International Journal of Technology Research and Management
ISSN (Online): 2348-9006
Vol 6 Issue 1 January 2019

Security Solution for Wireless Sensor Network by Implementing TEA Algorithm over Android

Anurag Punde[1], Mr. Rajesh K. Chakrawarti[2]
PG-Scholar, Department of Computer Science &Engineering, Shri Vaishnav Institute of Technology & Science, Indore, INDIA [1]
Assistant Professor, Department of Computer Science &Engineering, Shri Vaishnav Institute of Technology & Science, Indore, INDIA [2]
anurag.2485nova@gmail.com[1], rajesh_kr_chakra@yahoo.com[2]

Abstract: *from the past 3 decade's need of security has greatly increased as more and more new threats areemerging day by day. As modern age concern security is an essential phenomenon that come in various aspects, even in wireless sensor networks. Since these are low power and lesser computational capability we need such algorithm or way that must be cheaper in every other aspect except security. Here we are giving the demonstration of the way to provide security for such kind of devices using Android mobile as wireless sensor node over TEA Algorithm.*

Keywords: *Wireless Sensor Network (WSN), Tiny Encryption Algorithm (TEA), Encryption, Autonomous Sensor, Decryption.*

1. INTRODUCTION

Wireless sensor network (WSN) is highly distributed and self-configuring network that has small, light weighted nodes. We can use Wireless Sensor network in many application such as to constant monitoring and detection of events, in military, battlefield, surveillance, forest fire, flood detection, patient monitoring and etc.[1] Ad HOC wireless network also uses the wireless nodes but it is different from the wireless sensor network. There are certain limitations in the Ad HOC network such as resource limitation (Memory, Power and Processing), Now having unique global IDs, more prone to failure and etc. Currently, wireless sensor networks are beginning to be deployed at a hastened pace. It is not perverse to expect that in the next 10-15 years the world may be concealed with wireless sensor networks with the facility to access them using the Internet. This can be considered as the use of Internet becoming a physical network. A wireless sensor network is a collection of nodes organized into a cooperative network [2]. Each node comprises processing capability (one or more microcontrollers, CPUs or DSP chips), might include various types of memory (program, data and flash memories), have a RF transceiver (usually with a single omni directional antenna), have a energy source foundation (e.g., solar cells and batteries), and contain a range of sensors and actuators. The nodes commune wirelessly and regularly self-organize after being deployed in an ad hoc fashion. Systems of 1000s or even 10,000 nodes are estimated. Such systems can modernize the way we live and vocation. In current scenario this technology influences each and every area of life as we can encounter any kind of sensor in our daily life which are made to simplify many work which can take may be a life time to be done by a human. As crucial is this technology it is highly necessary to make it secure but only putting it in some secure environment is seemingly not enough as these devices used to perform communication task also. For that we have to provide some security mechanism so that these can be protected even when we are not stand-guarding it. For such concern we should provide such an dedicated algorithm which can be used for such purpose. The Tiny Encryption Algorithm (TEA) is basically a symmetric (private) key encryption algorithm created by David Wheeler and Roger Needham of Cambridge University. They published their work in 1994. This particular Algorithm was designed for basically easiness in understanding and implementation while considering performance, while looking for encryption strength on equivalence with more sophistically complicated and resource-intensive algorithms such as DES (Data Encryption Standard). Wheeler and Needham summarize their whole work as follows: "it is anticipated that this Algorithm can easily be translated into most of the modern languages. It takes little set up time and does enough rounds to make it secure. It can replace Data Encryption Standard (DES) and is tiny enough to write into almost any programming language“[3]. Academic research, one can learn of TEA’s relative merits.

2. PROBLEM DOMAIN

In the light of the overall computational power and even low level of battery supplied power consumption it would be torture for these poor devices to implement any other security measures. So in this regard we need such a procedure which is cheap in term of computational process, easy to implement, needless resources and provide tight security. Generally these systems suffers lots of security problems like DoS attacks, Black Hole (Sink) Attack, Worm Hole Attack, etc. hacking of communication lines etc. as these nodes can be far away from each other geographically. As these devices carry and transfers lots of important information and even need high power consumption at transmission time, any procedure which takes lesser time, energy and resources will be preferable. If we implement any algorithm directly in these devices it will be hard coded and making changes in such condition will be costly and nearly to impossible.

