New Key Exchange Protocol Based on Mandelbrot and Julia Fractal Sets

Mohammad Ahmad Alia and Azman Bin Samsudin,

School of Computer Sciences, Universiti Sains Malaysia.

Summary
In this paper, we propose a new cryptographic key exchange protocol based on Mandelbrot and Julia Fractal sets. The Fractal based key exchange protocol is possible because of the intrinsic 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 then used to calculate the shared key based on the existing private key and the received public key. The proposed protocol is designed to be resistant against attacks, utilizes small key size and comparatively performs faster then the existing Diffie-Hellman key exchange protocol. The proposed Fractal key exchange protocol is therefore an attractive alternative to traditional number theory based key exchange protocols.

Key words:
Fractals Cryptography, Key- exchange protocol, Mandelbrot Fractal set, and Julia Fractal set.

1. Introduction

Cryptography algorithms are classified into two categories, secret key (symmetric) algorithms and public key (asymmetric) algorithms. In general, cryptography protocol employs public key cryptosystem to exchange the secret key and then uses faster secret key algorithms to ensure confidentiality of the data stream [1, 2]. Secret key algorithm is used to encrypt and decrypt messages by using the same secret key. Public key algorithm on the other hand, works in a very different way. In public key encryption algorithm, there are two keys, both belong to the recipient. One key is known to the public, and is used to encrypt information that need to be send to the receiver who owns the corresponding private key.

The Diffie-Hellman (DH) algorithm was the first key exchange algorithm to utilize public-key concept to exchange the shared key. Public key based key exchange protocol rises above the difficulties faces by the secret key cryptosystem. This is because key management is much easier with the help of a key exchange protocol such as DH. In a secret key algorithm, both parties shared the same secret key. However the process of sharing the secret key between both the sender and the recipient, introduces a new set of problem – key distribution problem. Public key cryptosystem alleviates the key distribution problem by using two keys, a private and a public key. In most cases, by exchanging the public keys, both parties can calculate a unique shared key, known only to both of them [2, 3, 4].

This paper proposed a new Fractal (Mandelbrot and Julia Fractal sets) key exchange protocol as a method to securely agree on a shared key. The proposed method achieved the secure exchange protocol by creating a shared secret through the use of the Mandelbrot and Julia Fractal sets.

2.1. Fractals

A complex number consists of a real and an imaginary component. It is common to refer to a complex number as a "point" on the complex plane. If the complex number is $Z = (a + bi)$, the coordinate of the point is, a for the horizontal real axis, and b for the vertical imaginary axis. The unit of imaginary numbers is defined as $i = \sqrt{(-1)}$ [5].

Fractals on the other hand are fragmental geometric shape that is created interactively from almost similar smaller components [6]. From another perspective, Fractals are an example of a Chaos system, where by changing the initial parameters to the system, even slightly, can generate a totally new Fractal image altogether [7]. Figure 1 shows two examples of Fractal sets, Mandelbrot Fractal set and Julia Fractal set.

(a) (b)

Figure 1: Mandelbrot and Julia Fractal sets; (a) Mandelbrot image, (b) Julia image [4].

The Mandelbrot Fractal set (see Figure 1(a)) is the set of points on a complex plane. The Fractal image can be generated by applying Equation 1 recursively [5, 8].

$$Z_n = Z_{n-1}^{\ 2} + c\,;\, Z_0 = 0;\ c, Z_{n-1} \in C; n \in Z. \quad (1)$$

Similar to Mandelbrot Fractal set, Julia Fractal set (see Figure 1(b)) is a set of points on a complex plane defined recursively by Equation 2.

$$Z_n = Z_{n-1}^{\ 2} + c\,;\ c, Z_n \in C; n \in Z. \quad (2)$$

The difference between the Mandelbrot set and the Julia set is that the Mandelbrot set iterates Z^2+c with Z starting at 0 and varying c with every iteration, while Julia set iterates Z^2+c for fixed c and starting with non-zero value of Z [6, 10]. All points, Z_n, must reside on the Mandelbrot set or the Julia set, respectively. In our work, we are depending on the intrinsic connection between both of the Mandelbrot and the Julia Fractal sets. The connection between the Mandelbrot set and the Julia set is that each point c in the Mandelbrot set specifies the geometric structure of the corresponding Julia set [9].

