Available online at www.sciencedirect.com

ScienceDirect

Journal of Electrical Systems and Information Technology 2 (2015) 296–313

www.elsevier.com/locate/jesit

Two-phase hybrid cryptography algorithm for wireless sensor networks

Rawya Rizk*, Yasmin Alkady

Electrical Engineering Department, Port Said University, Port Said, Egypt

Received 25 May 2015; received in revised form 25 November 2015; accepted 25 November 2015
Available online 8 December 2015

Abstract

For achieving security in wireless sensor networks (WSNs), cryptography plays an important role. In this paper, a new security algorithm using combination of both symmetric and asymmetric cryptographic techniques is proposed to provide high security with minimized key maintenance. It guarantees three cryptographic primitives, integrity, confidentiality and authentication. Elliptical Curve Cryptography (ECC) and Advanced Encryption Standard (AES) are combined to provide encryption. XOR-DUAL RSA algorithm is considered for authentication and Message Digest-5 (MD5) for integrity. The results show that the proposed hybrid algorithm gives better performance in terms of computation time, the size of cipher text, and the energy consumption in WSN. It is also robust against different types of attacks in the case of image encryption.

Keywords: Advanced Encryption Standard; Cryptography; Elliptic Curve; Message Digest-5; XOR-Dual RSA

1. Introduction

Wireless sensor networks (WSNs) have a great vulnerability due to the broadcast nature and dangerous environment. Correspondingly, there are many solutions for the security issues such as routing security (Fouchal et al., 2014; Hayajneh et al., 2013; Lasla et al., 2014; Farouk et al., 2014), secure localization (Yu et al., 2013), and key management and cryptography (Mary Anita et al., 2015). Cryptographic algorithms are an essential part of the security architecture of WSNs.

WSNs suffer from many constraints such as low battery life and small memory. Due to these limitations, WSN is not able to deal with traditional cryptographic algorithms. Two main problems related to security algorithms arise in WSNs. First, the overload that security algorithms introduce in messages should be reduced at a minimum; every bit

* Corresponding author. Tel.: +20 1009067030.
E-mail addresses: r.rizk@eng.psu.edu.eg (R. Rizk), engyasminalkady@yahoo.com (Y. Alkady).
Peer review under the responsibility of Electronics Research Institute (ERI).

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jesit.2015.11.005

the sensor sends consumes energy and, consequently, reduces the life of the device. Second, the memory size which refers to size of an encrypted message and the key size should also be reduced (Faye and Myoupo, 2013).

Various cryptographic algorithms have been proposed to achieve the security requirements such as Authentication, Confidentiality, and Integrity. Authentication means preventing unauthorized parties from participating in the network. Confidentiality means keeping information secret from unauthorized parties. Integrity ensures the receiver that the received data is not altered in transit by an adversary. Data authentication can provide data integrity also.

Encryption is the process of encoding information in such a way that hackers cannot read it. There are two types of encryption techniques; symmetric and asymmetric. Symmetric cryptography, also called private-key cryptography uses only one key for encryption and decryption. Common symmetric encryption algorithms include Data Encryption Standard (***DES***) (Singh and Supriya, 2013) and Advanced Encryption Standard (***AES***) (Burr, 2003). Asymmetric key cryptography, also called public-key cryptography requires special keys to encrypt and decrypt messages. Common asymmetric encryption algorithms include ***RSA*** (Frunza and Asachi, 2007) and Elliptic Curve Cryptography (***ECC***) (Kodali and Sarma, 2013). ***ECDSA*** – Elliptic Curve Digital Signature Algorithm (Balitanas, 2009) and ***ECDH*** – Elliptic Curve Diffie Hellman (Johnson et al., 2001) are based on ***ECC***.

Both symmetric and asymmetric cryptographic techniques offer advantages and disadvantages. Symmetric encryption techniques provide cost-effective and efficient methods of securing data without compromising security however; sharing the secret key is a problem. On the other hand, asymmetric techniques solve the problem of distributing the key for encryption however; they are slow compared to symmetric encryption and consume more computer resources. Therefore, the best possible solution for encryption is the complementary use of both symmetric and asymmetric encryption techniques. Hybrid encryption attempts to exploit the advantages of both kinds of techniques while avoiding their disadvantages. Hashing creates a unique, fixed-length signature for a message or data set. It is commonly used to check data integrity. Message Digest-5 (***MD5***) (Hossain et al., 2012) algorithm is a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value. It has been utilized in a wide variety of security applications.

In this paper, a hybrid cryptography algorithm is proposed and presented. It is designed to provide data security and users authenticity. It includes two phases work at the same time. In Phase I, it takes the advantages of the combination of both symmetric and asymmetric cryptographic techniques using both ***AES*** and ***ECC*** algorithms. In Phase II, ***XOR-DUAL RSA*** is used since it is more robust and cannot be easily attacked. In addition, Hashing is also used for data integrity using ***MD5*** to be ensured that the original text is not being altered in the communication medium. The proposed algorithm has high operation speed, high security performance and strong usability.

The organization of this paper is as follows: Brief overviews of related works of some existing protocols are presented in Section 2. The proposed hybrid encryption algorithm is introduced in Section 3. Sections 4 and 5 present the numerical results and the simulation results of the proposed algorithm in WSNs; respectively. Section 6 presents the implementation of the proposed algorithm in the image protection application. Finally, the main conclusion is presented in Section 7.

2. Related work

To date, many cryptography algorithms have been proposed but a lot of them are not complete suitable for WSNs. In Subasree and Sakthivel (2010), a security algorithm architecture is proposed by *Subasree*. In this algorithm, the given plain text is encrypted using ***ECC*** and the derived cipher text is communicated to the destination through secured channel. Simultaneously, the Hash value is calculated through ***MD5*** for the same plain text, and then encrypted with ***DUAL RSA.*** The encrypted message of this hash value is also sent to the destination. In this algorithm, it is difficult to extract the plain text from the cipher text, because the hash value is encrypted with ***DUAL RSA*** and the plain text is encrypted with ***ECC***. The new hash value is calculated with ***MD5*** and then it is compared with decrypted hash message for its integrity. By which, it is ensured that either the original text being altered or not in the communication medium. This is the primitive feature of this algorithm however, there are two disadvantages. First, the message is encrypted by asymmetric encryption algorithms (***ECC*** and ***DUAL RSA***) that are slow compared to symmetric encryption. Second, if an attacker determines a person's private key, his or her entire messages can be read.

In *Dubal* security algorithm architecture (Dubal et al., 2011), the given plain text is encrypted with a key that is generated by ***ECDH***. The encryption algorithm used is ***DUAL RSA***. The derived cipher text is appended with the digital signature for more authentications, generated by the ***ECDSA*** algorithm. Simultaneously, the Hash value of this encrypted cipher text is taken through the ***MD5*** algorithm. Then, the generated cipher text and the signature are communicated to the destination through secured channel. On the other side, i.e., on decryption end, the hash value is first evaluated and integrated. This is compared with the signature, for the verification of the digital signature appended at the end of message. Thereafter, the decryption of cipher text is done by ***DUAL RSA*** (Sun et al., 2007). Hence, the plaintext can be derived. In this algorithm, the intruder may be trapped by both the encryption by the ***DUAL RSA*** with the key generated by ***ECDH*** algorithm and the appended signature. Hence, the message can be communicated to the destination with highly secured manner. However, the used asymmetric encryption algorithms (***DUAL RSA*** and ***ECDH***) are slow compared to symmetric encryption. In addition, the attacker may read the messages if he/she can determine the private key.