3. SOLUTION DOMAIN

In this section we are proposing the algorithm for encryption and decryption. As we also dive into the matter like understanding what Java is, what is the process behind the scene and how it is implemented over android platform? We will try to answer these simple questions.

4. THE ALGORITHM

British researchers from Cambridge University proposed extremely simple encryption algorithm, the TEA (Tiny Encryption Algorithm) It based on an alternative application of a large number of iterations with XORs and additions, rather than on preset tables. So it can achieve better performance with smaller code size and less complexity than standard encryption algorithms. The TEA encrypts 64 data bits at a time using a 128-bit key, which is the strongest encryption so far.

A. **Encryption Routine**

Figure 1 shows the structure of the TEA encoding routine. The inputs to this encoding routine are an original data block called Plaintext and a key K .The plaintext P = (Left [0], Right [0]) and the encrypted or cipher text C = (Left [64], Right [64]). The plaintext block is divided into two equal parts, Left [0] and Right [0]. Each half is used to encode the other half over 64 rounds of processing and then combined produce the cipher text block.

- Each round ihas inputs Left[i-1] and Right[i-1], derived from the earlier round, as well as a sub key K[i] derived from the 128 bit overall K.
- The sub keys K[i] are different from K and from each other.
- The constant delta = $(\sqrt{5} - 1) * 2^{31}$ = 9E3779B9, is derived from the golden number ratio to ensure that the sub keys are distinct and its precise value has no cryptographic significance.
- The round function differs slightly from a classical Fiestel cipher structure in that integer addition modulo 2^{32} is used instead of exclusive-or as the combining operator.

Algorithm:

```
void code (long* v, long* k)
{
unsigned long y = v[0], z = v[1], sum = 0, /* set up */
delta = 0x9e3779b9, n = 32 ; /* a key schedule constant */
while (n-->0)
{
 /* basic cycle start */
sum += delta ;
y += (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
z += (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ; /* end cycle */
}
    v[0] = y ; v[1] = z ;
```

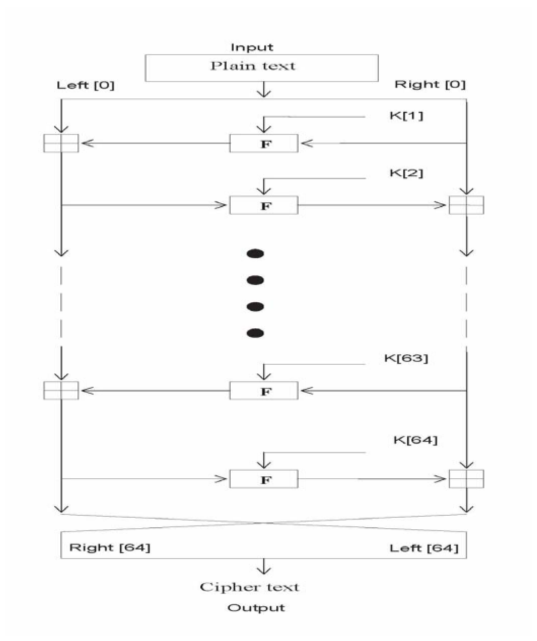


Figure 1: Abstract structure of TEA encryption routine

B. Decryption Routine

Decryption is essentially the same as the encryption process; in the decode routine the encrypted text is used as input to the Decode routine, but the sub keys K[i] are used in the reverse order. Figure 2 presents the assembly of the decryption routine. The in-between value of the decoding process is equal to the corresponding value of the encoding process with the two equal parts of the value swapped. For example, if the output of the nth encryption round is ELeft[i] || ERight[i] (ELeft[i] concatenated with ERight[i]). Then the corresponding input to the (64-i)th decryption round is DRight[i] || DLeft[i] (DRight[i] concatenated with DLeft[i]). After the last iteration of the encryption process, the two equal parts of the output are exchanged, so that the cipher text is ERight [64] || ELeft [64], the resultant outcome of that round is the final encrypted or cipher text C. Now this cipher text is used as the input to the decryption routine. ERight [64] || ELeft [64] is the input for round 1, which is equal to the 2-bit swap of the output of the 64thround.

