Asian Journal of Information Technology 6 (5): 567-575, 2007

A New Public-Key Cryptosystem Based on Mandelbrot and Julia Fractal Sets

Mohammad Ahmad Alia and Azman Bin Samsudin
School of Computer Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia

Abstract: In this study, we are proposing a new cryptographic public-key encryption protocol based on Mandelbrot and Julia fractal sets. The fractal based public-key encryption protocol is possible because of the strong connection between the Mandelbrot and Julia fractal sets. In the proposed protocol, Mandelbrot fractal function takes the chosen private key as the input parameter and generates the corresponding public-key. Julia fractal function is used to cipher the plaintext with receiver's public key and decipher the ciphertext based on the receiver's private key. The proposed protocol is designed to be resistant against attacks, utilize small key size and perform comparatively faster than the existing RSA public-key encryption protocol. The proposed fractal public-key encryption protocol is, therefore, an attractive alternative to the traditional number theory based public-key encryption protocol.

Key words: Fractals cryptography, public-key encryption protocol, Mandelbrot fractal set and Julia fractal set

INTRODUCTION

Encryption based cryptography algorithms are divided into two categories: Secret-key (symmetric) algorithm and public-key (asymmetric) algorithm. In general, a security protocol uses public-key cryptosystem to exchange the secret key between communicating nodes and then uses secret-key cryptosystems with the agreed secret key as the password to ensure confidentiality on the data transferred (Branovic *et al*., 2003; Menezes *et al*., 1996). Symmetric algorithms are used to encrypt and decrypt messages by using the same secret key. Public-key encryption algorithms work in a different way. In these algorithms, there is a pair of keys, one of which is known to the public and used to encrypt the plaintext. The corresponding ciphertext is then send to the receiver who owns the corresponding decryption key, also known as the private key.

RSA (Rivest *et al*., 1978) was the first public-key encryption protocol published based on the public-key characteristic proposed earlier by Diffie-Hellman (1976). RSA public-key encryption is based on the difficulty of factoring a number, resulted from a multiplication of two prime numbers (Stallings, 2003). This study proposed a new fractal (based on Mandelbrot and Julia fractal sets) public-key encryption protocol as a secured method to encrypt and decrypt information. The working of the proposed protocol depends on the strong connection between the Mandelbrot and Julia sets in their special functions, Mandelfn and Juliafn functions (Giffin, 2006) which generate the corresponding private and the public keys.

FRACTALS

A complex number consists of a real and imaginary number components (Fig. 1). It contains i, the imaginary unit, where $i^2 = -1$ (Patrzalek, 2006). Every complex numbers therefore can be represented in the form of a+bi, where a and b are real numbers. For example, Fig. 1 shows a point in a complex plane with coordinate 3 on real axis and 2 on the imaginary axis. The sum and product of two complex numbers are formulated as shown by Eq. 1 and 2.

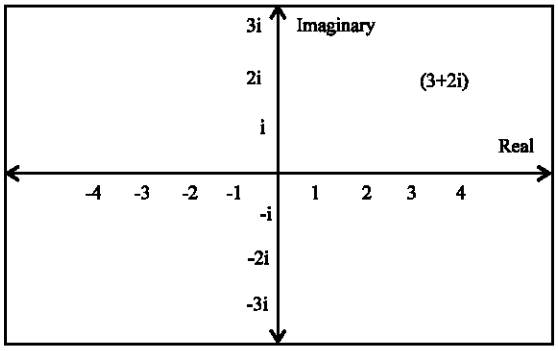


Fig. 1: Complex numbers planes

Corresponding Author: Azman Bin Samstudin, School of Computer Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia

$$(a+bi) + (c+di) = (a+c) + (b+d)i,\ a, b, d, \in Z;\ i^2 = -1 \quad (1)$$

$$(a+bi)\times(c+di) = (ac-bd) + (bc+ad)i,\ a, b, d, \in Z;\ i^2 = -1 \quad (2)$$

One of the interesting applications of the complex plane is fractal. Fractal was mode famous by Mandelbrot. In fact, the word fractal was derived from the Latin word fractus by Benoit Mandelbrot in 1960. As defined by Benoit, fractal is a fragment of geometric shape, created interactively from almost similar but smaller components (some changes in scale). From another perspective, fractals are irregular in shape (Jampala, 1992) and they are not cohering to the typical mathematical dimensions. Fractals can be characterized as having a non-integer dimension and generally they can be classified into two types; fractal curves, which the dimension of the fractal curves fall between the first and second dimensions (1-D and 2-D) and fractal surface, which shapes have a dimension between the second and third dimension (2-D and 3-D). There is another kind of fractals that is called fractal dimensions that can fall between 0.64th to 1.58th dimensions of the non-integer dimension (Mandelbrot, 1982). There are many applications of fractal. One prominent example is the use fractal to create a realistic image of nature such as the image of clouds, snow flakes, fungi and bacteria, mountains, river networks, systems of blood vessels and others (Patrzalek, 2006; Mandelbrot, 1982).