A hybrid algorithm architecture is proposed by Kumar (2012). In this algorithm, the given plain text is encrypted first with ***AES*** algorithm and then with ***ECC*** algorithm. The hash value of this encrypted cipher text is taken through the ***MD5***. On the other side, the Hash value is first evaluated and integrated. Thereafter, the decryption of cipher text is done by ***AES*** and ***ECC*** decryption algorithms. Hence, the plaintext can be derived. This algorithm is a combination of both symmetric and asymmetric cryptographic techniques. However, the execution time of this algorithm is long because the plaintext is encrypted sequentially by both ***AES*** and ***ECC***.

In Ren and Miao (2010), a hybrid algorithm is proposed. In this algorithm, ***DES*** algorithm is used for data transmission because of its higher efficiency in block encryption, and ***RSA*** algorithm is used for the encryption of the key of the ***DES*** because of its management advantages in key cipher. During the process of sending encrypted information, the random number generator uses 64-bit ***DES*** session key only once. It encrypts the plaintext to produce the cipher text. On the other hand, the sender gets public key from public key management center, and then uses ***RSA*** to encrypt session key. Finally, the combination of the session key from ***RSA*** encryption (Boneh and Durfee, 2000) and the cipher text from ***DES*** encryption are sent out. This algorithm is considered weak since using ***DES*** with ***RSA*** affects the security level.

In Zhu (2011), a hybrid algorithm architecture is presented. The plaintext is encrypted with symmetric cipher algorithm, and the key and digital signature belonged to the symmetric encryption algorithm are encrypted with asymmetric key algorithm. The sender encrypts the plaintext with the key belonged to the ***AES*** algorithm. To ensure the security of the cipher algorithm and simplify the key management, the sender uses the key only once. The receiver obtains the original information after signature verification. This algorithm suffers from low security level since that the message is encrypted in a single phase which leads to less complexity.

From the previous studies it's shown that the security algorithms that depend on asymmetric encryption algorithms such as *Subasree* and *Dubal* have critical weakness points since they are slow compared to symmetric encryption algorithms and consume huge power to encrypt all plaintext by public key. In addition, if an attacker determines a person's private key, his or her entire messages can be read. Although *Kumar* security algorithm is a hybrid algorithm, it has a critical weakness point. It wastes a lot of time for encryption and decryption processes since it encrypts the plaintext sequentially first by ***AES*** and again by ***ECC***. *Ren* and *Zhu* security algorithms are suffering from low security level since using ***DES*** with ***RSA*** in *Ren* affects the security level and using a single phase in *Zhu* leads to less complexity. Due to the hard constrains of WSNs, they cannot be able to deal with the majority of such security algorithms.

3. The proposed two-phase hybrid cryptography algorithm (*THCA*)

In this section, the proposed *THCA* is presented. It introduces a new method of merging both symmetric and asymmetric techniques by performing two parallel phases. These phases avoid the disadvantages of the existing hybrid algorithms by achieving high security level without increasing the execution time.

3.1. Encryption process

In the Encryption, the plaintext is divided into n blocks B_i. Each block consists of 128 bits. Then, it is divided into two parts m_i (0: n/2–1) blocks, and M_i (n/2: $n-1$) blocks. If n is not an integer number and has a fraction, *THCA* algorithm uses padding with null for the last block to be 128 bits. The encryption process is divided into two phases.

In Phase I, The first n/2 blocks are encrypted using (***AES*** and ***ECC)*** hybrid encryption algorithm. ***ECC*** algorithm is used for protecting secret key since it is the highest secure public key algorithm. Moreover, according to the mathematical problem on which ***ECC*** can be solved by fully exponential rather than sub-exponential for other public key systems, ***ECC*** needs smaller key size than other algorithms and that refers to less memory size (Kodali and Sarma, 2013). It allows the communication nodes to handle a larger number of requests with the smallest number of dropped packet. Since that ***ECC*** consumes more power than symmetric algorithm, using ***AES*** algorithm reduces the power consumption and raises the system performance (Lenstra, 2001). When using ***AES*** with ***ECC***, we are able to save power, and achieve speed up to 25% for encryption and nearly 20% for decryption (Tillich and Großschädl, 2005).

The first *n*/2 blocks are encrypted as the following:

m_i is encrypted using ***AES*** by the key k_i which is the secret key of ***AES*** encryption algorithm with size 128 bits. k_i is encrypted by ***ECC*** to produce K_j with length *L*.

$$m_i = \sum_{i=0}^{i=n/2-1} (Bi) \quad 0 \leq i \leq n/2-1 \tag{1}$$

$$K_j = ECC_{enc}(TC_{PK},\ k_{i-1}) \quad 0 \leq j \leq L-1 \tag{2}$$

where ECC_{enc} is Elliptic Curve encryption function. It ciphers the input with trust center public key (TC_{PK}) which is used as a function to authenticate the key.

$$C_i = E_{AES}(K_j,\ B_i) \tag{3}$$

where E_{AES} is the ***AES*** encryption function.

Phase II is performed in parallel of Phase I in order to increase the security level without increasing the execution time. In Phase II, the remaining *n*/2 blocks are encrypted using ***XOR-DUAL RSA*** algorithm. ***DUAL RSA*** allows for extremely fast encryption and decryption that is at most four times faster than standard ***RSA***. The ***XOR*** Encryption algorithm is a symmetric encryption algorithm that uses the same key for both encryption and decryption. ***XOR-DUAL RSA*** algorithm guarantees developing a stronger algorithm, as follows:

$$M_i = \sum_{i=n/2}^{i=n-1} (B_i) \quad n/2 \leq i \leq n-1 \tag{4}$$

In this algorithm, two large prime numbers are chosen randomly; *p* and *q*. Then, $x = p \times q$, $\phi(x) = (p-1) \times (q-1)$. A number relatively prime to ϕ is chosen; *d*. Then, *e* is calculated such that $e \times d = 1 \bmod \phi(x)$, and Public key (*e, x*) is used for encryption.

$$R_i = (Bi)^e \bmod x \tag{5}$$

ASCII for (B_i) is get and converted to binary

$$L_i = \mathrm{ASCII}\,(B_i) \tag{6}$$

where L_i is a function used to convert message block to ASCII. R_i is a ciphered text using ***DUAL RSA***.

$$C_i = (R_i)\,XOR\,(L_i) \tag{7}$$

MD5 is applied to the cipher texts c_i and C_i. It is the best performance of hashing function security (Tillich and Großschädl, 2005).

$$d_i = MD5\,(c_i) \tag{8}$$

$$D_i = MD5\,(C_i) \tag{9}$$

At the final stage of the encryption process, the two *n*/2 blocks are integrated to generate cipher text of *n* blocks and it is sent to the sink node. The corresponding hash values (d_i and D_i) with size 128 bits for each one are concatenated and sent to the sink node at the same time.

$$Q = c_i + C_i \tag{10}$$

$$H = d_i + D_i \tag{11}$$

The encryption algorithm is described in Algorithm 1.

Algorithm 1.

