

# The Exponent $\lambda(R)$ and Its Consequences on RSA

Andrew Misseldine

14 May 2009

## **Abstract**

In RSA systems, the private key is the multiplicative inverse of the published key modulo some secret number derived from the factorization of  $R$ , e.g.  $\phi(R)$  and  $\psi(R)$ . So essentially the secret of RSA depends not just on factorization techniques of  $R$  but also on order computation of elements in the group with secret order.

We will begin by presenting a general algorithm using the Carmichael function  $\lambda(n)$  to factor a semiprime or discover the private key for a RSA cryptosystem. The strengths, weaknesses, benefits and limitations of this algorithms will also be presented. Then we will discuss the algorithm's complexity as well as possible modifications to the algorithm.

# 1 Introduction

Cryptology is the mathematics of secrets. It is a mathematical science which hides information with the use of one-way functions, that is a function which is relatively easy to implement but relatively difficult to invert. The example in which this paper is based upon is the procedure of multiplication and factorization. Computationally if one is given two numbers, let us say two prime numbers, it is very easy to calculate their product. On the other hand, if one is given a number and told it is the product of two primes, it is extremely difficult to factor the product and discover the original two primes. The RSA cryptosystem is based on the difficulty of factoring. The general factorization problem has not yet been solved with any polynomial-time algorithm. If it were, then RSA would become obsolete. But still, RSA is not without its weaknesses, and there are many special cases of factoring or alternatives to factoring that can be used to attack a RSA key.

The general RSA schema works as following: A user, whom we will call Alice, secretly chooses two distinct prime number. We will call these secret primes  $p$  and  $q$ , and assume  $p < q$ . Alice then multiplies  $p$  and  $q$  together such that  $R = pq$ . This  $R$  will be published publicly by Alice. Alice may then compute  $\phi(R)$ , the Euler totient function or the number of relatively prime natural numbers less than  $R$ . Since  $\phi$  is multiplicative, if two natural numbers  $a$  and  $b$  are relatively prime, or coprime, then  $\phi(ab) = \phi(a)\phi(b)$ . Thus for the modulus chosen by Alice,  $\phi(R) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$ , since  $p$  and  $q$  are prime.

Next, Alice chooses another natural number  $e$  such that  $e$  is coprime with  $\phi(R)$ . Alice then computes the modular inverse of  $e$ , namely  $d = \frac{1}{e} \bmod \phi(R)$ . Now,  $e$  is also published publicly but  $d$  is kept secret. If an attacker to Alice's key  $(R, e)$ , we will call her Eve, would like to read encrypted messages sent to Alice, she will need this secret key  $d$ . If Eve were able to factor  $R$ , then she may follow this same procedure and compute  $d$ . But alternatively, if Eve knew the value of  $\phi(R)$ , she could also compute  $d$  without the factorization of  $R$  at all. In this paper we will discuss a method on how to find  $\phi(R)$  but skipping the factorization of  $R$ .

## 2 The Function $\lambda(R)$

There are a few preliminaries we need to clarify first of all. Consider the ring  $\mathbb{Z}_R$  with modular addition and multiplication as its operations. This ring is of order  $R$ , but the order of its multiplicative group, that is the set of all multiplicatively invertible elements under multiplication which we denote as  $\mathbb{Z}_R^*$ , is  $\phi(R)$ . This value will be our goal in this paper, and since it is the order of a naturally occurring group, we may use group theory to compute  $\phi(R)$  without the factorization of  $R$ . One important fact to mention is that  $1 \leq \phi(R) \leq R$ .

Of special interest to us is the theorem of Lagrange: Let  $H$  be a subgroup of a finite group  $G$ . Then the order of  $H$  is a divisor of the order of  $G$  (Fraleigh [2]). An immediate corollary of this theorem is that: Let  $g \in G$ , then the order of  $g$  divides the order of  $G$ . Also, we define the **exponent** of a group  $G$  to be the smallest positive integer  $m$  such that, for any element  $g \in G$ ,  $g^m = 1$  (Rotman [3]). We see immediately that for a finite group  $G$ , the exponent is the least common multiple of the orders of the elements of  $G$ . So, by Lagrange's Theorem, the exponent of  $G$  divides the order of  $G$ . We shall denote the exponent of  $\mathbb{Z}_R^*$  as  $\lambda(R)$ , also called the Carmichael function.

