Availability and Validity

1 Heterogeneous sharding

We shall describe a blockchain sharding scheme through which a single *relay chain* can validate numerous heterogeneous shard chains, with true shared security, minimal latency, good liveness, and relative efficiency in terms of computation, network, and storage. We refer to our shard chains by the term *parachains* because shards connotes a non-heterogeneous approach. There exist terms for heterogeneous approachs too, but they all conote bridged designes without shared security.

We let $\mathcal{V}$ denote the full validator list for the relay chain, and set $n := |\mathcal{V}|$. We expect here that relay chain validators act as bock producers for the relay chain. We assume that $n \geq 3f + 1$ where f bounds the number of validators controlled by any adversary.

An adversary who controls f of the n relay chain validators knows the upcoming block producer for its own upcoming block production slots of course, but we assume Bernoulli trials with probability $\frac{f(n-f)}{n^n}$ describe whether they know the honest block producer assigned to individual upcoming block production slots. In particular if $\frac{f}{n} < \frac{1}{3}$ then they know less than $\frac{5}{9}$ths of the upcoming honest block producers. We choose this assumption because it supports overall stronger block production mechanisms than proof-of-work or [?], although those would admit a better bound here.

Each parachain ρ has its own block production protocol run within its own infrastructure, and refer to its block producers as *collators*. We let $\mathcal{V}_{\rho,t} \subset \mathcal{V}$ denote the sublist of validators assigned to parachain ρ at time t. We need a bound n' such that $|\mathcal{V}_{\rho,t}| \geq n'$.

Notations $\mathcal{V} = \{V_1, V_2, ..., V_n\}$ is the validator set consisting of validators V_i. $\mathcal{P}V$ is a set of parachain validators and $\mathcal{P}V \subset \mathcal{V}$. For the sake of distinction, we denote parachain validators by $PV_i \equiv V_i$.

Definition 1 (Blob). *A blob is a tuple (B, π, M) where B is the parachain block, π is the proof of validity of B and M is the set of outgoing messages from that parachain.*

Definition 2 (Block Header). *Block header is a summary of the block B which links to $B \in PC$ and includes all the signatures by parachain validators who claims its validity. There exists a function* Reconstruct *which takes the block header B_{head} and block data B_{data} as an input and outputs B which B_{head} is linked to.*

Definition 3 (Erasure Code). *Erasure code consists of two algorithms:* Divide_k *and* Reconstruct *Given data D, the algorithm* Divide *outputs n pieces $d_1, d_2, ..., d_n$ and given k out of n pieces,* Reconstruct *outputs D.*

In other words, erasure code divides data into n pieces and at least k out of n pieces are enough to construct the data.

In the availability and validity protocol in Section 2, parachain validators divide a blob into n pieces. These n pieces corresponds to the B_{data} in Definition 2 and so the $\mathsf{Reconstruct}$ algorithm in Definition 2 corresponds to the $\mathsf{Reconstruct}$ algorithm of the erasure code in Definition 3.

1.1 Security Model

The active parties in the availability and validity scheme are validators, parachain validators, fishermen and collators.

- Parachain validators are responsible to validate a blob. They can be malicious.
- Collators are the full nodes of parachain and provide a PoV blob to the responsible parachain validators. They can be malicious and collude with parachain validators.
- Validators are responsible for maintaining the relay chain and finalizing the blocks in the relay chain. A validator can also be a parachain validator.
- Fishermen inspects any malicious activity. If any exists they announce and prove it.

In our security model, we assume that all parties except fishermen can be malicious. Given $|V| = n = 3f + 1$, at most number of f validators can be malicious.

Definition 4 (Proof of Validity (PoV))**.** *PoV consists of two algorithms:* Prove *and* Verify*:*

- Prove *takes the block B, the parachain PC and the outgoing messages from PC as an input and outputs the proof π.*
- Verify *takes the blob (B, π, M) as an input and outputs 1 (valid) or 0 (invalid).*

Correctness of PoV: PoV is correct if for all $B \in PC$ and M, $\mathsf{Verify}(B, \pi, M) \to 1$ where $\mathsf{Prove}(B, PC, M) \to \pi$.

Security of PoV: PoV is secure if an adversary generates a blob (B, π, M) for a parachain PC where $\mathsf{Verify}(B, \pi, M) \to 1$ and $B \notin PC$ with a negligible probability.

We note that $B \in PC$ (resp. $B \notin PC$) means that the block B is a valid (resp. invalid) block for a parachain PC.

Definition 5 (Unavailable Block)**.** *A block is unavailable if less than k erasure code pieces of its blob are obtained by honest validators.*

Definition 6 (Security of Availability)**.** *Assume that at most number of f validators are corrupted. If the probability of having a finalized block in the relay chain which includes an unavailable block is negligible, then the availability protocol is secure against malicious validators.*

Definition 7 (Security of Validity)**.** *Assume that at most number of f validators are corrupted. If the probability of having a finalized block in the relay chain which includes the header of an invalid block is less than a risk value, then the validity protocol is secure against malicious validators.*

2 Protocols

.

The protocol consists of two phases "Parachain phase" and "Relay Chain Phase". The parachain phase is executed between collators and parachain validators. In the end of this phase, the parachain validators validate the block and provide its erasure code pieces to the validators. Then, the relay chain phase begins. If the parachain phase is executed correctly, then the relay chain phase includes extra validation of a parachain block, adding the block header to the relay chain and finalizing that relay chain block. Otherwise, unavailability protocol is run between validators. The details are as follows:

Parachain Phase: We first describe the protocol by which collators of a parachain ρ submit a candidate block to the parachain validators assigned to $\mathcal{V}_{\rho,t}$, at some time t.

- For each parachain block B, a collator runs the $\mathsf{Prove}(B, PC, M)$ and obtains the proof π for the validity of block B. Then, he sends the blob $\mathbb{B} = (B, \pi, M)$ to the corresponding parachain validators to show that the block is valid and can be added to the relay chain.
- Each parachain validator $V \in \mathcal{V}_\rho$ checks the validity of the block by running the Verify algorithm for the blob $\mathbb{B}$. If it outputs 0, they ignore the block and store the block as invalid. Otherwise, V runs the $\mathsf{Divide}(\mathbb{B})$ to obtain the erasure code pieces $\mathcal{D} = \{d_1, d_2, ..., d_n\}$ of $\mathbb{B}$.
- V constructs a Merkle tree with leaves $d_1, d_2, ..., d_n$. Let us denote the root of the Merkle tree by r. At the end, V signs for each validator V_j the message $(d_j, r, \pi_i^{mt}, B_{info})$ and gives each signature σ_i^j to the distributor parachain validator. Here, π_i^{mt} is the proof that shows that d_i is one of the leaves of the Merkle tree with the root r and B_{info} is identifying information of B. We note that the distributor can be selected randomly or in order. Each parachain validator also gives the erasure code piece to at least one (trusted) collator in order to backup the piece.

Relay Chain Phase: This phase is to make sure that a valid and available block's header is in the relay chain.