The Proposed Encryption Algorithm.

```
Input: P (Plain text), k (secret key of AES encryption), s (128 bit size of block);
Output: Q (Cipher text), c_i (encrypted text using AES with ECC), C_i (encrypted text using XOR DUAL RSA), H (hashing value of cipher text);
1.   $n = P/s$;
2.   let $i = 0$;
3.     do{
4.     $m_i = \sum_{i=0}^{i=n/2-1}(Bi)$ first part of plain text;
5.     for($j = 0; j <= n-1; j{+}{+}$)
6.        {
7.        $K_j = ECC_{enc}(TC_{PK}, k_{i-1})$;
8.        }
9.     $c_i = E_{AES}(K_j, B_i)$;
10.      $d_i = MD5(c_i)$;
11.      $i{+}{+}$;
12.    }
13.      while($i<n/2$);
14.  $i = (n/2)$
15.  Let p and q two large prime numbers
16.  $x = p \times q$
17.  $\phi(x) = (p-1) \times (q-1)$
18.  Let d a relatively prime number to $\phi$
19.  $e \times d = 1 \bmod \phi(x)$
20.  Let (e, x) public key of DUAL RSA.
21.    do{
22.    $M_i = \sum_{i=n/2}^{i=n}(Bi)$ second part of plain text;
23.    $R_i = (B_i)^e \bmod x$;
24.    $L_i = ASCII(B_i)$;
25.    $C_i = (R_i)\ XOR\ (L_i)$;
26.    $D_i = MD5(C_i)$;
27.    $i{+}{+}$;
28.    }
29.    while($i < n$);
30.  $Q = c_i + C_i$;
31.  $H = d_i + D_i$;
```

3.2. Decryption process

In the decryption, the cipher text Q is divided into n blocks each block consists of 128 bits, Then, it is divided into two parts c_i (0: $n/2-1$) blocks and C_i ($n/2$: $n-1$) blocks. Hashing is used in order to identify whether the sink node receives the same cipher text or not. The hash values in both phases are compared. If they are the same, then the algorithm proceeds the decryption process. Else, it discards the message.

In the case of the hash values are the same at the source and sink nodes, the first $n/2$ blocks are decrypted using ***AES*** and ***ECC*** algorithms as follows:

$$c_i = \sum_{i=0}^{i=n/2-1}(Bi) \quad 0 \le i \le n/2-1 \tag{12}$$

$$k_i = ECC_{dec}(TC_{PK}, K_{j-1}) \quad \begin{array}{l} 0 \le i \le n/2-1 \\ 0 \le j \le L-1 \end{array} \tag{13}$$

The key of ***AES*** k_j with length L of bits is decrypted by ***ECC*** to produce k_i which has used to decrypt the cipher text using ***AES*** decryption scheme by D_{AES} (***AES*** decryption function).

$$m_i = D_{AES(K_j, c_i)} \tag{14}$$

m_i is the first part of the plain text. The remaining *n*/2 blocks are decrypted using ***XNOR-DUAL RSA*** algorithm as follows:

$$C_i = \sum_{i=n/2}^{i=n-1} (Bi) \quad n/2 \le i \le n-1 \tag{15}$$

Private Key (d, p, q) is used for decryption. To make decryption, first some parameters are computed $d_p = d \bmod (p-1)$, $d_q = d \bmod (q-1)$, $R_{pi} = R_i\, d_p \bmod p$, $R_{qi} = R_i\, d_q \bmod q$,

$$S_0 = (R_{qi} - C_{pi})p^{-1} \bmod q \tag{16}$$

$$S_i = R_{pi} + S_0\, P \tag{17}$$

ASCII for (C_i) is converted to binary.

$$W_i = ASCII\,(C_i) \tag{18}$$

where L_i is a function used to convert block of cipher text to ASCII.

$$M_i = S_i\, XNOR\, W_i \tag{19}$$

M_i is the second part of the plain text. At the final stage of the decryption process, the two *n*/2 blocks are integrated to produce plain text of *n* blocks.

$$P = m_i + M_i \tag{20}$$

The decryption algorithm is described in Algorithm 2.

Algorithm 2.

The Proposed Decryption Algorithm.

```
Input: $Q$ (Cipher text), $H$ (Hashing value of cipher text), $s$ (128 bit size of block), $L$ (key length), $d_i$, $D_i$, $K$ (encrypted key using ECC);
Output: $P$ (Plain text);
1.  $n = C/s$;
2.  let $i = 0$;
3.     do{
4.     $c_i = \sum_{i=0}^{i=n/2-1} (B_i)$ first part of cipher text;
5.     $d_{i''} = MD5(c_i)$;
6.     $D_{i''} = MD5\,(C_i)$;
7.     if $(d_i = d_{i''})\&(D_i = D_{i''})$
8.        {
9.          for($j = 0; j <= L - 1; j{+}{+}$)
10.             {
11.             $k_i = ECC_{dec}\,(TC_{PK}, K_{j-1})$;
12.             }
13.         $m_i = D_{AES}(K_j, c_i)$;
14.         $i{+}{+}$;
15.         }
16.       }
17.       while($i < n/2$);
18.   $i = n/2$;
```

Algorithm 2 (*Continued*)

```
19.  Give (d, p, q);
20.  $d_p = d \bmod (p - 1)$;
21.  $d_q = d \bmod (q - 1)$;
22.    do{
23.    $C_i = \sum_{i=n/2}^{i=n-1} (B_i)$ second part of cipher text;
24.    $R_{pi} = R_i d_p \bmod p$;
25.    $R_{qi} = R_i d_q \bmod q$;
26.    $S_0 = (R_{qi} - C_{pi}).\ p^{-1} \bmod q$;
27.    $S_i \vee= R_{pi} + S_0.\ P$;
28.    $W_i = ASCII\ (C_i)$;
29.    $M_i = S_i$ XNOR $W_i$
30.    i++;
31.      }
32.    while(i < n);
33.  $P = m_i + M_i$;
```

3.3. Strength of THCA

The strength of any cryptographic algorithm is based on several factors: the computational methods and the used key are two of them. In normal cryptographic approach the intruders may be able to identify cipher text patterns that are transmitted to the destination side. By analyzing the sequence of bit patterns; it is possible for the intruder to identify which type of encryption algorithm is used or they will identify the key used for encryption/decryption process.

In *THCA*, splitting the plain text improves the strength of the proposed algorithm. The intruder will not be able to identify which type of specific algorithm is applied to generate the cipher text. Thus, it is impossible to decrypt the cipher text. In addition, the two halves of the plain text are encrypted in parallel at the same time which reduces the time of both encryption and decryption.

When mixing ***AES*** with ***ECC*** in the first half of the plain text, the encryption process is done by symmetric algorithm (***AES***) which is faster than asymmetric algorithm. The secret key of ***AES*** is encrypted by ***ECC*** which is more complicated and then more secure. So that we obtain time reduction and power saving that are the advantages of symmetric encryption techniques in addition to the complexity which is the main advantage of asymmetric encryption techniques. Using ***XOR-DUAL RSA*** in the second half of the plain text allows our hybrid algorithm to be more robust and cannot be easily attacked.

In addition, Hashing is also used for data integrity using ***MD5*** to be ensured that the original text is not being altered in the communication medium. Then, the proposed algorithm has high operation speed, high security performance and strong usability. So, we can say that *THCA* is the true meaning of hybrid security algorithm.

4. Numerical results

In this section, the performance of the proposed *THCA* is measured in terms of size of the cipher text, time of encryption and decryption processes, and time complexity. The proposed algorithm is compared with the already existing algorithms that are presented in Section 2, *Subasree* (Subasree and Sakthivel, 2010), *Dubal* (Dubal et al., 2011), *Kumar* (Kumar, 2012), *Ren* (Ren and Miao, 2010), and *Zhu* (Zhu, 2011).

4.1. Size of cipher text

Table 1 describes the output of the encryption process. It shows the size of the cipher text in bytes. It is shown that *Kumar* algorithm has a largest size of cipher text whereas the other algorithms give a cipher text sizes that are equal or very close to the size of the plain text.

Table 1
Size of cipher text (byte).