From here on out, we will be concerned with order computation of elements in the group  $\mathbb{Z}_R^*$ , as this will give us clues to the order of  $\mathbb{Z}_R^*$ . At this time, we will assume we have a procedure which inputs an element from a finite group and returns the order of this element, and we will not consider the complexity of this algorithm. As was shown by Sutherland [5], the complexity of finding  $\lambda(R)$  is the same as finding the order of an arbitrary element. I refer the reader to his paper for this synopsis.

Now,  $\lambda(R)$  is the largest divisor of  $|G|$  we can find using the orders of elements. More importantly,  $\phi(R)$  is a multiple of  $\lambda(R)$ , that is  $\exists k \in \mathbb{N}$  such that  $k \cdot \lambda(R) = \phi(R)$ . Thus, if we iterate  $k$  and search the multiples of  $\lambda(R)$ , we will eventually find  $\phi(R)$ , which is much faster than simply searching all values less than  $R$  for  $\phi(R)$ . Ideally, we would like  $k = 1$ , but this situation is impossible. By the Fundamental Theorem of Finitely Generated Abelian Groups,  $\mathbb{Z}_R \cong \mathbb{Z}_p \times \mathbb{Z}_q$ . This then implies that  $\mathbb{Z}_R^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^* \cong \mathbb{Z}_{p-1} \times \mathbb{Z}_{q-1}$  (Shoup [4]). But  $2 \mid \gcd(p-1, q-1)$ , and hence  $\mathbb{Z}_{p-1} \times \mathbb{Z}_{q-1}$  is not cyclic. If  $k = 1$ , then  $\mathbb{Z}_R^*$  would be cyclic.  $\Rightarrow \Leftarrow$ . So, our best value of  $k$  would be  $k = 2$ . Unfortunately,  $k$  can be very big—big enough to make an exhaustive search of the set  $\{r \in \mathbb{N} \mid \lambda(R) \leq r \cdot \lambda(R) \leq \phi(R)\}$  unreasonable. We will address this problem later.

### 3 An Algorithm to Compute $\phi(R)$

We now discuss how to find  $\phi(R)$  given  $\lambda(R)$ .

We start out with a system of equations, namely:  $R = pq$  and  $\phi(R) = (p-1)(q-1)$ . When we solve for  $p$ , we have

$$p = \frac{(\phi(R) - 1 - R) - \sqrt{(\phi(R) - 1 - R)^2 - 4R}}{2}.$$

We next define a function

$$p(x) = \frac{(x - 1 - R) - \sqrt{(x - 1 - R)^2 - 4R}}{2}.$$

From here, we will evaluate  $p(k \cdot \lambda(R))$ . Now, if  $p(k \cdot \lambda(R)) \in \mathbb{N}$ , then  $k \cdot \lambda(R) = \phi(R)$ . This is how we find  $\phi(R)$ .

We now need to address the problem for big  $k$ . Obviously, we do not have to try every natural number for  $k$ . For example, since  $\phi(R) < R$ , there is no need to check  $p(k \cdot \lambda(R))$  if  $k \cdot \lambda(R) \geq R$ . We are now interested in placing reasonable bounds on  $\phi(R)$ . Remember that for a semiprime  $R$ ,

$$\begin{aligned} \phi(R) &= (p-1)(q-1) \\ &= pq - p - q + 1 \\ &= R - p - q + 1 \\ (*) &= R - p - \frac{R}{p} + 1 \\ &= \frac{pR}{p} - \frac{R}{p} - p + 1 \\ (**) &= \frac{(p-1)R}{p} - (p-1) \end{aligned}$$

We will first nail down a pretty good upper bound to  $\phi(R)$ . Consider equation (\*) from above and rewrite it as a function  $f : \mathbb{R} \rightarrow \mathbb{R}$  with  $f(x) = R - x - R/x + 1$ . We see that this function is a rotated parabola and  $f'(x) = R/x^2 - 1$ . Hence, the maximal value of  $f(x)$  is when  $x = \sqrt{R}$ . Then we have that

$$\begin{aligned}\phi(R) &\leq R - \sqrt{R} - \frac{R}{\sqrt{R}} + 1 \\ &= R - \sqrt{R} - \sqrt{R} + 1 \\ &= R - 2\sqrt{R} + 1\end{aligned}$$

We will denote this upper bound as  $U(R)$ .