Julia and Mandelbrot sets: Other than imitating the image of nature, fractal geometry has also permeated many area of science, such as astronomy, physics and biological sciences. Fractal geometry has also been classified as one of the most important techniques in computer graphics (Patrzalek, 2006). Julia fractal set (Fig. 2), developed by

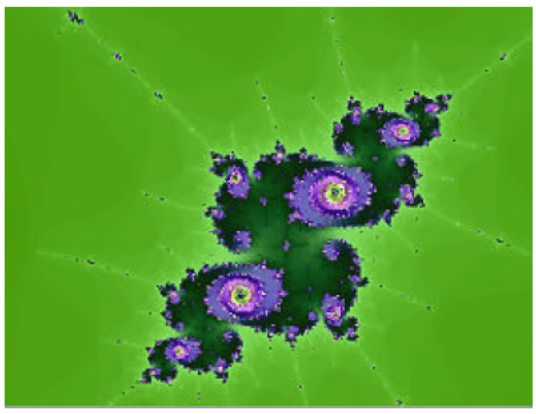


Fig. 2: Julia fractal image

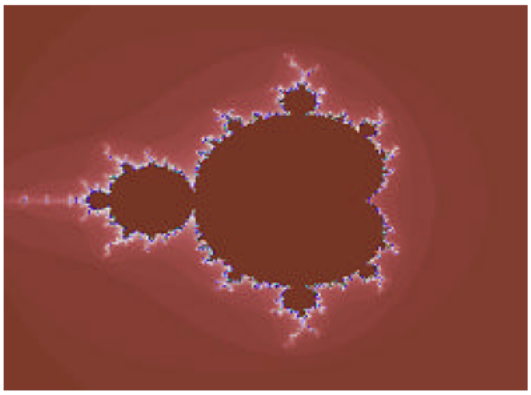


Fig. 3: Mandelbrot fractal image

Gaston Julia (Mandelbrot, 1982) is the set of points on a complex plane. Julia fractal image can be created by iterating the recursive Julia function Eq. 3.

In 1982, Benoit Mandelbrot began his refinement on Julia fractal set. He was looking for the connection on the value *c* from the Julia fractal equation (Lazareck *et al*., 2001). As the result, Mandelbrot fractal was defined and it was defined as the set of points on a complex plane by applying Eq. 4 iteratively (Fig. 3). Although Mandelbrot fractal set iterates z^2+c with z starting at 0 and Julia set iterates z^2+c starting with varying non-zero z which is a slight difference from Mandelbrot equation, but actually they are both using the same basic fractal equation as we can see from Eq. 3 and 4. The connection between Mandelbrot fractal set and Julia fractal set is that, each point c in the Mandelbrot is actually specifies the geometric structure of a corresponding Julia set (Alia and Samsudin, 2007).

$$z_n = z_{n-1}^{\ 2} + c; c, z_n \in C; n \in Z \quad (3)$$

$$z_n = z_{n-1}^{\ 2} + c;\ z_0 = 0;\ c, z_{n-1} \in C; n \in Z \quad (4)$$

PUBLIC KEY

As mentioned earlier, the concept of public-key cryptosystem was developed by Diffie-Hellman (1976) and the first encryption protocol based on the public-key concept is the RSA algorithm which was published by Revist *et al*. (1978), Al-Tuwaijry and Barton (1991). In RSA, one key is known to public and is used to encrypt the information by the sender. The other key is known as a private key and is used to decrypt the encrypted data received by the receiver. There are many other public-key