Size of plain text (bytes)	*Subasree*	*Dubal*	*Kumar*	*Ren*	*Zhu*	*THCA*
609	609	673	846	602	609	641
25,615	25,615	25,645	35,142	25,610	25,615	25,647
35,080	35,080	35,192	48,226	35,070	35,080	35,112
61,386	61,386	61,486	84,340	61,369	61,386	61,418
184,162	184,162	184,262	253,008	184,143	184,162	184,194

4.2. Time of encryption and decryption processes

The encryption time is the time that an encryption algorithm takes to produce a cipher text from a plaintext. The decryption time is the time that an decryption algorithm takes to produce a plaintext from a cipher text.

Tables 2 and 3 show the time of encryption and decryption processes for different sizes of plain text; respectively. It is clear that, *THCA* achieves the least time for both encryption and decryption. This is due to that the plain text of the proposed algorithm is split into two different parts and these parts are encrypted and decrypted simeltaneously. The times shown in the tables are the maximum time of processing the two parts. It is shown that *Zhu* algorithm has the same time of *THCA* since that the message in this algorithm is encrypted in a single phase which leads to less security level. Achieving less encryption time results in highest throughput since that the throughput of encryption can be calculated as the total plain text over the encryption time. Then, *THCA* can realize the highest throughput.

4.3. Time complexity

In *THCA*, the time complexity of encryption process is calculated as follows: $Max[O(\log_2(n+1)+2n+\sqrt{n}+8)$ AND $O(\log(n^2)+\log(n)+3n+6)]$ that is equal to $O(\log(n^2)+\log(n)+3n+6)$ which is the integration of $O(\log(n^2)+\log(n)+n+2n+2+1+3)$. It consists of seven terms. The first term $(\log(n^2))$ denotes time complexity of ***DUAL RSA*** (Sun et al., 2007). The second and third terms $(\log(n)+n)$ refer to time complexity of ***XOR.*** The forth and fifth terms $(2n+2)$ refer to time complexity of two for loops. The sixth term (1) refers to time complexity of ***MD5***. It is a constant value since it is a probabilistic comparison algorithm (Erickson, 2008). The seventh term (3) refers to the variables. It can be neglected since it is very small with respect to n. Then, the total time complexity of encryption can be summerized to $O(\log(n^2)+\log(n)+3n)$ that can be reduced further to $O(n)$.

Table 2
Time of encryption (ms).

Size of plain text (bytes)	*Subasree*	*Dubal*	*Kumar*	*Ren*	*Zhu*	*THCA*
609	2063	2032	1500	1432	998	998
25,615	3683	6305	1518	1490	1022	1022
35,080	5651	15,643	1526	1468	1059	1059
61,386	15,351	120,608	4219	3019	3143	3143
184,162	105,889	198,700	5752	4970	3814	3814

Table 3
Time of decryption (ms).

Size of plain text (bytes)	*Subasree*	*Dubal*	*Kumar*	*Ren*	*Zhu*	*THCA*
609	1078	1016	966	756	562	562
25,615	1085	4053	972	821	713	713
35,080	1082	13,227	980	953	824	824
61,386	1197	13,227	991	864	891	891
184,162	2087	18,578	1099	1075	907	907

Table 4
Time complexity of encryption and decryption.

Algorithm	Encryption process	Decryption process
Subasree	$O(log(n^2) + 4n)$	$O(\log(2n^3) + 4n)$
Dubal	$O(\log(n^2) + \log 2(n) + \sqrt{n} + 4n)$	$O(\log(2n^3) + \log 2(n) + \sqrt{n} + 4n))$
Kumar	$O(\log 2(n+1) + \sqrt{n} + 4n)$	$O(\log 2(n+1) + \sqrt{n} + 5n)$
Ren	$O(\log(n^2) + \sqrt{n} + 4n)$	$O(\log(n^3) + \sqrt{n} + 4n)$
Zhu	$O(\log 2(2n+1) + \sqrt{n} + 4n)$	$O(\log 2(2n+1) + \sqrt{n} + 4n)$
THCA	$O(\log(n^2) + \log(n) + 3n)$	$O(\log(n) + \log(2n^3) + 2n)$

The time complexity of decryption process of *THCA* is calculated as follows: $Max[O(\log_2(n+1) + 2n + \sqrt{n} + 10)$ AND $O(\log(n) + \log(2n^3) + 3n + 7)]$ that is equal to $O(\log(n) + log(2n^3) + 2n + 6)$ which is the integration of $O(log(2n^3) + log(n) + n + 2n + 2 + 1 + 4)$. It consists of seven terms. The first term ($\log(2n^3)$) denotes time complexity of decryption of ***DUAL RSA***, the second and third terms ($\log(n) + n$) refer to time complexity of ***XNOR***, the forth and fifth terms ($2n + 2$) refers to time complexity of two for loops, the sixth term (1) refers to time complexity of ***MD5***, the seventh term (7) refers to time complexity of the variables. Then, it can be summerized to $O(\log(n) + \log(2n^3) + 2n)$ that can also be reduced further to $O(n)$.

Table 4 shows the time complexity of *THCA* compared with the existing algorithms for encryption and decryption process. Note that the time complexity shown in the table before final abbreviation that yields to $O(n)$ in all algorithms. However, it is shown that *THCA* has the least amount of processing time because of that both the two phases of the algorithm are encrypted and decrypted simultaneously.

5. Simulation Results of WSN

In order to prove the results of the proposed protocol, it is tested as the security protocol in WSN. The simulation is done using the network simulator NS2.

5.1. Simulation environment

It is assumed in the topology of the WSN that it consists of twenty nodes. The nodes are located randomly in the network. Different scenarios are assumed for transmission of data between different nodes. Each node must have the information about the other nodes present in the WSN. This information is first transmitted in the form of small packet. This packet contains the information about the source address. If any intermediate node receives a packet, it forwards this packet to the next neighboring node. When this packet reaches the final node it checks all the address present in this packet and then transmits reply back to the source node. The size of the packet increases gradually as the intermediate nodes add their address to the packet. After transmission of packet, every sensor node has the idea of the location of every other sensor node in the network. Therefore, the communication can be done from one node to the other node.

In some situations, the links present between the sensors nodes fail or the sensor nodes move from their actual location and thereby resulting in breakage of the link. In some other cases, improper packets may be propagated over the link between any two nodes. In addition, some packets may be dropped due to delay of execution time (time out). When such insecure packets are dropped, the link will not be used for a certain time and the network uses an alternate path.

5.2. Energy consumption

Because of the power source limitation of WSNs, all processes and communication protocols regarding sensor networks must minimize energy consumption so that sensor lifetime may be maximized. The evaluation of energy consumption considers both the energy consumed during the execution of cryptographic algorithms and the energy of communication. The energy required for the calculation of cryptography algorithm is simply the product of the average power consumption and the execution time of this algorithm. The execution time was determined through simulations.

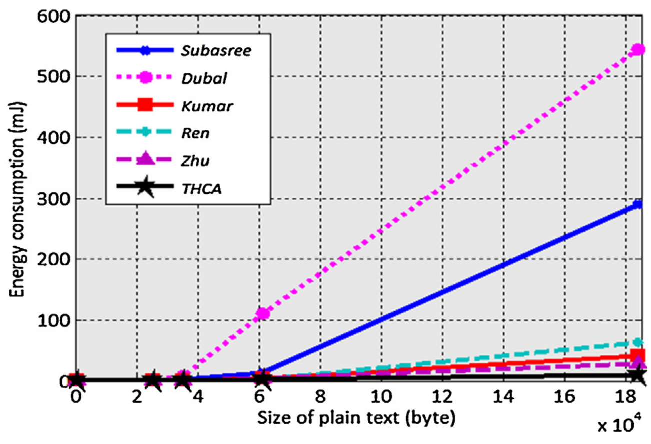


Fig. 1. Energy consumption of *THCA* with the comparison of the other algorithms.

