

Fast Decryption Methods for the RSA Cryptosystem

T. L. Grobler and W. T. Penzhorn

Abstract—In this paper we present three new decryption algorithms for the RSA cryptosystem. These algorithms increase the speed of RSA decryption by exploiting the Chinese Remainder Theorem (CRT). Two of the three algorithms use special prime generation to construct a special decryption key. The third method, Wiener, works by choosing special CRT decryption exponents to construct the decryption key.

Index Terms—RSA, Wiener, Fast Decryption Algorithms, Chinese Remainder Theorem, Cryptosystem, Hamming weight.

I. INTRODUCTION

The RSA algorithm is arguably the most widely used public-key algorithm. [6]. Areas of application include browser security, the secure exchange of session keys, internet banking, and credit and debit card payments. Many applications involve the use of smart cards, for example, for the secure storage of secret keys. RSA is also used by certificate authorities.

The decryption speed of the RSA algorithm is substantially slower than the encryption speed, due to the much longer decryption exponent. However, the more computationally demanding decryption is often performed by computationally-constrained devices with limited resources, such as a smart card. Hence, the need has been recognized to optimize the decryption algorithm, so as to increase the decryption speed.

In this paper we will give a brief review of the RSA algorithm in section II. This will be followed by section III wherein a brief description of the CRT and its use by Quiscater et al [4] to increase the speed of RSA decryption is given.

Then the three new algorithms will be introduced in sections IV, V and VII. A low Hamming weight prime difference generator needed by the algorithms in sections IV and V is discussed in section VI. In section VIII these new algorithms are compared to each other theoretically. The paper is concluded by a results section which will evaluate each algorithms performance. These new algorithms improve on the technique developed by Quiscater et al [4].

T. L. Grobler is with the University of Pretoria, RSA (012) 4303777; fax: (012) 4303777; cell: 0720204679 e-mail: s22003569@tuks.co.za

W. T. Penzhorn is a professor at the University of Pretoria, RSA (012) 4262953; cell: 0834172979; e-mail: walter.penzhorn@up.ac.za

II. THE RSA ALGORITHM

The RSA algorithm operates briefly as follows [6].

- Generate two large prime numbers p and q .
- Calculate the modulus:

$$n = p \times q \quad (1)$$

- Calculate:

$$\phi(n) = (p-1)(q-1) \quad (2)$$

where $\phi(n)$ denotes the Euler Totient function.

- Select an encryption exponent e such that the

$$\gcd(\phi(n), e) = 1, 1 < e < \phi(n) \quad (3)$$

Proposed values for e are $2^{16}+1$ and 3.

- The key pair (e, n) is known as the public key.

- Calculate the decryption exponent d :

$$de \equiv 1 \pmod{\phi(n)} \quad (4)$$

d is thus the multiplicative inverse of e modulo $\phi(n)$

The Extended Euclidean algorithm can be used to calculate multiplicative inverses [3].

- The key pair (d, n) is known as the private key.

- Encrypt the plain text M as follows:

$$C \equiv M^e \pmod{n} \quad (5)$$

where C is the encrypted message and $M < n$.

- Decrypt the ciphertext C as follows:

$$M \equiv C^d \pmod{n} \quad (6)$$

User A encrypts the message with the public key (e, n) of user B using (5). User B is now the only one who can decrypt this message using his own private key (d, n) and (6).

III. THE CHINESE REMAINDER THEOREM

The Chinese Remainder Theorem (CRT) for the case of only two moduli is described below.

Let m_1, m_2 be pairwise relatively prime positive integers.

Then the given system of congruences

$$x \equiv a_1 \pmod{m_1} \quad (7)$$

$$x \equiv a_2 \pmod{m_2}$$

has a unique solution modulo $M = m_1 m_2$ [7].

The solution x to the simultaneous congruences in (7) can be calculated using (8) as discussed in [7].

$$x \equiv (a_1 m_2 c_1 + a_2 m_1 c_2) \pmod{M} \quad (8)$$

where c_1 is the multiplicative inverse of $m_2 \pmod{m_1}$ and c_2 is the multiplicative inverse of $m_1 \pmod{m_2}$.

