MVFactor: A Method to Decrease Processing Time for Factorization Algorithm

Kritsanapong Somsuk
Department of Computer Science, Faculty of Science
Khon Kaen University, KKU
Khon Kaen, Thailand
kritsanapong@kkumail.com

Sumonta Kasemvilas
Department of Computer Science, Faculty of Science
Khon Kaen University, KKU
Khon Kaen, Thailand
sumkas@kku.ac.th

***Abstract*—RSA is the most well-known public key cryptosystem. The security of RSA is based on the difficulty of factoring the modulus which is the product of the two large prime numbers. If the modulus is factorized, RSA is broken. VFactor is a new factorization algorithm which can factorize the modulus faster than Fermat's Factorization algorithm (FFM) and Trial Division algorithm (TDM). The runtime of VFactor depends on the difference between the two large prime numbers but not on the size of the modulus. In this paper, we propose implementing Modified VFactor (MVFactor) modified from VFactor in order to decrease the processing time in computation. Key to MVFactor is the decrease of computation time of computing the product of two integers. Experiments have shown that when using MVFactor, the computation speed increases in comparison to VFactor. However, if the difference between the two prime numbers is large, then the computation time of MVFactor is more decreased than the computation time of VFactor.**

Keywords—RSA; VFactor; Integer Factorization; Computation Time

I. INTRODUCTION

RSA cryptosystem [1] proposed in 1978 is the most well-known public key cryptosystem. It is widely used to secure the information in the insecure channel [10]. RSA algorithm uses a pair of keys to encrypt and decrypt a message: One key is used to encrypt a message, called a public key, *e*; and the other key is the inverse of the public key modulo the Euler' s totient function, $\Phi(n)$, used to decrypt the encrypted message, called a private key, *d*. Generally, *d* is kept secret for only parties but *e* is disclosed for intended recipient. The principle of RSA is that: it is very easy to find the two large prime numbers and the product of these prime numbers, but it is more computationally difficult to factorize the modulus, *n*, which is the product of two prime numbers. The security of RSA is based on the difficulty of integer factorization. Assume that we can factorize $n = p*q$, *p* and *q* are prime numbers, after that $\Phi(n)$ is computed: $\Phi(n) = (p - 1)(q - 1)$ and we can recover d: $e*d \bmod \Phi(n) = 1$. Thus RSA is broken.

Many factorization algorithms were introduced, such as Trial Division algorithm (TDM) [9], Fermat's Factorization algorithm (FFM) [3], Quadratic Sieve (QS) [4, 8], Pollard's p-1 algorithm [3, 9] and Monte Carlo Factorization algorithm [9, 11]. Some factorization algorithms cannot finish all trivial and nontrivial values of *n*, such as Pollard's p-1 algorithm [3].

Latter, Prashant Sharma et al [2] proposed the new method to factorize *n*, called VFactor, in 2012. It is an algorithm which can finish all trivial and nontrivial values of *n* [3]. The computation time of VFactor is based on the difference between the two large prime factors but not on the size of *n*. The speed of VFactor is faster than FFM and TDM but in some cases, the computation speed of TDM is faster than VFactor only when *n* is less than 10^{12} [2]. However, if the difference between the two prime numbers is little, VFactor can factorize more quickly. The idea of VFactor is that we have to find the product of two integers: $m = x*y$, *y* is a maximum odd number which is less than or equal to *floor(sqrt(n))* and $x = y + 2$, if *m* is equal to *n*. We can conclude that *x* and *y* are factors of *n*. Otherwise, if *m* is less than *n*, we continue computing $x = x + 2$ and $m = x*y$ until *m*, which is equal to *n*, is found. However, if *m* is more than *n*, we continue computing $y = y - 2$ and $m = x*y$ until *m*, which is equal to *n*, is found.

In this paper, a Modified VFactor (MVFactor) modified from VFactor is proposed to implement. This method is not computed: $m = x*y$ if the least significant digit of *x* or *y* is *5* because we can strongly confirm that it is not a prime and *m* is not equal to *n*, except *y* is equal to *5*. Experiments have shown that MVFactor can factorize faster than VFactor. In addition, if the difference between the two prime numbers is large, then the computation time of MVFactor is more decreased than the computation time of VFactor.

II. RSA SCHEME

RSA is a type of public key cryptosystem that is based on the difficulty of integer factorization. It is used for encryption/decryption message and digital signature. The process of RSA is divided into three parts as follows:

1. Key generation process: Generate keys that are a public key and a private key for encryption and decryption, respectively.
   1.1 Generate the two large primes, *p* and *q*, randomly which should have the same size.

