Security Q&A

The security of our network is of paramount concern to us, thus we are starting a series of posts that discuss our robust design. For a start, this article answers two important questions that came up during our recent interactions with developers.

[

800×510

](https://cdn-images-1.medium.com/max/800/1*2gGlbjkyxyPWMBTX1fWB5w.jpeg)

Q: How do you mitigate if a malicious group creates a signed block with bad cross-shard transactions ?

I will start with explaining the minimal requirements of our system. At the first day of the launch, Elrond protocol will have at least 800 validators, 400 for a shard and another 400 for metachain. We have a requirement that any shard has to contain at least 400 validators, otherwise the shard will not be created, or it will be merged with another.

A few more highlights:

- BFT requirement / assumption 75% of the total nodes are good actors;
- Probabilities are calculated with the assumption that 25% of nodes are malicious. On shard level we accept a maximum of 33% of malicious nodes. Calculations are done with 10 shards, 4000 nodes from which 1000 are malicious.
- The initial validator to shard allocation is random, randomness source for comes from the metachain. When a new validator comes in, it is allocated at random to an existing shard. At most 30% of the nodes are reshuffled at the end of every epoch;
- The consensus group size in metachain is 400, the leader changes at every round (5 seconds), in order to sign a block ⅔+1 has to sign it, which is 267 validators;
- The consensus group size in a shard is 63, it is selected at random from the 400 buffer, changing at every round. ⅔*63 + 1 equals 43 — needed validators to sign a block.
- The random seed is a chain of seeds, un-biasable, un-predictable, unchangeable. The leader of the current block signs the random seed of the previous block with his private key, using BLS single signature scheme, that number is hashed and becomes the random seed for the next group selection;
- BLS single signature — a signed message with a private key result is fixed;
- Block finality: block N is final only if block N-1, block N-2 … block N-K are signed. Metachain only notarized final blocks. Currently we have chosen K = 1;

The probability that a malicious super majority (>67%) to be selected for the same round in the same consensus is 10^-9, even if 33% of the nodes from the shard are malicious. In that case they can propose a block and sign it — let's call it BLOCK M, but it will not be notarized by metachain. Metachain notarizes BLOCK M, only if BLOCK M+1 is built on top of it. In order to create BLOCK M+1 the next consensus group has to agree with BLOCK M. Only a malicious group will agree with BLOCK M, so the next group must have a malicious super majority again. As the random seed for group selection cannot be tampered with, the probability of selecting one malicious super majority group is ~10^-9 — to be exact 5.38 * 10^-10.

The probability of signing two consecutive malicious blocks equals with selecting two subgroups with at least (⅔*63 + 1) members from the malicious group consequently. The probability for this is: ~10^-18.

Furthermore, the consequently selected groups must be colluding, otherwise the blocks will not be signed.

We demonstrated that the protocol is provably secure against invalid transactions.

Q: How do you mitigate shard-takeover ? Is shard-takeover possible ?

The protocol is designed in such a way that it is provably secure and the probability of shard takeover is extremely low: 2.88*10^-78

. A sharded architecture has to be constructed in such a way, that it makes impossible for a shard to be taken over.

Shard takeover resolution

:

Fallback solution

, even for this impossible cases. Status: research done, two possible solutions, implementation planned after first test net launch.

Solution 1:

At creation of each block, the leader will add a proof that money was not created out of thin air. This proof will be verified by metachain and by each destination shard as well.

[

800×453

](https://cdn-images-1.medium.com/max/800/1*uVDEUUnoKAcMcKdqvHXgXw.png)

Source: Sharded Chains as Data Layers

Solution 2:

When the leader proposes a block he adds in the header also a merkle proof for a few accounts that changed their balance during the block execution. The selection of these accounts needs to be deterministic, e.g take accounts from first transaction in each miniblock and provide an aggregated proof that goes from all these accounts to the state root hash registered in the block. When one invalid block is proposed by a malicious majority, the state root is tampered with an invalid result (after including invalid changes to the state tree).

By providing the combined merkle proof for a number of accounts, this allows a challenge for that proof to be raised by an honest node. The honest nodes will provide the block of transactions, the previous reduced merkle tree with all affected accounts before applying the challenged block and the SC states. If the proofs are not provided in the bounded time frame, the challenge is considered incomplete and all messages will be dropped. No slashing occurs, challenged block is considered valid.

The cost of one invalid challenge is the entire stake of the node.

