Analyzing and Optimizing the Combined Primality test with GCD Operation on Smart Mobile Devices

Hosung Jo[1] and Heejin Park[2] * **

[1] Department of Electronics and Computer Engineering,
Hanyang University, Seoul, Korea
ustog@hanyang.ac.kr

[2] Division of Computer Science and Engineering,
Hanyang University, Seoul, Korea
hjpark@hanyang.ac.kr

Abstract. Fast generation of a large prime is important to enhance security. However, primality test of prime generation takes long time. Therefore, primality tests are combined to speed up. A combination of trial division and Miller-Rabin test (TD combined test hereafter) is the most popular combination. GCD operation and Miller-Rabin test combination (GCD combined test hereafter) is also popular and easily alternative to TD combined test. Although the performance of TD combined test has been analyzed and optimized, GCD combined test has scarcely been analyzed. In this paper, we present a probabilistic analysis on the total running time of GCD combined test for smart mobile devices. Our analysis and corresponding experiment result are very accurate with only 1-2% error. In addition, we propose a method to determine the optimal number of small primes used in GCD primality test to speed up.

Keywords: Prime generation, Primality test, RSA, Public-key cryptosystems

1 Introduction

Cryptosystems such as RSA [1] and ElGmal [2] and signature schemes such as DSS [3] use large primes in order to provide strong security. However, generating large primes requires heavy computation, so efficiently generating large primes is important. Especially, a fast generating of large primes on smart mobile devices is more important because the smart mobile devices do not have enough computation power as much as a general processor.

Prime generation algorithm consists of a random number generation and a primality test. Random number generation makes a n-bit odd positive integer

* This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2011-0026496)

** Corresponding author, E-mail: hjpark@hanyang.ac.kr randomly and primality test examines whether the generated odd random number is a prime or not. Since the primality test is much-more time-consuming than the random number generation, developing fast primality test is important in order to develop an efficient prime generation algorithm.

The primality test is divided into deterministic primality tests and probabilistic primality tests. A deterministic primality test certifies that a random number is a prime with probability 1 such as Trial division [4], GCD operation [5], elliptic curve analogue [6], and Maurer's algorithm [7]. A probabilistic primality test certifies a random number is a prime with very high probability such as Fermat test [5], Miller-Rabin test [8, 9], and Solovay-Strassen test [10]. Practically primality tests are combined to speed up. A popular combination is a combination of trial division and Miller-Rabin test (or Fermat test). The performance of this combination is analyzed by Maurer et al. [7]. They proposed a probabilistic analysis of the expected running time for the combination. In addition, they showed how to compute the optimal number of primes used in the trial division. The primality test combining GCD operation and Miller-Rabin test is an alternative to TD combined test. However, the performance of this combination has scarcely been analyzed. In this paper, we introduce a probabilistic analysis of the expected running time for this combination and we show how to compute the optimal number of primes used in GCD operation.

This paper is organized as follows. Section 2 introduces some preliminaries briefly. Section 3 describes our contribution: the analysis of GCD combined test and how to compute the optimal number of primes used in the GCD operation. Finally, we conclude in Section 4.

2 Preliminaries

Trial division divides a random number r by all primes less than or equal to $\sqrt{r}$. GCD operation computes the greatest common divisor of two integers a and b. If the GCD of r and the product of all primes less than or equal to $\sqrt{r}$ is 1, r is a prime. However, when r is large, the running time of both primality tests are very slow. so they are combined with Miller-Rabin test (or Fermat test). Prime number generation algorithm that uses TD combined test consists of random number generation, trial division, and probabilistic primality test and the following is the pseudocode of it.

TD Combined Algorithm (n, k)

```
1. Random Number Generation
     - Generate a $n$-bit odd random number $r$
2. Trial division on $r$ with $k$ small primes
     - Divides $r$ by $k$ small primes
     - If $r$ is divided by any prime, go to Step 1.
3. Miller-Rabin test on $r$
     - Perform Miller-Rabin Test on $r$
     - If $r$ passes, $r$ is a prime. Otherwise, go to Step 1.
```

Maurer [7] introduced a probabilistic analysis of the expected running time of TD combined algorithm. Let N_{Trial} be the number of trials until finding a prime. Let T_{RND}, T_{TD}, and T_{MR} be the average running times of random number generation, trial division, and Miller-Rabin test respectively. Then, the total running time,T_{Total}, is as follows.

$$T_{Total} = N_{Trial} \cdot (T_{RND} + T_{TD} + T_{MR}) \tag{1}$$

In here, if the length of r is n-bit, $N_{Trial} = \frac{n \cdot \ln 2}{2} \approx 0.347n$. Let T_{div} and T_{mr} be the running times of one division and one MR test respectively. k is the number of small primes used in trial division and p_i is the ith odd prime number, $p_1 < p_2 \ldots < p_k$. Then, T_{TD} and T_{MR} are as follows.

$$T_{TD} = T_{div} \cdot (1 + \sum_{1 \leq j \leq k} \prod_{1 \leq i \leq j} (1 - \frac{1}{p_i})) \tag{2}$$

$$T_{MR} = T_{mr} \cdot (\prod_{1 \leq i \leq k} (1 - \frac{1}{p_i})) \tag{3}$$

Therefore, T_{Total} with k small primes is as follows.