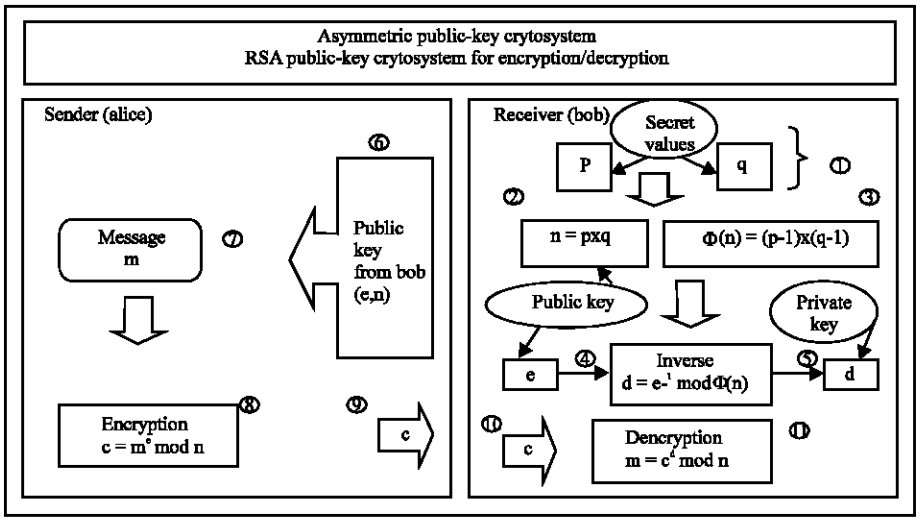


Fig. 4: RSA public-key protocol

encryption algorithms published since RSA. Among them are ElGamal (1985), Elliptic Curve (Koblitz, 1987; Rabin and Micheal, 1979) etc.

RSA public-key encryption: The RSA scheme is the most widely used public-key encryption algorithm. It can be used to provide both secrecy and digital signatures. The RSA security is based on the intractability of the integer factorization problem. As shown in Fig. 4, there are three integers e, d and n used in the encryption and decryption algorithm, where n = p×q, with p and q are large primes. Below are the details of the RSA algorithm.

The RSA algorithms

Algorithm for key generation (generated by receiver, Bob): Bob must do the following (referring to Step 1 to 5 on Fig. 4):

1. Choose two prime numbers (p, q) randomly, secretly and roughly of the same size.
2. Compute the modulus n as follows: n = p×q.
3. Compute the Φ (n), as follows: Φ (n) = (p-1)×(q-1).
4. Choose the public key e, such that 1<e<Φ (n) and GCD (e, Φ(n)) = 1.
5. Compute the decryption key d, where $d = e^{-1} \bmod \Phi(n)$.
- Determine the public keys (e, n) and determine the private keys (Φ(n), d).

Algorithms for encryption and decryption

Encryption (sender-Alice): Alice must do the following (referring to Steps 6 to 9 on Fig. 4):

Table 1: Working example of the RSA public-key encryption protocol

Keys generation (Bob generates the keys)			
1	p = 5;	4	Ø (n) = 24;
2	q = 7;	5	e = 5;
3	n = 35.	6	d = 5.
Encryption (Alice encrypts the plaintext)		Decryption (Bob decrypts)	
7	m = 4;	9	C = 9;
8	$c = 4^5 \bmod 35 = 9$.	10	$m = 9^5 \bmod 35$; m = 4.

6. Obtain the public keys (e, n).
7. Determine the message m to be encrypt such that 0<m<n.
8. Encrypt the message as follows, $c = m^e \bmod n$.
9. Send c to Bob (receiver).

Decryption (receiver-Bob): Bob must do the following (referring to Step 10 and 11 on Fig. 4):

10. Obtain c from Alice.
11. Recover the message as follows, $m = c^d \bmod n$.

Table 1 shows the working example of RSA public-key algorithm. In this example, the receiver must generate the public and private keys, which p is initialized to 5 (Table 1, Step 1) and q is initialized to 7 (Table 1, Step 2). p and q are prime numbers and used to compute the value n = 5×7 and the secret value Φ(n) = (5-1)×(7-1) (Table 1, Steps 3 and 4). Also the receiver must choose e which is initialized to 5 as the public key (Table 1, Step 5). Then the receiver has to calculate his own decryption key d as shown by Table 1, Step 6. Sender will choose his message and produces the cipher value, c, after executes $c = m^e \bmod n$, as shown by Table 1, Steps 7 and 8. Table 1, Steps 9, 10 and 11, show the deciphering of m, after executing the equation, $m = c^d \bmod n$ ($m = 9^5 \bmod 3 = 4$).

PUBLIC-KEY ENCRYPTION BASED ON THE MANDELBROT AND JULIA FRACTALS SETS

Mandelbrot and Julia fractal shapes contain complex number points, computed by the recursive functions (Eq. 3 and 4). In this study we are using Mandelbrot and Julia properties to design a new public-key encryption protocol. In this study, with the aids of Fig. 5, we briefly explain the propose idea of the fractal encryption protocol.