The lower bound is a little more tricky. Consider equation (\*\*):  $\phi(R) = \frac{(p-1)}{p}R - (p-1)$ . If we choose a prime number  $p_0 \leq p$ , then  $\phi(R) \geq \frac{(p_0-1)}{p_0}R - (p_0-1)$ . Certainly,  $2 \leq p$ , and thus  $\phi(R) \geq \frac{1}{2}R - 1$ . Denote this lower bound as  $L(R)$ . Now,  $L(R)$  is not a very tight lower bound and for a large enough  $R$ , the difference  $U(R) - L(R)$  can still be quite large.  $p_0$  simply needs to be less than  $p$  and  $q$ . We define  $L(R, p_0) = \frac{(p_0-1)}{p_0}R - (p_0-1)$ . Then  $\phi(R) \geq L(R, p_0)$  if  $p_0 \leq p, q$ . So, if Eve had some knowledge on the size of  $p$  and  $q$ , she could choose a better  $p_0$  than 2. For example, if a RSA key generating program constructs prime numbers of at least 50 digits each, then Eve could choose  $p_0$  to be  $10^{49}$  or the next prime number after  $10^{49}$ . Note that  $p_0$  need not be prime. Since  $\mathbb{Z}_R^*$  is not cyclic,  $\lambda(R) < \phi(R)$  and  $2 \mid \frac{\phi(R)}{\lambda(R)}$ . Thus,  $\lambda(R) \leq \frac{1}{2}\phi(R)$ . Then  $\lambda(R) < L(R, p_0)$  if  $p_0 > 2$ .

Hence, we know that  $L(R, p_0) \leq \phi(R) \leq U(R)$ . Also,  $k \cdot \lambda(R) = \phi(R)$ . Also, since  $p$  and  $q$  are both odd primes,  $p-1$  and  $q-1$  are even numbers, and hence  $4 \mid \phi(R)$ . Define  $\Lambda(R) = \text{lcm}(\lambda(R), 4)$ . Therefore, we have our primary result.

**Theorem 1** : If  $p_0 \leq p, q$ , then  $\phi(R) \in \{x \in [L(R, p_0), U(R)] \mid x = k \cdot \Lambda(R), k \in \mathbb{N}\}$ .

The algorithm is summarized as following:

1. Compute  $\lambda(R)$ .
2. Compute  $\Lambda(R)$ .
3. Compute  $U(R)$ .
4. Compute the largest  $k$  such that  $k \cdot \Lambda(R)$  is less than  $U(R)$ .
5. Evaluate  $p(k \cdot \Lambda(R))$ . If  $p(k \cdot \Lambda(R)) \in \mathbb{N}$ , then return  $k \cdot \Lambda(R)$ .
6. Else set  $k := k - 1$  and repeat line 5.

## 4 Conclusions

Please note that the algorithm does not depend on  $L(R, p_0)$ . The lower bound is needed to determine an upper bound on the set mentioned in Theorem 1. The efficiency of this algorithm depends entirely on line 1, which will be discussed later, and the size of the set in Theorem 1.

Let  $\mathcal{S} = \{x \in [L(R, p_0), U(R)] \mid x = k \cdot \Lambda(R), k \in \mathbb{N}\}$ . By its definition, we see that  $|\mathcal{S}| \leq \lceil \frac{U(R) - L(R, p_0)}{\Lambda(R)} \rceil$ . By my investigation,  $\mathcal{S}$  appears to be small on the average. Using the computer algebra program Maple, I was able to gather data to support this conjecture. In the Maple code, I would generate two random primes  $p$  and  $q$ , calculate their product  $R$ , calculate  $\phi(R)$ , calculate  $\lambda(R)$ , and calculate  $\lceil \frac{U(R) - L(R, p_0)}{\Lambda(R)} \rceil$ . The code would then repeat this process for a predetermined number of rounds. After this, it would calculate the median length of  $R$  and the median size of  $\lceil \frac{U(R) - L(R, p_0)}{\Lambda(R)} \rceil$ . Once in a while  $\lceil \frac{U(R) - L(R, p_0)}{\Lambda(R)} \rceil$  could be very large, much larger than the rest of the sample data. This means that the mean average does not accurately display the typical size of  $\mathcal{S}$ .