Since the two primes p and q are known to the receiver, he can derive the following set of equations.

$$d_p \equiv d \pmod{p-1} \text{ and } d_q \equiv d \pmod{q-1}$$

$$C_p \equiv C \pmod{p} \text{ and } C_q \equiv C \pmod{q} \quad (9)$$

$$M_p \equiv C_p^{d_p} \pmod{p} \text{ and } M_q \equiv C_q^{d_q} \pmod{q}$$

with $p < q$, d is the decryption exponent, C is the cipher text.

Quiscater et al [4] have shown that the plaintext can be efficiently decrypted by (10) if one applies the CRT to (6):

$$\tilde{M} \equiv [((M_q + q - M_p) \cdot A) \pmod{q}] \cdot p + M_p \quad (10)$$

where \tilde{M} is the decrypted message and A is a constant integer such that $0 < A < q-1$ and $A \cdot p \equiv 1 \pmod{q}$. The integer A can be calculated with the Extended Euclidean algorithm [3]. These equations are used to decrypt any RSA cipher text.

The decryption speed is about four times faster due to the fact that the moduli are reduced to half the bit-size of the modulus n . This means that the computations are being done with smaller numbers. The variables d_p and d_q will be used in the remainder of the text and will be referred to as the CRT decryption exponents. Since the primes p and q are only known to the receiver, the CRT decryption algorithm can only be used by the receiver to decrypt a received message.

New algorithms taking advantage of these two new exponents d_p and d_q are discussed in the remainder of the text. If d (decryption key) is chosen so that it remains large but leads to two smaller CRT decryption exponents, this would give an increase in the execution speed of RSA decryption without affecting the security thereof. The chosen d must remain large, because Wiener [8] has shown that choosing d smaller than half the bit size of the modulus can lead to security problems. Choosing d has another negative effect. It slows encryption dramatically, because the encryption exponent can no longer be chosen equal to $2^{16} + 1$. Now the encryption exponent is actually of the same order as the modulus. The decryption algorithms discussed below increase the speed of RSA decryption by choosing d .

IV. FAST DECRYPTOIN ALGORITHM I (FDA I)

Decryption algorithm I utilizes the following steps.

- Generate two primes p and q such that $r = q - p$ and $|r| \geq c$, and $H(r) > h$ (11)

where $|x|$ denotes the bit size of x , $H(x)$ denotes the Hamming weight of x , c and h are specific constants. The suggested values for these constants can be found later on in this section.

- Now calculate the decryption exponent using $d \equiv (kq + r) \pmod{(p-1)(q-1)}$ (12)

where k is an odd positive integer.

- As before calculate the encryption exponent by evaluating (13) by using the Extended Euclidean algorithm [3].

$$e \equiv d^{-1} \pmod{(p-1)(q-1)} \quad (13)$$

if e does not exist go back to the first step.

Using the first two equations from (9) the following can be deduced of d_p and d_q when d is chosen by (12).

$$\begin{aligned} d_p &\equiv d \pmod{p-1} \\ &\equiv kq + r \pmod{p-1} \\ &\equiv k(p+r) + r \pmod{p-1} \\ &\equiv k(r+1) + r \pmod{p-1} \end{aligned} \quad (14)$$

and

$$\begin{aligned} d_q &\equiv d \pmod{q-1} \\ &\equiv kq + r \pmod{q-1} \\ &\equiv k + r \pmod{q-1} \end{aligned} \quad (15)$$

For the special case when $k = 3$

$$\begin{aligned} d_p &\equiv 4r + 3 \pmod{p-1} \\ d_q &\equiv r + 3 \pmod{q-1} \end{aligned} \quad (16)$$

The equations in (16) are used to analyze the theoretical efficiency of decryption algorithm I. The values of c and h need to be determined in order to implement decryption algorithm I. The bit size of the prime difference between p and q is recommended in [1] to be at least 412 bits for a 1024 bit modulus. Using this as guideline appropriate values for c can be calculated using the following equation

$$|r| \geq \frac{412}{1024} \cdot |n| = \frac{103}{256} \cdot |n| \quad (17)$$

where $|n|$ equals the bit size of the modulus, and $|r|$ equals the bit size of the prime difference. Suggested values for h can be calculated by choosing the cryptosystem security level equal to 128 bits. Choosing the security level equal to 128 bits doesn't really weaken RSA, because the key size of AES is also only 128 bits long.