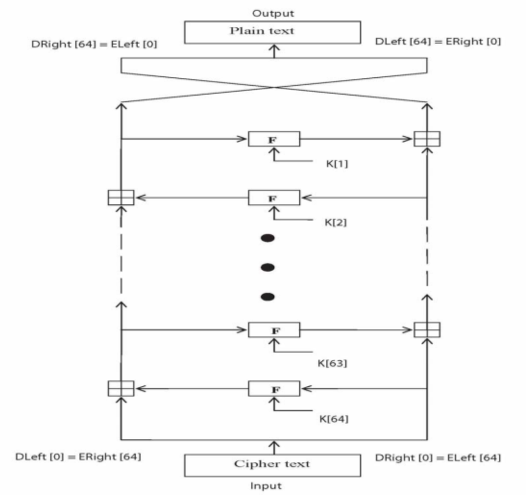


Figure 2: The abstract structure of TEA decryption routine.

JAVA: Java is an open source, platform independent language which is highly suitable for various kind of project development

Android platform

Android is a SDK facility provided by Google. It is firstly launched as a developmental SDK with the eclipse IDE for app developers around the world. Afterwards google stop its support to the eclipse SDK of android development when Intellij Company launched its own IDE called Android Studio. As newer versions of android are coming every year android SDK supporting library is also increasing in sizes. Developers who want to cover up most of the devices should start with the most possibly the lower version of SDK. Android studio can also be connected with genymotion which is a lightweight emulator for running the android applications as emulator which is embedded with android studio takes lots of time and resources.

MS-ACCESS

MS-ACCESS is a database functionality given in the Office suite developed by Microsoft Company. It is very powerful and user friendly software which is also widely used database. It is simply GUI based software where user can perform all the operations associated with the database functionality. SQL is the basic query language which is used to perform operations on the database, which is the same as the other database softwares. To connect the MS-ACCESS to any other language development platform it requires

JDBC/ODBC connection which can be done easily and is the same functionality provided by Microsoft by default. Any other database software can be used in place of it but we have selected it especially considering its user friendliness and simpleton.

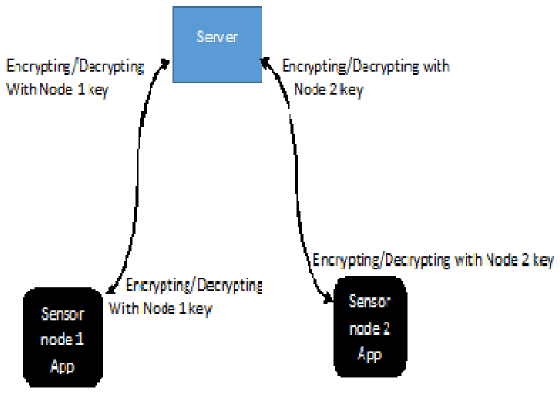

Server
Encrypting/Decrypting With Node 1 key
Encrypting/Decrypting with Node 2 key
Encrypting/Decrypting with Node 2 key
Encrypting/Decrypting With Node 1 key
Sensor node 1 App
Sensor node 2 App

Fig 3: The Actual Communication between 2 Wireless sensor nodes using deployed App

The Process

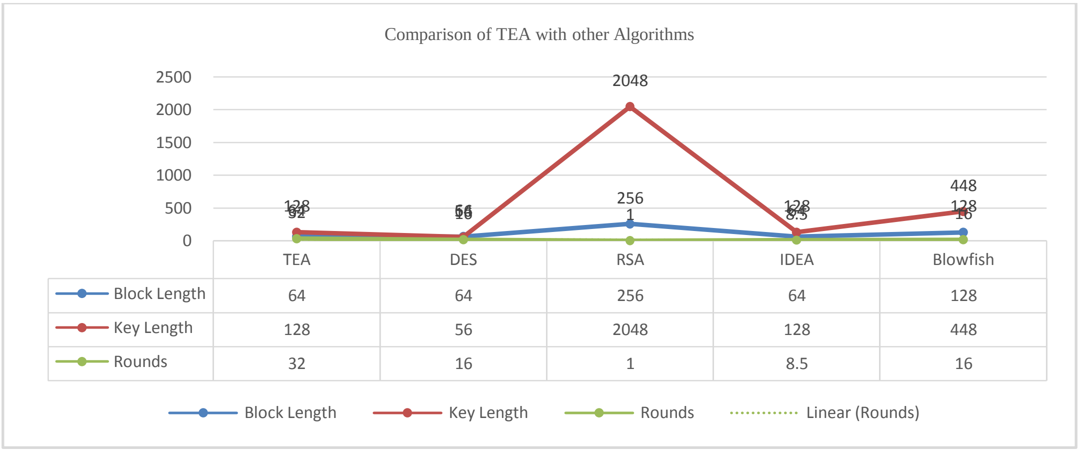
First of All both the participating Nodes have to register themselves with the server by entering an ID and password. The Communication process will commence as the user or system will open this deployed App/solution. After preparing the out-going message which is to be sent to other node this App/Solution will encrypt the message using the key provided by the user at the time of registration with the app. After the encryption it will go to the server with which both the nodes were registered, here the server will decrypt the message with Node1's key and encrypt it with Node2's key and send it to node 2. The App installed on Node2 will take the message and decrypt it with Node2's key and with that it can read the message send to it by Node1 Securely. This Process will run same for Node2. The depiction is given below for the same.