   1.2 Compute $n = p*q$.
   1.3 Compute $\Phi(n) = (p-1)(q-1)$.
   1.4 Select an integer e randomly, $1<e<\Phi(n)$, such that $gcd(e, \Phi(n)) = 1$.
   1.5 Compute an integer d, $1<d<\Phi(n)$, such that $e*d \bmod \Phi(n) = 1$ using Extended Euclid Algorithm [5].
2. Encryption process: Encrypt the plaintext using a public key before sending to the insecure channel.
   2.1 $c = m^e \bmod n$, m is a plaintext and c is a ciphertext sending in the insecure channel.
3. Decryption process: Decrypt the ciphertext using a private key to recover the plaintext.
   3.1 $m = c^d \bmod n$.

III. VFactor

The security of RSA cryptosystem is based on integer factorization. To break RSA cryptosystem, a cryptanalyst has to factorize *n*. Currenly, there are many factorization algorithms used to break RSA such as TDM and FFM. VFactor is a new, fast factorization algorithm which can finish all trivial and nontrivial values of *n*. The speed of VFactor is faster than TDM and FFM [2]. The runtime of this algorithm is based on the difference between two prime factors. The scheme of VFactor is as follow:

Input: the modulus *n*: $n = p*q$

```
Let $root = floor(sqrt(n))$;
If ($root\%2 == 0$)
    $root = root - 1$;
EndIf
$y = root$;
$x = y + 2$;
$m = x * y$;
While($m != n$)
    If($m < n$)
        $x = x + 2$;
    Else
        $y = y - 2$;
    EndIf
    $m = x*y$;
EndWhile
```

Output: *x* and *y*, where *x* and *y* are prime numbers which are the factors of *n*

Fig. 1. VFactor algorithm

IV. Our Proposed Method

For VFactor, if the least significant digit of *x* or *y* is *5*, we can strongly confirm that it is not a prime, because of divisibility by 5 of a decimal number. This is not true in the case where the value of *y* is *5* that is a prime. So, for the value of *y* is more than *5*, if the least significant digit of *x* or *y* is *5*, then we do not compute the product of *x* and *y*, because we can strongly confirm that it is not equal to *n*. In the case that the least significant digit of *x* is *5*, we do not compute: $m = x*y$ and continue computing $x = x + 2$ and $m = x*y$ until a value of *m*, which is equal to a value of *n*, is found. Nevertheless, in the case that the least significant digit of *y* is *5*, we do not compute: $m = x*y$ and continue computing $y = y - 2$ and $m = x*y$ until a value of *m*, which is equal to a value of *n*, is found. The algorithm of MVFactor is as follow:

Input: the modulus *n*: $n = p*q$

```
If($n \% 5 == 0$) // Check, 5 is a prime factor of n
    $y = 5$;
    $x = n / y$;
Else        // MVFactor starts here
    Let $root = floor(sqrt(n))$;
    If($root \% 2 == 0$)
        $root = root - 1$;
    EndIf
    $y = root$;
    $x = y + 2$;
    $m = x * y$;
    $xt = x \% 10$; // xt is the least significant digit of x
    $yt = y \% 10$; // yt is the least significant digit of y
    While($m != n$)
        If($m < n$)
            If($xt == 3$) /* We do not use $xt = 5$ because x is
not a prime */
                $xt = xt + 4$;
                $x = x + 4$;
            Else
                $xt = xt + 2$;
                $x = x + 2$;
            EndIf
/* If the least significant digit of x is 9, then the next
should be 1 after computing $x = x + 2$ */
            If($xt == 9$)
                $xt = 1$;
            EndIf
        Else
            If($yt == 7$){ /* We do not use $yt = 5$ because y
is not a prime */
                $yt = yt - 4$;
                $y = y - 4$;
            Else
                $yt = yt - 2$;
                $y = y - 2$;
            EndIf
/* If the least significant digit of y is 1, then the next
should be 9 after computing $y = y - 2$ */
            If($yt == 1$)
                $yt = 9$;
            EndIf
        EndIf
        $m = x * y$;
    EndWhile
EndIf
```

Output: *x* and *y*, where *x* and *y* are prime numbers which are the factors of *n*

Fig. 2. MVFactor algorithm

Example 1: VFactor VS. MVFactor, given $n = 72851 = 277 * 263$.