On the other hand choosing the Hamming weight too small reduces the level of security (less possible values to test before finding p and q) due to an available simple factoring attack on the modulus. Consider the following identity

$$\left(\frac{q+p}{2}\right)^2 - pq = \left(\frac{q-p}{2}\right)^2 \quad (18)$$

with $r = q - p$. To solve p and q the following can be done. Substitute $q - p$ with all possible values of r . Then solve the

term $\left(\frac{q+p}{2}\right)^2$. The correct answer is obtained if $\left(\frac{q+p}{2}\right)^2$

is a perfect square. Once the value for $q + p$ and $q - p$ are known p and q are easily determined. Thus the security of RSA relies on the number of possible values of r . The number of possible values for r if $H(r) = h$ can be calculated using (19):

$$\binom{|r|-2}{h-1} > 2^{128} \quad (19)$$

where $\binom{x}{y}$ denotes x combination y or expressed

mathematically as $\binom{x}{y} = \frac{x!}{(x-y)!y!}$. The above equation

estimates the number of different values r can assume when there are a maximum of h ones allowed in r . The bit size is reduced by two to ensure that r is an even positive integer and that the most significant bit is a one (same reason why h is reduced by one). Equation (19) can also be used to calculate the constant h if the bit size of r is known. By using (17) and (19) the suggested values for the constants c

an h can be calculated. The suggested values for c and h for standard RSA modulus sizes are given in table I.

From (16) it is clear that d_p and d_q are smaller than the standard CRT exponents derived by Quisquater. The new CRT exponents are however not a lot smaller. The true reason for the speed increase of decryption algorithm I is the low Hamming weight of the new exponents. This is true, because a lower hamming weight produces fewer multiplications when one uses the standard square-and-multiply exponentiation algorithm [3]. It is important to note from (12) that decryption algorithm I has a decryption exponent with size just larger than half the modulus. This insures that the restriction of Wiener [8] is satisfied but only barely. If one increases k the size of the decryption exponent will become larger with the added cost of increasing the size and hamming weight of the CRT decryption exponents making the algorithm slower.

TABLE I
THE CONSTANTS c AND h FOR STANDARD RSA MODULUS SIZES

Modulus size	c	h
1024	412	24
1536	618	21
2048	824	20

V. FAST DECRYPTION ALGORITHM II (FDA II)

Decryption algorithm II is utilized via the following steps:

- Generate two primes p and q such that

$$r = q - p \text{ and } |r| \geq c, \text{ and } H(r) > h \quad (20)$$

where $|x|$ denotes the bit-size of r , $H(x)$ denotes the Hamming weight of x , and c and h are specific constants. The suggested values for these constants can be found in table I.

- Calculate the decryption exponent using

$$d \equiv (p^2 q) \bmod (p-1)(q-1) \quad (21)$$

- As before calculate the encryption exponent by evaluating (22) using the Extended Euclidean algorithm [3].

$$e \equiv d^{-1} \bmod (p-1)(q-1) \quad (22)$$

Using the first two equations from (9) the following can be deduced of d_p and d_q when d is chosen according to (21).

$$\begin{aligned} d_p &\equiv d \bmod (p-1) \\ &\equiv (p^2 \cdot q) \bmod (p-1) \\ &\equiv q \bmod (p-1) \\ &= (r+p) \bmod (p-1) \\ &\equiv (r+1) \bmod (p-1) \end{aligned} \quad (23)$$

and

$$\begin{aligned} d_q &\equiv d \bmod (q-1) \\ &\equiv (p^2 q) \bmod (q-1) \\ &\equiv p^2 \bmod (q-1) \\ &= (q-r)^2 \bmod (q-1) \\ &\equiv (r-1)^2 \bmod (q-1) \end{aligned} \quad (24)$$

The last equation from (23) as well as the last equation from (24) is used for the theoretical analysis of decryption algorithm II. Decryption algorithm II is implemented using the exact same steps as decryption algorithm I. The only major difference is that a different equation is used to calculate d . Another difference is that FDA II always produces an e and thus there is no need for looping. This is true because $p^2 q$ and $(p-1)(q-1)$ will always be relatively prime. It is important to note from (21) that the decryption exponent will be the same bit size as the modulus and not half the bit size as decryption algorithm I. The values determined for c and h stay the same for both of the decryption algorithms. A study on decryption algorithm I/II were conducted in [2].