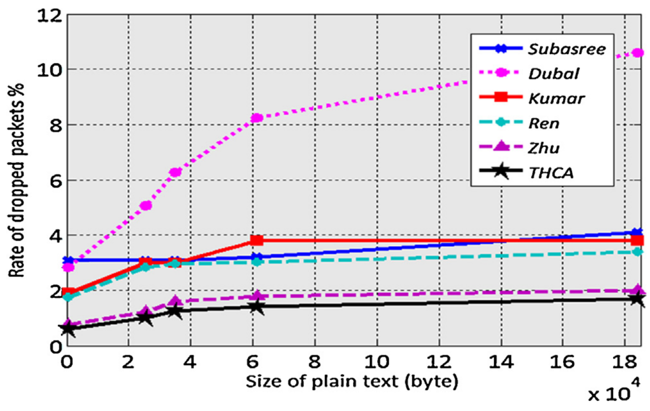


Fig. 2. Rate of dropped packets of *THCA* and the existing algorithms.

Communication energy depends on the distance between sending and receiving nodes and the time required for sending the cipher text, which it is proportional to the size of the plain text.

Fig. 1 shows the energy consumption of *THCA* compared to the other algorithms. It consumes about 10 (mJ) at 184,162 plain text size (byte) as opposed to 63, 31, 544, 290 (mJ) for *Ren*, *Kumar*, *Dubal*, and *Subasree*; respectively. It is shown that the *THCA* achieves the least energy consumption which is the demand to guarantee the lifetime of sensor networks.

5.3. Rate of dropped packets

Fig. 2 shows the rate of dropped packets. It is shown that the *THCA* achieves the least rate of packet dropping compared to the other protocols. This is due to that, the proposed *THCA* checks authentication using ***DUAL RSA*** and then protects the network from unsecured nodes. In addition, the number of dropped packets due to time out is decreased in the *THCA* since it has the least execution.

6. Implementation of *THCA* on images

In this section, the proposed *THCA* is tested on image encryption to prove its robustness against different types of attacks. It is applied on image protection application using magic cube theory. The magic cube (Bashir et al., 2012) divides the original image into six sub-images and these sub-images are divided into a number of blocks and attached to the faces of a magic cube as shown in Fig. 3. Then the attached image is fed to *THCA* which is applied to the pixels of the image to encrypt the scrambled image.

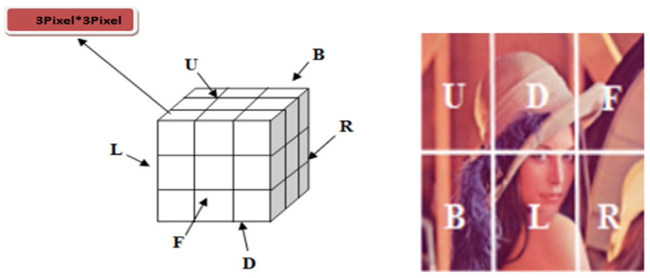


Fig. 3. Mapping the six sub-images on the magic cube faces.

6.1. *Description of the cube mapping*

1. The original image is resized to a size of $M \times N$ so that the resized image can be divided into six sub-images of the same size and with no overlapping.
2. The sub-images have the size $(M/3) \times (N/2)$. The six faces are marked as Up (U), Front (F), Right (R), Left (L), Down (D) and Back (B).
3. The six sub-images are divided into a number of blocks with the same number of pixels.
4. According to the implementation of *THCA*, The sub-images (U, D, F) are encrypted by first part of *THCA* which uses (***AES*** with ***ECC***) for encryption while sub-images (B, L, R) are encrypted by the second part which uses (***XOR-DUAL RSA***).
5. At the receiver side, the original image is retrieved by mapping of the six sub-images on the magic cube faces.

6.2. *Security analysis*

A good encryption procedure should be robust against all kinds of attacks. Some security analysis was performed on the proposed *THCA* in the case of using image, including the most important like statistical analysis and key space analysis.

6.2.1. *Statistical analysis*

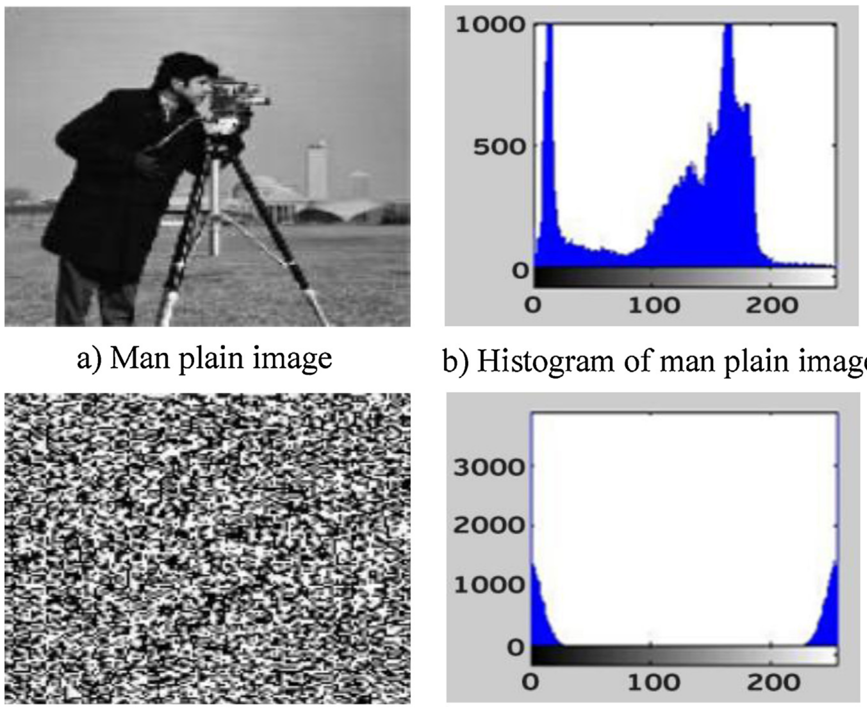
An ideal cipher should be robust against any statistical attack. To prove the robustness of the proposed algorithm, we performed statistical analysis by calculating the histograms and the correlations of two adjacent pixels in the plain image/cipher image.

6.2.1.1. Histograms analysis. The histogram of the image shows how pixels in the original images are distributed by graphing the number of pixels at each gray level (Abderrahim et al., 2012; Ahmed et al., 2007). A color histogram represents the number of pixels that have colors in each of a fixed list of color ranges that used for three-dimensional space like RGB channels. We calculated and analyzed the histograms of the several original and encrypted images that have widely different contents.

Fig. 4 shows the histogram of Man plain image and the Man encrypted image. It is shown that they are significantly different. Fig. 5 shows the histogram of Lena plain image. Fig. 5 (e), (g) and (i) illustrate the Histogram of Lena's image of Red, Green and Blue channels respectively. The histograms of the cipher images are shown in Fig. 5 (d), (f), (h) and (j). It is also shown that they are significantly different from that of the original images and have no statistical resemblance to the plain images.

It is clear that the histogram of the encrypted images are significantly different from the respective histogram of the original images and hence does not provide any clue to employ any statistical attack on the proposed *THCA* in the case of using image.