Algorithm: VFactor

$root = floor(sqrt(n)) = 269$
So, $y = 269, x = 271$
$271 * 269 = 72899$, that is more than n, compute $y = y - 2 = 267$
$271 * 267 = 72357$, that is less than n, compute $x = x+2 = 273$
$273 * 267 = 72891$, that is more than n, compute $y = y - 2 = 265$
$273 * 265 = 72345$, that is less than n, compute $x = x+2 = 275$
$275 * 265 = 72875$, that is more than n, compute $y = y - 2 = 263$
$275 * 263 = 72325$, that is less than n, compute $x = x+2 = 277$
$277 * 263 = 72851$, here is equal to n, thus the factors of n are 277 and 263

According to Example 1 of VFactor, the iterations of computing the product of two integers are 7.

Algorithm: MVFactor

$root = 269$
So, $y = 269, x = 271$
$271 * 269 = 72899$, that is more than n, compute $y = y - 2 = 267$
$271 * 267 = 72357$, that is less than n, compute $x = x+2 = 273$
$273 * 267 = 72891$, that is more than n, compute $y = y - 2 = 265$. Because the least significant digit of y is 5, we continue computing $y = y - 2 = 263$.
$273 * 263 = 71799$, that is less than n, compute $x = x+2 = 275$. Because the least significant digit of x is 5, we continue computing $x = x + 2 = 277$.
$277 * 263 = 72851$, here is equal to n, thus the factors of n are 277 and 263

According to Example 1 of MVFactor, the iterations of computing the product of two integers are only 5 which is less than VFactor.

Example 2: VFactor VS. MVFactor, given $n = 69133 = 269 * 257$.

Algorithm: VFactor

$root = floor(sqrt(n)) = 262$, that is an even number, thus we have to compute $root = 262 – 1 = 261$
So, $y = 261, x = 263$
$263 * 261 = 68643$, that is less than n, compute $x = x+2 = 265$
$265 * 261 = 69165$, that is more than n, compute $y = y - 2 = 259$
$265 * 259 = 68635$, that is less than n, compute $x = x+2 = 267$
$267 * 259 = 69153$, that is more than n, compute $y = y - 2 = 257$
$267 * 257 = 68619$, that is less than n, compute $x = x+2 = 269$
$269 * 257 = 69133$, here is equal to n, thus the factors of n are 269 and 257

According to Example 2 of VFactor, the iterations of computing the product of two integers are 6.

Algorithm: MVFactor

$root = 261$
So, $y = 261, x = 263$
$263 * 261 = 68643$, that is less than n, compute $x = x+2 = 265$. Because the least significant digit of x is 5, we continue computing $x = x + 2 = 267$.
$267 * 261 = 69687$, that is more than n, compute $y = y - 2 = 259$
$267 * 259 = 69153$, that is more than n, compute $y = y - 2 = 257$
$267 * 257 = 68619$, that is less than n, compute $x = x+2 = 269$
$269 * 257 = 69133$, here is equal to n, so the factors of n are 269 and 257

According to Example 2 of MVFactor, the iterations of computing the product of two integers are only 5 which is less than VFactor.

From two examples above, we conclude that MVFactor can decrease iterations to compute the product of two integers in comparison to VFactor. That is, the computation time of MVFactor is decreased.

V. Results

Experiments have shown the comparison between VFactor and MVFactor. We choose BigInteger Class in Java to implement two algorithms, because BigInteger is a data type which is unlimited in size. In order to control the same settings, all experiments were conducted on 2.53 GHz an Intel® Core i3 with 4 GB memory.

TABLE I. COMPARISON BETWEEN VFACTOR AND MVFACTOR

n	**V - Factor**		**MV – Factor**	
	Com-puting m = x*y (Iterations)	***Computation Time (s)***	***Com-puting m = x*y (Iterations)***	***Computation Time (s) (Speed Up (%))***
25426686165792287 = 137462537 * 184971751	23754607	3.73	19003687	*3.15 (15.54%)*
780115113938271913 = 840219481 * 928465873	44123196	6.81	35298558	*5.76 (15.42%)*
5889267638331574111 = 2160246761 * 2726201351	282977295	43.03	226381837	*36.05 (16.22%)*
163216432381925762729 = 16955697601 * 9626052329	3664822636	804.69	2931858109	*653.14 (18.83%)*

According to Table I, the iterations of computing the product of two integers of MVFactor are less than VFactor. That is, the computation time of MVFactor is decreased. For example, in Table I, $n = 163216432381925762729 = 16955697601 * 9626052329$, the iterations of computing the product of two integers of VFactor are 3664822636 but MVFactor are only 2931858109. That is, if we use MVFactor, the iterations of computing the product of two integers are decreased to 732964527 (3664822636 – 2931858109). In this case, MVFactor is faster than VFactor which takes 804.69 seconds about 18.83%.

