HW09

Riccardo Zucchelli 1984963

October 2024

1 Introduction

The objective of this homework is to create a protocol that is able to make two people play a game of *"Rolling Dice"*, and make the match secure against cheating by both players.

The rules

Rules are pretty simple: it's a two-player game in which the opponents have a set of K dice that will be thrown simultaneously. Each dice produces randomically a value between 1 and 6, and the result of the throw is summed up to calculate a score for the game. The game is won by whoever got the highest score, and if they are the same the game is a tie. A game can be repeated a fixed number of times for a match, and whoever wins the most games wins the match.

Game phase

The game is structured in 2 major phases, assuming that handshaking part has been already established:

- **Dice throwing**: Every player throws their dice using a pseudo-random algorithm K times, to ensure fairness. Once every dice has been thrown their values are temporarily stored, and the score is calculated by summing every value. The score will need to be securely stored so that manumission of it wouldn't be possible without the other party noticing it;
- **Reporting**: Once dice are thrown and the results calculated, the values are shown to the other so that they can both determine the fair winner of the game. This part includes the verification of the score of each player, that will need to be done via cryptography.

The goal of this is to also not rely on any third party, so that the entire game can be peer-to-peer.

2 Vulnerability assessment

In this kind of game there is plenty of space for a person to cheat. We'll go through some of the ways and methods in which this can be done.

Cheating by manipulating memory

It's very difficult to check for memory manipulation before the checks run, in fact it's the main objective of what multi-billion dollar companies try to create with *Anti-cheats*. These tools usually work by means of inserting special values on random addresses that a memory manipulator would change and also by aggressively monitoring the user's pc, while sometimes invading their privacy, and for this reason this particular way of cheating won't be looked at, but we can assume that memory can be readable.

Cheating by reporting fraudulent score

Any player could in theory ignore the results provided by the dice thrower, and instead send to the other side an arbitrary score value. This is our main focus, as it undermines the integrity of the game and can be conducted without requiring sophisticated tools or techniques. To address this, the protocol must ensure that scores are both authentic and verifiable independently by both parties, by providing to the other part also the seed used to generate that dice score.

It could be easily implemented by having a third-party that verifies the match between the two, but that is out of scope for the homework.

3 Implementation

To address this, the protocol leverages the concept of **commitment schemes**. A commitment scheme ensures that a party commits to a value (such as the seed used for generating dice rolls) without revealing it, while being unable to change the value later. This guarantees fairness and verifiability. The process can be structured as follows:

1. Each player generates a random seed to drive their pseudo-random dice roll generation. This seed, combined with a unique identifier (such as a timestamp or player ID), is hashed using a secure cryptographic hash function to produce a commitment.

2. The commitment is shared with the opponent before rolling the dice, ensuring that neither player can adjust their rolls after seeing the other's results.

3. After both commitments are exchanged, the players reveal their seeds along with the results of their dice rolls. Each player can verify the integrity of the other's data by checking if the revealed seed, when hashed, matches the original commitment.

This approach ensures that both players' dice rolls remain unpredictable before commitment, and any attempt to alter results post-commitment becomes detectable. By including details such as the session key and the timestamp associated with the match, the protocol also mitigates replay attacks and rainbow tables, where an adversary might reuse a previously valid commitment or just precalculate values.

4 Limitations

While the proposed implementation effectively mitigates many forms of cheating, it is not without limitations. One primary concern is the reliance on both players honestly following the protocol, as the system assumes they will correctly generate and reveal seeds. A malicious player could, for instance, delay revealing their seed or intentionally provide incorrect information to disrupt the game. Additionally, the use of pseudo-random number generation, even with a carefully chosen seed, can introduce vulnerabilities. For instance, using a combination of timestamp, username, and session key as the seed might seem secure, but a malicious player could exploit this setup. By precomputing multiple dice roll outcomes for various seeds within the allowable time frame, they could selectively report the seed that produces the most favorable results. This type of manipulation undermines the fairness of the game and highlights the importance of designing the protocol to prevent seed exploration or retrospective seed selection.