2.2. Diffie-Hellman Key Exchange

The Diffie-Hellman key agreement protocol was developed by Diffie and Hellman in 1976 and was published in a paper called "New Directions in Cryptography". The Diffie-Hellman key agreement protocol is also known as the "Diffie-Hellman-Merkle" protocol. The Diffie-Hellman protocol is used to exchange a secret key between two users over an insecure medium without any prior communication between them. Normally, the exchanged secret is then used as the secret key for subsequent communication (see Figure 2).

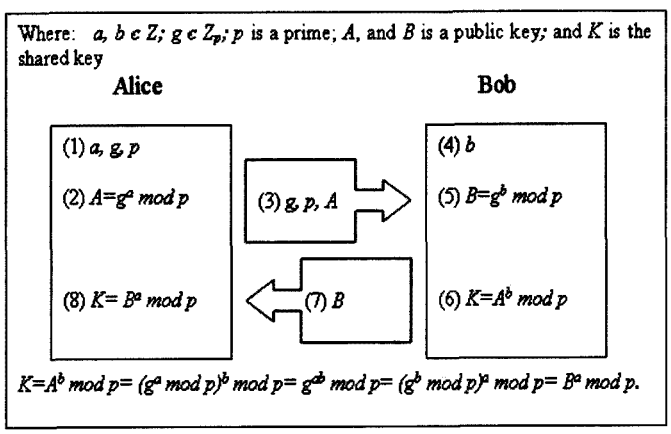


Figure 2: Diffie -Hellman key exchange protocol

3. Key Exchange Protocol Based on the Mandelbrot and Julia Fractals Sets

Mandelbrot and Julia Fractal sets are set of complex number, generated by complex number equations as discussed in Section 2.1. By utilizing Fractals strong properties, we have designed a new key exchange protocol based on Mandelbrot and Julia Fractal Sets. Figure 3 shows the proposed key exchange protocol.

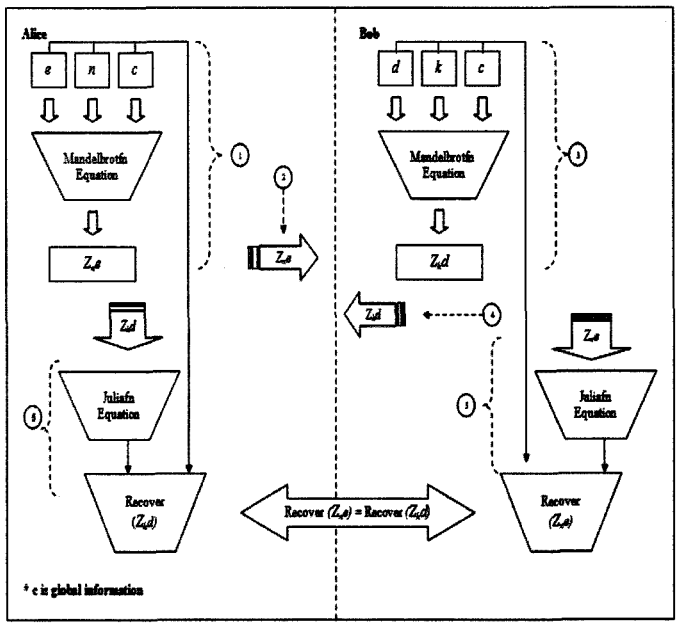


Figure 3: Fractal key exchanging protocol

In the proposed protocol, c is the global value known to the public. e and n are the private values for Alice while k and d are the private values for Bob. The private values and the global information c are then used as the inputs to the Mandelbrot function, which will in turn produces the public keys, Z_ne and Z_kd, for Alice and Bob, respectively. The public values will be used in the exchange process between Alice and Bob. With the private information and the other party's public key as the inputs to the Julia function, both parties will be able to generate the same secret key, $(Z_ne)_kd = (Z_kd)_ne$.