$$T_{Total} = 0.347n \cdot (T_{RND} + T_{div} \cdot (1 + \sum_{1 \leq j \leq k} \prod_{1 \leq i \leq j} (1 - \frac{1}{p_i})) + T_{mr} \cdot (\prod_{1 \leq i \leq k} (1 - \frac{1}{p_i})) \tag{4}$$

The running time T_{Total} depends on k and the optimal value k_{opt} which makes the running time fastest is as follows [7].

$$g_{opt} = \frac{T_{mr}}{T_{div}} \tag{5}$$

3 GCD Combined Test

3.1 GCD Algorithms

The running time of GCD combined algorithm can be modeled as follows where T_{gcd} is the running time of GCD operation.

$$T_{Total} = 0.347n \cdot (T_{RND} + T_{gcd} + T_{mr} \prod_{1 \leq i \leq k} (1 - \frac{1}{p_i})) \tag{6}$$

We first introduce two famous GCD algorithms: Euclid's GCD algorithm and Binary GCD algorithm by J. Stein.

```
$Euclid(a, b)$

1. If $b$ is 0, return $a$
2. Otherwise, return $Euclid(b, a \bmod b)$
```

```
$BinaryGCD(a, b)$

1. If $a > b$, and both are odd, $gcd(a, b) = gcd(\frac{(a-b)}{2}, b)$
2. If $a$ is odd and $b$ is even, $gcd(a, b) = gcd(a, \frac{b}{2})$
3. If $a$ is even and $b$ is odd, $gcd(a, b) = gcd(\frac{a}{2}, b)$
4. If both a and b are even, $gcd(a, b) = 2gcd(\frac{a}{2}, \frac{b}{2})$
5. If $a < b$, $swap(a, b)$
6. return BinaryGCD($a$, $b$)
```

In practice, two algorithms are combined to use. Euclid's algorithm is used until two integers have similar bit-lengths. Then Binary GCD algorithm is used.

```
$CombinedGCD(a, b)$

1. If $b = 0$, return $a$
2. If the difference of bit length of ($a$ and $b$) $> 2$, $CombinedGCD(b, a \bmod b)$
3. If the difference of bit length of ($a$ and $b$) $\leq 2$, $BinaryGCD(a, b)$
```

The running time of $CombinedGCD(a, b)$ is a sum of the running time for Euclid's GCD in step 2 and for Binary GCD in step 3. Let T_{Euc} and T_{bgcd} be the running time for Euclid's GCD and Binary GCD, respectively. Then,

$$T_{gcd} = T_{Euc} + T_{bgcd} \tag{7}$$

In $CombinedGCD(a, b)$, Euclid's GCD does 2 divisions on average, so the running time is $O((1 + \log q)\log b)$, which is the running time for dividing a by b where q is the quotient $\lfloor a/b \rfloor$. The running time of $BinaryGCD(a, b)$ is proportional to the bit length of the bigger of a and b, i.e., $O(\log a)$.

3.2 Analysis of GCD Combined Test

In GCD combined algorithm, we compute GCD of r and Π_k. While the bit length of r is fixed to n, the size of Π_k is varying as k increases. We divide the analysis of T_{gcd} into two cases when $r > \Pi_k$ and when $r < \Pi_k$.

When $r > \Pi_k$, $T_{gcd} = T_{Euc}(r, \Pi_k) + T_{bgcd}(r \bmod \Pi_k, \Pi_k)$. Since the running time for $Euc(a, b)$ is $O((1 + \log q)\log b)$, $T_{Euc}(r, \Pi_k)$ is asymptotically as follows.

$$O((1 + \log(\frac{r}{\Pi_k}))\log \Pi_k) = O(\log \Pi_k + \log \Pi_k \log r - (\log \Pi_k)^2) \tag{8}$$

Therefore, the running time of $T_{Euc}(r, \Pi_k)$ can be represented as a quadratic function of $\log \Pi_k$ like this: $T_{Euc}(r, \Pi_k) = x(\log \Pi_k)^2 + y(\log \Pi_k) + z, x < 0$. The running time $T_{bgcd}(r \bmod \Pi_k, \Pi_k)$ is $O(\log \Pi_k)$ because ($r \bmod \Pi_k < \Pi_k$), and therefore it is a linear function of $\log \Pi_k$.

$$T_{bgcd} = u \cdot (\log \Pi_k) + v \tag{9}$$

When $r < \Pi_k$, T_{gcd} is $T_{gcd}(k) = T_{Euc}(\Pi_k, r) + T_{bgcd}(r, \Pi_k \bmod r)$. Since $r < \Pi_k$, the time complexity of $Euc(\Pi_k, r)$ is like this.

$$O((1+\log(\frac{\Pi_k}{r}))\log r) = O(\log r + \log r \log \Pi_k - (\log r)^2) \tag{10}$$

Because $\log r$ is a constant, $Euclid(\Pi_k, r)$ is a linear function of $\log \Pi_k$ like this: $T_{Euclid} = u'(\log \Pi_k)+v'$. Since $(r > \Pi_k \bmod r)$, $T_{bgcd}(r, \Pi_k \bmod r) = O(r)$, which is constant time. Overall, we get the following lemma.