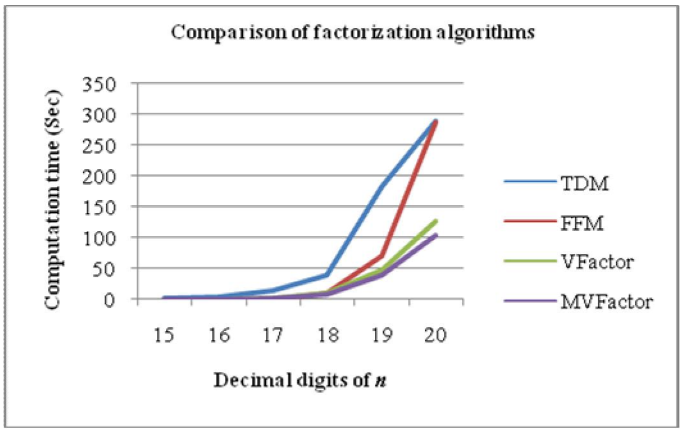

Fig. 3. Comparison of factorization algorithms

Fig. 3 shows a plot of comparison of factorization algorithms between computation time and decimal digits of n during 15 to 20 digits by using TDM, FFM, VFactor and MVFactor. In Fig. 3, we choose prime factors of n with the same size for experiments and we can see that MVFactor use less time than VFactor, FFM and TDM to factorize n by several examples.

VI. Conclusion

In this paper, MVFactor modified from VFactor is proposed. This method computes the product of two integers only if the least significant digit of two integers is not equal to *5*. We can strongly confirm that the product of two integers is not equal to n when the least significant digit of one out of two integers is *5*, because it is not a prime. Experiments have shown that MVFactor can find the factors of n much faster than VFactor. However, if q is far from p, the computation time of MVFactor is more decreased than the computation time of VFactor. The future work, we wish to improve MVFactor to factorize the bigger size of n in the proper time. For example, we may reduce more iterations to compute the product of two integers. The multiplication is not computed when we know that, one out of two multipliers is not a prime number.

References

[1] R.L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public key cryptosystems", Communications of ACM. 21 164-158 (1978).

[2] Prashant Sharma, Amit Kumar Gupta, Ashish Vijay, "Modified Integer Factorization Algorithm using V-Factor Method", Advanced Computing & Communication Technologies. (2012) 423 – 425.

[3] B. R Ambedkar, Ashwani Gupta, Pratiksha Gautam, Sarabjeet S.Bedi, "An Efficient Method to Factorize the RSA Public Key Encryption ", Communication Systems and Network Technologies. (2011) 108 – 111.

[4] Qingfeng Huang, Zhi-Tang Li, Yejing Zhang, Chuiwei Lu, "A Modified Non-Sieving Quadratic Sieve For Factoring Simple Blur Integers", Multimedia and Ubiquitous Engineering. (2007) 729 – 732.

[5] Jianqin Zhou, Jun Hu, Ping Chen, "Extended Euclid algorithm and its application in RSA", Information Science and Engineering. (2010) 2079 – 2081.

[6] A. K. Lenstra and H. W. L. Jr., editors. "The development of the number field sieve", volume 1554 of LNCS. Springer,1993.

[7] Bell, E. T. "The Prince of Amateurs: Fermat". New York: Simon and Schuster. (1986) 56-72.

[8] Garrett Paul B.,Making, "Breaking codes:An Introduction to Cryptology", Prentice Hall, 2000.

[9] David Bishop, "Introduction to Cryptography with java Applets", Jones and Bartlett Publisher, 2003.

[10] R.S. Vignesh, S. Sudharssum, K.J.J. Kumar, "Limitations of Quantum & the Versatility of Classical Cryptography: A Comparative Study", Environmental and Computer Science, 2009. ICECS '09. (2009) 333-337.

[11] J. Pollard, "Monte Carlo methods for index computation (mod p)", Math. Comp., Vol. 32, pp.918-924, 1978.

[12] Ren-Junn Hwang, Feng-Fu Su, "An Efficient Decryption Method for RSA Cryptosystem", Advanced Information Networking and Applications, 2005. AINA 2005. (2005) 585-590.

[13] W. Diffie and M. E. Hellman, "New Directions in Cryptography", In IEEE Transactions on Information Theory, volume IT–22, no. 6, pages 644–654, November 1976.