3.1. Mandelbrot and Julia Fractal Sets Key-Exchange Protocol

In this section we will describe the proposed key exchange algorithm in detail. The first step in this protocol is to generate the public key and the private key by using Mandelbrot and Julia functions. The equation used in our proposed protocol is the Mandelbrot function, "*Mandelfn*" (see Equation 4) and Julia function, "*Juliafn*" (see

Equation 5). *Mandelfn* is one of the many Mandelbrot functions, and similarly *Juliafn* is a specific form of Julia functions. An image generated from the *Mandelfn* function is shown in Figure 4.

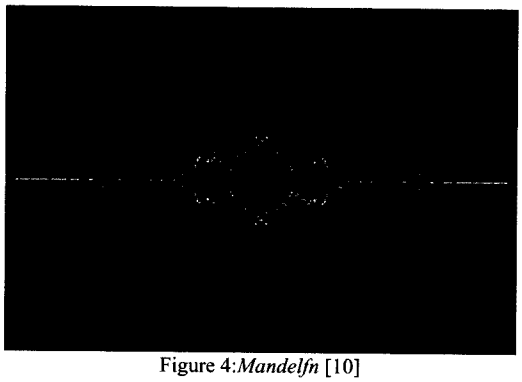


Figure 4:*Mandelfn* [10]

$$Z(0) = c; c, Z \in \mathbf{C}. \quad (3)$$

$$Z(n+1) = c \times f(Z(n)); c, Z \in \mathbf{C}; n \in Z. \quad (4)$$

It is easy to generate variation of Fractal images based on *Mandelfn* and *Juliafn* functions. For example, we can substitute the function *f( )* in Equation 5 and 6 with some known functions such as *sin( ), cos( ), exp( ),* etc., to generate different variation of the functions. However, the generated values from *Mandelfn* must always reside within the Mandelbrot set, and similarly, the values generated by the *Juliafn* must reside within the Julia set [10].

