On El-Gamal and Fermat's Primality Test

IRVAN JAHJA / 13509099
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509099@std.stei.itb.ac.id

May 6, 2012

Abstract

We discuss the feasibility of using the simple Fermat's Primality Test to generate prime required as a key to the public key El-Gamal cryptosystem. This option is appealing due to the simplicity of Fermat's Primality Test, but is marred with the existance of Carmichael Numbers.

1 Introduction

1.1 Background

El-Gamal is one of the most used public key cryptosystem. This accounts to its simplicity of implementation as well for being one of the earliest relatively secure public key cryptosystem to be published. For instance, GNU Privacy Guard software and recent versions of PGP are major user of this system.

One of the major challenges in this algorithm is the generation of sufficiently large prime, used to generate the Cyclic Group G required in the entire computation of the algorithm. One of the most popular method is the Rabin-Miller Primality testing, but it suffers from the lack of ability to detect Carmichael Numbers.

There has been numerous studies on Carmichael Number, proving many of its properties [1] [8] [16] [10] [7] [4]. The study of El-gamal itself is equally numerous [22] [19], beginning with its classic paper [6].

1.2 Roadmap

The paper will begin with some background theories on fermat primality testing as well as the El-Gamal Cryptosystem itself. Then, the paper will discuss how El-Gamal reacts with non prime p, and then discuss the case with Carmichael Numbers. The paper then closes with a possible way to take advantage of the behavior of El-Gamal with composite p.

Throughout this paper, pseudocodes in Python-like languages will be presented as to make things less abstract.

2 Background Theory

2.1 Fermat Primality Testing

Fermat Primality Testing is one of the fastest primality testing algorithm. Interested readers are referred to the book by Cormen et. al [20], and we will only give a brief explanation in this section.

The Fermat's Primality Testing is based on the Fermat's Little Theorem:

If p is prime and $1 \leq p < p$, then $a^{p-1} \equiv 1 \pmod{p}$.

Which was given by Pierre D. Fermat and whose formal proof were only given 40 years after it was published.

To test whether n is prime, the Fermat's Primality Testing repeatedly pick a and tests the following equivalence:

$$a^{n-1} \equiv 1 \pmod{n}.$$

If an a is found for which the equation above fails to hold, then n is declared composite and a is said to be its witness.

The Fermat Little Theorem, however, suffers from the numbers known as Carmichael Numbers, which are composite numbers for which the equation $a^{n-1} \equiv 1 \pmod{n}$ holds for all a.

2.2 Carmichael Number

The properties of Carmichael Numbers have been well-studied. There are an infinite number of Carmichael Numbers [2]. Furthermore, from [16], Carmichael Numbers n has the following properties:

- n is square-free
- n has at least three prime factors
- $p|n$ implies $p-1|n-1$ and $p < n^{0.5}$

Although there are an infinite number of Carmichael Numbers, it is not known whether there also exists an infinite amount of Carmichael Numbers with exactly three factors. For instance, up to 10^18, there are 35585 Carmichael Numbers with exactly 3 prime divisors [15]. For comparison, there are 24739954287740860 primes [5].

There are numerous number of Carmichael Numbers $C(X)$, with its asymptotic growth being at least $X^{0.332}$ [9]. An upperbound on the number of Carmichael Number has also been established, but is rather difficult to phrase concisely [7]. This numerous number of Carmichael Numbers inhibit the usefulness of the Fermat Primality Testing to test in general whether a number is prime.

In general, there is no known easy way to distinguish a Carmichael number with a prime number except by testing its primality using some other methods [13].

2.3 Other Primality Tests

PRIMES, the problem of deciding whether a number N is prime, has been proven to be in P [1], giving an $O(\log^1 2N)$ algorithm. Lenstra and Hendrick proceeds to improve its complexity to $O(\log^6 N)$ unconditionally [11]. These discoveries have been made recently and until such it was widely believed that PRIMES is not in P.