Theorem 1 *The running time of CombinedGCD(r, Π_k) is as follows.*

$$T_{gcd}(k) = \begin{cases} u(\log \Pi_k)^2 + v(\log \Pi_k) + w & (r > \Pi_k) \\ u'(\log \Pi_k) + v' & (r < \Pi_k) \end{cases}$$

for some u, v, w, u' and v' where Π_k is $p_1 p_2 ... p_k$ and p_i is a prime.

3.3 Comparison of the Expected and Measured Running Times

To examine the accuracy of our expected running time, we implemented GCD combined algorithm and measured the running time for generating 512-bit primes in 100,000 times because 512 bit primes are widely used on smart mobile devices.

When $r > \Pi_k$, $T_{gcd}(k)$ is computed by Theorem 1. However, $T_{Euc}(r, \Pi_k)$ is very small enough to neglect. The following table shows that the running time of $T_{Euc}(r, \Pi_k)$ and $T_{bgcd}(r \bmod \Pi_k, \Pi_k)$ when r is 256, 512 or 1024 bit.

	256 bit	512 bit	1,024 bit
$T_{Euc}(\mu s)$	0.96 - 1.46	1.14 - 4.28	1.58 - 10.8
$T_{bgcd}(\mu s)$	36	102	314

Thus, when $r > \Pi_k$, $T_{bgcd}(r \bmod \Pi_k, \Pi_k)$ approximates to a linear function of $\log \Pi_k$ like this: $T_{gcd}(k) = u(\log \Pi_k) + v$. Finally, we can expect the running time of combined GCD by computing coefficients (u, v) and (u', v'). We compute the regression using 4 sample points for each case. The regression analysis shows a similar result with experimental results.

$$T_{gcd}(k) = \begin{cases} 201.3916(\log \prod_{i=1}^{k} p_i) - 3,993.05 & (r > \Pi_k) \\ 17.17009(\log \prod_{i=1}^{k} p_i) - 96,861.38 & (r < \Pi_k) \end{cases} \tag{11}$$

The expected total running time can be computed by measuring T_{RND} and T_{mr}. The following table compares the expected running times and measured running times when 512 bit prime is generated.

$r(bit)$	$\Pi_k(bit)$	k (primes)	Expected (ms)	Measured (ms)	Error(%)
512	127	25	214	215	0.4
512	256	43	192	193	0.5
512	509	74	183	185	1.0
512	1028	131	169	171	1.1
512	2046	232	160	162	1.2
512	4092	471	157	159	1.2

The error rate is from 0.3% to 1.2%. The result shows that a probabilistic analysis and running time expectation for total running time of GCD combined algorithm is quite good.

3.4 Computing k_{opt} for optimal running time

Because the running time is decreasing for a while and increasing later, an optimal point exists. We will find a integer k_{opt} that satisfies the following equation.

$$T_{Total}(k+1) - T_{Total}(k) \approx 0 \tag{12}$$

From the equation (16), we will get the followings to determine k_{opt}.

Theorem 2 *The GCD combined algorithm is fastest when k satisfies the below equation.*

$$\frac{a}{T_{mr}} \simeq \frac{1}{p_{k+1}\log(p_{k+1})}\left(\prod_{i=1}^{k}\left(1-\frac{1}{p_i}\right)\right)$$

4 Conclusion

In this paper, we proposed a probabilistic analysis model for expected running time of GCD combined test. Furthermore, we proposed a method to determine k_{opt} in order to make the running time fastest. Finally, we showed that our analysis is quite accurate by comparing the expected running time and the measured running time.

References

1. Rivest, R.L., Shamir, A. and Adleman, L.: A method for obtaining digital signatures an public-key cryptosystems, *Communications of the ACM*, 21 (2) 120-126 (1978).
2. ElGmal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory*, 31 (4) 469-472 (1985).
3. National Institute for Standards and Technology, Digital Signature Standard(DSS), *Fedral Register*, 169 (1991).
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C.: *Introduction to Algorithms, 2nd ed, MIT press*, (1991).
5. Menezes, A.J., van Oorschot, P.C. and Vanstone, S.A.: Handbook of Applied Cryptography, *CRC Press*, (1997).
6. Atkin, A.O.L. and Morain, F.: Elliptic curves and primality proving, *Mathematics of Computation*, 61 29-63 (1993).
7. Maurer, U.M.: Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters, *Journal of Cryptology*, 8 (3) 123-155 (1995).
8. Miller, G.L.: Riemann's Hypothesis and Tests for Primality, *Journal of Computer Systems Science*, (1976).
9. Rabin, M.O.: Probabilistic Algorithm for Primality Testing, *Journal of Number Theory*, 12 128-138, (1980).
10. Solovay, R. and Strassen V.: A fast Monte-Carlo test for primality, *SIAM Journal on Computing*, 6 84-85 (1977).