$$Z(n+1) = c * f(z(n)) \quad (5)$$

$$Z(n+1) = c \times f(Z(n)); \; Z, c \in \mathbf{C}; n \in Z. \quad (6)$$

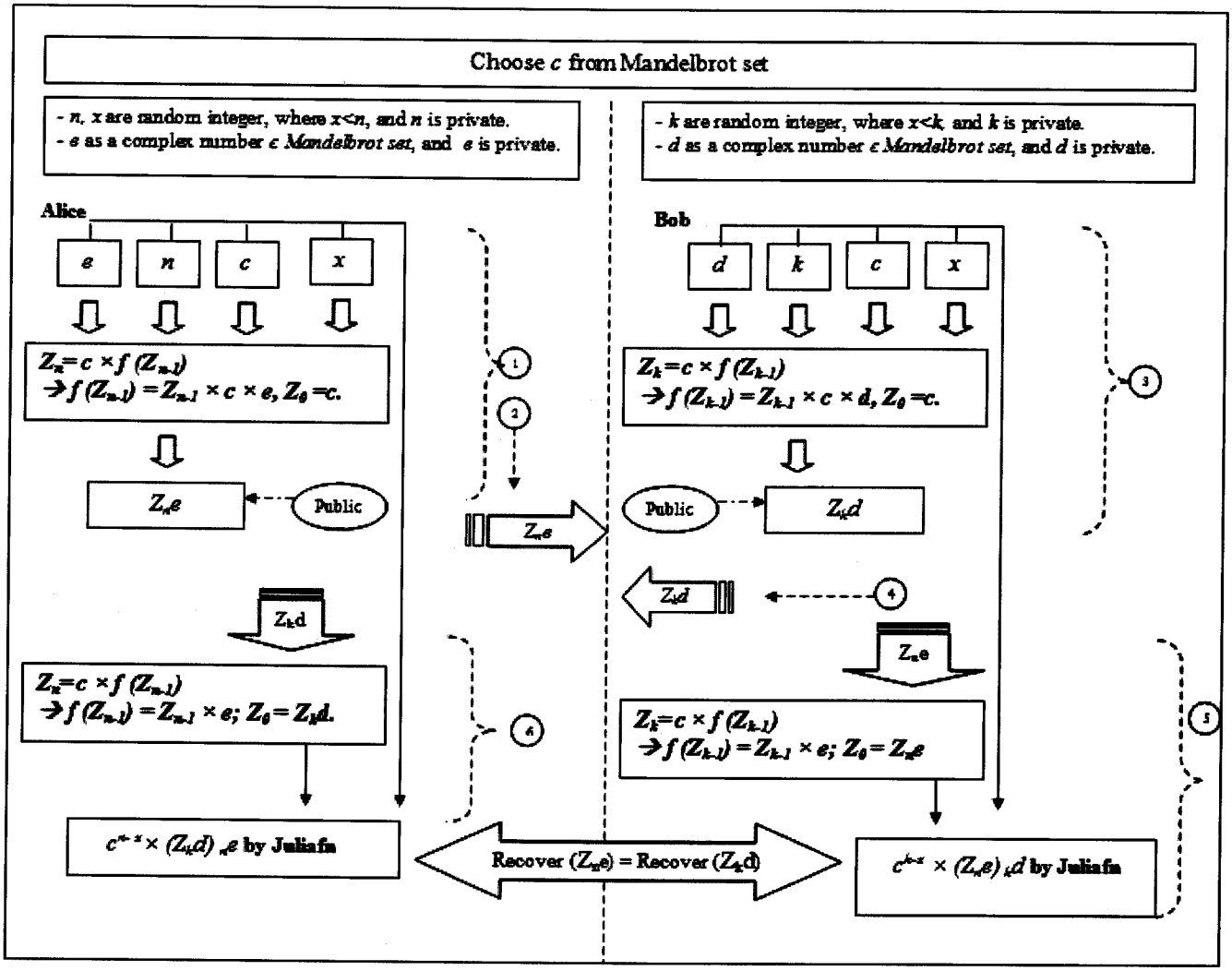


Figure 5: Fractal Key Exchanging Algorithm

As shown by Figure 3, Fractal key exchange protocol involves two parties, Alice and Bob. Alice must generate the public key based on her private key as describe earlier. The generated public key is then send to Bob. Bob on the other hand will do the same and send his public key to Alice. To produce the public key, we use Mandelbrot function, *Mandelfn*. For Alice, the generated public key is Z_ne, as describes by Equation 7 (see step 1 from Figure 5).

$$Z_n e = Z_{n-1} \times c^2 \times e; Z,c,e \in C; n \in Z. \quad (7)$$

Similarly for Bob, the generated public key, Z_kd, (see step 3 from Figure 5) is calculated by using *Madelfn* equation as shown by Equation 8.

$$Z_k d = Z_{k-1} \times c^2 \times d; Z,c,d \in C; k \in Z. \quad (8)$$

Note that, it is impossible to find the private values from the published public keys, since the iteration, n, and the variation constant e, are unknown to the public. Hence, we can identify that the hard problem for the proposed Fractal key exchange is through its key selection. This is true since the complex value produces by *Mandelfn* depends on the number of iterations, n, as well as the variation constant, e, which makes the *Mandelfn* values jump path chaotically. This will prevent attack on the private values, given that e is being represented appropriately. We are suggesting e to be represented by a 128-bit value which should give 2^{128} possibilities for every values of n that are being brute force.

After exchanging the public keys (see steps 2 and 4 from Figure 5) and executing the *Juliafn* function (steps 5 and 6 from Figure 5), both Alice and Bob will arrived at the same secret value, $(Z_ne)_kd = (Z_kd)_ne$. Both $(Z_ne)_kd$ and $(Z_kd)_ne$ are equals, based on a known Fractal property as shown by Equation 9.

$$c^{n-x} \times (Zkd)_n e = c^{k-x} \times (Z_n e)_k d; \quad Z,c,e,d \in C; n,x,k \in Z. \quad (9)$$