Rabin Miller, a probability primalistic test, achieves respectable accuracy using $\log^4 N$ complexity, and is widely used [17] [14]. A deterministic variant of this algorithm is also present and its currently tightest complexity is due to [3]. It is worth noting that this complexity is not strictly polynomial in $\log N$, unlike the one due presented in [1].

2.4 El-Gamal

El-Gamal [6] is an asymmetric public key cryptosystem based on the difficulty of finding discrete logarithms [12]. Discrete Logarithm is stated formally as follows.

Given a, b, and p, find x such that

$$a^x \equiv b \pmod{p}.$$

El-Gamal is widely used. For instance, GNU Privacy Guard software and recent versions of PGP are major user of this system.

As with many public key cryptosystems, the El-Gamal algorithm consists of three distinct steps:

- Key Generation - Generates both public and private key from a given prime p
- Encryption - The process to encrypt a message using the public key in such way that only the holder of the private key is able to decipher the message.
- Decryption - The process of inverting the encrypted message to obtain the original message.

Their respective pseudocodes are presented in the accompanying python-like pseudocodes.

It is worth noting that it is possible to create a reversed variant of El-Gamal, (signature scheme instead of encrypting message) – that is, to create a message that can only be deciphered by the public key and cannot be constructed without creating the private key. This is known as the El-Gamal Signature Scheme.

2.5 Inverse

Finding inverse quickly is important for the El-Gamal cryptosystem, in the decryption phase. Algorithms to find inverses quickly are well-known - Fermat little theorem provides one such alternative for prime p. Recall that Fermat Little theorem states for a prime p and any integer $1 \leq a < p$:

$a^{p-1} \equiv 1 \pmod{p}$

Hence,

$a^{p-2} * a \equiv 1 \pmod{p}$

Therefore, the inverse of $a \bmod p$ is a^{p-2}, which can be computed efficiently using modular exponentiation.

This equation is, however, only valid for prime p. The following theorem generalizes inverses for non primes p:

Theorem 1. *If a is relatively prime to p, then there exists an integer b such that $a * b \equiv 1$ (mod p).*

```
# Produces the private and public keys
def generate_key(p)
  g = random number between 0 and p-1
  x = random number between 0 and p-1
  h = g**x
  public key = (p, g, h)
  private key = (x)
```

Figure 1: Key Generation

```
# Encrypts a message m that can only be read by private key owner
# Assumes that m < p
def encrypt(p, g, h, m)
  y = random number between 0 and q-1
  c1 = g**y
  s = h**y
  c2 = m * s
  return (c1, c2)
```

Figure 2: Encryption

```
# Decrypts c1 and c2 into m
def encrypt(p, g, h, x, c1, c2)
  s = c1**x
  m = c2 * inverse(s)
```

Figure 3: Encryption

To proof this, we will show the famous Bezout's Identity. For a given a and b, form all values $ax + by$, and pick the smallest positive number d amongst all of them. Now, the remainder of dividing either a or b by d is also of the form $ax' + by'$, since we assumed $d = ax + by$. However, since this remainder must be strictly smaller than d, it follows from the fact that d is the smallest positive $ax + by$ that d is zero, so that d divides both a and b.

If c is another divisors of both a and b, c also divides $ax+by = d$. This implies that d is the greatest common divisor of a and b.

Hence, if a is relatively prime to p, there exists x and y such that $ax + py = 1$. Hence, $ax \equiv 1 \pmod{p}$, and so x is the inverse of a, completing the proof.

This theorem is founded for a long time, for instance in [21].

The extended euclid algorithm is one of the possible implementations to compute the inverses of numbers efficiently. Interested readers are referred to the excellent book due to Cormen et. al [20].

