, the 3

honest actors must all agree, so the attack succeeds with probability 1 - 1/2^3

, or 87.5\%

! This is a very big boost from 31\%

.

- Suppose we have 20

validators. For P = 20\%

, the freezing probability is 94\%

. For P = 40\%

, it jumps to 98.7\%

. (compare with 87\%

in the no-junta case)

- Suppose we have 100

validators. As we are already at 99.9\%

without coordination, having a junta is the least of our worries.

In the Splitting Attack, it is very difficult to detect either (1) that the attack is happening or (2) who the members of Z

are. This is amplified if we allow some noise (for example, having some members of Z

each epoch vote differently at random).

The gain of coordination is smaller for larger numbers of validators, where freezing is already very likely. As “honest” validators are probabilistically splitting the vote, the junta’s members (and even its existence) are hard to detect. As before, this analysis changes if the honest (non-Z

) validators have some defaults, though the defaults can be theoretically leveraged.

Contrarian Attack

The next obvious thing to do is to give the junta more power when they adapt to the information in other validator’s votes, which are by default broadcast over the network. Suppose the junta wish to “act last” after all honest validators, their optimal strategy is no longer to split their vote, but to vote against the winning checkpoint. With strictly more information than in the setup of The Splitting Attack, we would expect much higher freezing probability; in particular, if the junta has P > 1/3

, the freezing probability should go to 1

!

In this model, which we call the Contrarian Attack

(keeping the basic voting scheme as previous models):

- The set of non-Z

validators vote for b_1

(and also b_1'

) uniformly at random. Suppose there are M

of them, each with stake (1-P)/M

. As before, we approximate with a Gaussian centered at 0

with a variance of (1-P)^2/M

.

- The members of Z

sees the honest validators’ votes first, then vote for the “losing” checkpoint to maximize the splitting of the vote.

Now, the freezing probability is the probability that a N(0, (1-P)^2/M)

Gaussian ends up in (-1/3-P, 1/3+P)

, after which the junta adjusts the vote by P

towards 0

to end in the (-1/3, 1/3)

deadlock region: \frac{1}{\sqrt{2\pi(1-P)^2/M}}\int_{-1/3-P}^{1/3+P} e^{-Mx^2/2(1-P)^2} dx.

- Suppose we have 5

(total) validators. P=20\%

means the other 4

validators need to agree. This is equivalent to the P = 40\%

Splitting Attack, so the freezing probability is also 87.5\%

. P=40\%

obviously gives 100\%

freezing probability.

- Suppose we have 20

validators. For P = 20\%

, the freezing probability is 99.7\%

. We wish to mention with even just P = 5\%

, we get 92.6\%

. For P = 10\%

, we get 96.7\%

!

- Suppose we have 100

validators… we probably do not need to analyze this case, given the previous sections.

One obvious “solution” is to have the voting done by some sort of commitment scheme (depending on purpose, this may itself be useful for other things, such as avoiding forerunning in certain smart contracts). This would mean that it is no longer possible for Z

to check “which chain is currently winning in votes.” Thus, a commitment scheme-ish assumption reduces the problem to the regime for the Splitting Attack, which is another reason to study the Splitting Attack.

While the Contrarian Attack is strictly stronger (in terms of freezing probability) than the Splitting Attack, it also gives away more information for identifying the junta. Just as the designers make tradeoffs between efficiency and security, the junta can make tradeoffs between detectability and freezing power, making it make sense to study both the Contrarian and Splitting Attacks. If the junta does prioritizes freezing power and goes with the Contrarian Attack, here’s one way in which the cat-and-mouse game may escalate:

- Honest validators may realize that the same people are voting in a big clump in the same direction, and setup a punishment scheme.
- One defense by the junta would be to let some of the junta vote differently from the others (for example, if P=15\%

and the honest validators have a 60\%-25\%

vote split between b_1

and b_1'

, the junta can give 5\%

to b_1

and 10\%

to b_1'

, thus making at least 1/3

of its members less suspicious.