- After receiving the signatures from parachain validators in the parachain phase, the distributor sends for each validator V_j all signatures $\{\sigma_j^k\}_{PV_k \in \mathcal{PV}'}$ where $\mathcal{PV}' \subseteq \mathcal{PV}$ representing the set of parachain validators who sent a signature for V_j.
- If V_j receives at least κ signatures for B, it adds it to the block list which keeps the data to be included to the relay chain. Later on, if this validator is the designated validator in BABE to produce the block, it creates block headers B_{head} (Definition 2) for the blocks in the block list. B_{head} including $\{\sigma_j^k\}_{PV_k \in \mathcal{PV}'}$. These signatures are necessary to show which parachain validators approve for B. Then, he produces the relay chain block which includes the header. See BABE for details.
- At the same time, each validator V_j examines each block in the relay chain if it includes unavailable block headers. If V_j sees a block header B_{head} in a block of the best relay chain whose erasure code piece d_j is not received by V_j (unavailable code piece), then he

asks for it first from parachain validators. If a parachain validator sends valid d_j, r, π_j^{mt}, then the protocol ends. Otherwise, it asks for it from collators of the corresponding parachain. If V_j receives a valid d_j, r, π_j^{mt}, the protocol ends. If V_j does not receive any response in agp time after seeing B_{head}, he runs the unavailability protocol in Section 2.1.

2.1 Unavailability

We integrate our availability and secondary validity check protocols directly with GRANDPA, in that an honest node $V \in \mathcal{V}$ cannot vote for relay chain blocks B unless

- V possesses their own piece for all the blobs referenced by B, and also
- V witnessed ...

We run the availability and secondary validity check protocols only for parachain blocks included in relay chain blocks, not for all parachain blocks proposed by parachain validators. In this way, we reduce the damage done by spammy parachains, at least beyond their own assigned parachain validators.

At the same time, we avoid complex anti-spam or prioritisation logic since only GRANDPA require this prior voting restriction. In fact, if we later require prioritisation logic then this trick isolates it inside relay chain block production.

We cannot entirely escape the availability question within BABE however: We have several forks $C_1, ..., C_k$ for which at most f validators possess all their chunks, but no fork for which $f+1$ validators possess all their chunks. Yet, each block producers V possess all their chunks for at least one fork C_i. If BABE were oblivious to availability, then V extends C_i, and GRANDA stalls under this configuration.

Instead, we define an availability grace period agp after which an unavailability subprotocol alters BABE's chain selection rule:

If a validator $V \in \mathcal{V}$ cannot obtain some piece x within agp time after seeing x referenced by some relay chain block header B_{head}, then V announces via gossip the unavailability of the piece x.

If any validator $V' \in \mathcal{V}$ observes unavailability announcements for pieces of some blob y from at least $f+1$ different validators. then V' shall not propose relay chain containing y in BABE. In this situation, there might be prevotes but never any precommits in GRANDPA for chains contain in y, so y cannot possibly be finalised by GRANDPA. We should consider if the GHOST chain weighting rule used by BABE and by GRANDPA for prevotes should weigh unavailability announcements, but doing so should only impact performance.

We considered V' abandoning any fork C for which at least $f+1$ different validators gossiped unavailability announcements for possibly distinct blobs in C, instead of for the same blob. Any truly unavailable pieces eventually trigger both conditions but they trigger this variant with possibly distinct blobs first, and with more false positives. We optimise for the honest case here and caution that more false positives results in more chains, so spammy parachains create could more load on the availability system.

Any validator should revoke their past unavailability announcements for some missing piece x whenever they eventually obtain x, again by gossiping the revocation. We also define some super availability grace period, longer than $|\mathcal{P}V|\mathsf{agp}$, after which time, if $2f+1$ of the validators announced unavailability of some specific piece, then the parachain validators who signed for that block are slashed. We revert this slash whenever revocations later reduce the unavailability announcements below $f+1$.

... As soon as receiving an unavailability announcement, the assigned validator sends the erasure code piece with the proof, if he has it. ... The details of how the assigned validator is selected and how he obtains the erasure codes are given in Section 2.2. ...

2.2 Validity Check

With the unavailability protocol, we can detect the block headers whose pieces are missing. Beyond this, we also need to detect whether invalid block headers are in the relay chain blocks. For example, if all parachain validators are malicious, they can sign for an invalid block. We can catch this only by reconstructing the blob with the available pieces. In the validity check protocol, we describe how extra validators (different from parachain validator) are assigned to check the validity of blocks and how malicious activities are detected.

We first give some parameters before giving the details of the protocol. The first parameter $\#V\mathsf{check}$ defines the total number of validator checks needed for a block header.

$$\#V\mathsf{check} = |\mathcal{P}V| + \mu + \lceil r_a + r_f \rceil$$

Here, μ specifies the minimum number of checks, r_a is the unavailability report factor from collators and r_f is the invalidity report factor from fishermen who claim invalidity for the relevant block.

There are a three-factor validity check during the availability protocol:

1. **Fisherman Check:** All fishermen collect blobs from collators in order to check the validity all the time. Whenever they discover an invalid block, they announce it to the validators by signing their claim. These claims are added to the block list by the validators. So, these claims are going to be in the relay chain. They do not provide any proof. If their claims are wrong then they are slashed. Otherwise, they are rewarded. The correctness of the fishermen claims are determined with the validity check protocol.

2. **Parachain Validator Check:** All parachain validators check the validity of the block as described in the parachain phase of the availability protocol. If one parachain validator PV_i sees that an invalid block header is in a block of the best relay chain, then he announces the invalidity. If a parachain validator has never seen the blob of the block header in the relay chain or he does not sign for that block, he runs the extra validity check protocol given below.

3. **Non-Parachain Validator Check:** The extra validity checks are done by the non-parachain validators. Now, we explain how these validators are selected.

   Selection: The selection process is private and no one knows the selected ones until they make it public. Assume that there exists m parachains in Polkadot. Each validator V_j checks first the following if required extra check is more than n/m (i.e., $\mu + \lceil r_a + r_f \rceil \geq n/m$):

$$ID_{PC} = \mathtt{VRF}_{\mathsf{sk}_j^v}(r_t) \mod m \tag{1}$$

Here, $ID_{PC} \in \mathbb{N}$ is the identifier of a parachain PC and r_t is the VRF randomness of the block producer of slot t in the relay chain (See BABE for the details of r_t). For the current time T where $T > \mathsf{agp}$, if less than $\#V\mathsf{check}$ amount of checks are completed with condition (1) or $\mu + \lceil r_a + r_f \rceil < n/m$, each validator V_j checks the following condition:

$$H(ID_{PC}||\mathtt{VRF}_{\mathsf{sk}_j^v}(r_t)) < \frac{\mu + \lceil r_a + r_f \rceil - \theta}{|\mathcal{V}| - |\mathcal{P}V|} + \tau \tag{2}$$

where $\tau = T - \mathsf{agp}$ and H is a hash function that maps a value between 0 and 1. Here, if $\mu + \lceil r_a + r_f \rceil \geq n/m$, θ is the number of validators in condition (1) for ID_{PC}. Otherwise, $\theta = 0$.

