A New Digital Signature Scheme 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: This paper describes a new cryptographic digital signature scheme based on Mandelbrot and Julia fractal sets. Having fractal based digital signature scheme is possible due to the strong connection between the Mandelbrot and Julia fractal sets. The link between the two fractal sets used for the conversion of the private key to the public key. Mandelbrot fractal function takes the chosen private key as the input parameter and generates the corresponding public-key. Julia fractal function then used to sign the message with receiver's public key and verify the received message based on the receiver's private key. The propose scheme was resistant against attacks, utilizes small key size and performs comparatively faster than the existing DSA, RSA digital signature scheme. Fractal digital signature scheme was an attractive alternative to the traditional number theory digital signature scheme.

Keywords: Fractals Cryptography, Digital Signature Scheme, Mandelbrot Fractal Set, and Julia Fractal Set

INTRODUCTION

Cryptography is the science of information security. Cryptographic system in turn, is grouped according to the type of the key system: symmetric (secret-key) algorithms which utilizes the same key (see Fig. 1) for both encryption and decryption process, and asymmetric (public-key) algorithms which uses different, but mathematically connected, keys for encryption and decryption (see Fig. 2). In general, cryptography protocol employs public-key algorithm to exchange the secret key and then uses faster symmetric algorithms to ensure secrecy of the data stream [1, 2].

Public-key scheme is based on the idea that a user can possess two keys, one key is known to the public and the other is private to the owner. The public-key algorithm uses the public key to encrypt the data to be sent, and then at the recipient side, uses the private key to decrypt the ciphered data. Digital signature is a verification mechanism based on the public-key scheme that is focusing on message authenticity. The output of the signature process is called the digital signature [2]. Digital signatures are then used to provide authentication of the associated input, which is called a message [3, 4] (see Fig. 3). In digital signature public-key algorithms, the private key is used to sign a message, while the public key is used to verify the authenticity of the message.

Fig. 1: Secret-key scheme.

Fig. 2: Public-key scheme.

Digital signatures scheme used to provide the following [4]:
- Data integrity (the assurance that data has not been changed by unauthorized party).
- Message authentication (the assurance that the source of data is as claimed).
• Non-repudiation (the assurance that an entity cannot deny commitments).

In 1976, the first notion of a digital signature scheme was given by Whitfield Diffie and Martin Hellman, although at that time they only conjectured the existence of such scheme [5, 6]. Soon after that, in 1978, Rivest, Shamir, and Adleman invented the first digital signature scheme which is called RSA digital signature algorithm [7]. Subsequently, there are a few more proposed digital signature algorithms such as ElGamal signature scheme [8], Undeniable signature [9] and others.

**Fractals:** A complex number is a number of the form \( a + bi \), where \( a \) and \( b \) are real numbers, and \( i \) is the imaginary unit defined as \( i^2 = -1 \) [12]. The real number, \( a \), is considered as the real component, and the unit \( i \) and \( b \) are considered as the imaginary component of the complex number. The sum and product of two complex numbers are formulated as shown by Equations 1 and Equation 2.

\[
(a+b)+c+d) = (a+c)+(b+d)i; \quad a,b,d,c \in \mathbb{Z}; i^2 = -1.
\]  
\[
(a+bi)(c+di) = (ac-bd)+(bc+ad)i; \quad a,b,c \in \mathbb{Z}; i^2 = -1.
\]  

In the late 19th century, Henri Poincaré, Felix Klein, Pierre Fatou and Gaston Julia used the iterated function in the complex plane, but their findings were dwarfed by the absent of visualization effect. In the 1960’s the modern computer graphics era had shed new light on the study of iterated function, which soon gave the birth to the field of fractal geometrics.

Fractal was made famous by Mandelbrot. In fact, the word fractal itself was coined by Benoit Mandelbrot in 1960. The word fractal came from a Latin word “fractus” meaning “broken” or “fractured”. 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 viewpoint, fractals are irregular in shape [13, 10] and they do not cohere with the typical mathematical dimensions. One of the unique things about fractals is that they have no integer dimension, instead they have a real and a imaginary part as described earlier. Because of the imaginary part, generally fractal can be classified into two types: fractal curves, in which the dimension of the fractal curves fall between the first and second dimensions (1-D and 2-D), and fractal surface, in which shapes have a dimension between the second and third dimension (2-D and 3-D). There is another kind of fractal that is called “fractal dimensions” which can fall between the 0.64th to the 1.58th dimensions of the non-integer dimension [15]. The fractal dimension is a statistical quantity that shows how the fractal is filling the space completely during the zooms down to finer scales. There are many applications of fractal. One major example is the use of fractal to create a realistic image of nature such as the image of clouds, snow flakes, fungi, bacteria, mountains, river networks, systems of blood vessels and others [12, 16].