VI. LOW HAMMING WEIGHT PRIME DIFFERENCE PRIME GENERATOR

To implement decryption algorithm I/II one requires a low Hamming weight prime difference prime generator. In other words one needs an algorithm that will calculate two primes p and q such that the difference r between them has a low Hamming weight. An efficient algorithm is given below. It generates a prime difference r by spreading a lesser hamming weight than required evenly.

- Generate a prime p .
- Generate the difference r by (later r will become a prime difference).
 1. Set $l = h - m$, where h is the required Hamming weight of the prime difference r , l is a new Hamming weight to enable faster computation of p and q , m ($m \neq 0$) is a positive integer ($m = 6$ is sufficient).
 2. Set $t = \frac{c}{l}$, $r = 0$, $r_0 = 0$, $r_{i-1} = 1$, $i = 1$.
Where $|r| = c$, and r_x is the x^{th} bit position of r .
 3. While ($|r| > i$ and $H(r) < l$)
 - $i = i + R(t)$, where $R(t)$ generates a random positive integer greater or equal to 0 and smaller than t .
 - Set $r_i = 1$.
 - $i = i + (t - R(t))$, the same $R(t)$ as in the first bullet of step 3.
- Calculate $q = p + r$.
- While q is not prime $q = q + 2$.
- Set $r = q - p$. (Now r has become a prime difference)
- If $H(r) \neq h$, go back to step 1 (generate a prime p) else return p and q .
- Outputs two primes p and q with difference r . The prime difference has a size c and Hamming weight h .

VII. WIENER'S DECRYPTION ALGORITHM

The method suggested in [8] is implemented via the following steps.

- Generate two random primes p and q such that $|p| = |q| = \frac{|n|}{2}$. Where n is the modulus. Ensure that the $|\text{lcm}[p-1, q-1]| > 0.5|n|$.
- Generate d_p randomly such that $|d_p| = \frac{|n|}{k}$. Where k is any positive real number such that $k \neq 0$, $k \ll |n|$.
- Generate a random number f such that $|f| < |d_p|$ and $\text{gcd}(p-1, q-1) | f$.
- Calculate d_q with $d_q = d_p + f$.
- Calculate $A = \frac{(p-1)}{\text{gcd}(p-1, q-1)}$, $B = \frac{(d_q - d_p)}{\text{gcd}(p-1, q-1)}$ and $C = \frac{(q-1)}{\text{gcd}(p-1, q-1)}$.
- Now calculate the multiplicative inverse y by using the Extended Euclidean algorithm[3] and $Ay \equiv 1 \pmod{C}$.
- Now calculate the integer D using $D \equiv yB \pmod{C}$ with $D < C$ and $D < yB$.
- Now calculate d using equation 25
$$d \equiv (D(p-1) + d_p) \pmod{\text{lcm}[p-1, q-1]} \quad (25)$$
Where $\text{lcm}[f, g]$ denotes the least common multiple of f and g .
- Generate e by using equation 22 and the Extended Euclidean algorithm[3]. If e does not exist go back to step 2.

The derivation of (25) is given below

Wiener proposed a method of generating d by choosing d_p and d_q small. Thus d_p and d_q are known and d must be determined. The first two equations of (9) can be rewritten as the following system of simultaneous congruences.

$$\begin{aligned} d &\equiv d_p \pmod{(p-1)} \\ d &\equiv d_q \pmod{(q-1)} \end{aligned} \quad (26)$$

With $q > p$ and $d_q > d_p$. The above system can be solved only if $\text{gcd}(p-1, q-1) | \{d_q - d_p\}$.

(26) can not be solved directly using (8) because the relatively prime constraint is not satisfied. Instead (25) was derived using only basic modular arithmetic (modular addition, multiplication etc.). Solving the above system can be accomplished by following the steps below.

- Rewrite the first congruence of (26) into the following form

$$d = (p-1)x + d_p \quad (27)$$

where x is an integer.

- Substituting equation 27 into the second congruence of (26) leads to

$$(p-1)x + d_p \equiv d_q \pmod{(q-1)} \quad (28)$$

- Subtracting d_p on both sides of the above congruence leads to the following Diophantine equation (linear congruence). Solving Diophantine equations is documented in [5].

$$(p-1)x \equiv (d_q - d_p) \pmod{(q-1)} \quad (29)$$

Which can only be solved if $\text{gcd}(p-1, q-1) | \{d_q - d_p\}$.