Table 1 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 [11] to simulate the 64-bit complex numbers. In this example, the global information, c, is initialized to a complex value (-0.1155056)+(-0.359816)i, and x is initialized to 3 (The value of x is used to reduce the final calculation (see Equation 9) and can be set to 0, if desired). At the beginning, Alice and Bob need to choose their private keys (see Table 1, row 1). Then they have to calculate the corresponding public keys as shown by Table 1, row 2, by using the Mandelbrot function, *Mandelfn*. These values are Z_ne (Alice's public key) and Z_ne (Bob's public key). Table 1, row 3, shows both parties exchanging their public keys. Following this process is the calculation of the shared value by using Julia function, *Juliafn*. Alice will produce the shared value, $(Z_kd)_ne$, after executes the *Juliafn* with n,e (Alice's private key) and Z_kd (Bob's public key) as the input parameters. Similarly Bob will produce the shared value $(Z_ne)_kd$ by executing the *Juliafn* function with k,d (Bob's private key) and Z_ne (Alice' public key) as the input parameters. This process is shown by row 4 of Table 1. In row 5 of Table 1, we show that both shared values are indeed the same.

Table 1: Example of fractals based key exchange protocol

No	Description	B	A
1	Random integer no.	k=7;d=-0.50001002+-0.50001002;	n=6;e=-0.84000007+-0.84000007;
2	Compute by Mendel's fn formula	Z_kd=-0.000019571131854664384825S + 0.000056890057067247429519S.	Z_ne=-0.125328392603999767482 + -0.956577009556459717744.
3	A send Z_ne, B send Z_kd	-0.125328392603999767482 + -0.956577009556459717744	-0.000195711318546643848258 + 0.000056890057067247429519S.
4	Compute by Julia's fn formula: A $(Z_ne)_kd$, B$(Z_kd)_ne$	-0.001665788303148910387964 + 0.000180504674723535012322	-0.001665788303148910387964 + 0.000180504674723535012322
5	A=B?	-0.001665788303148910387964 + 0.000180504674723535012322	-0.001665788303148910387964 + 0.000180504674723535012322

Table 2: Key space comparison between DH and Fractal based key exchange protocols

Key size	Fractals keyspace	DH keyspace
8-bit	256	54
16-bit	65536	6542
32-bit	4294967296	1936355250
64-bit	18446744073709551616	4158285338936617711
128-bit	3.40282366920938463463374607431774e+38	3.83534127596395259474259325670866+36
192-bit	6.27710173538668076383578942320777e+57	9.04775962842136964947976353980536+52
256-bit	1.15792089237316195423570985008696+77	6.52549504402785146581992180213036+74
512-bit	1.34078079299425970995740249982066+154	3.77800352227868776256529195323656+151

4. 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 Fractals key exchange depends on the size of the key. For example in 128 bits key, there are 2^{128} possible key values. Diffie Hellman (DH) keys are fundamentally different from Fractals keys. The DH protocol depends on large prime numbers (see Figure 6). The DH key space for 128 bit is limited by how many primes existed in the finite field of Z_p, where p is the largest prime that can be represented by a 128-bit value. Therefore, DH key space is considerably smaller than the Fractals key space for a given finite field [4]. Table 2

shows the key space for both DH and the proposed Fractal key exchange algorithms for a given key size. The key space for DH was calculated based on the number of primes existed for particular key sizes. The calculation was based on this Equation 10 [12].

