**Discussion**

**5.1 Discussion Methodology**

After collecting and analysing the data provided by the real and pseudorandom generators sampled, the final step was to evaluate each implementation’s overall performance. As well as these individual evaluations, where possible the generators were compared to each other, since another aim of the investigation was to attempt to identify which generators performed better than others. A key point that was considered during this discussion was made in Section 4.9, in which it was stated that when reviewing all the test data gathered and attempting to compare suitable generators the datasets able to not necessarily outperform but remain constantly effective across many tests should be considered the more effective solutions.

**5.2 C# Evaluation**

The C# implementations were broken down into two main categories: the Rand functions and the Cryptographic function. The Rand implementations, both seeded and unseeded, performed well in the dice, coin, and card simulations which showed that the generators effectiveness remained relatively unaffected by input and output limitations. During Chi-Squared testing, all the Rand implementation datasets ranked between the desired distribution ranges, with the dice simulation data performing just slightly better than the coin simulation data. When producing shuffled deck data, the distribution of suits for both implementations was acceptable, with most groups being between two or three cards large. None of these tests presented any clear weighting or favoured outputs within the functions. During each of the dealing tests, the hands provided aligned with the expected odds of a poker deck, with the most common hands being High Card or Pairs. The distribution of values, seen for instance in the Kolmogorov-Smirnov test, with the numeric sequence Rand datasets followed the expected trend of pseudorandom data and while never being the top performers in any of the tests conducted, both the seeded and unseeded Rand implementations consistently passed. Unfortunately, the same couldn’t be said about the Cryptographic implementation which performed poorly in many of the numeric sequence tests conducted. Despite this however, it is still believed that the function can produce a valid pseudorandom sequence and the main issue presented by the tests was that they weren’t designed for the cryptographically secure sequences being produced. The purpose of such a function is to offer the user a non-reproducible collection of bits that can be used for encryption, not a mass-produced collection of values simulating randomness.