- Equation (29) can be solved using the following steps

1. Divide (29) by $\text{gcd}(p-1, q-1)$ on both sides of the congruence

$$\frac{(p-1)}{\text{gcd}(p-1, q-1)} x \equiv \frac{(d_q - d_p)}{\text{gcd}(p-1, q-1)} \pmod{\frac{(q-1)}{\text{gcd}(p-1, q-1)}} \quad (30)$$

2. Calculate the multiplicative inverse y by using the Extended Euclidean algorithm

$$Ay \equiv 1 \pmod{C} \quad (31)$$

$$\text{Where } A = \frac{(p-1)}{\text{gcd}(p-1, q-1)} \text{ and}$$

$$C = \frac{(q-1)}{\text{gcd}(p-1, q-1)}$$

3. Multiply y by $B = \frac{(d_q - d_p)}{\text{gcd}(p-1, q-1)}$ to obtain

$$x \equiv yB \pmod{C} \equiv D \pmod{C} \quad (32)$$

where $D < C$ and $D < yB$. Equation (32) can be written as

$$x = vC + D \quad (33)$$

where v is an integer.

- After substituting x into (27) and solving d (27) becomes

$$\begin{aligned} d &= [(p-1)Cv] + [D(p-1) + d_p] \\ &= \left[\frac{(p-1)(q-1)}{\text{gcd}(p-1, q-1)} v \right] + [D(p-1) + d_p] \end{aligned} \quad (34)$$

$$d \equiv (D(p-1) + d_p) \pmod{\text{lcm}[p-1, q-1]}$$

This concludes the derivation of (25).

The size of d_p and d_q is given by the following equation

$$\frac{|n|}{k} \quad (35)$$

Equation (35) is used for the theoretical analysis of Wiener. This is so, because the CRT exponents will have bit sizes equal to (35) due to the way Wiener constructs the CRT decryption exponents in steps 2,3 and 4 of the algorithm.

There are a few differences between Wiener and the fast decryption algorithms. Some of these differences are mentioned below. Wiener requires no generation of special primes. The other difference is Wiener begins by selecting d_p and d_q small, while the fast decryption algorithms do not. The decryption algorithms rely on the characteristics of the prime difference r to generate CRT decryption exponents that have an extremely low hamming weight. Wiener relies more on choosing the actual size of the CRT decryption exponents directly to increase the speed of RSA decryption. One more thing is that from step 2 and (25) one can see that the decryption exponent will satisfy the restriction in [8].

VIII. THEORETICAL ANALYSIS OF THE CRT AND ITS KEY GENERATORS

As discussed in [3] the general equation in terms of operation cost (small multiplications) for decrypting cipher text is given by

$$\frac{3}{2}(|n|)(|n|^2) = \frac{3}{2}|n|^3 \quad (36)$$

where n represent the modulus of RSA encryption and is equal to pq . The $\frac{3}{2}$ factor is a factor that comes from the standard square-and-multiply algorithm [3]. When employing the CRT the modulus and exponent is reduced by a factor of two. But there are now two of these equations instead of one. When employing the CRT equation 36 becomes

$$2\left(\frac{3}{2}\right)\left(\frac{|n|}{2}\right)\left(\frac{|n|}{2}\right)^2 = \frac{3}{8}|n|^3 \quad (37)$$

Equation (37) is a first-order estimation of the cost of using the CRT. Dividing (36) by (37) gives the theoretical improvement of the CRT over straight-forward RSA decryption. The factor is equal to 4. This means the CRT improves RSA decryption by 75% (maximum). For decryption algorithm I (37) can be written as

$$2(h+c)\left(\frac{|n|}{2}\right)^2 \quad (38)$$

where c and h were defined in (11). The substitution of the exponent can be made because of (16). This expression relates the CRT decryption exponents to the prime difference. Noting that h is small, as shown in Table I, it can be assumed to be zero without loss of generality due to the fact that $h \ll c$. After replacing c with (17), (38) becomes

$$\frac{103}{512}|n|^3 \quad (39)$$

Equation (39) is a reasonable estimate of the computational cost of decryption algorithm I. Because all of the key generation methods only influence the highest order term of the CRT model, (39) can estimate the improvement on standard CRT with reasonable accuracy. Dividing (37) by (39) gives the improvement over standard CRT by decryption algorithm I. The factor equals $1\frac{89}{103}$.