This will allow metachain to try and apply the changes as it already has previous state tree (just for affected accounts) and will be able to detect the inconsistency, either invalid transaction, or invalid state root. This can be traced and the consensus group can be slashed. At the same time the challenger can be rewarded with part of the slashed amount. We need to benchmark what is the size of such a proof for maximum allowed transactions in the block. As well if the problem comes regarding a smart contract operation, then the execution of a SC is needed, but for this we can still ensure at least that no funds have been created without validating the correctness of SC execution.

The solution is further optimized by sending the reduced merkle tree proof only on challenge

, the challenger presents both the reduced proof before applying the invalid block and the block itself.

However a malicious group can even hide the block from other nodes — non-malicious ones. In this case the honest nodes, even if they would be aware that new blocks have been produced(by seeing new headers notarized by metachain), they could not raise challenges because they do not have access to the block data. It is impossible to prove that.

The solution is to make mandatory the propagation of each produced block to the sibling shard, and have them confirm the reception, send confirmation to the metachain. In this case, we could have challenges being raised also from the sibling shard, as those nodes have access to the blocks and could also verify them. Another advantage with this is that there is another channel where honest nodes can request the data if they are denied by their own shard nodes. The communication overhead is further reduced by sending only the intrashard miniblock to the sibling shard. The cross shard miniblocks are always sent on different topics accessible by interested nodes. In the end, challenges can be raised by multiple honest nodes.

Another protection is given by how the P2P topics and messages are set up. The communication from one shard toward the metachain is done through a defined set of topics / channels — metachain will not accept any other messages from other channels.

This solution introduces some delay in metachain only in case of challenges,

which are very low in number and highly un-probable since if detected (high probability of being detected) the nodes risk their entire stake.The metachain consensus will execute verification of multiple such proofs coming from different shards. We only notarize the blocks from shards where there are no outstanding challenges, and for the others with challenges try to process them as soon as possible, with a first come first served priority, and do the slashing either for the challenger if there was a false alarm or for the group if the challenge was validated.

Furthermore, if multiple such challenges are validated the metachain can trigger an early end-of-epoch command. In this case the malicious nodes are instantly slashed and 30% of the total nodes reshuffled — those who still have enough stake. This ensures that no shard takeover will block the protocol for a long time period and there are enough nodes in each shard

that can ensure security. Otherwise if we just slash a few times and do not end the epoch, each time we remove 43 nodes from the shard we are left with fewer and fewer nodes and the overall shard security decreases.

So we described the solution in case the impossible happens, now we prove that the probability is practically 0 for a malicious super majority group to be formed inside a selected shard.

For shard takeover the malicious group needs to have ⅔+1 members in a single shard from the consensus buffer of 400. This equals 267 validators. Taken into consideration the above assumptions, 25% of the total nodes are malicious, the probability of the malicious groups has super majority in one shard is 2.88*10^-78

.

If we consider having 33% of total network nodes as malicious, the probability of having 267 malicious nodes (metachain take over attack) is 2.84*10^-47

. Same, if we consider having 400 malicious nodes inside of a shard (full take over attack), in which there is no honest node able to raise a challenge, the probability is practically 0 (10^-211

).

If there are less malicious members than ⅔+1, but more than ⅓ of the validators in one shard (less than 267, more than 133), then the shard will only stagnate, it will not create any transactions, as none of the blocks will get its finality status, good nodes will not construct over bad blocks.

If there are even less than ⅓ of the validators in one shard we reach the situation described and explained in the first question.

In conclusion, we demonstrated that the system is provably secure against shard take-over attacks.

Furthermore, in order to propose a bad transaction a malicious group would need to have 400 malicious nodes inside of a shard (full take over attack), in which there is no honest node able to raise a challenge, the probability is practically 0 (10^-211

) and 400 more in the sibling shard (** probability 10^-211).

Because of the fisherman challenge, we might have a 99% Fault Tolerant Sharded System, needing only 1 honest node / 2 shards.

Medium – 19 Jun 19

Security focused Q&A

The security of our network is of paramount concern to us, thus we are starting a series of posts that discuss our robust design. For a…

Reading time: 7 min read

Elrond's official outlets

:

- Elrond Github: https://github.com/ElrondNetwork
- Elrond Community Platform: https://community.elrond.com
- Twitter: https://twitter.com/elrondnetwork
- Official website: www.elrond.com