In the proposed protocol, sender and receiver must agree and use a public domain value, c. The receiver, Bob, will generate e and n as the private keys, while the sender, Alice, generates k and d as her private keys. Sender and receiver use their private values as well as the value c as inputs to the Mandelbrot function to produce the public-keys z_nd and z_ke. Then Bob and Alice must exchange the public keys. Alice will obtain Bob's public key, z_nd and uses this value together with her private key and the plaintext, as inputs to the Julia function to produce the ciphertext V, which will then send to Bob. Bob must obtain Alice's public-key, z_ke and the ciphertext V from Alice which will be used as input values together with his own private key to Julia function, to decipher the ciphertext V.

Mandelfn and Juliafn function of the Mandelbrot and Julia fractal sets: In this study, we use a specific Mandelbrot function, Mandelfn and similarly, a specific Julia function, Juliafn (Alia, 2007). An example of image generated from the Mandelfn and the Juliafn is shown in Fig. 6. In Mandelfn and Juliafn functions, we can substitute the function f( ) in Eq. 5 and Eq.6 with well known equations such as sin( ), cos( ), exp( ), etc. However, the value which is generated by Mandelfn must reside within the Mandelbrot set and similarly, the value generated by *Juliafn* must reside within Julia set (Giffin, 2006). In our protocol we set f( ) as shown by Eq. 7 for Mandelfn function and Eq. 8 for Juliafn function.

$$z(n+1) = c \times f(z(n)) \tag{5}$$

$$f(z(n)) = z_{n-1} \times c \times e; z, c, e \in C; n \in Z \tag{6}$$

In this study we will describe fractal public-key encryption in details (Fig. 7). The first step of the proposed protocol is to generate the private key and public-key by using Mandelbrot function Mandelfn Eq. 7 and Julia function Juliafn Eq. 8.

$$z(n+1) = c \times f(z(n)); \; z(0) = c; \; c, z \in C; \; n \in Z \tag{7}$$

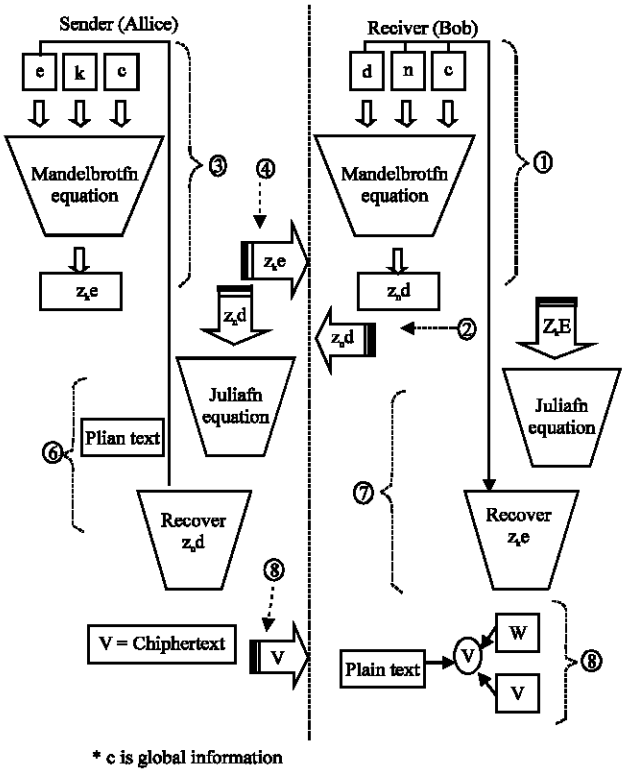


Fig. 5: Fractal public-key encryption protocol

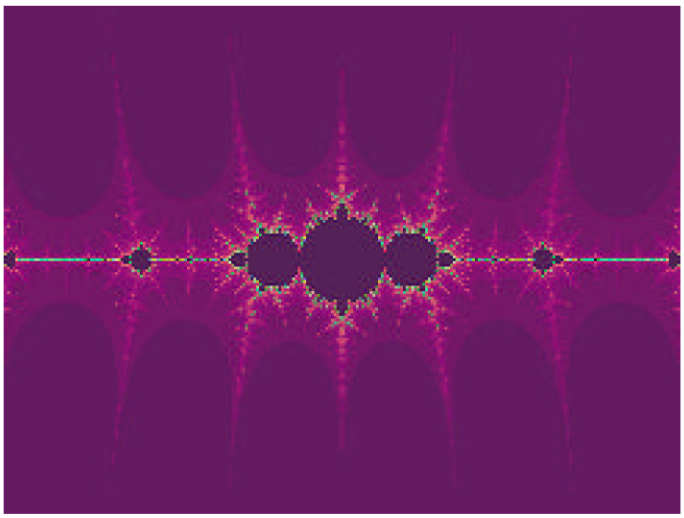


Fig. 6: Mandelfn and Juliafn image (Giffin, 2006)