However, if a block producer produces multiple blocks with the same VRF randomness r_t (equivocation), then a validator V_j is selected for the extra check if one of the below two conditions together with (1) and (2) are satisfied:

$$ID_{PC} = \mathtt{VRF}_{\mathsf{sk}_j^v}(r_t||H_B) \mod m \tag{3}$$

$$H(ID_{PC}||\mathtt{VRF}_{\mathsf{sk}_j^v}(H_B)) < \frac{\#V\mathsf{check} - |\mathcal{P}V|}{|\mathcal{V}| - |\mathcal{P}V|} \tag{4}$$

where H_B is the hash of the relay chain block whose VRF randomness is r_t. Remark that we have greater number of checks for equivocation because it is a sign of malicious activity. Malicious validators can see who is in condition (1) or (2) and if there is no honest validator with the conditions (1) or (2), they can equivocate which includes an invalid block. Therefore, we need different conditions than the condition (1) and the condition (2).

When V_j satisfies the condition (1), (2), (3) or (4), V_j has a right to check the validity. For this, it follows the extra validity check protocol.

We note that if number of $\#V\mathsf{check}$ validators satisfy the condition (1) or (2) for a parachain with the identifier ID_{PC} and no equivocation exist, then satisfying condition (3) or (4) does not get rewarded.

Extra Validity Check: An assigned validator V_j asks for the blob for the extra validity check or the erasure code pieces in the following order until receiving all information to check the validity of the block header:

1. Parachain validators who signed for it: asked for the blob
2. Other parachain validators who did not sign for it: asked for the blob
3. Validators: asked for the erasure code piece with the proof. Assuming that V_j has a set $\mathcal{D}'$ of a correct erasure codes where $|\mathcal{D}'| < k$. He asks for at least number of $k - |\mathcal{D}'|$ pieces in $\mathcal{D} \setminus \mathcal{D}'$ from the validators.
4. Collators: asked for the erasure codes or the blob. We note that V_j can ask for the blob or the erasure code, if he is connected to the collators network.

In the end, if V_j's attempt in option 1 and 2 above is unsuccessful and obtains all missing pieces instead of blob $\mathbb{B}$, he adds them to the set $\mathcal{D}'$ and reconstructs $\mathbb{B} = (B, \pi, M)$ with the algorithm $\mathsf{Reconstruct}(\mathcal{D}')$.

In any case, either obtaining $\mathbb{B}$ from PVs or the reconstruction algorithm, V_j checks the validity of the block by running $\mathsf{Verify}(\mathbb{B}) \rightarrow b$.

- if $b = 0$ (the blob is invalid), V_j signs that the block is invalid. Whenever a validator receives a valid signature from another validator saying that the block is invalid, this validator obtains the blob the same way that extra validators obtain and follow the same process.
- if $b = 1$ (the blob is valid), V_j signs that the block is valid. V_j also runs the $\mathsf{Divide}((B, \pi, M))$ to obtain the erasure codes $\mathcal{D} = \{d_1, d_2, ..., d_n\}$ and constructs a Merkle tree whose leaves are $d_1, d_2, ..., d_n$. In the end, he gives away the missing piece of the validator with the Merkle tree proof whenever a validator announces unavailability.

In the end of the validity protocol, if at least $f + 1$ signatures are given by the validators saying that the block is invalid, then it is considered as invalid.

2.3 Slashing and Rewarding

Fisherman

Slashing: If at least $\#V\mathsf{check}$ validators sign saying that the block is valid, then fishermen with the claim saying the block is invalid are slashed. All the claims from these fishermen are ignored in future.

Rewarding: If $f + 1$ validators sign to say the block is invalid, the fishermen with the claim saying that the block is invalid are rewarded.

Validators

Slashing: If at least $f + 1$ validators sign for the invalidity of a block and less than $f + 1$ validators sign for the validity of the block, then we slash all validators who signed for the validity.

If at least $f + 1$ validators sign for the validity of a block and less than $f + 1$ validator sign for invalidity, we slash the ones who signs for the invalidity.

We note that $f + 1$ validity signatures and $f + 1$ invalidity signatures are not possible given that at most f validators are malicious and proof of block algorithms are correct (See Definition 4).

Rewarding: All parachain validators and extra validators who signed for the final decision (either valid or invalid) of a block receive a reward.

Eligibility of the reward can be easily checked for the parachain validators and extra validators who satisfies the conditions when $\tau = 0$. Latter, we just need to verify the VRF proof. However, it is not easy to verify the condition for $\tau > 0$, because this value is subjective. Therefore, we may not need to be very precise for this τ value, if the signature of the validator is correct.

3 Security of Availability and Validity

Theorem 1 (Availability)**.** *Assuming the Merkle tree constructed from collision resistant hash functions and a block in GRANDPA can be finalized with at least $2f + 1$ votes, then the availability protocol is secure.*

Proof. Assume that an unavailable block is finalized. It means that this finalized block includes a block header of which at most f honest validators has the correct erasure code piece (See Definition 5). If this block is finalized, it means that

- either not all honest validator who does not have the erasure code piece announce the unavailability. If they would announce it, since their number is at least $2f+1-f = f+1$ (i.e., honest parties - honest parties who do not have the erasure code), the other honest parties do not finalize it and malicious parties do not have enough number to finalize it. Therefore, we can assume that there is at least one honest party who do not announce the unavailability of its erasure code in this case, because he thinks that he has one but actually it is not correct. If there exists a one honest party who has incorrect erasure code, but having the proof that it is correct (provided by parachain validators), then it means that the collision resistance assumption of Merkle tree is broken.
- or $f+1$ parties announces unavailability it means that GRANDPA finalize the block with at most $2f$ parties. So, this implies that the security of GRANDPA is broken by finalizing a block with less than $2f+1$ votes.

Since the GRANDPA is secure and the Merkle tree is collision resistant, the unavailability protocol is secure. □

Theorem 2 (Validity)**.** *Assuming that the availability protocol is secure, the signature scheme is EF-CMA secure, the hash function H in Conditions (2) and (4) is a random oracle, the PoV protocol is secure and correct (Definition 4), the VRF is secure and a block in GRANDPA can be finalized with at least $2f+1$ votes then the invalid block is finalized in the relay chain with probability less than risk probability.*

Proof. (Sketch) TODO: Security reduction, Rewrite the security arguments

If there is one honest PV and there is not any $f+1$ unavailability report, then at least $2f$ of the honest parties has the piece. It means that PV can collect all the pieces from the honest parties in order to reconstruct the blob and check its validity. Therefore, if there is at least one honest party in PV, it can detect the invalidity and announces it. In the end, we will have more than f invalidity signatures since the other parties also check invalidity after seeing the signature saying that the block is invalid.

Therefore, we analyze the case where all parachain validators are malicious. If there is at least one honest validators who is assigned for extra validity check then the same case happens as having at least one honest parachain validators. Therefore, we need to find out the probability of having all parachain and extra-check validators are malicious given that all parachain validators are malicious.