$$\text{No. of prime in } [0,n) = n / \log n \; ; n \in Z \; . \qquad (10)$$

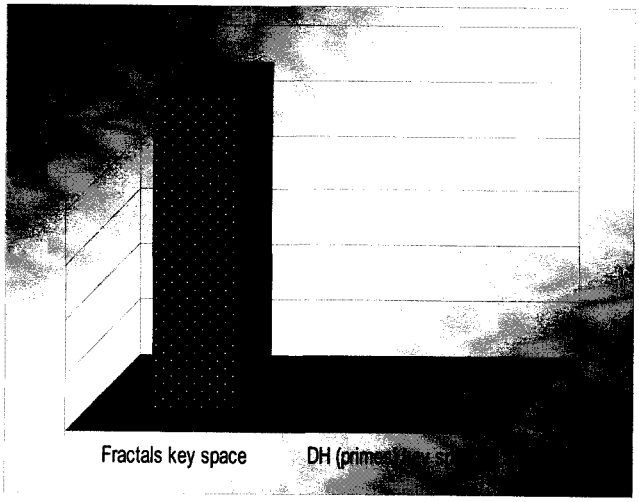


Figure 6: DH and Fractal key space

5. Performance Evaluation Based on Equivalent Key Sizes for Fractals and Diffie Hellman Key Exchange Protocols

We compare the performance of the Fractal based key exchange protocol against the well known DH key exchange protocol. Table 3 shows the performance for both DH and Fractal based key exchange protocols. Both protocols were coded in Turbo C and GMP, and run on a computer with 1.6 GHz Intel® M Pentium processor and 256MB.

The comparison [13] between Fractal and DH key exchange protocols shows that Fractal key exchange protocol performs better as shown by Figure 7 and 8. The Fractal based key exchange protocol provides higher level of security at a much lower cost, both in term of key size and execution time.

6. The Security of Fractals Key Exchange

The security of key exchange protocol is based on the strength of the algorithm and the size of the key used. Both Fractal and DH protocols can provide equal strength in security, both in terms of the algorithm complexity and the key size used. However, Fractal key exchange algorithm is more efficient than DH since the algorithm used small key size and execute faster

Table 3: Performance evaluation between DH and Fractal based key exchange protocols

Keys Generation	64-bit	30 Milliseconds	512-bit	101 Milliseconds
Exchanging		10 Milliseconds		13 Milliseconds
Keys Generation	128-bit	144 Milliseconds	1024-bit	240 Milliseconds
Exchanging		75 Milliseconds		100 Milliseconds
Keys Generation	192-bit	8763 Milliseconds	3840-bit	9914 Milliseconds
Exchanging		6910 Milliseconds		1522 Milliseconds
Keys Generation	256-bit	60187 Milliseconds	7680-bit	66205 Milliseconds
Exchanging		59246 Milliseconds		9893 Milliseconds