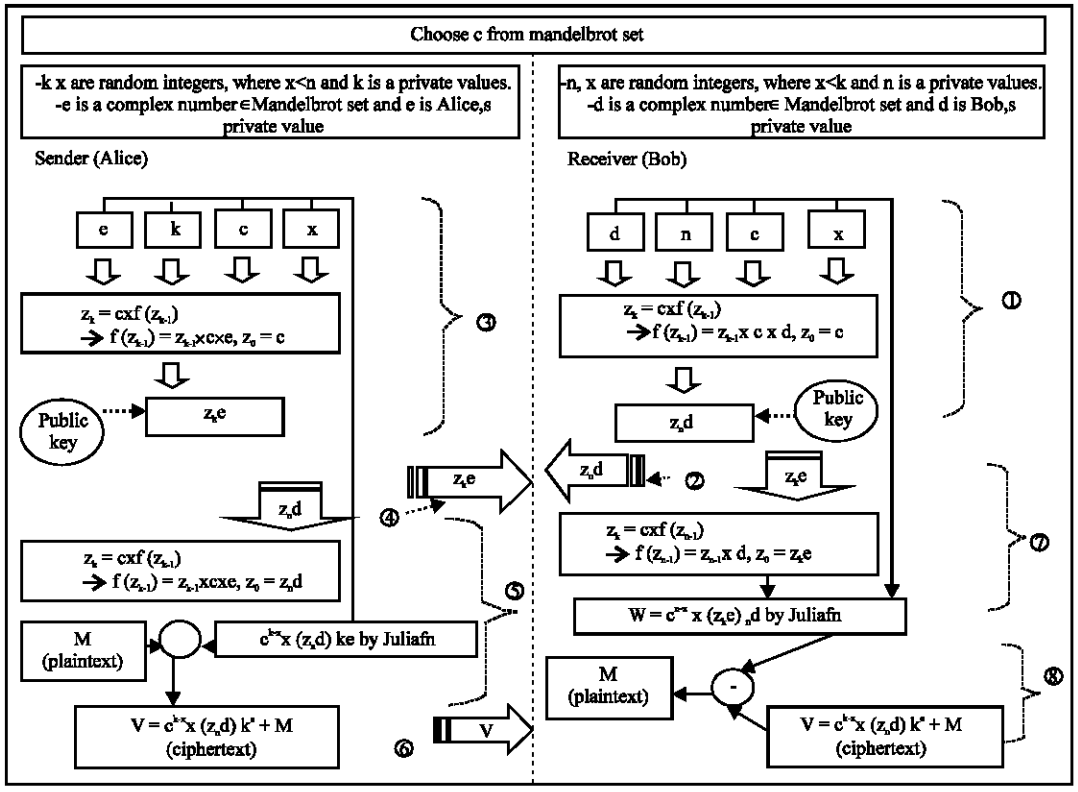


Fig. 7: Fractal public-key encryption algorithm

$$z(n+1) = c \times f(z(n));\ z(0) = y;\ \ y,c,z \in C; n \in Z \quad (8)$$

As shown in Fig. 5 earlier, fractal public-key encryption protocol involves sender and receiver. The receiver must generate the public key from the chosen private key and then send the public key to the sender. The sender will then generates his public key by using Mandelfn function and send it to the receiver.

$$Z_n d = z_{n-1} \times c^2 \times d; z,c,d \in C; n \in Z \quad (9)$$

$z_n d$ is the generated public-key, generated by the receiver by executing Eq. 9 (Step 1 of Fig. 7). The receiver's private key is the value (d, n). Similarly for the sender, with the private value of (e, k), the sender will produce the corresponding public key, $z_k e$ (Step 3 from Fig. 7) generated by using Mandelfn as shown by Eq. 10.

$$Z_k e = Z_{k-1} \times c^2 \times e;\ z,c,e \in C; k \epsilon Z \quad (10)$$

In step 5 and 6 (Fig. 7), executing Juliafn by the sender will encrypt the plaintext to produce the ciphertext V. The ciphertext V, will then send to the receiver. Similarly (Fig. 7, Step 7), the receiver will execute Juliafn to produce W which then is used to recover back the plaintext M (Fig. 7, Step 8).