There are three cases that the extra check validators are selected: Either with only condition (1) or with condition (1) and (2) or with condition (4). Let's analyze the success of malicious validators in these three cases. Below, $\mu' = \mu + \lceil r_a + r_f \rceil$ which is the required number of non-parachain validator validity check and $f' < n/3 - n/m$ is the number of non-parachain malicious validators and $|PV| = n/m = nc$ where $c1/n$.

$$\mathsf{Pr}[\text{Cond. (1)}] = \mathsf{Pr}[\text{Cond. (3)}] = p_{vrf} = \frac{1}{m}$$

$$\mathsf{Pr}[\text{Cond. (2) at time } \tau] = p_{\tau} = \frac{\mu'}{n - nc} + \tau$$

$$\mathsf{Pr}[\text{Cond. (4) }] = \frac{\mu'}{n - nc}$$

1. The malicious validators generate an invalid block when they see that at least μ' malicious validators satisfy the condition (1). This can happen with probability

$$\sum_{i=\mu'}^{f'} \binom{f'}{i} \frac{1}{m^i}(1 - \frac{1}{m})^{f'-i} \tag{5}$$

which is not negligible. When this happens, their attack succeeds if no honest validator satisfies the condition (1). Then, the attack probability is the following:

$$\begin{aligned} \mathsf{Pr}[\mathsf{attack1}] &= \mathsf{Pr}[\text{no honest satisfies (1)}] \\ &\leq (1 - \frac{1}{m})^{2n/3} \\ &\leq e^{-2n/3m} \end{aligned} \tag{6}$$

2. The malicious validators generate an invalid block. In this case their attack succeeds if no honest validator satisfies the condition (1) and the condition 2 until a time where at least μ' malicious validators satisfy the condition 2. For simplicity, we divide time in discrete values $[\tau_0 = 0, \tau_1 = u, \tau_2 = 2u, ..., \tau_k = ku]$. τ_k can be computed by malicious validators before the validation process begins. In any case, the only way not to be caught is not to have any honest extra-check validator until ku. The probability of this attack is the following:

$$\begin{aligned} \mathsf{Pr}[\mathsf{attack2}] &= \mathsf{Pr}[\mathsf{attack1}] \prod_{i=0}^{k} (1 - p_{\tau_i})^{2f+1} \\ &\leq e^{-2n/3m} \prod_{i=0}^{k} (1 - \frac{\mu' + \tau_i n(1-c)}{n(1-c)})^{\frac{2n}{3}} \\ &= e^{-2n/3m} \exp(\sum_{i=0}^{k} -\frac{2}{3}\frac{\mu' + \tau_i n(1-c)}{1-c}) \\ &= \exp(-2/3(\frac{n}{m} + \frac{(k+1)\mu'}{1-m/n} + nu\frac{k(k+1)}{2}) \end{aligned} \tag{7}$$

3. In this attack, the adversary equivocates the block. Here, we can assume that the adversary knows who satisfies the condition (1) since if the malicious validator is the block producer, he first produces the block and sees who validates this block. If no honest validator checks it with condition (1), he equivocates it. We know that probability of no honest party satisfies the condition (1) is given in inequality (6). When this happens, the attack succeeds if no honest party satisfies the condition (3) and (4). The probability of attack 3 is

$$\begin{aligned}\mathsf{Pr}[\mathsf{attack3}] &= \mathsf{Pr}[\text{no honest check in cond. (4) and (3)}] \\ &\leq \mathsf{Pr}[\mathsf{attack1}](1 - \frac{\mu'}{n - nc})^{\frac{2}{3}n} \\ &\leq \exp(-2/3(\frac{n}{m} + \frac{\mu'}{1 - m/n}))\end{aligned}$$

□

3.1 Parameter Selection

Remark that all the attacks in the proof of Theorem 4 can succeed if all parachain validators are malicious and this happens with the probability $\frac{1}{3^{n/m}}$.

All attacks above depend on two parameters, μ and n/m. Therefore, we need to specify these parameters so that the attack probabilities are less than the risk probability. The risk probability is critical here and we should find a way to define it. Informally, the risk function is defined as follows: When the certain conditions are satisfied to execute attack 1, attack 2 or attack 3, whatever the adversary gain when he executes the attack needs to be smaller than whatever he loses when the attack is unsuccessful.

$$\mathsf{Pr}[\mathsf{attackX}]\mathsf{Gain} < (1 - \mathsf{Pr}[\mathsf{attackX}])\mathsf{Lost} \tag{8}$$

We can assume that Gain at most equals to all stake of validators (I am not sure how logical to define it like this) and Lost equals to stake of parachain validators. In this case, we obtain

$$\frac{\mathsf{Pr}[\mathsf{attackX}]}{1 - \mathsf{Pr}[\mathsf{attackX}]} < \frac{sn/m}{sn} = \frac{1}{m} \tag{9}$$

The relation between n and m: When we compare the probability of attacks, clearly, attack 1 is more probable than the others. Therefore, we need to make sure that probability of attack 1 satisfies the relation in (9). In order to have this relation, we should have the following inequality between n and m:

$$n > \frac{3m\ln(m+1)}{2} \tag{10}$$

Selection of μ: Since attack 1 is the most probable attack, we also need to make sure that the necessary condition to execute attack 1 should happen very rarely. Remember that the necessary condition to execute the attack 1 is having at least μ malicious validators in condition (1). To bound this probability, we consider the case when all parachain validators are malicious which can happen with probability $\frac{1}{3^{n/m}}$. Assuming that the set of parachain validators changes in each epoch, the probability of satisfying the necessary condition in one slot to execute attack 1 (Equation (11)) should be less than $\frac{1}{\zeta}$ where ζ is the expected number of malicious slots in one epoch.

$$p'_1 = \mathsf{Pr}[\text{at least } \mu \text{ malicious in cond. (1)}] = \sum_{i=\mu}^{f'} \binom{f'}{i} \frac{1}{m^i}(1 - \frac{1}{m})^{f'-i} \tag{11}$$

According to BABE practical results (e.g., $c = 0.034, T = 3$ and $k = 54$), the epoch length should be around 20000 slots and expected number of malicious slots are around 215. With this constraint, we need to have the parameter μ as shown in Figure 1 with respect to m.