I gathered samples from moduli  $R$  as small as 24 digits long and sample moduli as large as 778 digits long. No matter the length of  $R$ , the median upper bound on  $|\mathcal{S}|$  would always calculate to be 2. Because  $\lceil \frac{U(R) - L(R, p_0)}{\Lambda(R)} \rceil$  would average to be 2 and this upper bound was defined by the ceiling function, it is reasonable to assume that for majority of the samples  $|\mathcal{S}| = 1$ . Then in this case, by Theorem 1,  $\mathcal{S} = \{\phi(R)\}$ . This is our main conclusion from this project; that if  $p_0$  is chosen well,  $\mathcal{S} = \{\phi(R)\}$ . There was, though, a significant amount of sample moduli  $R$  which generated extremely large  $\mathcal{S}$ . But throughout all my code, I set  $p_0 = 1001$ . For moduli of at least 700 digits, this is clearly not a good choice for a lower bound on  $p, q$ . If Alice chose a prime this small, nearly all factoring algorithm could factor Alice's modulus. Hence, for larger  $R$ , we can reasonably use a much larger  $p_0$ . This would dramatically shrink the size of  $\mathcal{S}$  and make many more cases computationally possible. If I had more time to investigate, I would be curious to know, if given a large sample of moduli  $R$ , how small can  $p_0$  be with respect to the average length of digits of  $R$  and still have that 68%, 95%, 99.7% of the sample satisfy the inequality  $\lceil \frac{U(R) - L(R, p_0)}{\Lambda(R)} \rceil \leq 2$ ?

There is another idea I had to deal with the problem of  $|\mathcal{S}|$  being too large. If  $\lambda(R)$  is very small, then it means that the  $\gcd(p - 1, q - 1)$  is very large. Now,  $\lambda(R) = \text{lcm}(p - 1, q - 1)$ . So the smallest  $\lambda(R)$  can be is  $\lambda(R) = q - 1$ . If this is the case, then  $\lambda(R) + 1 = q$  and we have found a factorization of  $R$ . Now,  $\lambda(R)$  is a multiple of  $q - 1$ , and hence if  $\lambda(R)$  is small compared to  $\phi(R)$ , then  $q - 1$  is large compared to  $\lambda(R)$ . Hence, we may factor  $\lambda(R)$  and test its divisors  $d$  to see if  $d = q - 1$ . Unless  $p$  and  $q$  are both safe primes, that is  $p - 1$  and  $q - 1$  have large prime divisors, this procedure may be more reasonable to do. That is, the total number of divisors of  $\lambda(R)$  may be fewer than  $|\mathcal{S}|$ .

The main flaw in the algorithm is line 1. Unfortunately, there is no fast order computation algorithm for an arbitrary group. In fact, for a generic group, factoring is faster than order computation [5]. But there is always hope for an alternative method to emerge. Though no quick order computation algorithm is yet known for generic groups, are there special classes of groups with quick order computation? Could such an algorithm be written for a class of groups for which RSA is popularly implemented, e.g. the modular ring  $\mathbb{Z}_R$  or elliptic curve groups? (Note for elliptic curve groups we would be interested in finding  $\psi(R) = (p + 1)(q + 1)$  instead of  $\phi(R)$ ). Could there be an alternative method of evaluating  $\lambda(R)$  other than order computation? Could a Legendre-like covert channel be implemented which leaked information about orders of elements or  $\lambda(R)$ ?

Unfortunately, the complexity of the algorithm presented in this paper depends in the average case entirely on order computation. At the time of writing this paper, order computation is still impractical. My hope is that this paper may give future researcher ideas or guidance on how the exponent of a group may be used in Cryptology.

## References

- [1] Devore, Jay L. *Probability and Statistics for Engineering and the Sciences* Sixth Edition. Thomson Learning, Inc. 2004.
- [2] Fraleigh, John B. *A First Course in Abstract Algebra* Seventh Edition. Pearson Education, Inc. 2003.
- [3] Rotman, Joseph J. *Advanced Modern Algebra*. Pearson Education, Inc. 2002.
- [4] Shoup, Victor. *A Computational Introduction to Number Theory and Algebra*, Cambridge University Press, 2005.
- [5] Sutherland, Andrew V. *Order Computation in Generic Groups*. Massachusetts Institute of Technology. 2007.