1 Introduction

Integer factorization is known to be a computationally hard to solve problem. The state-of-art algorithms for integer factorization has subexponential complexity. Some of the most popular cryptographic algorithms are by the successful factorization of the modulus. With the wide spreed of cheap general purpose computers, availability of supercomputers and the design of hybrid hardware for factorization, the recommendation of RSA's modulus size has reached 4096 bits.

A typical factorization algorithm is trial division which has been shown to have exponential complexity. There has been many attempts at designing new algorithms most of which relied on the difference of squares, Pierre de Fermat's method or other variations of it. Over time, improvements on those gave us the quadratic sieve and the number field sieve algorithms. The quadratic sieve is known to be the fastest integer factorization algorithms for numbers less than 100 digits. On the other hand, the Number Field Sieve (NFS) has a better asymptotic complexity and is faster for larger numbers.

Generally, factoring algorithms are of two types deterministic and non-deterministic. Non-deterministic algorithms are not guaranteed to end, and could run for an indeterminate amount of time. Pollard's ρ factoring method **??** is one of the earlier attempts at designing an algorithm that doesn't guarantee a result but sometimes can be faster than other deterministic algorithms like trial division. Deterministic algorithms guarantee the end of an algorithm at most in a worst-case running time, have been a topic of research for a long time. A lot of progress in the design and improved of deterministic general integer factoring algorithms has been made over time resulting in the current NFS.

Solving the factorization problem breaks many proposed public key cryptosystems most famous of which is the RSA cryptosystem. Such cryptosystems are used in many critical real life application and their compromise is not acceptable. There has been a debate on whether solving the RSA problem is equivalent to factoring, but it is definite that factoring RSA numbers breaks the cryptosystem.

In this work, we use a modified Fermat's factorization to factor RSA numbers by making use of relations between the product and the sum of two prime numbers. These relations allow us to nearly halve the search space of the algorithm. We show that such relations can be applied multiple times without loss in the reduction of tries.

In section **??** we prove that there are only $\frac{h-1}{2}$ or $\frac{h+1}{2}$ possible values of sums for any product of two primes modulo a certain odd prime h. After that we show how to compute these values with different methods depending on the algorithms requirement. In section **??**, we show how to use the ideas from section **??** with the formulas introduced in [**?**]. Section **??**, generalizes and studies the use of multiple prime filed mappings to further reduce the search space.

1.1 Problem statement

Let $n = pq$ where p and q are distinct odd primes, given n find p and q. To solve this we make use of the following proven in [**?**]:

1. $\sqrt{r^2 - 4n}$ is an integer iff $r = p + q$, for any r between $2\lceil\sqrt{N}\rceil$ and n.

2. If $r = p + q$, then $\sqrt{r^2 - 4n} = p - q$ and $p = \frac{(p+q)+(p-q)}{2}$

To factor n we search for an integer number r in the interval $[\lceil 2\sqrt{n}\rceil, n]$ then evaluate equation **??**. If we find r that produces an integer from **??**, then a system of two linear equations can be written and solving for p and q results in $p = [(p + q) + (p - q)]/2$.

2 Background and Related Work

2.1 The RSA Cryptosystem

RSA is a well-known cryptosystem that was introduced in 1977 by Ronald Rivest, Adi Shamir, and Leonard Adleman [**?**]. Rivest, Shamir, and Adleman invented RSA to provide confidentiality and authenticity to digital information travelling through insecure communication channels. RSA is one of the first cryptosystems to implement asymmetric encryption using public key cryptography. Before RSA, people relied on symmetric encryption whereby two parties must share the same key in order to communicate secretly. This shared key requirement is especially difficult in practice. Moreover, if one person wants to privately communicate with multiple people, this person needs to share a different key with each one of them adding more to the complexity of the problem. Public key cryptography solves this problem by using two different keys. One key is used for encryption, and that key is distributed publicly, while the other private key is used for decryption and it must be kept secret. If Alice wants to communicate with Bob in a secure way, she would use Bob's public key to encrypt her message and then send the resulting cipher text to Bob. Using his private key, Bob then decrypts the cipher text to obtain the original message from Alice.

The inventors of RSA needed a special type of mathematical functions called one way trapdoor functions in order to implement public key cryptography. Such function is easy to compute for any given input but hard to invert without an additional piece of information called the trapdoor. The function they used was modular exponentiation where the encryption of a message m can be easily computed by raising m to some exponent e and taking the remainder after dividing by some number n called the modulus. The integer n is the product of two large prime numbers p and q and e is an odd integer such that $e \geq 3$. The pair (n, e) is distributed as the public key. To reverse this computation, another exponent d is needed to undo the effect of e and that integer d is the trapdoor. The integer d is calculated such that $ed = 1\phi(n)$ where ϕ is Euler's totient function and $\phi(n) = (p - 1)(q - 1)$. The exponent d must be kept secret along with the integers p and q. To break RSA, one must find d given only (n, e) which requires factoring n to find its prime factors p and q.

2.2 The RSA Problem

Breaking RSA is known as the RSA problem. It is formally defined as follows: Compute M given a public key (n, e) and a ciphertext $C = M^e n$ [?]. It is believed to be as difficult as the integer factorization problem, however, no definite proof has been found. Clearly a solution to the integer factorization problem also solves the RSA problem, however, it is still unknown if the opposite is also true. On the one hand, some research shows that breaking RSA is not equivalent to integer factorization for a very small public exponent [?]. On the other hand, multiple researchers suggest that the RSA problem and integer factorization are equivalent [?, ?].

The fastest known method for factoring large numbers is the General Number Field Sieve [?]. Another method is the Fermat Factoring Attack which can be used if the factors p and q are very close to each other [?]. Moreover, Elliptic Curves have been used to factor large integers [?].

There are other ways to break RSA without factoring n. One way is to solve the discrete logarithm problem to find the private exponent d that satisfies the equation $M = C^d n$. This is also a difficult problem that has not been solved efficiently yet [?]. Other attacks target the RSA cryptosystem itself. The most notable attacks include: Common Modulus, Low Private Exponent, Low Public Exponent, Hastad's Broadcast Attack, Franklin Reiter Related Message Attack, Coppersmith's Short Pad Attack, Partial Key Exposure Attack, and other implementation attacks all explained by Dan Boneh in [?].

Despite the large number of attacks against RSA, it is still considered secure if implemented properly and large key sizes are used.