5. COMPARISON

On the internet. Everything now we share with the help of internet. The two major concerns regarding the sharing of data and information is to maintain the data and information confidential and secure from unauthorized users to access it. As the eavesdropping is done usually while transmitting the confidential information by an attacker, from very beginning various algorithms concerning this issues were designed which are our main focus of comparing our algorithm with those in which we found that the key sharing between the two parties unlikely essential. In this comparative study, we are concerning about the algorithms that provide the confidentiality and security to our data. So that, the unauthorized users cannot easily fetches the information. In this manner we can prevent our data. A formula used to turn ordinary data, or "plaintext," into a secret code known as "cipher text." Every one of these algorithm uses a sequence of bits or string known as a "key" to perform the calculations. The larger the key (the more bits), the bigger the number of potential patterns can be produced, thus making it harder to break the code and descramble the contents. Most of the encryption algorithms uses the block cipher technique, which codes fixed blocks of input that are typically varies from 64 to 128 bits in size. Several Algorithms use the stream technique, which works with the continuous stream of input. Many of the algorithms were suggested by many researchers such as 3DES algorithm, Blowfish, IDEA, DES, summer and many more. All of these algorithms works on some dataset. Despite of it, they have their own way to generate the key though which they convert the plaintext or the original text in to some different form called cipher text. The process of converting the original text to some unreadable text is known as encryption. At the receiver ends, when receiver receives the unreadable message from source than it required same key to decode it. It means the same key will be used to decrypt the unreadable message to the original text. This type of process is known as decryption. Every algorithm will generate the key and it is definitely uses the different concept to not only generate the key but also they examine how to share the key. Various factors were involved in the comparative study of various algorithms like block length, Key length, Rounds taken by the algorithm to encrypt and decrypt the given data and to produce the cipher, time taken by the algorithms etc. The study is been shown by the comparison tables and charts.

Table 1:Characteristic sizes of the focused ciphers

Cipher	**TEA**	**DES**	**RSA**	**IDEA**	**Blowfish**
Block Length	64	64	256	64	128
Key Length	128	56	512-2048	128	32-448
Rounds	32	16	1	8.5	16



	TEA	DES	RSA	IDEA	Blowfish
Block Length	64	64	256	64	128
Key Length	128	56	2048	128	448
Rounds	32	16	1	8.5	16