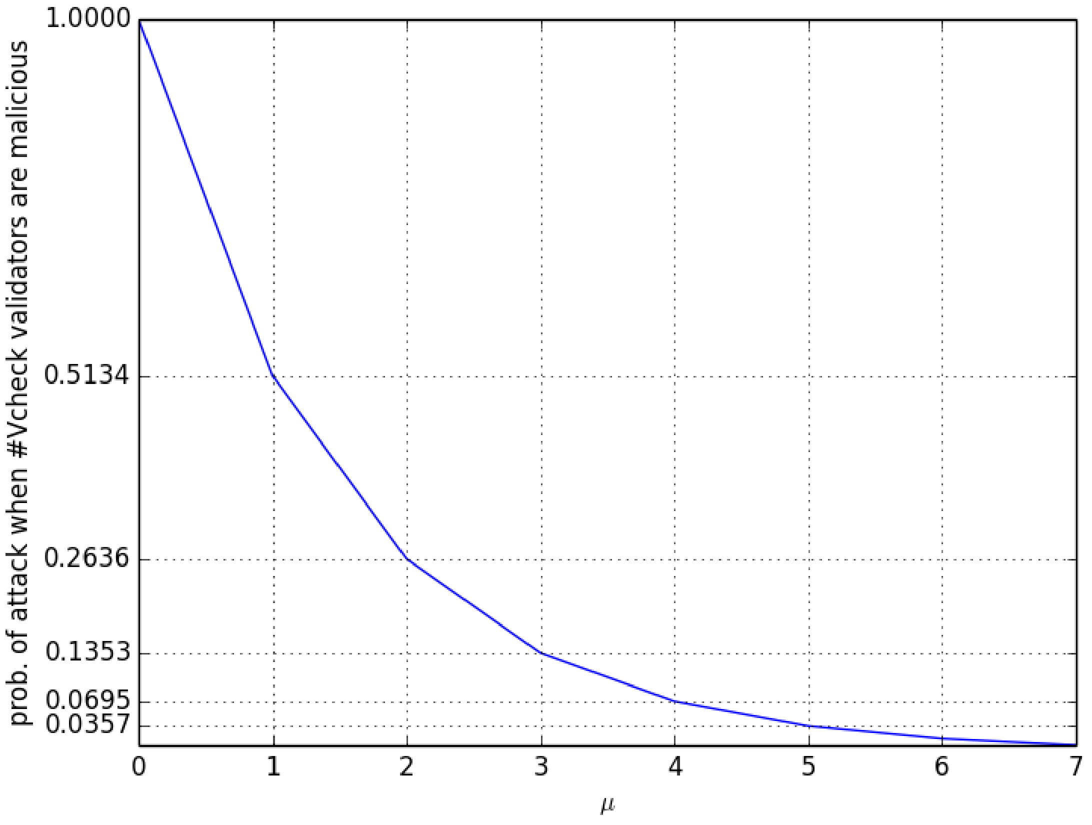

Figure 1: The parameter μ according to number of parachains

Now, let's analyze attack 2 and attack 3 with the parameterization above.

Actually, there is not any condition for attack 2 which means that it can be executed in any block. The reason of this is that there exists always a $k \in \mathbb{N}$ which let at least μ malicious validators being selected for the extra check in every block. When the adversary executes the attack 2, we assume that $x < \mu$ malicious validators satisfy the condition (1) and at least $\mu - x$ malicious validator satisfy the condition (2) until the time τ_k. The best case for the adversary in order to have the highest probability in attack 2 (See (7)) is when $k = 0$. Therefore, the best case probability is the following:

$$\begin{aligned} p'_2 &= \sum_{x=0}^{\mu-1} \mathsf{Pr}[\text{x malicious is in cond. (1)}]\mathsf{Pr}[\text{at least } \mu - x \text{ malicious in cond (2) when } k = 0] \\ &= \sum_{x}^{\mu-1} \binom{f'}{x} \frac{1}{m^x}(1 - \frac{1}{m})^{f'-x} \sum_{i=\mu-x}^{f'} (\frac{\mu}{n - n/m})^i (1 - \frac{\mu}{n - n/m})^{f'-i} \end{aligned} \tag{12}$$

We can see that with the selection of the parameter μ according to p'_1 in Equation (11), $p'_2 > 1/\zeta$. This means that the malicious parties will have more chance to be in the best case in one epoch. However, since the probability of attack 2 is very low considering the gain/lost relation, any rational malicious validator do not execute the attack 2.

Lastly, we analyze attack 3. The condition for attack 3 is to have no honest validator satisfying the condition (1) which can happen with the probability

$$p'_3 = \mathsf{Pr}[\text{no honest in condition (1)}] = (1 - \frac{1}{m})^{2n/3}. \tag{13}$$

In the analysis of attack1, we fix this probability to almost $1/m$. Similar to attack 2, if $m < \zeta$, then parachain validators are malicious, they have more chance to execute the attack 3 during their period. With the same probability as in attack 2, executing attack 3 is too risky for the adversary considering its gain and lost.

The claim that any rational adversary doe not execute attack 2 or attack 3 can be clearly seen in the Figure 2.

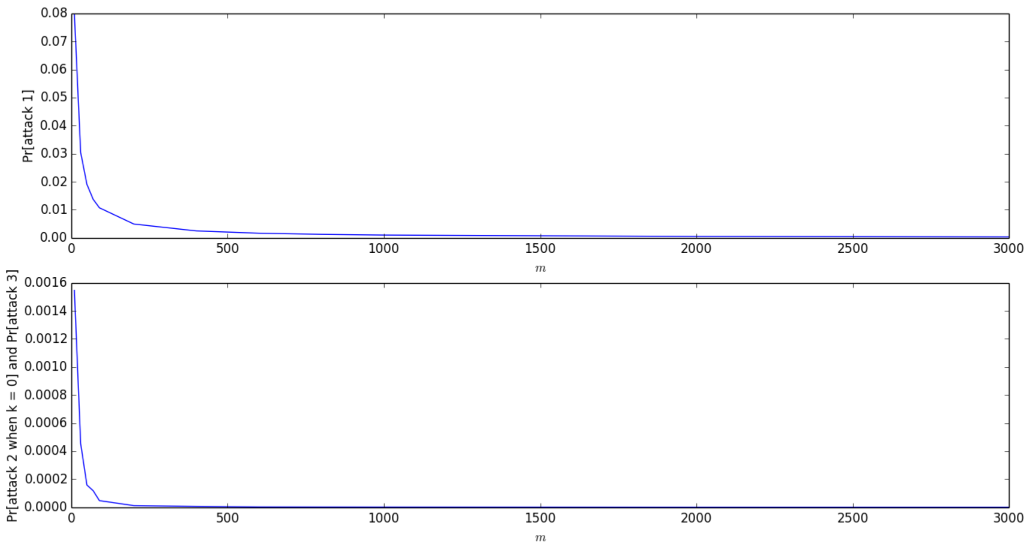

Figure 2: The success probabilities of attack 1, 2 and 3

We give the total attack probabilities (i.e., $p'_X \mathsf{Pr}[\mathsf{attackX}]$)is given in the Figure 3. As it can be seen the success of the attacks are very low. Remember that all these attacks can be happen when all parachain validators are malicious. So, in this case, they have this probability. Otherwise their success probabilities are 0.