Which means decryption algorithm I improves standard CRT decryption by approximately 46%.

The analysis of decryption algorithm II will be done next. As can be seen from the last equation of (23) and the last equation from (24) only d_p will be of size c and have hamming weight approximately equal to h . From the last equation of (24) it is clear that d_q will be half the modulus and have no special hamming weight characteristics.

These observations make it possible to write (37) as

$$\left(\left(\frac{3}{2}\right)\left(\frac{|n|}{2}\right) + c + h\right)\left(\frac{|n|}{2}\right)^2 \quad (40)$$

Again assuming h equals 0 due to the fact that $h \ll c$ and therefore has little effect on the amount of long multiplications generated and substituting c with (17) leads to

$$\frac{295}{1024}|n|^3 \quad (41)$$

Again dividing (37) by (41) gives the appropriate improvement factor of decryption algorithm II on standard CRT decryption. The factor equals $1\frac{89}{295}$. This implies decryption algorithm II improves standard CRT decryption by about 23%. When the Wiener algorithm is used, the size of the CRT decryption exponents have to be chosen. Choosing it one third of the modulus (37) can be written as

$$2\left(\frac{3}{2}\right)\left(\frac{|n|}{3}\right)\left(\frac{|n|}{2}\right)^2 = \frac{1}{4}|n|^3 \quad (42)$$

Dividing (37) by (42) gives the improvement factor for the Wiener algorithm with $\frac{|n|}{3}$, which is equivalent to 1.5.

This leads to a 33% reduction in the execution time of CRT decryption. When it is one fifth of the modulus then (37) can be written as

$$2\left(\frac{3}{2}\right)\left(\frac{|n|}{5}\right)\left(\frac{|n|}{2}\right)^2 = \frac{3}{20}|n|^3 \quad (43)$$

Dividing (37) by (43) gives the improvement factor of the Wiener algorithm with $\frac{|n|}{5}$. The factor equals 2.5. This leads to a 60% reduction in execution time of CRT decryption.

IX. RESULTS

The results of a simulation are summarized in Table II.

TABLE II
EXECUTION TIMES FOR DECRYPTION ALGORITHMS

Algorithm	Security level			
	512 bits	1024 bits	1535 bits	2048 bits
Standard RSA	4.08ms	25.6ms	77.2ms	166.88ms
RSA with CRT	1.88ms	8.78ms	23.14ms	49.06ms
Wiener $\frac{ n }{3}$	1.54ms	5.66ms	15.92ms	32.82ms
Wiener $\frac{ n }{5}$	0.64ms	3.14ms	9.4ms	20ms
FDA I	Not applicable	4.68ms	12.82ms	25.94ms
FDA II	Not applicable	6.58ms	17.88ms	36.86ms

The algorithms were implemented on an Intel Celeron 2.4GHz with 352MB RAM, running the Microsoft Windows XP Home edition version 2002. The application used the MIRACL library and was compiled by C++Builder6.

The following represents table II graphically.

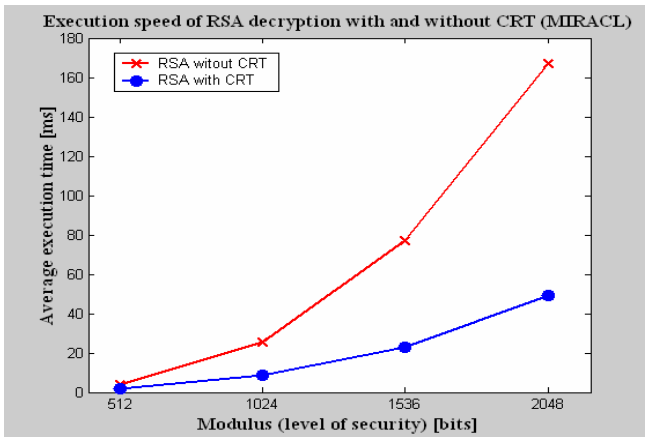


Fig.1. The execution time of unoptimised stock RSA decryption and the execution time of RSA implemented through the CRT.

