Progress Report

The goal of the project:

Computers are limited regarding the biggest integer they can store and process. What happens when we need to compare or check for equality between integers that are too big for a standard computer? The literature over the years has develop many algorithms to solve this problem. The goal of this project is to implement and analyse the efficacy of the best algorithm we know.

How do we represent integers we can't write explicitly on a computer? We use arithmetic circuits with operations {+, *}. Consider the following example:
x0 = 1
x1 = x0 + x0 = 2
x2 = x1 * x1 = 2*2 = 2^2
x3 = x2 * x2 = 2^2 * 2^2 = 2^(2^2)
…
xn = xn-1 * xn-1 = 2^(2^(n-1))
We see that using arithmetic circuits we can create very big numbers with few operations.

How do we compare two arithmetic circuits? The algorithm described by Arnold Schönhage simply chooses a random integer and calculates the module of each operation of the two circuits, if the final residues are equal then there is a high probability the two integers are equal. The algorithm is not deterministic, there is a random component that affects greatly the results we obtain. Depending on the random integer we choose we can get false positives. The project focuses on experimenting with the algorithm and trying to understand the random component, so we get satisfying results.

The methods that are being used:

To analyse the algorithm many tests have been develop that will give as input two arithmetic circuits and many random numbers of different sizes, so we can compare the relation between the size of the random component and the probability of failure. The biggest integer we can make with the smallest number of operations is 2^(2^n), note that all arithmetic circuits start with x0 = 1. We have tested many pairs or circuits; observation indicates that the algorithm struggles the most when we compare 2^(2^n) to 2^(2^(n-a)) for $1 < a < n$.

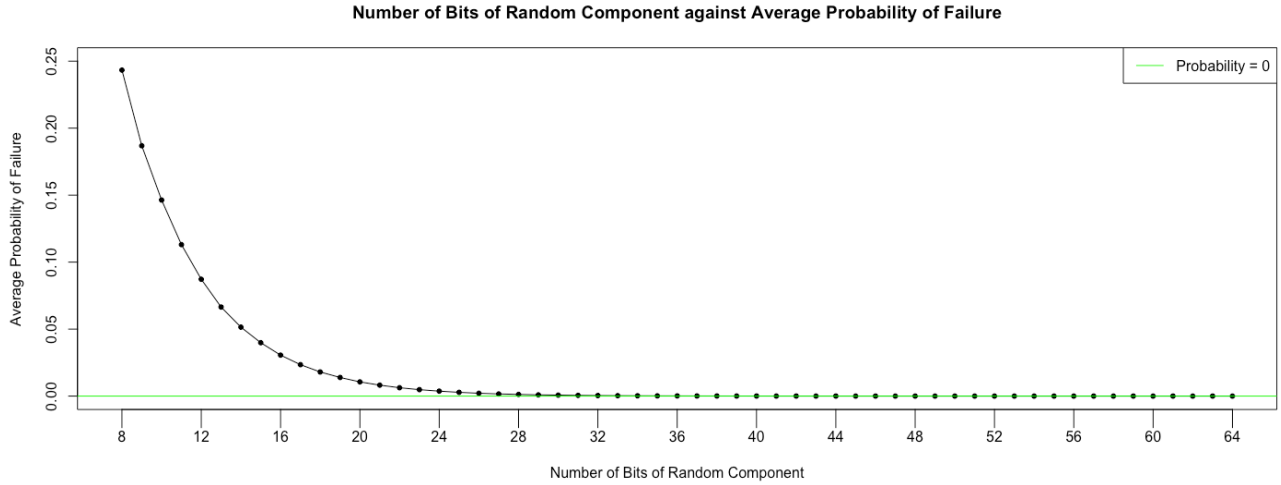
How do we choose the random integer? The ideal case is finding a relation between the size of the input (the two arithmetic circuits) and the size of the random integer needed for getting good probabilities of success. We have classified integers in ranges depending on how many bits are needed to represent them: $2^{(b-1)} \leq n \leq 2^{(b)} - 1$ where b = bits. Hence, we use integers in different ranges and calculate the probability of failure.