It is impossible to mount a ciphertext attack on the proposed protocol because of the iteration, k and the variation constant e, which are unknown to the public. Hence, we can identify that the hard problem for the proposed fractal public-key encryption is through the chaos property of the fractal function which in this case depends on the key selection. This is true since the generated complex value ($z_n d$ and $z_k e$) produced by *Mandelfn* depends on the number of iterations, n, as well as the variation constant, d and e which makes the *Mandelfn* values jump path chaotically. This will prevent attack on the private values, given that d and e are being represented appropriately. We are suggesting the value of d and e to be represented by a 128-bit value which should give 2128 possibilities for every value of n that is being brute force.

After exchanging the public keys (Step 2 and 4 from Fig. 7) and executing the Juliafn function (Step 5 and 7 from Fig. 7), sender Alice and receiver Bob had completed the proposed secured encryption and decryption protocol. The process from Fig. 7, Step 5 is also being illustrated by Eq. 11. The corresponding decryption process, which is Step 7 of Fig. 7 is further illustrated by Eq. 12.

$$V = c^{k-x} \times (z_n d)_k e + M; \quad V,c,e,d \in C;\ n,x,k \in Z; M \in R \quad (11)$$

$$W = c^{n-x} \times (z_k e)_n d; \quad W,c,e,d \in C;\ n,x,k \in Z \quad (12)$$

Table 2 shows a working example of the proposed protocol. In this example each complex number is being represented by a 64-bit value. We use GMP (http://swox.com/gmp/, 2006) to simulate the 64-bit complex numbers. In this example, the public information, c, is initialized to a complex value (-0.022134) + (-0.044) i and variable x is initialized to 3 (The value of x is used to reduce the final calculation, Eq. 13 and 14. The value x can be set to 0, if desired). At the beginning, receiver and sender need to choose their private keys (Table 2). Then they have to calculate the corresponding public keys by using the Mandelbrot function, Mandelfn, as shown by Table 2. These values are $z_n d$ (receiver's public key) and $z_k e$ (sender's public key). Table 2, shows both parties exchanging their public keys. Following this process is the calculation of the ciphertext by using Julia function, Juliafn. Sender will produce the cipher value, V, after executes the Juliafn with input parameters k and d (sender's private key) as shown by Table 2. Table 2 shows the decrypted value M, after the Juliafn is executed with parameters n and e (receiver's private key).

Table 2: Example of fractals based public-key encryption protocol

No.	Description	Reciever	Sender
1		Choose c = -0.022134+-0.044 from Mandelbrot set	
2	Generate the private keys	n = 4 d = 0.0134078079299425970 - 0.013407807929942597	k = 6 e = 0.013407807 +- 0.04340780792994 m = 0.0098765+0.00134078
3	Generate the public keys $z_n d$ and $z_k e$ by using Mandelfn	$z_n d$ = 0.0231483882363480530143 + 0.00465248587376768622462	$z_k e$ = 0.000000003216724132211132396 34 + 0.000000002283531128803132964 51
4	Exchange the public keys $z_n d$ and $z_k e$ between sender and reciever	$z_k e$ = 0.000000003216724132211132396 34 + 0.000000002283531128803132964 51	$z_n d$ = 0.0231483882363480530143 + 0.00465248587376767622462
5	Sender must find v = cipher text by Juliafn and then send V to the reciever.	V = 0.008459434591738412561 26+	V = 0.002045673250788008458726
6	Reciever must calculates W by executing Juliafn to decrypt V	M = 0.0098765 + 0.00124078	.....

Table 3: Key spce comparison between fractal based public-key encryption and RSA

Key size	Fractals key space	SA (primes) key space
8-bit	256	54
16-bit	65536	6542
32-bit	4294967296	193635250
64-bit	18446744073709551616	415828533893661771
128-bit	3.40282366920938463463 3633e+38	3.835341275963952949425+36
192-bit	6.277101735386680763835e+57	9.047759628421369649479 7e+52
256-bit	1.157920892373161954235e+77	6.525495044027851465819 9e+74
512-bit	1.340780792994259709 95e+154	3.778003522278687762 56e+151

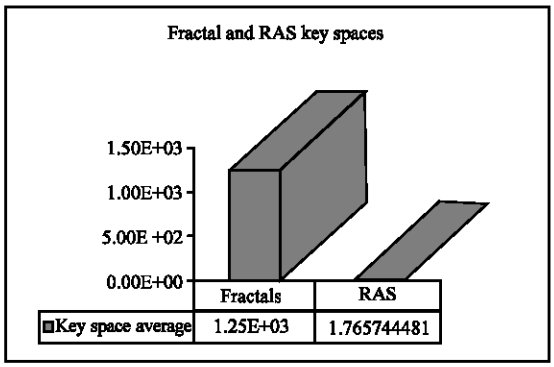


Fig. 8: Key space comparison between fractal key and RSA encryption implementation