3 A Simple Carmichael Detection

We first notice that since a Carmichael number has at least three distinct divisors

Theorem 2. *There exists at least one divisor p of a Carmichael number N such that $p \leq N^{1/3}$.*

Proof: If this were not true, then all divisors of N are greater than $N^{1/3}$. Since a Carmichael Number has at least three divisors, $N = \prod D_i \geq D_0 * D_1 * D_2 \geq N^{1/3*3}x > N$, which is a contradiction.

Hence, we obtain a simple way to detect whether N is carmichael. It is sufficient to test that amongst the first $N^{1/3}$ numbers, there are no number which divides N.

4 El-Gamal and Non-Primality

The only dependency of the El-Gamal algorithm with the primality of p is in the process of inverting s. Indeed, this is the only operations amongst the other operations (exponentiation, multiplication, etc.) that is non- trivial.

Recall that $s = h^y = g^{xy}$. Hence, for s to be relatively prime to p, it is sufficient to let g be relatively prime to p. Hence, our algorithm proposes that the initial g should be made relatively prime to p.

With this change, what are the consequences? We claim that the algorithm will work as intended. Indeed at no other stage of the algorithm the fact that p is not prime makes difference. Hence, the algorithm is indeed stay correct.

Our worries is then whether or not the problem of discrete logarithm can be efficiently solved for Carmichael Numbers. Citing [23], the problem of discrete logartihm over composite number is still difficult as long as the factorization of the composite number remains unknown.

Recalling that Carmichael numbers must have at least three distinct prime divisors, the problem of factorizing it becomes significantly easier. Indeed, the above discussion on Carmichael detecting already yields an $O(N^{1/3})$ algorithm, which may get even weaker since there are no strict upper bound on the number of divisors of a Carmichael Number.

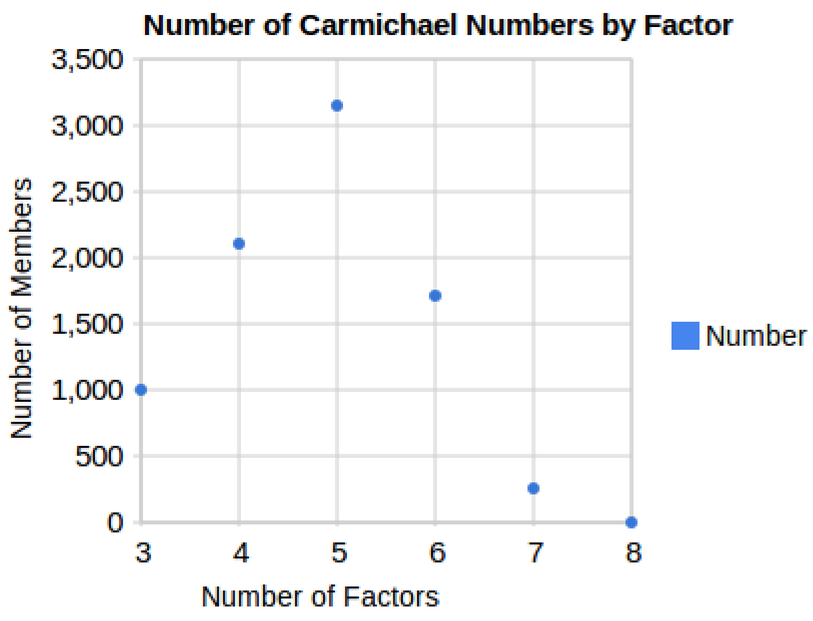
We demonstrated this fact by performing experiment by listing all Carmichael numbers below 10^{12}, show in the table below.