What has been accomplished so far:

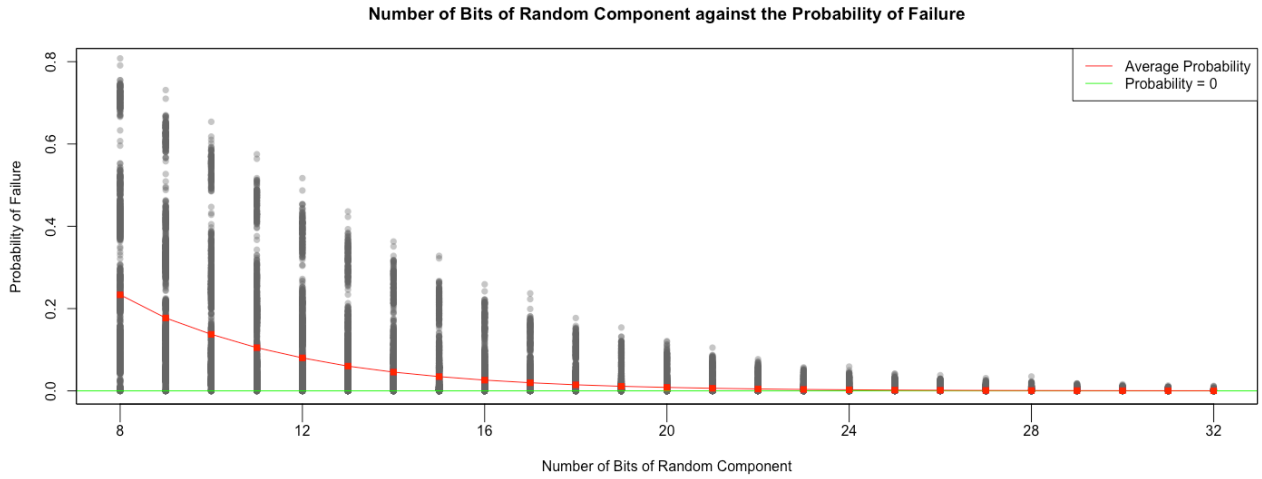
The algorithm has been implemented and tested. Many automatic tests have been created that grab random integers in different ranges and calculate the probability. Big data sets have been generated and analyse.

We have obtained very interesting results:

The biggest data set compared 2^(2^128) vs. 2^(2^127); 2^(2^128) vs. 2^(2^126); … ; 2^(2^128) vs. 2^(2^1); … … ; 2^(2^127) vs. 2^(2^126); 2^(2^127) vs. 2^(2^125); … ; 2^(2^127) vs. 2^(2^1); … … … … ; 2^(2^2) vs. 2^(2^1). It randomly selected 1000 integers in each range going from 8 bits to 64 bits and calculated the probability of failure for each pair or circuits in each range.



We also compared 2^(2^64) vs. 2^(2^63); 2^(2^64) vs. 2^(2^62); … ; 2^(2^64) vs. 2^(2^1); … … ; 2^(2^63) vs. 2^(2^62); 2^(2^63) vs. 2^(2^61); … ; 2^(2^63) vs. 2^(2^1); … … … … ; 2^(2^2) vs. 2^(2^1). It randomly selected 1000 integers in each range going from 8 bits to 32 bits and calculated the probability of failure for each pair or circuits in each range.



We clearly see that the bigger the random integer we choose the smaller the average probability of failure becomes. The worst probability also decreases as the size of the random integer increases.

What remains to be done to complete the project:

To complete the project more testing needs to be done so we can have more examples and more data to support possible hypothesis about the relationship between the input and the size of the random integer. All the writing needs to be completed as well specially the findings and the background.