KEY SIZE

The chaotic nature of the fractal functions ensures the security of the proposed protocol. However, to prevent a brute force attack, the choice of the key size becomes crucial. The key space in fractal public-key encryption depends on the size of the key. For example in 128 bits key, there are 2128 possible key values. RSA keys are fundamentally different from fractal keys. The RSA protocol depends on large prime numbers (Fig. 8). The 128-bit RSA key space is limited by how many primes exist in the finite field of Z_p, where p is the largest prime that can be represented by a 128-bit value. Therefore, RSA key space is considerably smaller than the fractal key space for a given finite field (Diffie, 1976). Table 3 shows the key space for both RSA and the proposed fractal public-key encryption algorithms for a given key size. The key space for RSA was calculated based on the number of primes existed for particular key sizes. The calculation was based on Eq. 13 (Caldwell, 2006).

$$\text{No. of prime in } [0,n) = n/\log n;\ n \in Z. \quad (13)$$

PERFORMANCE EVALUATION BASED ON EQUIVALENT KEY SIZES FOR FRACTAL AND PUBLIC-KEY ENCRYPTION PROTOCOL

We compare the performance of the fractal based public-key encryption protocol against the well known RSA public-key encryption protocol. Table 4 shows the performance for both approaches. Both protocols were coded in Turbo C with GMP library and run on a computer with 1.6 GHz Intel® M Pentium processor and 256MB RAM. Also, we used Miller-Rabin algorithm (MediaWiki, 2006) for primality test which was coded using C and GMP as well.

Table 4: Performance evaluation between fractal based public-key encryption and RSA encryption protocols

	Fractal encryption		RSA	
Description	Key size	Time (Milliseconds)	Key Size	Time (Milliseconds)
Key generation		35		580
Encryption	64-bit	5	512-bit	7
Decryption		5		10
Key generation		144		3575
Encryption	128-bit	50	2304-bit	20
Decryption		40		630
Key generation		8763		10465
Encryption	192-bit	3565	7680-bit	79
Decryption		3485		15462
Key generation		60187		36442
Encryption	256-bit	39507	15360-bit	300
Decryption		36933		108386

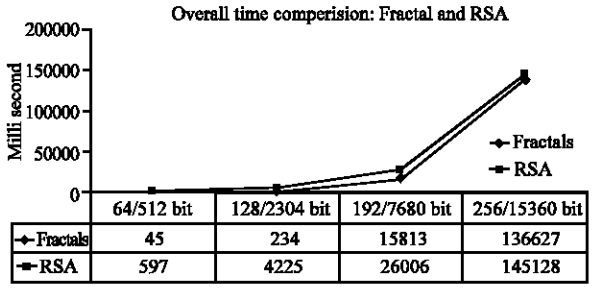
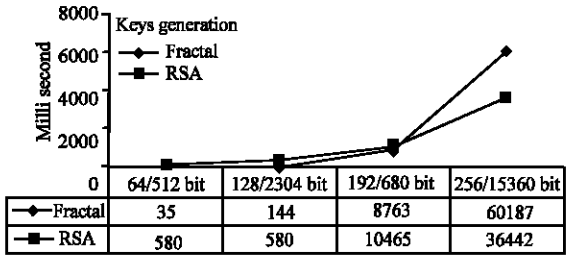


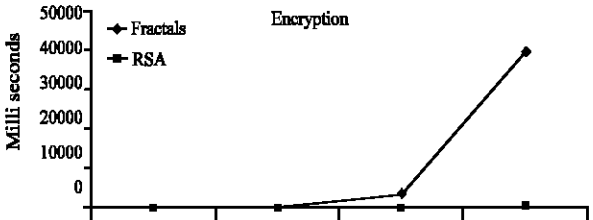
Fig. 9: Overall time comparison between fractal and RSA public-key encryption algorithm time

The comparison between fractal and RSA public-key encryption protocols shows that fractal key encryption protocol performs better than RSA in general. Note that, in our implementation we increased the number of iterations k and n (Fig. 7) proportionate with the key size to get suitable comparisons as shown by Fig. 9-12. As those Figures indicate, the fractal based public-key encryption\decryption protocol provides higher level of security at a much lower cost, both in term of key size and execution time.