**Julia and Mandelbrot Fractal 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 [12]. One of the interesting fractal sets is the Julia fractal set. Julia fractal set (see Fig. 4), developed by Gaston Julia [10], is the set of points on a complex plane. Julia fractal image can be created by iterating the recursive Julia function (see Equation 3). Later in 1982, Benoit Mandelbrot expanded his ideas in the fractal geometry of nature by refining the Julia fractal set. He was looking for the connection on the value \( c \) from the Julia fractal equation [14]. As the result, Mandelbrot fractal was defined, and it was defined as the set of points on a complex plane by applying Equation 4 iteratively (see Fig. 5). 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 Equation 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. On the other hand if \( c \) is in the Mandelbrot set, the Julia set will be connected. However, if \( c \) is not in the

![Fig. 3: Digital signature scheme.](image)
Mandelbrot set, the Julia set will become a Cantor dust specified in a U.S. Government Federal Information Processing Standard [17]. The algorithm is called Digital Signature Standard (DSS). Fig. 6 illustrates the steps in the DSA digital signature algorithm. The DSA can be viewed as a variant of the ElGamal signature scheme. Both signature schemes are based on the same mathematical problem - discrete logarithm problem. DSA base its security on the complexity of the discrete logarithm problem in field of $Z_p$, where $p$ is a prime [18].

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

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

Fig. 4: Julia fractal image.

Fig. 5: Mandelbrot fractal image.

**DIGITAL SIGNATURE**

The well known digital signature schemes can be classified according to the inherited mathematical problems. As now, there are three different NP-hard problems where the most known digital signature schemes are based from:

1. **Integer Factorization (IF) schemes.** The security in integer factorization schemes are based on the complexity of the integer factorization problem. Examples of IF scheme implementation are RSA digital signature scheme [7] and Rabin digital signature scheme [16].

2. **Discrete Logarithm (DL) schemes.** Discrete logarithm schemes are based on the complexity of the discrete logarithm problem in a finite field. Examples of DL scheme implementation are ElGamal [8] and DSA [17].

3. **Elliptic Curve (EC) schemes.** The security in elliptic curve schemes are based on the complexity of the elliptic curve discrete logarithm problem. Examples of EC scheme is the elliptic curve digital signature [18].

**DSA Digital Signature Algorithm:** In 1991, the U.S. National Institute of Standards and Technology (NIST) proposed the digital signature algorithm (DSA) and was

The **DSA Algorithms:** Key generation algorithm (generated by receiver, Alice)

Alice must do the following (referring to Steps 1 to 6 on Fig. 6):

1. Choose a prime numbers ($p$), where $2L-1 < p < 2L$ for $512 \leq L \leq 1024$ and $L$ a multiple of 64.
2. Choose a prime numbers ($q$), where $q$ divisor of ($p - 1$), and $2159 < q < 2160$.
3. Compute $g$ as follows: $g = \frac{h(p-1)}{q} \mod p$, where $1 < h < (p - 1)$, and $g > 1$.
4. Choose a random integer $x$, with $0 < x < q$.
5. Compute $y$ as follows: $y = gx \mod p$.
6. Send $(p, q, g, y)$ to Bob (verifier).

**Signing and verifying algorithms:** Signing (sender - Alice)

Alice must do the following (referring to Steps 7 to 11 on Fig. 6):

7. Determine the message $m$ to be signed such that: $0 < m < p$.
8. Choose a random integer $k$, with $0 < k < q$.
9. Compute $r$ as follows: $r = (gk \mod p) \mod q$.
10. Compute $s$ as follows: $s = \frac{(k-1) \times (SHA-1(m) + x \times r)}{q} \mod q$.
11. The signature is $(r, s)$.
Send the signature \((r, s)\) and the message to the receiver.

- \(k^{-1}\) is a multiplicative inverse of \(k\) in \(\mathbb{Z}_q\).

**Verifying (receiver - Bob):** Bob must do the following (referring to Steps 12 and 16 on Fig. 6):

1. \(w = s^{-1} \mod q\).
2. \(u_1 = ((SHA-1(m)) \times w) \mod q\).
3. \(u_2 = (r \times w) \mod q\).
4. \(v = (g^{u_1} \times y^{u_2}) \mod p \mod q\).