6.2.1.2. Correlation coefficient analysis. In addition to the histogram analysis, we also analyzed the correlation between two vertically adjacent pixels, two horizontally adjacent pixels and two diagonally adjacent pixels in plain



a) Man plain image

b) Histogram of man plain image

c) Encrypted image

d) Histogram of encrypted image

Fig. 4. Histograms of the plain image and the corresponding cipher image.

image/cipher image; respectively. First, we randomly selected 2000 pairs of two adjacent pixels from an image. Then, we calculated their correlation coefficient (r_{xy}) using the following formula (Abderrahim et al., 2012):

$$r_{xy} = \frac{Cov(x, y)}{\sqrt{D(x)}\sqrt{D(y)}} \tag{21}$$

where

$$Cov(x, y) = \sum_{i=1}^{N} (x_i - x')(y - y') \tag{22}$$

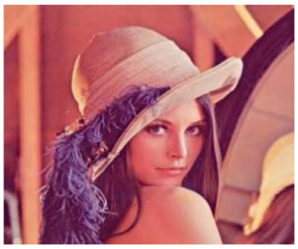
$$x' = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{23}$$

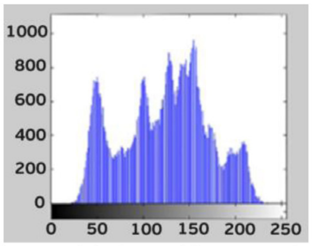
$$y' = \frac{1}{N}\sum_{I=1}^{N} y_i \tag{24}$$

$$D(x) = \sum_{i=1}^{N} (x_i - x')^2 \tag{25}$$

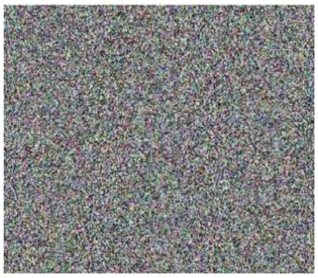
$$D(y) = \sum_{i=1}^{N} (y_i - y')^2 \tag{26}$$

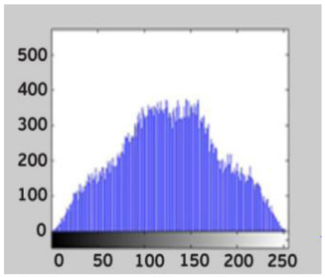
where x_i is the intensity of the ith pixel in original image, y_i is the intensity of the ith pixel in ciphered image, x' is the mean intensity of original image (the sum values divided by number of selected pixels), y' is the mean intensity of ciphered image, and N is the number of selected pixels.


a) Lena Plain-image

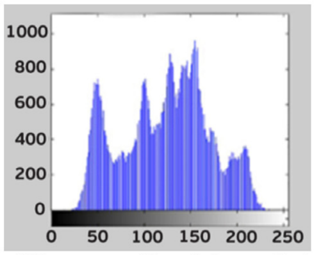


b) Lena Plain-image histogram after converted from RGB scale into gray scale

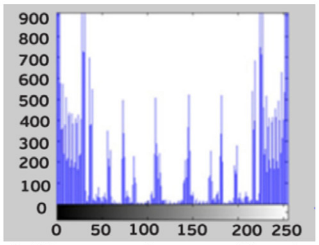

c) Encrypted image



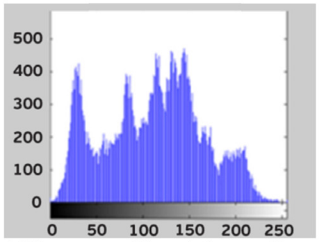
d) Histogram of Lena Encrypted image



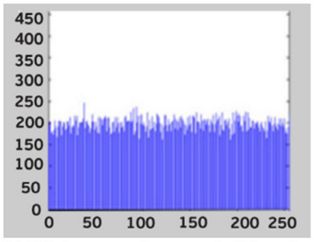
e)Histogram of Lena's image Red color component



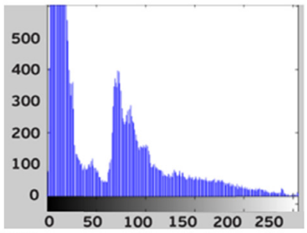
f) Histogram of encrypted Lena's image Red color component



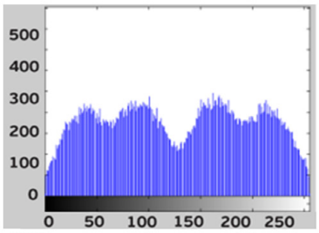
g) Histogram of Lena's image Green color component



h) Histogram of encrypted Lena's image Green color component



i) Histogram of Lena's image Blue color component



j) Histogram of encrypted Lena's image Blue color component

Fig. 5. Histograms of the colored plain image and the corresponding cipher image in RGB channels.

Table 5
Correlation analysis in man plain image/man cipher image.

Algorithm Direction of adjacent pixels	Plain image	Cipher image
Horizontal	−0.0468	0.2951
Vertical	−0.0488	0.09935
Diagonal	−0.0353	0.09614

Table 6
Correlation analysis in lena plain image/lena cipher image.

Algorithm direction of adjacent pixels	Plain image	Cipher image
Horizontal	0.9898	0.0303
Vertical	0.9805	0.0302
Diagonal	0.9769	0.0311

The correlation coefficient has the value 1 if the two images are absolutely identical, 0 or very near to zero if they are completely uncorrelated, or -1 if they are completely anti-correlated, for example if one image is the negative of the other. It is clear from Table 5 that the cipher Man image is highly independent of the Man plain image.

In Table 6, the correlation coefficients of two horizontally adjacent pixels are 0.9898 and 0.0303; respectively for both Lena plain image/Lena cipher image of *THCA*. Similar results for vertical and diagonal directions are obtained. It is clear from Table 6 that there is a negligible correlation between the two adjacent pixels in the cipher image. However, the two adjacent pixels in the plain image are highly correlated as shown in the Fig. 6.

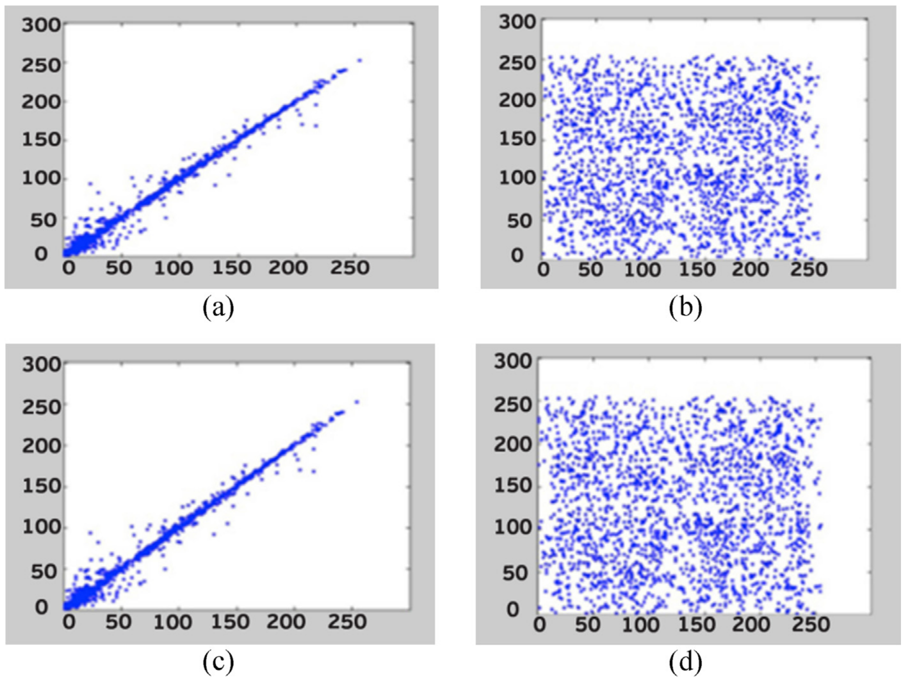


Fig. 6. Correlation of two adjacent pixels: (a) distribution of two horizontally adjacent pixels in the Man plain image, (b) distribution of two horizontally adjacent pixels in the Man encrypted image, (c) distribution of two horizontally adjacent pixels in Lena Plain image, (d) distribution of two horizontally adjacent pixels in Lena encrypted image.

