CSD 2nd Project Report

Authors:
Pedro Afonso 70357 Senhong Chen 70369

Scoring the Relays:

When considering the security of each relay, we are considering our adversary model to be each individual country, which we individually score based on the alliances set provided in the input.

We assume each country is only as truthful as the least trusted alliance they belong to. If they don't appear on the alliance list we attribute a default value of trust of 0.05, because we don't really want to consider countries which aren't specified by the client, but we don't want to eliminate those relays completely. If we can't find a relay's country we give it a security score of 0.

For scoring guards, their base security value is equal to the trust score of the relay's country. However, if the guard country and the client country are the same, a penalty is applied equal to the trust score of the country and the alliance respectively in order to protect the client's identity.

Scoring exits works similarly to guard scoring for calculating the base score, but we score exits based on, not only the client location, but also the destination location and a selected guard. This time we apply a penalty for the countries of the pairs client-exit, guard-exit and guard-destination, in order to mitigate possible correlation attacks

Guards Selection Strategy:

The system will iterate over all nodes in the database, calculate each node's score based on its various attributes, and finally return a sorted list of tuples (guard, score).

After scoring, the system uses the **filter_relays** function to classify all candidate guards into "**safe**" and "**acceptable**" categories based on their scores and bandwidth limits, stopping once the **max_bandwidth** is reached. The system prioritizes selecting guards from the "**safe**" group; if there are not enough "**safe**" nodes, it supplements from the "**acceptable**" group. If the total number of candidates in both groups remains insufficient, the system enters an edge handling phase where filtering parameters are relaxed to allow more suboptimal but viable guard nodes to be selected, ensuring both connection availability and security.

Exit Selection Strategy:

For each of the previously selected guards, we check every relay to check if they can be used as exits. A relay may be used as an exit if their exit

policies accept the destination ip address, the client may also provide a port for their destination address to further filter exits. Any candidate exit node that belongs to the same family, has the same ASN as the guard, or is the guard itself will be immediately assigned a score of zero and excluded from the selection.

We score each valid exit for each guard and only keep the one that has the highest score for a given guard. We could keep more valid exits, but we decided that, in order to improve performance for the algorithm, we should keep only the most trusted one.

Once we have a set of guard-exit pairs, we score them once again. This time, we multiply the minimum trust score between the guard and the exit with the minimum of their bandwidth, but the trust score is elevated by 4, because it is more important that a node is trusted than their bandwidth. We then select the pair with the highest score.

Middle Selection Strategy:

The middle relay is not as important as the other ones in terms of security, so we only consider bandwidth. Given a guard and exit relays, a random relay is chosen out of all the valid relays for the pair. A relay is a suitable middle if it is not in the same family or ASN as the other two nodes. This random choice is weighed by each relay's bandwidth.

Handling of edge cases:

We have implemented a progressive filtering mechanism to ensure the construction of **guard-exit** pairs even in edge cases. This mechanism performs up to five selection attempts, during which it gradually relaxes the trust score thresholds, specifically, the **accept_upper** and **accept_lower** parameters—to expand the candidate pool. Additionally, by allowing the system to choose between safe and acceptable nodes, it can fall back to acceptable options when a sufficient number of safe nodes are not available. After the 5 attempts, it is assumed that no valid path can be found, so the program writes a message stating it was unable to find a safe path.

Conclusion:

Our solution balances security, performance, and reliability through a trust-based scoring system and dynamic relay selection. We prioritize guards and exits for security, while simplifying middle selection for efficiency. Edge cases are handled with progressive filtering to ensure route availability. Trade-offs were made deliberately to maintain secure paths without sacrificing connectivity.