**Final Project: integer factorization/primality testing method**

**Weijie Sun**

Abstract:

Followed by the fundamental theorem of arithmetic, every integer greater than 1 has a unique prime factorization. The reason I want to explore the integer factorization/primality testing methods. The integer factorization method is widely used in cryptography. I will use the Trail division method as the base line, and compare the other two method complexity and performance with the trial division.

1. **Algorithm**

1.1Trial division (base line)

Trial division is the easiest integer factorization algorithms. For the precondition of this algorithm, the prime number generation has been given. In this algorithm the time complexity depends on the prime number generation and the required time grows exponentially with the digits of the number.

* 1. Fermat’s factorization method.

Since if an odd number can be represent as N = a\*\*2 – b\*\*2, then N = (a-b)(a+b). The basic idea is find the closed integer a which a\*\*2 – N is a root of square.

* 1. Euler’s factorization method.

The basic idea for this method is if N = a\*\*2 + b\*\*2 = c\*\*2 + d\*\*2, then N can represent as N = A\*B.

1. **Project**:

2.1 Try to write the programs for Trial division, Fermat’s factorization method, and Euler’s factorization method.

2.2 Compare the complexity and performance by different type of number, include the very big prime number, very big two prime number’s production, and a normal very big number.

2.3 Compare the complexity and performance by mathematic method.

2.4 Analysis the best and worst case for each method according to 2.2 and 2.3.