- If \(v = r\), then the signature is verified.
- If \(v\) does not equal \(r\), then the message should be considered as invalid.

**RSA Digital Signature Scheme:** In the RSA digital signature algorithm, the private key is used to sign the message. The signed message will be sent to the receiver with the sender’s electronic signature. Fig. 7 shows the steps of the RSA digital signature algorithm. To verify the contents of digitally signed data, the recipient generates a new verification key from the signed message that was received, by using his public key, and compares the verified value with the original message value. If the two values match, then the message is verified and authenticated.

**Signature and verification algorithms:** Signature (sender - Alice): Alice must do the following (referring to Steps 6 to 8 on Fig. 7):

1. Determine the message \(m\) to be signed such that \(0 < m < n\).
2. Sign the message as follows, \(s = md \mod n\).
3. Send the signature \(s\) with the message \(m\) to Bob (receiver).

**Verification (receiver - Bob):** Bob must do the following (referring to Steps 9 and 12 on Fig. 7):

4. Obtain the keys \((d, n)\).
5. Obtain \(s, m\) from Alice.
6. Compute \(u\) as follows, \(u = se \mod n\).
7. Verify the message \(m\) as follows: \(m = u^{-1}\).

**MATERIALS AND METHODS**

**Digital Signature Based on the Mandelbrot and Julia Fractal Sets:** Mandelbrot and Julia fractal shapes (see Fig. 4 and 5) consists of complex number points, computed by the recursive functions as defined earlier in Subsection 2.1 (Equation 3, and 4). In this Section, with the aids of Fig. 8, we are going to explain in brief the proposed idea of the fractal digital signature scheme based on fractal set. The detail explanation of the proposed method will be given in the following Subsection. As mentioned earlier Mandelbrot and Julia properties were used in the design of the new proposed digital signature scheme.

In the proposed algorithm, sender and receiver must agree and use the 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 \(z_{de}\) and \(z_{de}\). Then Bob and Alice must exchange their public keys. Alice will obtain Bob’s public key, \(z_{de}\) and uses these values together with her private key and the plaintext, as inputs to the Julia function to produce the signature \(s\), which will then send with the message to Bob. Bob must obtain Alice’s public key, \(z_{de}\), the signature \(s\) and the message \(m\) from Alice which will be used as input values together with his own private key to Julia function, to verify the message \(v\).
Fig. 8: Fractal digital signature algorithm.

**Mandelfn and Juliafn:** The fractal based digital signature used a specific Mandelbrot function, Mandelfn and similarly, a specific Julia function, Juliafn. Fig. 9 and 10 show images which were generated from the Mandelfn and the Juliafn. In Mandelfn and Juliafn functions, we can substitute the function $f(\,)$ (see Equation 5 and 6) with well known functions such as $\sin(\,)$, $\cos(\,)$, $\exp(\,)$, etc. However, the value which is generated by Mandelfn must belong to the Mandelbrot set, and likewise, the value generated by Juliafn must belong to the Julia set. In the proposed scheme we set $f(\,)$ as shown by Equation 7 for Mandelfn function and Equation 8 for Juliafn function.

$$z \ (n + 1) = c \times f(z \ (n)).$$  \hspace{1cm} (5)

$$f(z(n)) = z_{n-1}, z_{n+1}, c, e \in C; n \in Z.$$  \hspace{1cm} (6)

**Proposed Fractal Digital Signature Scheme:** In the following we will describe the fractal digital signature scheme in details (see Fig. 11). The first step of the proposed scheme is to generate the private key and public key by using Mandelbrot function Mandelfn (Equation 7) and Julia function Juliafn (Equation 8).

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

$$z(n+1)=c\times f(z(n)); \ \ z(0)=y; \ y,c,z \in C; n \in Z. \hspace{1cm} (8)$$

As shown in Fig. 8 earlier, fractal digital signature scheme involves a sender and a 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 generate his public key by using Mandelfn function and send it to the receiver.

$$z_n d = z_{n-1} \times c^2 \times d; z,c,d \in C; n \in Z. \hspace{1cm} (9)$$

As indicates by Fig. 11, $z,d$ is the generated public key, generated by the receiver by executing Equation 9 (see Step 1 of Fig. 11). 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, $z_e$ (Step 3 from Fig. 11), generated by using Mandelfn. The Mandelbn is given in Equation 10.

$$z_k e = z_{k-1} \times c^2 \times e; z,c,e \in C; k \in Z. \hspace{1cm} (10)$$

In Steps 5 and 6 (Fig. 11), executing Juliafn by the sender will sign the message $m$ to produce the signature $s$. The signature $s$ with the message $m$, will then send to the receiver. Similarly (Fig. 11, Step 7), the receiver will execute Juliafn to produce $v$ which then is used to verify the message $m$ (Fig. 11, Step 8).