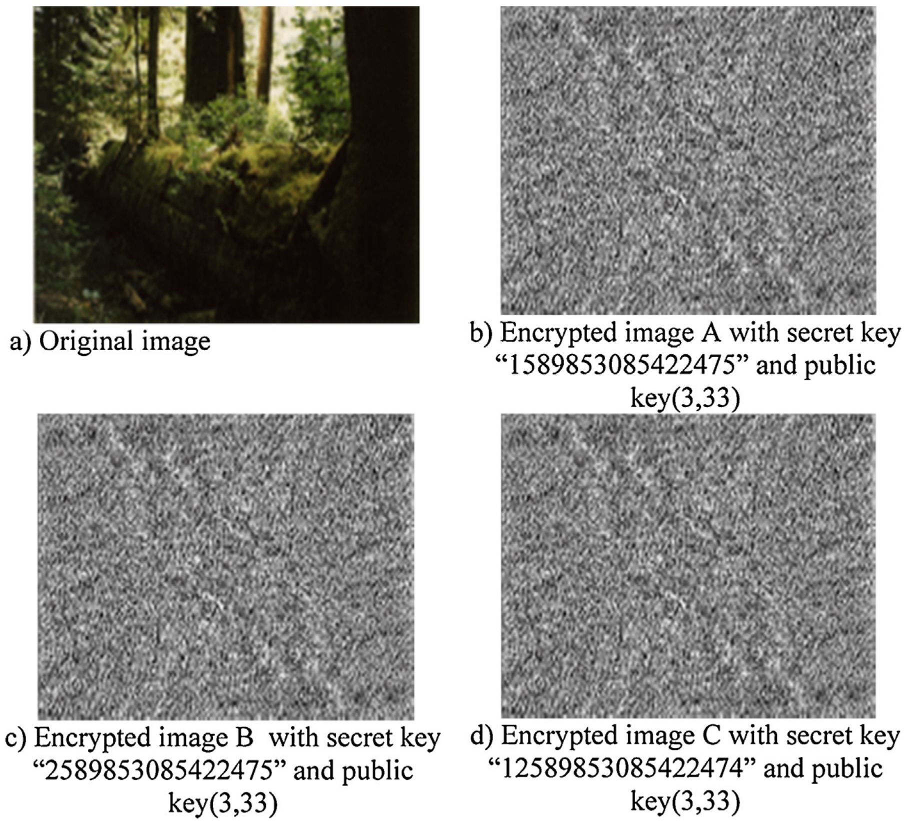

a) Original image
b) Encrypted image A with secret key "1589853085422475" and public key(3,33)
c) Encrypted image B with secret key "2589853085422475" and public key(3,33)
d) Encrypted image C with secret key "12589853085422474" and public key(3,33)

Fig. 7. Key sensitive Test (1) with *THCA*.

6.2.2. *Key space analysis*

A good image encryption algorithm should be sensitive to the cipher keys, and the key space should be large enough to make brute-force attacks infeasible. *THCA* has two types of keys, the first key is the secure key which is used to encrypt the first half of plain image and the second key is the public key which used to encrypt the second half of plain image. Then, for the proposed *THCA*, key space analysis and testing have to be carefully performed.The change of a single bit in the secret key or private key should produce a completely different encrypted image, which means that the cipher image cannot be decrypted correctly although there is only a slight difference between encryption and decryption keys. This guarantees the security of *THCA* against brute-force attacks to some extent. For testing the key sensitivity of *THCA*, the following steps were performed in Test (1) shown in Fig. 7:

1. First, the original image (Fig. 7(a)) is encrypted by using the test secure key "1551917990046475381" which is equivalent to "1589853085422475" (in hexadecimal) and public key (3, 33). The resultant image is referred as encrypted image A as shown in Fig. 7(b).
2. Then, the most significant bit of the secret key (in hexadecimal) is changed, so that the original secret key becomes, say "2704839494653322357" which is equivalent to "2589853085422475" (in hexadecimal) and the same public key (3, 33). The resultant image is referred as encrypted image B as shown in Fig. 7(c).
3. Then, the least significant bit of the secret key (in hexadecimal) is changed, so that the original secret key becomes, say "1551917990046475380" which is equivalent to "1589853085422474" (in hexadecimal) and the same public key (3, 33). The resultant image is referred as encrypted image C as shown in Fig. 7(d).
4. Finally, the three ciphered images A, B and C encrypted by the three slightly different keys are compared.

Table 7
Correlation coefficients between the corresponding pixels of the three different encrypted images obtained by using slightly different secret keys.

Image 1	Image 2	Correlation coefficient
Encrypted image A Fig. 7(b)	Encrypted image B Fig. 7(c)	0.0309
Encrypted image B Fig. 7(c)	Encrypted image C Fig. 7(d)	0.0358
Encrypted image C Fig. 7(d)	Encrypted image A Fig. 7(b)	0.0342

Fig. 7 shows the original image as well as the three encrypted images produced in the above steps. It is not easy to compare the encrypted images by simply observing these images. So for comparison, the correlation coefficients between the corresponding pixels of the three encrypted images have to be compared. For this calculation, the formula in (21) is used except that in this case x and y are the values of corresponding pixels in the two encrypted images to be compared. Table 7 shows the results of the correlation coefficients between the corresponding pixels of the three encrypted images A, B and C. It is clear from the table that there is no correlation exists among the three encrypted images even though these have been produced by using slightly different secret keys. Key sensitivity analysis shows that changing one bit in encryption key will result in a completely different cipher image.

Moreover, Fig. 8 shows the results of Test (2) that presents some attempts to decrypt an encrypted image with slightly different secret keys than the one used for the encryption of the original image. Fig. 8(a) and (b) shows the original image and the encrypted image produced using the secret key "1589853085422475" (in hexadecimal) and private key (7,3,11), respectively. Whereas Fig. 8(c) and (d) shows the images after the decryption of the encrypted image with the same secret key "1589853085422475" (in hexadecimal) and the slightly different secret key "1589853085422474"