4 Security of Availability and Validity

Theorem 3 (Availability)**.** *Assuming the Merkle tree constructed from collision resistant hash functions and a block in GRANDPA can be finalized with at least $2f + 1$ votes, then the availability protocol is secure.*

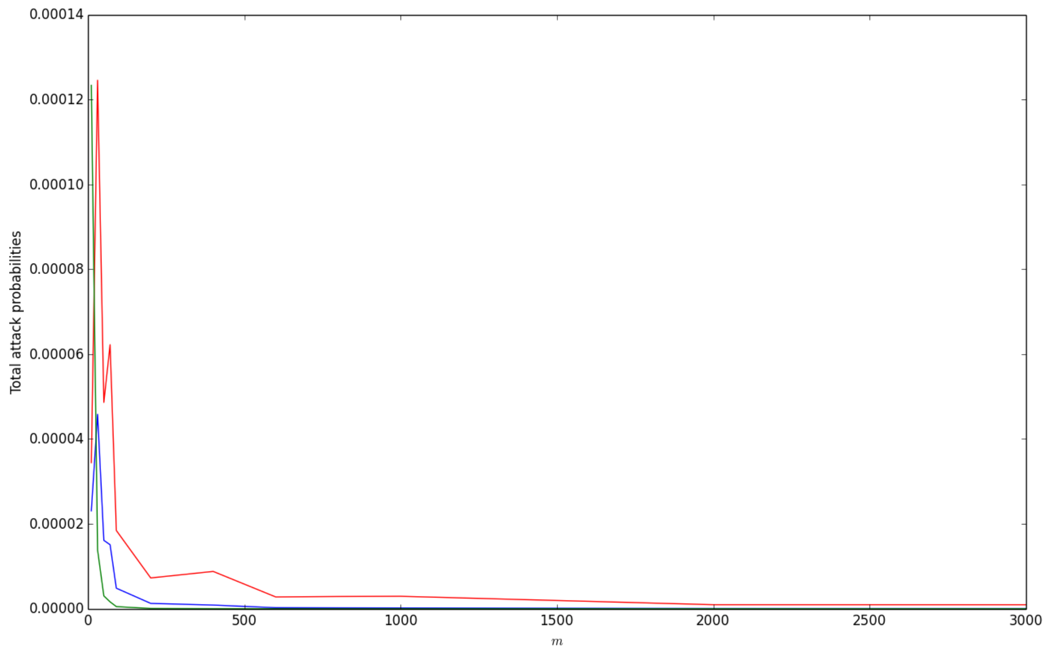

Figure 3: The total attack probabilities. Red is for attack 1, blue is for attack 2 and green is for attack 3.

Proof. Assume that an unavailable block is finalized. It means that this finalized block includes a block header of which at most f honest validators has the correct erasure code piece (See Definition 5). If this block is finalized, it means that either

- not all honest validator who do not have the erasure code piece announce the unavailability. If they would announce it, since their number is at least $2f+1-f=f+1$ (i.e., honest parties - honest parties who do not have the erasure code), the other honest parties do not finalize it and malicious parties do not have enough number to finalize it. Therefore, we can assume that there is at least one honest party who do not announce the unavailability of its erasure code in this case, because he thinks that he has one but actually it is not correct. If there exists one honest party who has incorrect erasure code, but having the proof that it is correct (provided by parachain validators), then it means that the collision resistance assumption of Merkle tree is broken.

- or $f+1$ parties announce unavailability means that GRANDPA finalized the block with at most $2f$ parties. So, this implies that the security of GRANDPA is broken by finalizing a block with less than $2f+1$ votes.

Since the GRANDPA is secure and the Merkle tree is collision resistant, the unavailability protocol is secure. □

Theorem 4 (Validity)**.** *Assuming that the availability protocol is secure, the signature scheme is EF-CMA secure, the hash function H in Conditions (2) and (4) is a random oracle, the PoV protocol is secure and correct (Definition 4), the VRF is secure and a block in GRANDPA* *can be finalized with at least $2f+1$ votes then the invalid block is finalized in the relay chain with the probability less than risk probability.*

Proof. (Sketch) TODO: Security reduction, Rewrite the security arguments

If there is one honest PV and there is not any $f+1$ unavailability report, then at least $2f$ of the honest parties has the piece. It means that PV can collect all the pieces from the honest parties in order to reconstruct the blob and check its validity. Therefore, if there is at least one honest party in PV, it can detect the invalidity and announces it. In the end, we will have more than f invalidity signatures since the other parties also check invalidity after seeing the signature saying that the block is invalid.

Therefore, all parachain validators have to be malicious not to be caught. If there is at least one honest validators who is assigned for an extra validity check with either of the conditions, then the same case happens as having at least one honest parachain validator. Therefore, an invalid block cannot be detected (the attack succeeds) if all parachain validators are malicious and all extra check validators are malicious as well.

Remark that the extra check conditions (1) and (2) with $\tau = 0$ and (4) and (3) guarantees that the expected number of checks by parachain validators and the extra check validators is $\#V\mathsf{check}$. Therefore, we can consider that the parachain validator and extra-validator selection mechanism actually randomly samples $\#V\mathsf{check}$ validators for each parachain. So, the probability of all selected validators for a particular parachain being malicious is bounded by $(\frac{1}{3})^{\#V\mathsf{check}}$. See 1 for the details.

□

Theorem 5. *Given that $\#V\mathsf{check}$ malicious validators are eligible to check the validity of a block header, the invalidity attack is bounded by the probability $\exp(-\frac{2}{3}(\mu+r))$ if $\mu+r < n/m$ and the probability $\exp(-\frac{2n}{3m})$ if $\mu+r \geq n/m$.*

Proof. Below, we give the probability of any validator being in conditions (1), (2), (3), (4):

$$\mathsf{Pr}[\text{Cond. (1)}] = \mathsf{Pr}[\text{Cond. (3)}] = p_{vrf} = \frac{1}{m}$$

$$\mathsf{Pr}[\text{Cond. (2) at time } \tau] = p_\tau = \frac{\mu'}{n-nc} + \tau$$

$$\mathsf{Pr}[\text{Cond. (4) }] = \frac{\mu'}{n-nc}$$

We assume that $|\mathcal{P}V| = \frac{n}{m}$. The number of $\#V\mathsf{check} - |\mathcal{P}V| = \mu + r$ malicious validators can be selected for the extra check with the following cases:

1. $\mu + r \geq \frac{n}{m}$ and at least $\mu + r$ malicious validators are in condition (1) and condition (2) until the time τ_k for the parachain whose validators are malicious. The attack probability is highest if at least $\mu + r$ malicious validators are in (1). In this case, the attack succeeds if no honest validator satisfies the condition (1)

$$\mathsf{Pr}[\mathsf{attack1}] = (1 - \frac{1}{m})^{2n/3} < e^{-2n/3m}$$

2. $\mu+r < \frac{n}{m}$ and at least $\mu' = \mu+r$ malicious validators are in condition (2) until the time τ_k. For simplicity, we divide time in discrete values $[\tau_0 = 0, \tau_1 = u, \tau_2 = 2u, ..., \tau_k = ku]$. τ_k can be computed by malicious validators before the validation process begins. The probability of this attack is the following

$$\begin{aligned}
\mathsf{Pr}[\mathsf{attack2}] &= \prod_{i=0}^{k}(1-p_{\tau_i})^{\frac{2n}{3}} \\
&\leq \prod_{i=0}^{k}(1-\frac{\mu'+\tau_i n(1-c)}{n(1-c)})^{\frac{2n}{3}} \\
&= \exp(\sum_{i=0}^{k} -\frac{2}{3}\frac{\mu'+\tau_i n(1-c)}{1-c}) \\
&= \exp(-2/3(\frac{(k+1)\mu'}{1-m/n} + nu\frac{k(k+1)}{2})
\end{aligned}$$

Remark that $\mathsf{Pr}[\mathsf{attack3}]$ is maximum when $k = 0$. When $k = 0$, $\mathsf{Pr}[\mathsf{attack3}] \leq \exp(-\frac{2\mu' n}{3(n-m)}) \leq \exp(-\frac{2}{3}(\mu+r))$.