**Fig. 9: Mandelbn image with the sine function ($\sin(\,)$) \hspace{1cm} [19].

**Fig. 10: Juliafn image with the cosine function ($\cos(\,)$) \hspace{1cm} [20].

**Fig. 11: Fractal digital signature algorithm.**
After exchanging the public keys (see Steps 2 and 4 from Fig. 11) and executing the Julia function (Steps 5 and 7 from Fig. 11), sender Alice and receiver Bob had completed the proposed secured digital signature scheme. The process from Fig. 11, Step 5 is also being illustrated by Equation 11. The corresponding signature process, which is Step 7 of Fig. 11, is further illustrated by Equation 12.

\[
\begin{align*}
  s &= c^{x \times (z_d e)} e \times m; \\
  s, c, e, d &\in C; n, x, k \in Z; m \in R. \\
  \end{align*}
\]

(11)

\[
\begin{align*}
  v &= c^{e \times (z_e d)} d \times m; \\
  v, c, e, d &\in C; n, x, k \in Z; m \in R. \\
  \end{align*}
\]

(12)

Security Analysis of the Proposed Scheme: It is impossible to mount an attack on the proposed scheme 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 digital signature is through the chaos property of the fractal function which in this case depends on the private key selection. This is true since the generated public keys \((z_d, d)\) and \((z_e, e)\) produced by Mandelbrot function depends on the number of iterations, \( n \), as well as the variation constant, \( d \) and \( e \), which makes the Mandelbrot 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 \( 2^{128} \) possibilities for every values of \( n \) (or \( k \)) that is being brute force.

Working Example of the Proposed Scheme: Table 1 shows a working example of the proposed scheme. In this example each complex number is being represented by a 64-bit value. We use GMP [23] 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, see Equation 13 and Equation 14. The value \( x \) can be set to 0, if desired.

At the beginning, receiver and sender need to choose their private keys (see Table 1, Row 2). Then they have to calculate the corresponding public keys by using the Mandelbrot function, Julia, as shown by Table 1, Row 3. These values are \( z_d \) (receiver’s private key) and \( z_e \) (sender’s private key). Table 1, Row 4, shows both parties exchanging their public keys. Following this process is the calculation of the signature by using Julia function, Julia. Sender will produce the signature, \( s \), after executes the Julia function with input parameters \( k \) and \( d \) (sender’s private key) as shown by Table 1, Row 5. Table 1, Row 6, shows the signature of the hashed message \( M \) by using the security hash algorithm SHA-1 \((m = SHA-1(M))\), after the Julia is executed with parameters \( n \) and \( e \) (receiver’s private key) by the receiver.

A hash function is a reproducible method of turning the data relatively into a small digest. Hash function takes a random sized input message to produce a fixed sized digest. The outcome of the resulted digest is based on the hash technique used (SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512, which are designed by the National Security Agency (NSA) and published by the NIST as a U.S. government standard) [25].

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Sender</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Choose ( z = 0.159 - 0.129 ) from Mandelbrot set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Generate the private keys ( c \times z = 0.013875020867921 m = 0.002931905074 )</td>
<td>( n \times 10 )</td>
<td>( n \times 10 )</td>
</tr>
<tr>
<td>3</td>
<td>Generate the public keys ( z_d ) and ( z_e ) by using Mandelbrot ( \Delta x = 0.013875020867921 + 0.000929592692592 ) ( \Delta y = 0.000929592692592 )</td>
<td>( n \times 10 )</td>
<td>( n \times 10 )</td>
</tr>
<tr>
<td>4</td>
<td>Exchange the public keys ( z_d ) and ( z_e ) between sender and receiver ( \Delta x = 0.000929592692592 ) ( \Delta y = 0.000929592692592 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Sender must find ( \Delta e ) by using Julia and then send ( s ) to the receiver ( \Delta x = 0.000929592692592 ) ( \Delta y = 0.000929592692592 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Receiver must calculate ( \Delta s ) by using Julia to verify the signed message ( \Delta x = 0.000929592692592 ) ( \Delta y = 0.000929592692592 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Example of fractals based digital signature scheme.