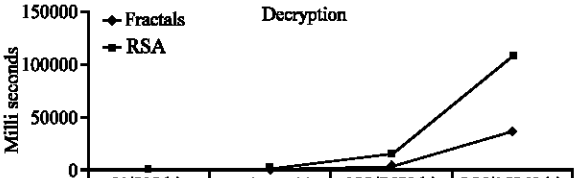
	64/512 bit	128/2304 bit	192/680 bit	256/15360 bit
Fractal	35	144	8763	60187
RSA	580	580	10465	36442

Fig. 10: Fractal and RSA keys generation time



Fractals	5	50	3565	39507
RSA	7	20	79	300

Fig. 11: Fractal and RSA encryption time



	64/512 bit	128/2304 bit	192/7680 bit	256/15360 bit
Fractals	5	40	3485	36933
RSA	10	630	15462	108386

Fig. 12 Fractal and RSA decryption time

THE SECURITY OF FRACTALS PUBLIC-KEY ENCRYPTION PROTOCOL

The strength of the algorithm and the size of the key used, play the main role in the security of public-key encryption protocol. Both fractal and RSA protocols can provide equal strength in security, both in terms of the algorithm complexity and the key size used. Nevertheless, fractal public-key encryption algorithm is more efficient than RSA since the algorithm used small key size and executes faster.

CONCLUSION

This study has shown the possibility of establishing a fractal based public-key encryption, emanating from the logical connection between the Mandelbrot and Julia fractal sets. The security of the proposed fractal public-key encryption depends on the number of iterations which convert the initial value *c* in the Mandelbrot fractal equation to the starting value of z in Julia fractal equation. Adding the key e and d during the iteration of Mandelbrot and Julia functions introduces the needed complexity of the proposed protocol. As the result, the proposed public-key encryption protocol requires small key size and performs faster when compared to RSA.

ACKNOWLEDGMENT

The authors would like to thank the Universiti Sains Malaysia (USM) for supporting this study.

REFERENCES

Alia, M. and A. Samsudin, 2007. New Key Exchange Protocol Based on Mandelbrot and Julia Fractal Sets. Int. J. Computer Sci. Network Security, 7: 302-307.

Al-Tuwaijry, F.A. and S.K. Barton, 1991. A high speed rsa processor, IEEE. Conf., pp: 210-214.

Branovic, I.R., Giorgi and E. Martinelli, 2003. Memory performance of public-key cryptography methods in mobile environments, ACM SIGARC workshop on memory performance: Dealing with Applications, Systems and Architecture (MEDEA-03), New Orleans, LA, USA., pp: 24-31.

Caldwell, C.K., 2006. How many primes are there, http://primes.utm.edu/, The University of Tennessee at Martin.

Diffie, W. and M.E. Hellman, 1976. New directions in cryptography. IEEE. Trans. Inform. Theory, 22: 644-654.

ElGamal, T., 1985. A Public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE. Trans. Inform. Theory, or CRYPTO 84: 10-18, Springer-Verlag, 31: 469-472 .

Giffin N., 2006. Fractint, TRIUMF project at the University of British Columbia Campus in Vancouver B.C. Canada.

http://swox.com/gmp/, 2006, GMP Arithmetic Without Limitations, release: 4.2.1.

Jampala, S., 1992. Fractals: Classification, Generation and Applications, Circuits and Systems. IEEE. Conf., pp: 1024-1027.

Koblitz, N., 1987. Elliptic curve cryptosystems, in Mathematics of Computation, pp: 203-209.

Lazareck, L., G. Verch and J.F. Peter, 2001. Fractals in Circuits, IEEE. Conf., pp: 589-594.

Mandelbrot, B., 1982. Fractal Geometry of Nature, San Francisco: W.H. Freeman.

MediaWiki: Literate Programs, 2006. Miller-Rabin, http://en.literateprograms.org/Miller-Rabin_primality_test_(C,_GMP).

Menezes, A.P., Van Oorschot and S. Vanstone, 1996. Handbook of Applied Cryptography. CRC Press, 516: 4-15.

Patrzalek, E., 2006. Fractals: Useful Beauty General Introduction to Fractal Geometry, Stan Ackermans Institute, IPO, Centre for User-System Interaction, Eindhoven University of Technology.

Rabin and Michael, 1979. Digitalized signatures and public-key functions as intractable as factorization. MIT Laboratory for Computer Science.

Rivest, R.A., Shamir and L. Adleman, 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM., 21: 120-126.

Stallings, W., 2003. Cryptography and Network Security Principles and Practices, Pearson Education, (3rd Edn).