3. This case considers equivocation: Here, we can assume that the adversary knows who satisfies the condition (1) and (2) since if the malicious validator is the block producer, he first produces the block and sees who validates this block. If no honest validator checks it with condition (1), he equivocates it. When this happens, the attack succeeds if no honest party satisfies the condition (3) or (4). Hence, the probability of attack 4 is

$$\begin{aligned}
\mathsf{Pr}[\mathsf{attack4}] &= \mathsf{Pr}[\text{no honest check in cond. (4) and (3)}] \\
&\leq \mathsf{Pr}[\mathsf{attack1}](1-\frac{\mu'}{n-nc})^{\frac{2}{3}n} \\
&\leq \exp(-2/3(\frac{n}{m}+\mu+r))
\end{aligned}$$

□

5 Practical Results

As it can be seen from the proof of theorem 4, malicious validators can attack the validity protocol with probability $(\frac{1}{3})^{\#V\mathsf{check}}$. Remember that $\#V\mathsf{check} = |\mathcal{PV}| + \mu + r$ where $r = \lceil r_a + r_f \rceil$. If we assume that parachain validators change in every x minutes, the malicious parties needs to wait $x3^{\#V\mathsf{check}}$ minutes in order to succeed the attack. For example, if $x = 5$ minutes, then the total time for an attack for each $\#V\mathsf{check}$ is given in Figure 4:

As it can be seen, if $\#V\mathsf{check}$ is more than 14, the adversary needs to wait more than 50 years for the attack. Therefore, in terms of security, it is important to have minimum $\#V\mathsf{check}$ validators.

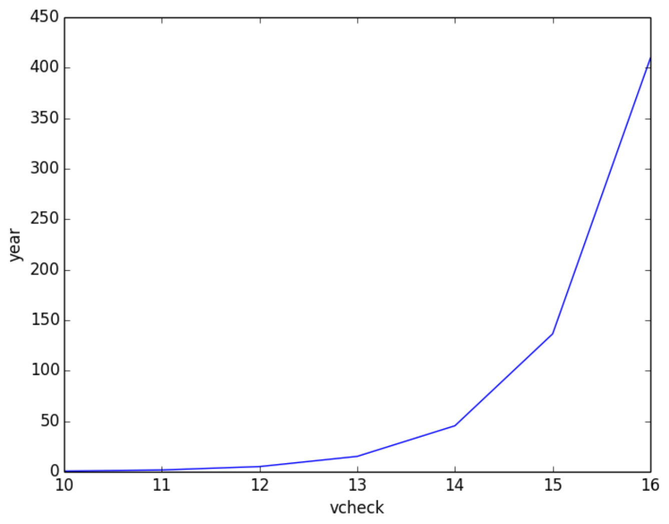

Figure 4: It shows the waiting time for an adversary for a successful attack given that every parachain validators changes every 5 minutes

The parameter μ: The possible μ (minimum number of extra check) values:

- If $\mu = 0$, then $|pv| = 14$ in order to make sure that minimum $\#V\mathsf{check}$ is always equal to 14. Given that $|pv| = n/m$ where m is the number of parachain validators $n = 14*m$.
- If $\mu > 0$, then $|pv| < 14$. It means that we need less validators for the security than in the case where $\mu = 0$. Less validators imply less network delay and less network delay implies more secure BABE. On the other hand, if μ increases, it means that more validators need to reconstruct a blob in order to check the validity. So, validators need to do more work. In terms of scalability of the relay chain, we need to decide μ.

As rule of thumb, if we don't care about the number of validators as far as it is not the maximum number of validators (N) that Polkadot can handle, then if $14 * m < N$, $\mu = 0$. Otherwise, $\mu = 14 - N/m$.

Another disadvantage of having $\mu = 0$ is that parachain validators have a less risky attack when all of them are malicious. Imagine a parachain with a few collators. We can assume that they may be malicious and collaborate with the malicious validators. In this case, the validators will not have any report. So, there will be 0 extra check. As soon as all parachain validators are malicious in this malicious parachain, they can add invalid block headers and cannot be caught. The security argument says here that they need to wait around 50 years for this, so the attack is not possible. However, in the real life, since the attack does not have any risk, the collators can bribe parachain validators with their stake and parachain validators validate an invalid blob.

Theorem 5 gives the risk that malicious validators take at when they put an invalid report. If $\mu + r \leq n/m$, the invalid block will not be detected with probability $\exp(-\frac{2}{3}(\mu + r))$. In the worst case scenario, if no report is received then the attack probability is $\exp(-\frac{2}{3}(\mu))$. We give in Figure 5 that the probability of a successful attack in the case of $\#V\mathsf{check}$ malicious validators are selected. As it can be seen in Figure 5, their risk is exponentially increasing when μ increases. In order to make the risk close to 0 even if no report received, μ should be greater than 4.

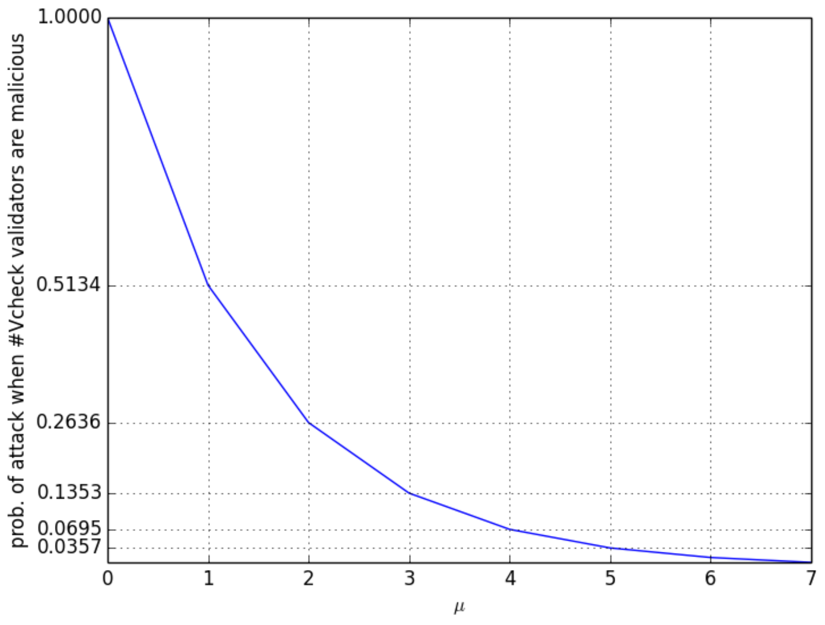

Figure 5: The risk of malicious validators when they attack even if $\#V\mathsf{check}$ validators are malicious

Fisherman and Collator Reports: In the validity scheme, we rely on the invalidity reports of fishermen and unavailability reports of collators to find the parameter $\#V\mathsf{check}$. However, we need to bound $\#V\mathsf{check}$ so that these reports do not make too many validators to check the validity. Let us call this bound μ_{max} (i.e, $\mu + \lceil r_f + r_a \rceil \leq \mu_{max}$). This bound is necessary because regular malicious reports can slow down the process easily.