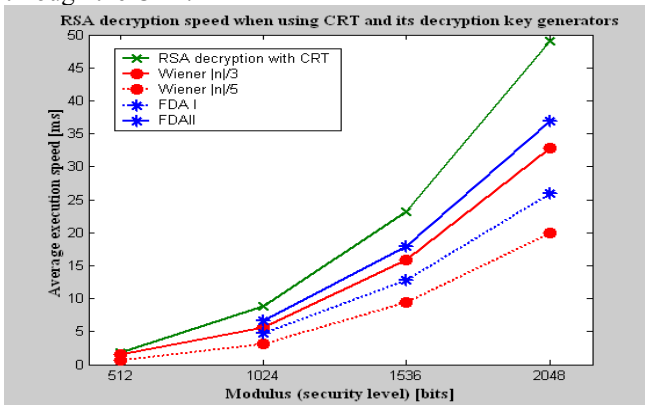


Fig.2. The execution times of RSA decryption with the aid of the CRT and its decryption key generators.

Take note that plain RSA and CRT decryption (Fig 1) is drawn separately from CRT decryption and the algorithms presented in this paper (Fig 2), because the improvement on plain RSA by the algorithms represented in this paper are so high that if shown in one graph it would make the graph unclear.

In summary it can be seen from Table II that the improvement due to each decryption key generator on CRT decryption are as indicated in Table III. Table III does not represent the improvement on unoptimised stock RSA but shows the improvement of the algorithms in this paper on CRT decryption. This was done, because it has become standard to implement RSA by using CRT decryption, so any algorithm you design must improve on CRT decryption to be of any use in the industry. Also to calculate the theoretical and practical improvements on plain RSA can easily be done by using section VIII and Table II.

TABLE III

PRACTICALLY OBTAINABLE SPEED INCREASE ON CRT DECRYPTION

X. CONCLUSION

The Wiener algorithm can increase the speed of CRT decryption by 60%. However, it has to be remembered that the security of Wiener remains an open question [8]. There has been no study up to date that has either proved or disproved Wieners algorithm. If it is compromised it would be useless in the industry. The reason being attackers would

Algorithm m	Wiener $\lfloor \frac{n}{3} \rfloor$	Wiener $\lfloor \frac{n}{5} \rfloor$	FDA I	FDA II
%	33%	60%	46%	23%

easily be able to break RSA in that case. No weakness could be identified by the current study and analysis of this method. So the best method to use until the Wiener algorithm is proven would be decryption algorithm I. This gives a 46% speed increase. Fast decryption algorithm I/II are known to be safe [2]. The Wiener algorithm can also be extended to include low hamming weight if the algorithm of section VII is rewritten. In addition the Wiener algorithm as described in section VII can be made faster if it remains secure for even larger values of k .

ACKNOWLEDGMENT

I would like to thank G. Joseph for helping me with the MIRACL library.

REFERENCES

- [1] ANSI X9.31: Digital signatures using reversible public key cryptography for the financial services industry (rDSA). *American National Standards Institute*, 1998.
- [2] G. Joseph. "Design and Implementation of High-Speed Algorithms for Public Key- Crypto Systems". *Master of Engineering (Electronics) in the faculty of engineering, built environment & information technology at the University of Pretoria*, 2005.
- [3] A.J. Menenzes, P.C. van Oorschot and S.A. Vanstone. "Handbook of applied cryptography". London: CRC Press, 1997.
- [4] J.-J. Quisquater and C. Couvreur. "Fast decipherment algorithm for RSA public-key cryptosystems", *Electronic Letters*, no. 18, pp. 905-907, 1982.
- [5] H. Riesel. "Prime Numbers and Computer Methods for Factorization". Birkhäuser Boston, Inc, 1985.
- [6] R.L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *CACM*, vol. 21, pp. 120–126, 1978.
- [7] H.K. Rosen. "Elementary Number Theory and its Applications". AT&T Bell Laboratories, 1993.
- [8] M. J. Wiener. "Cryptanalysis of short RSA secret exponents". *IEEE Trans. Information Theory*, IT-36 (3), pp. 553-558, 1990.

Trienko L. Grobler was born on the 22 February 1983 in Pretoria, South-Africa.

He matriculated at High School S.P.C.R. Swart in 2001 and completed his BEng(Computer Engineering) degree at the University of Pretoria in 2005.