- A “re-raise” (abusing poker terminology)) by the honest validators would be trying to identify the junta by timestamps of voting, not just by direction.
- The natural “re-re-raise” by the junta would be to spread out the timing and the voting direction, which leads to a more sophisticated attack that we now introduce as the Smoke Attack.

The Smoke Attack

The Splitting Attack emphasizes undetectability and maximal damage given minimal information from the perspective of the junta. The Contrarian Attack emphasizes damage. The following Smoke Attack

is one of many possible ways to combine the strengths of these two attacks.

- Z

observes the chain and obtains a distribution of how the honest validators vote. Z

then models a time interval [0,T]

of when most/all of the validator votes will come in.

- Z

generates a random distribution of “voting times” for members of Z

that is spread out among [0,T]

, using a distribution as close to the empirical distribution of legitimate votes as possible. WLOG, we can name the members of Z

z_1, \ldots, z_N

in some random order, and voting times (t_1, \ldots, t_N)

voting at times 0 \leq t_1 < \cdots < t_N \leq T

respectively. %This gives rise to a \rho \in S_N

(the set of permutations on N

elements), which is the ordering of the members of Z

.

- In the most basic version of the attack, at each time t_i

, the junta member z_i

votes for the checkpoint that is currently “losing”. To decrease detection probability at the cost of freezing probability, z_i

may vote for the block that is currently not winning with a high probability instead.

Depending on parameters and distribution of honest voting times, we would get different freezing probabilities that would be very easy to simulate (It should be easy to analyze the difference combinatorially for easy distributions, such as a uniform distribution. Fearing that this will only be a mathematician’s game, I pass for now. However, I’ll do it in the next version of this note if people find it helpful). However, the analysis of the Splitting Attack gives us a universal

lower bound on freezing power of this attack because in the Smoke Attack, Z

must be strictly more successful at avoiding consensus than the Splitting Attack, because the honest actors are doing the same thing in the two attacks while Z

has additional information. Similarly, the Contrarian Attack gives a universal upper bound on the freezing power of this attack as Z

obtains maximal information that way.

Conclusion

First of all, we do not claim anything genius in this note. People have probably already thought about equivocation a bit. The math isn’t particularly hard. The outcomes of equivocation aren’t even that bad (we can use defaults, we can use social consensus, etc.). We just think some analysis like ours seem very important to have and to build on. At least, it seems like good security hygiene.

Again, one potential interpretation of our results is that we are emphasizing

the need to have a well-thought-out default; in particular, one which does not reduce easily to the random voter model under attacks

. This would make all attacks in this note obsolete, which would be a great outcome despite it making us look dumb! =D Otherwise, we would expect the high freezing probabilities that we see in our work.

Finally, we wish to mention that in working on this analysis, we thought of attacks for creating

equivocation that are similar to these attacks we analyze during

equivocation. Here’s a sketch of a situation where equivocation may be nontrivally possible (which itself may be worth future work): As long as we have the fairly weak assumption that the culture/protocol makes it “legitimate,” or at least not

suspicious, for proposers to propose a block in addition to a chain with length L-1

when the longest proposed chain is length L

(for example, the proposers can falsify timestamps and/or lie that “both chains were the same length to me when it was my turn to propose”), or we just simply have block proposal propagations be fairly inefficient, so it is expected that blocks frequently fork, then a junta with above-average latency can theoretically keep two chains pretty close by always adding to the shorter chain. This idea is remarkably similar to the kind of attacks we propose, except performed at the block-generation layer instead of the validation layer. A hand-wavy philosophical but possibly true consequence: since the idea of building 2 chains of similar size is very similar to the idea of building 2 validation blocs of similar size, we expect the models and approximations appearing in this note would show up again if/when we analyze equivocation creation,

which is another reason to do these intellectual exercises.

Acknowledgments

: We thank Albert Ni and Paul Christiano for enlightening conversation during this. Any stupidity, however, is due to myself.