RESULTS

The chaotic nature of the fractal functions ensures the security of the proposed scheme. However, to prevent a brute force attack, the choice of the key size becomes essential. The key space in fractal digital signature depends on the size of the key. For example in 128 bits key, there are \( 2^{128} \) possible key values, as is the case in the symmetric scheme. RSA and DSA keys are basically different from fractal keys. The RSA and DSA algorithms depend on large prime numbers (see Fig. 12) [23]. The 128-bit RSA and DSA keys space are 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. Hence, RSA and DSA keys size space are considerably smaller than the fractal key space for a given finite field [45]. Table 2 shows the key size for RSA, DSA and symmetric scheme, regarding to the resistance to brute force attacks. The keys space for RSA and DSA were calculated based on the number of primes in the finite field of \( Z_p \).
primes existed for particular key sizes. The computation
was based on Equation 13 [24].

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

**Performance evaluation based on equivalent key sizes for fractal and digital signature schemes:** We compare the performance of the fractal based digital signature scheme against the well known RSA and DSA digital signature schemes. Table 3 shows the performance of fractal, RSA and DSA digital signature approaches. These algorithms 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 [25] for primality test which was coded using C and GMP.

The comparison between fractal, RSA and DSA digital signature schemes show that fractal key digital signature performs better than RSA and DSA in general. Note that, in our implementation we increased the number of iterations \( k \) and \( n \) (see Fig. 11) proportionate with the key size to get suitable comparisons as shown by Fig. 13, 14, 15, and 16. As those Figure indicate, the fractal based digital signature scheme provides higher level of security at a much lower cost, both in term of key size and execution time.

The strength of the algorithm and the size of the key used, play the main role in the security of digital signature scheme. Fractal, RSA and DSA schemes can provide equal strength in security, all in terms of the algorithm complexity and the key size used. Nevertheless, fractal digital signature algorithm is more efficient than RSA and DSA since the algorithm used smaller key size and executes faster. Note that the performance of the propose scheme is amplified further in a multi-verification scenario, which is most likely to happen in real implementation.

![Fractal, RSA and DSA Key spaces](image)

Fig. 12: Key space comparison between fractal key, RSA and DSA digital signatures implementation.

![Fig. 14: Fractal, RSA and DSA keys generation time.](image)

![Table 2: Key space comparison between symmetric, RSA and DSA schemes.](image)

<table>
<thead>
<tr>
<th>Symmetric scheme</th>
<th>RSA/DSA Key size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key size</td>
<td>Time (Milliseconds)</td>
</tr>
<tr>
<td>Key generation</td>
<td>26</td>
</tr>
<tr>
<td>Signature</td>
<td>7</td>
</tr>
<tr>
<td>Verification</td>
<td>6</td>
</tr>
<tr>
<td>Key generation</td>
<td>32</td>
</tr>
<tr>
<td>Signature</td>
<td>10</td>
</tr>
<tr>
<td>Verification</td>
<td>9</td>
</tr>
<tr>
<td>Key generation</td>
<td>101</td>
</tr>
<tr>
<td>Signature</td>
<td>16</td>
</tr>
<tr>
<td>Verification</td>
<td>26</td>
</tr>
<tr>
<td>Key generation</td>
<td>145</td>
</tr>
<tr>
<td>Signature</td>
<td>30</td>
</tr>
<tr>
<td>Verification</td>
<td>26</td>
</tr>
<tr>
<td>Key generation</td>
<td>656</td>
</tr>
<tr>
<td>Signature</td>
<td>7945</td>
</tr>
<tr>
<td>Verification</td>
<td>3952</td>
</tr>
<tr>
<td>Key generation</td>
<td>60187</td>
</tr>
<tr>
<td>Signature</td>
<td>36054</td>
</tr>
<tr>
<td>Verification</td>
<td>36057</td>
</tr>
</tbody>
</table>

Table 3: Performance evaluation between Fractal based digital signature and RSA digital signature scheme.

![Fig. 14: Fractal, RSA and DSA keys generation time.](image)
**Fig. 15:** Fractal, RSA and DSA signature time.

**Fig. 16:** Fractal, RSA and DSA verification time.

**Fig. 13:** Overall time comparison between fractal, RSA and DSA digital signature algorithm time.

**ACKNOWLEDGMENTS**

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

**REFERENCES**


**CONCLUSION**

This paper shows the possibility of establishing a fractal based digital signature, derived from the logical connection between the Mandelbrot and Julia fractal sets. The security protection of the proposed fractal digital signature depends partially on the number of iterations needed to 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 will establish the needed complexity of the proposed scheme. As a result, the proposed signature scheme requires small key size and performs faster when compared to RSA and DSA.