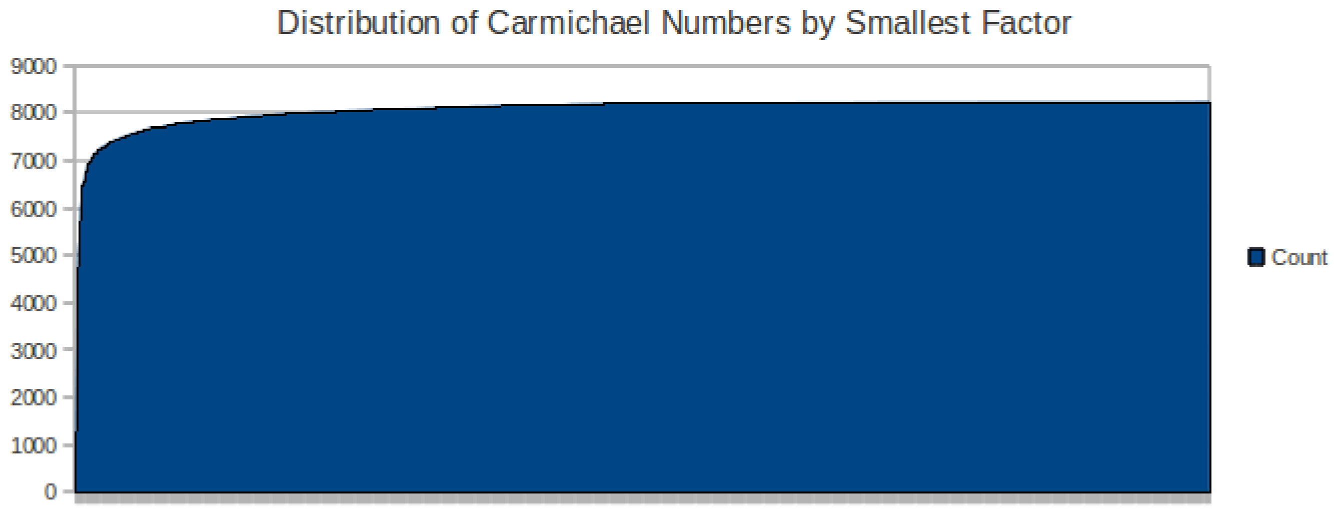
Factors	Count	Percentage
3	1000	12.134%
4	2102	25.507%
5	3156	38.296%
6	1714	20.798%
7	262	3.179%
8	7	0.085%
All	8241	100.000%

From the first 8247 Carmichael numbers, we observed that all of them has a factor less than or equal to 6917, which is less than $10^{12*1/3}$, consistent with our analysis. However, the distribution of these numbers show that there are extremely few numbers amongst the Carmichael Numbers whose smallest factor is of size $O(10^{12*1/3})$, as shown in the accompanying figure. In particular, we observed that more than 85% of those carmichael numbers have a smallest factor less than 100. We conjecture than this trend will continue for larger Carmichael Numbers, hence factoring a Carmichael Number in most cases should proof to be easy.

Hence, we advise that it is not pointless to double check the numbers passing the Fermat Primality Test with slower primality checking methods, such as the venerated Rabin-Miller method [17] [14]. We have to reiterate that the number of Carmichael Number is significant that checks like this is of importance if security is to be guaranteed. However, since the number of

Number of Carmichael Numbers by Factor





Carmichael Numbers in the face of Prime Numbers is very small, this double check is in average will be executed at most once, and very rarely twice, and so retains the efficiency of the Fermat's Primality Check.

To conclude, from our discussions so far, the correctness of El-Gamal remains unaffected by the non-primality of p. However, from the viewpoint of security as described in [23], a particularly bad choice of non-prime p may lead to security breach.

5 Alternative p

Our discussion has shed light on the possibility of another choice for p. Since El-Gamal do not necessarily require p to be prime, we may as well choose a p to be a product of two large primes, similar to how we choose the key required for the RSA algorithm [18]. It is arguably easier to find two primes of smaller value. Furthermore, we can apply this recursively to the two primes to obtain a recursive algorithm that may stop its iteration at any depth – corresponding to the degree of security we desire.

The algorithm is presented as a very simple and succinct pseudocode, which is a function on the number of bits of p that we desire.