When a fisherman sends a report of invalidity, but later on it is found to be valid, the fisherman is slashed. Therefore, the reliability of a fisherman report can be measured by how much it is staked. Considering this, r_f can be computed as follows:

$$r_f = \frac{\sum_i \mathsf{stake}_{f_i}}{v_{gain}}$$

Here, stake_{f_i} is the stake that a fisherman f_i puts for this report and v_{gain} is the DOT value that a validator receives in each block. In a nutshell, a fisherman needs to stake at least the same amount that the validator earns for each block in order to make a validator to execute an extra validity check. If the fisherman report is not valid, then the fisherman pays for this extra check. So, a malicious fisherman has to spend $\mu_{max} v_{gain}$ for a report to slow down the validator network. We believe that this model discourages fishermen to send false reports.

However, we cannot measure the reliability of unavailability reports as fisherman's reports since it is not possible to show the correctness or incorrectness of any unavailability reports. Malicious collators do not lose anything by just sending fake unavailability reports. In order to solve this issue, we assign a parameter $\alpha \in (0,1)$ for a parachain that defines the proportion of honest collator assumption. Depending on α, we have define r_a for two cases:

- if $\alpha > 1/2$, $r_a = \mu_{max}^{x/\alpha}$

  $$r_a(x) = \begin{cases} \mu_{max}^{x/\alpha} & \text{if } x \leq \alpha \\ \mu_{max} & \text{otherwise} \end{cases}$$

  where
  $$x = \frac{\text{total unavailability reports}}{\text{all collators}} = \frac{\sum_{c_i \in C_P} st_{c_i}}{|C_P|}.$$

  Here, $st_{c_i} \in \{0,1\}$ and it is 0 if the block is available. Otherwise, it is 1. The reason of using a function such as $\mu_{max}^{x/\alpha}$ is to make sure that we have extra checks close to maximum check only if the number of unavailability reports are close to the number of honest collator assumption. Thanks to this, if the number of honest collators are majority, the malicious collators who want to slow down Polkadot cannot achieve to make always maximum number of checks with only unavailability reports. In general, these type of parachains can be investigated by fishermen as far as the blocks are available to honest collators. Therefore, we expect that if there is any invalid block, the fishermen catch it and send a report. If the blocks are unavailable to honest collators (so fishermen too), validators checks the invalidity with the maximum capacity.

- if $\alpha < 1/2$, then it is critical to give more importance to the unavailability reports from this parachain. Therefore, we use the following linear function:

  $$r_a(x) = \begin{cases} \mu_{max} \frac{x}{\alpha} & \text{if } x \leq \alpha \\ \mu_{max} & \text{otherwise} \end{cases}$$

So, $\mu' = \mu + \lceil r_a + r_f \rceil$ if $\mu + \lceil r_a + r_f \rceil \leq \mu_{max}$. Otherwise, $\mu' = \mu_{max}$.

The parameter μ_{max}: Given the fact that we have maximum number of extra checks when there are many invalidity or unavailability reports, the parameter μ_{max} needs to be big enough so that the probability of having at least one honest extra check is almost 1. This probability can be bounded by $1 - (\frac{1}{3})^{\mu_{max}}$. Therefore, we can have $\mu_{max} = 15$.

In Figure 6, we compute the number of extra checks required depending on the α-parameter of a parachain. As it can be seen, the parachain that we trust less requires more extra-checks. The parachain where we assume honest majority reaches the maximum check when the unavailability reports are close to the number of honest collators.

A Proof Details

Lemma 1. *For any slot number s, block producer bp, epoch randomness r for the epoch that includes s, integer $k > 0$, and parachain P, with probability at least $1-(f/n)^{-k}$ over the epoch*

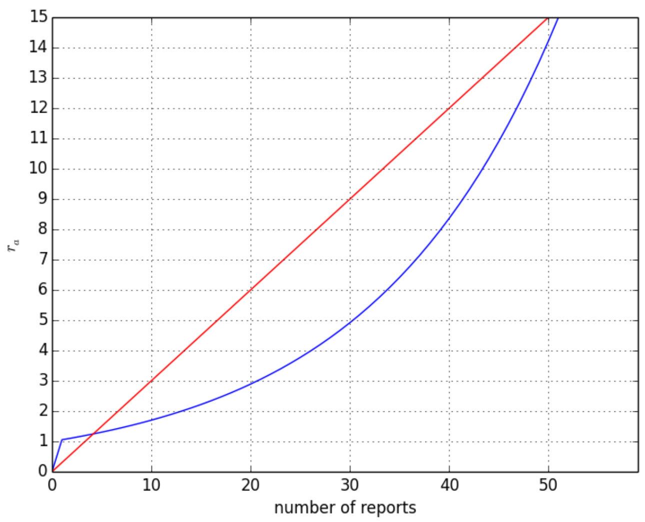

Figure 6: The red and blue graph shows the number of required extra checks when $\alpha \le 1/2$ and $\alpha > 1/2$. Here, we assume that the total number of collators are 100.

randomness and the random oracles for the hashes and VRFs, for any block that bp produces in slot s that includes a parachain header from P, if we eventually have $\#V\mathsf{check} \ge k$ for this parachain header, then some honest validator checks the parachain block.

Proof. Consider generating an ordering ℓ of validators as follows, first we have the parachain validators in a uniformly random order, then all validators who satisfy condition (1) (using bp's VRF for slot s) in a uniformly random order, then we order all remaining validators in increasing order of the hash $H(ID_{PC}||\mathsf{VRF}_{\mathsf{sk}_j^v}(V))$ used in condition (2), breaking ties due to collisions in a uniformly random order. We claim that distribution of ℓ is as a uniformly random permutation of the validator set. To see this note that the only operation for which the different validators are treated differently are the VRFs and we assume that they are random oracles. The parachain validators are a set of n/m validators chosen uniformly at random using r. Each validator satisfies condition (1) with the same probability independently and so the distribution of the set of validators satisfying (1) conditioned on its size is a set of that size chosen uniformly at random from the non-parachain validators. Also the hashes in (2) are independently and identically distributed.

Next we show that for any block bp produces in slot s which includes a parachain header for P, if any honest validator ever sees that $\#V\mathsf{check} \ge k$, then any honest validators in the first k validators in ℓ eventually check that block. All honest parachain validators and honest validators who satisfy (1) check. If all honest validators in the fist k validators in ℓ satisfy these conditions we are done. So suppose that v is an honest validator in the first k validators in ℓ who doesn't satisfy either of these conditions. v will still check if τ is large enough and they don't see attestation from $\#V\mathsf{check}$ validators who either are parachain validators, satisfied condition (1) or had a smaller hash in condition (2). But noting that all such validators are before v in ℓ so there can be at most $k-1$ of them. Thus v only ever counts $k-1$ attestations towards the number needed $\#V\mathsf{check}$ which is eventually $\ge k$. So when $\#V\mathsf{check} \ge k$ and τ is large enough, v checks.

Finally we need to show that the probability that the first k validators in ℓ are honest is at least $(f/n)^k$. Since the first k validators in ℓ are distributed as a set of k validators chosen uniformly at random, this probability is $\prod_{i=0}^{k-1}(f-i)/(n-i) \leq (f/n)^k$. $\square$