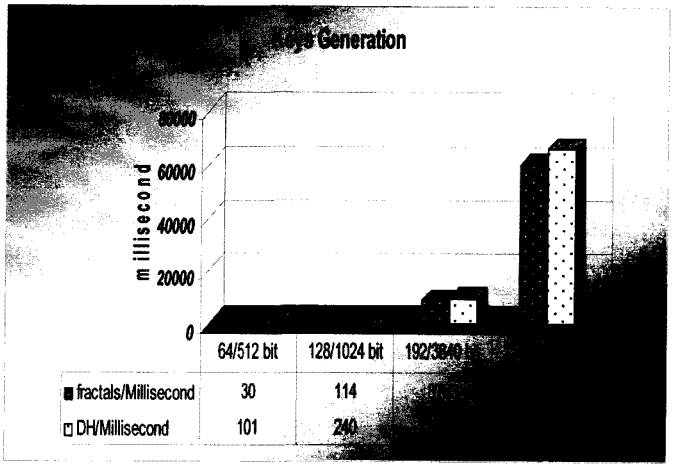


Figure 7: Fractal and DH key generation time

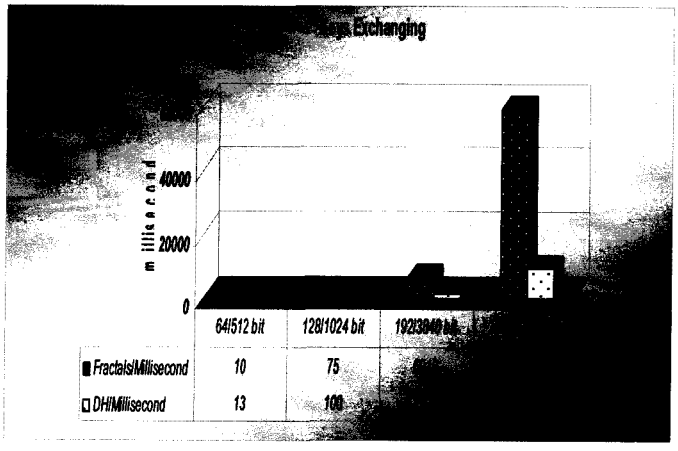


Figure 8: Fractal and DH key exchange running time

7. Conclusion

This paper has shown the possibility of using Fractal sets (Mandelbrot and Julia function sets) in Cryptographic keys exchange protocol. The Fractal based key exchange protocol is made possible because of the intrinsic connection between the Mandelbrot and Julia Fractal sets. The security of the proposed Fractal key exchange depends on the number of iterations which convert the initial value c in the Mandelbrot Fractal equation to the starting value of Z for Julia Fractal equation. Adding the key e during the iteration of Mandelbrot and Julia functions introduces the needed complexity of the proposed protocol. As the result, the proposed key exchange protocol requires small key size and performs faster compared to Diffie-Hellman, one of the most used key exchange protocols currently.

Acknowledgments

The authors would like to express their thanks to Universiti Sains Malaysia (USM) for supporting this study.

References

[1] I. Branovic, R. Giorgi, E. Martinelli, "Memory Performance of Public-Key Cryptography Methods in Mobile Environments," *ACM SIGARCH Workshop on Memory performance: Dealing with Applications, systems and architecture (MEDEA-03)*, New Orleans, LA, USA, pp. 24-31 , Sept. 2003.

[2] A. Menezes, P. Van Oorschot, and S. Vanstone, "Handbook of Applied Cryptography," *CRC Press*, pp.4-15, 516, 1996.

[3] K. Palmgren, "Diffie-Hellman Key Exchange – A Non-Mathematician's Explanation," *http://www.securitydocs.com/library/2978*, 02/02/2005.

[4] W. Stallings, "Cryptography and Network Security Principles and Practices,"*Pearson Education*, 3rd edition.

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

[6] P. Bourke, "An Introduction to Fractals," *http://local.wasp.uwa.edu.au/~pbourke/fractals/fracintro/*, May 1991.

[7] S. Spencer "About Fractals," *http://avatargraphics.com/fractalland/fractalfaq*.html, pp. 1-2, 1997.

[8] S. Zakeri, "On Biaccessible Points Of The Mandelbrot Set," *Institute for Mathematical Sciences, Stony Brook*

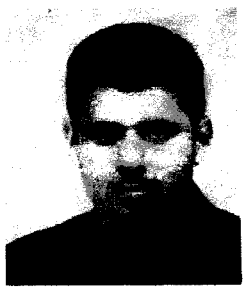
[9] M. C. Taylor, J. Louvet, "Sci.fractals FAQ," *Computing Services Mount Allison University Sackville, Canada*, 1998.

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

[11] http://swox.com/gmp/,"GMP Arithmetic without Limitations," release: 4.2.1, 2006.

[12] C. K. Caldwell, "How Many Primes Are There," *http://primes.utm.edu/*, The University of Tennessee at Martin,2006.

[13] A R&D team, "The Enduring Value of Symmetric Encryption," white paper, *Aladdin securing the Global Village*, pp.6-8, August 2000.



Mohammad A. Alia received the B.S. degree in Computer Sciences and M.S. degree in Information Technology from Al-Zaytoonah University in 2000 and 2003, respectively. During 2000-2004, he stayed at Al-Zaytoonah University-Jordan as an instructor of Computer Sciences and Information Technology. Then, he worked as a lecturer at Al-Quds University in Saudi Arabia form 2004 - 2005. Currently he is a PhD student at the School of Computer Sciences, Universiti Sains Malaysia.

Azman Samsudin is a lecturer at the School of Computer Sciences, Universiti Sains Malaysia. He received the B.Sc. degree in Computer Science from the University of Rochester, USA, in 1989. He obtained his M.Sc. and Ph.D. degrees in Computer Science from University of Denver, USA, in 1993 and 1998, respectively. His research interests are in the field of Cryptography, Interconnection Switching Networks, and Parallel Distributed Computing.