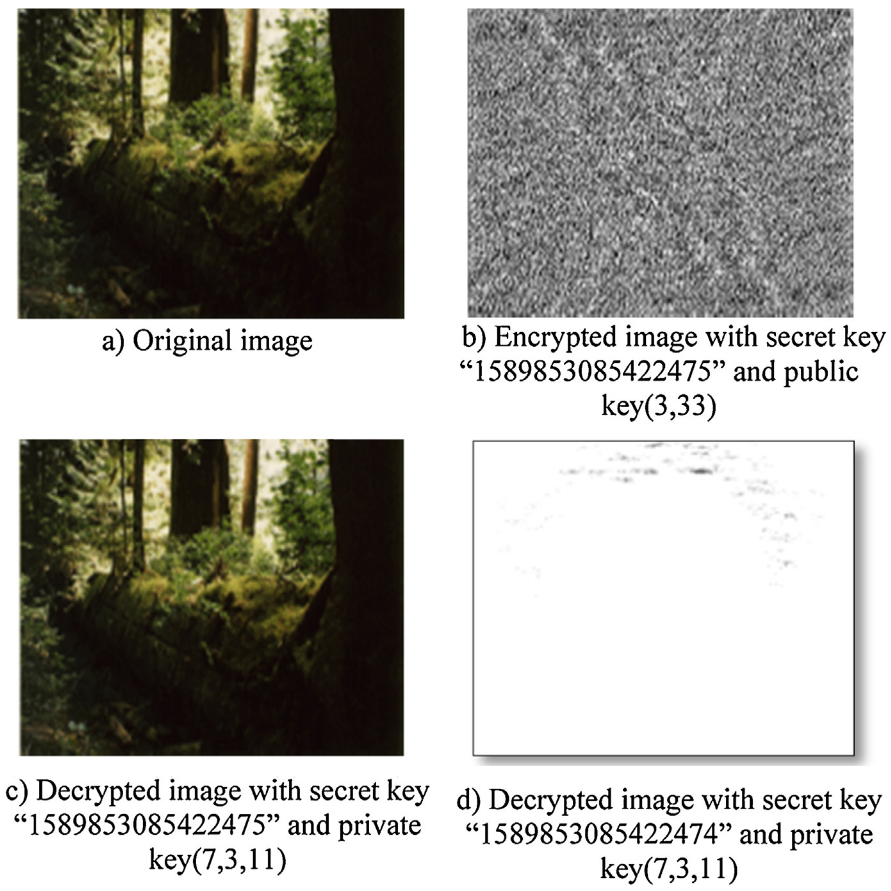

a) Original image
b) Encrypted image with secret key "1589853085422475" and public key(3,33)
c) Decrypted image with secret key "1589853085422475" and private key(7,3,11)
d) Decrypted image with secret key "1589853085422474" and private key(7,3,11)

Fig. 8. Key sensitive Test (2) with *THCA*.

(in hexadecimal); respectively. It is clear that the decryption with a slightly different key fails completely and hence the proposed *THCA* is highly key sensitive.

7. Conclusion

In this paper, a robust hybrid security algorithm for WSNs is proposed. It is designed in order to solve several problems as practical implementation, short response time, efficient computation and the strength of cryptosystem. The proposed *THCA* tries to trap the intruder by splitting the plain text and then applies two different techniques. First, it takes the advantages of the combination of both symmetric and asymmetric cryptographic techniques using both ***AES*** and ***ECC*** algorithms. Second, ***XOR-DUAL RSA*** is used since it is more robust and cannot be easily attacked. In addition, Hashing is also used for data integrity using ***MD5*** to be ensured that the original text is not being altered in the communication medium. The performance of *THCA* is compared with other existing security algorithms. It offers better security for a shorter encryption and decryption time and smallest cipher text size. Thereby, it decreases the processing overhead and achieves lower energy consumption that is appropriate for all WSN applications. The proposed *THCA* is implemented in the case of image encryption. It is shown that it is robust against different types of attacks.

References

Abderrahim, N., Benmansour, F., Seddiki, O., 2012. Integration of chaotic sequences uniformly distributed in a new image encryption algorithm. Int. J. Comput. Sci. (IJCSI) 9 (2).

Ahmed, H., Kalash, H., Farag Allah, O., 2007. An Efficient Chaos-Based Feedback Stream Cipher (ECBFSC) for image encryption and decryption. Informatica 31 (1), 121–129.

Balitanas, M., 2009. WiFi protected access-pre-shared key hybrid algorithm. Int. J. Adv. Sci. Technol. 12.

Bashir, A., Basari, A., Almangush, H., 2012. Novel Image Encryption using an Integration Technique of Blocks Rotation based on the Magic cube and the AES Algorithm. Int J. Comput. Sci. 9 (4).

Boneh, D., Durfee, G., 2000. Cryptanalysis of RSA with private key d less than N. IEEE Trans. Inf. Theory 46 (4), 1339–1349.

Burr, W., 2003. Selecting the advanced encryption standard. IEEE Secur. Priv. 1 (2), 43–52.

Dubal, M.J., Mahesh, T.R., Ghosh, P.A., 2011. Design of a new security protocol using hybrid cryptography architecture. In: Proceedings of 3rd International Conference on Electronics Computer Technology (ICECT), vol. 5, India.

Erickson, J., 2008. Hacking: The Art of Exploitation, ch. 7, 2nd ed. W. Pollock of Publisher, USA, pp. 393–450.

Farouk, F., Rizk, R., Zaki, F.W., 2014. Multi-level stable and energy efficient clustering protocol in heterogeneous wireless sensor network. IET Wirel. Sens. Syst. 4 (4), 159–169, http://dx.doi.org/10.1049/iet-wss.2014.0051.

Faye, S., Myoupo, J.F., 2013. Secure and energy-efficient geocast protocols for wireless sensor networks based on a hierarchical clustered structure. Int. J. Netw. Secur. 15 (1), 121–130.

Fouchal, H., Hunel, P., Ramassamy, C., 2014. Towards efficient deployment of wireless sensor networks. Secur. Commun. Netw., http://dx.doi.org/10.1002/sec.1059.

Frunza, M., Asachi, Gh., 2007. Improved RSA encryption algorithm for increased security of wireless networks. In: ISSCS International Symposium, vol. 2.

Hayajneh, T., Doomun, R., Almashaqbeh, G., Mohd, B.J., 2013. An energy-efficient and security aware route selection protocol for wireless sensor networks. Secur. Commun. Netw., http://dx.doi.org/10.1002/sec.915.

Hossain, Md.A., Islam, Md.K., Das, S.K., Nashiry, Md.A., 2012. Cryptanalyzing of message digest algorithms MD4 and MD5. Int. J. Cryptogr. Inf. Secur. (IJCIS) 2 (1).

Johnson, D., Menezes, A., Vanstone, S., 2001. The elliptic curve digital signature algorithm (ECDSA). Int. J. Inf. Secur. 1 (1), 36–63.

Kodali, R., Sarma, N., 2013. Energy efficient ECC encryption using ECDH. In: Emerging Research in Electronics, Computer Science and Technology, Lecture Notes in Electrical Engineering, vol. 248. Springer, pp. 471–478.

Kumar, N., 2012. A Secure Communication Wireless Sensor Networks Through Hybrid (AES+ECC) Algorithm, vol. 386. von LAP LAMBERT Academic Publishing.

Lasla, N., Derhab, A., Ouadjaout, A., Bagaa, M., Challal, Y., 2014. SMART: secure mlti-paths routing for wireless sensor networks. In: Ad-hoc, Mobile, and Wireless Networks, Lecture Notes in Computer Science, vol. 8487, pp. 332–345.

Lenstra, A., 2001. Unbelievable security matching AES security using public key systems. Adv. Cryptol. ASIACRYPT 2248, 67–86.

Mary Anita, E.A., Geetha, R., Kannan, E., 2015. A novel hybrid key management scheme for establishing secure communication in wireless sensor networks. Wirel. Pers. Commun., http://dx.doi.org/10.1007/s11277-015-2290-9.

Ren, W., Miao, Z., 2010. A hybrid encryption algorithm based on DES and RSA in bluetooth communication. In: Proceedings of the 2nd International Conference on Modeling, Simulation and Visualization Methods, China, pp. 221–225.

Singh, G., Supriya, 2013. A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. Int. J. Comput. Appl. 67 (19), 33–38.

Subasree, S., Sakthivel, N.K., 2010. Design of a new security protocol using hybrid cryptography algorithms. IJRRAS 2 (2), 95–103.

Sun, H., et al., 2007. Dual RSA and its security analysis. IEEE Trans. Inf. Theory, 2922–2933.

Tillich, S., Großschädl, J., 2005. Accelerating AES using instruction set extensions for Elliptic Curve cryptography. In: Computational Science and Its Applications – ICCSA, vol. 3481., pp. 665–675.

Yu, N., Zhang, L., Ren, Y., 2013. A novel D–S based secure localization algorithm for wireless sensor networks. Secur. Commun. Netw., http://dx.doi.org/10.1002/sec.909.

Zhu, S., 2011. Research of hybrid cipher algorithm application to hydraulic information transmission. In: Proceedings of International Conference on Electronics, Communications and Control (ICECC), China.