Note that since factorization methods are known to run in approximately the smallest divisor of N, the security of this algorithm depends greatly on threshold. In particular, it is not difficult that it is possible to crack this algorithm in $O(2^{threshold})$, so extra caution should be exercised in picking the value of $threshold$ as to avoid this issue.

6 Conclusion

Although the majority of El-Gamal implementation relies on prime p, this is not necessary for the correctness of the algorithm to continue to hold. Hence, it is absolutely okay, from the viewpoint of correctness, to use Fermat's Primality Test to generate the large prime required, as in the event it becomes a Carmichael Number, the algorithm's correctness stay intact. However, this may not be the case from the viewpoint of security, as the discrete logarithm problem boils down to integer factorization in the event that the moduli is non-prime [23].

Furthermore, this property may be further exploited by using a divide and conquer algorithm to construct a p from several smaller primes that are still big and make it difficult to factorize.

7 Appendix

7.1 Declaration of Non Plagiarism

I hereby confirm that this paper is the product of my own work and is not an excerpt and/or translation of the work of other entities.

```
Bandung, May 6th 2012




Irvan Jahja
13509099
```

References

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. 2004.

[2] W. R. Alford, Andrew Granville, and Carl Pomerance. There are infinitely many Carmichael numbers. *Ann. of Math. (2)*, 139(3):703–722, 1994.

[3] E. Bach. Explicit bounds for primality testing and related problems. 55(191):355–380, 1990.

[4] R. Balasubramanian and S. V. Nagaraj. Density of carmichael numbers with three prime factors. *Math. Comput.*, 66(220):1705–1708, October 1997.

[5] M. Deleglise and J. Rivat. Computing &pgr;(x): the meissel, lehmer, lagarias, miller, odlyzko method. *Math. Comput.*, 65(213):235–245, January 1996.

[6] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[7] Paul Erdos. On pseudoprimes and carmichael numbers. *Pub. Math. Deprecen*, 4:20–206, 1956.

```
def compute_p(number_of_bits)
  if number_of_bits <= threshold:
    return generate_prime(number_of_bits)
  else:
    return compute_p(number_of_bits / 2) * compute_p(number_of_bits / 2)
```

Figure 4: Encryption

[8] Andrew Granville and Carl Pomerance. Two contradictory conjectures concerning carmichael numbers. *Math. Comput.*, 71(238):883–908, April 2002.

[9] Glyn Harman. On the number of carmichael numbers up to x, 2005.

[10] Everett W. Howe. Higher-order carmichael numbers. *MATH. COMP*, 69:1719, 1998.

[11] Hendrik W. Lenstra, Jr. Primality testing with gaussian periods. In *Proceedings of the 22nd Conference Kanpur on Foundations of Software Technology and Theoretical Computer Science*, FST TCS '02, pages 1–, London, UK, UK, 2002. Springer-Verlag.

[12] Kevin S McCurley. The discrete logarithm problem, 1990.

[13] Zachary S. McGregor-Dorsey. Methods of primality testing.

[14] Gary L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13(3):300–317, December 1976. invited publication.

[15] R.G.E. Pinch. On using carmichael numbers for public key encryption systems, 1997.

[16] Richard G. E. Pinch. The carmichael numbers up to 10^16. *Math. Comp*, 61:381–391, 1998.

[17] M O Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980.

[18] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[19] Claus Peter Schnorr and Markus Jakobsson. Security of signed elgamal encryption (extended abstract), 2000.

[20] Ronald L. Rivest Clifford Stein Thomas H. Cormen, Charles E. Leiserson. Introduction to algorithms (second edition), 2001.

[21] J.-P. Tignol. *Galois' theory of algebraic equations*. John Wiley & Sons, Inc., New York, NY, USA, 1987.

[22] J. Wu and D. R. Stinson. On the security of the elgamal encryption scheme and damg ˚ ard's variant.

[23] Masao Kasahara Yasuyuki Murakami. A discrete logarithm problem over composite modulus, 1993.