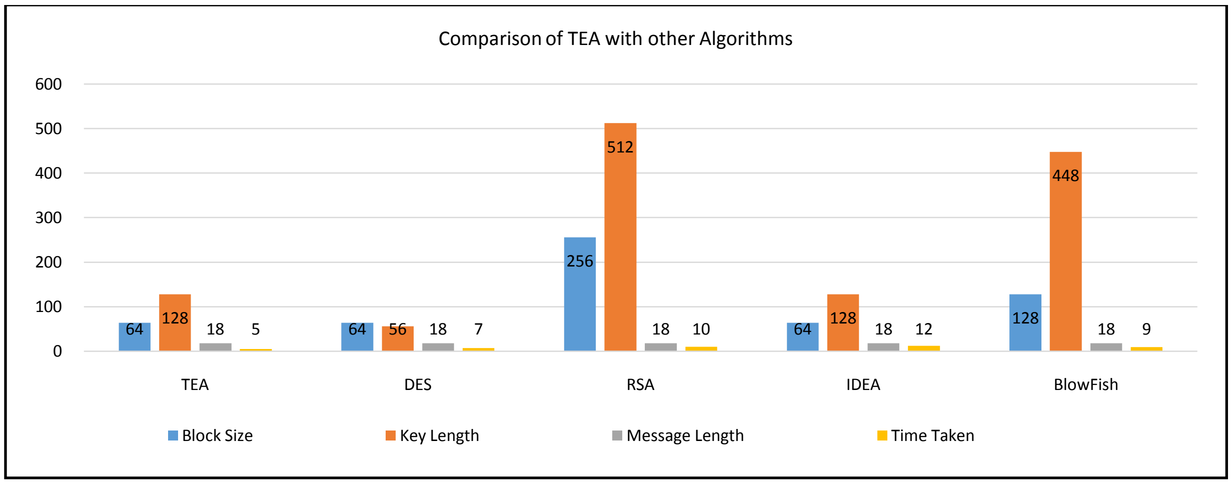
Fig4: Comparison chart of TEA with other algorithms

Table 2. Execution Time Taken in Encryption/Decryption by various Algorithms

S.No.	Cipher	Block Size	Message Length	Time Taken (Encryption/Decryption) (In Seconds)
1	TEA	64	16	5
2	DES	64	16	7
3	RSA	256	16	10
4	IDEA	64	16	12

5	Blowfish	128	16	9

Following charts were drawn from the table given above:



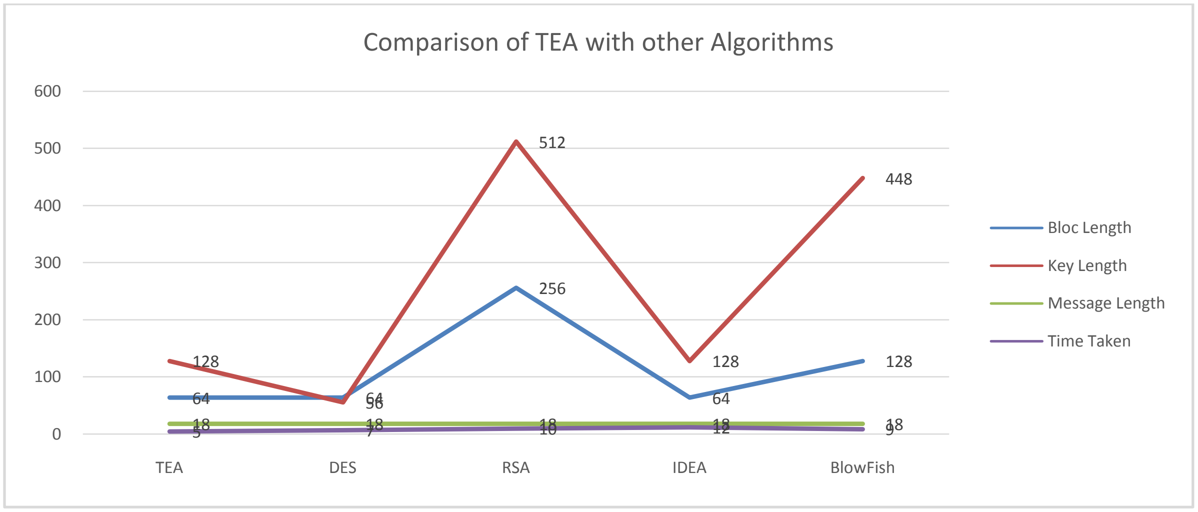


Fig 5: Comparison chart of TEA with other algorithms

6. RESULT & DISCUSSION

It is being observed that at firstly the sensor nodes weren't able to achieve the required security as the mechanism for the same was highly costly as per system resources. After analysis it is found that TEA algorithm is satisfying the required parameters by following which it can be applied to the sensor nodes without slightest difficulty. The cipher text is strong enough and requires less computational power so it will not be burden on the sensor nodes having low power issue. The Android system was taken due to their high availability as well as it will be cheaper to maintain and illustrate the implementation as we might needed to modify the code during development. Taking Java is adding another advantage like security which is highly recommended. The most important factor is the tiny encryption algorithm plays a crucial role as we can see that it is providing good results in terms of security. As this whole process is running on the server side there will be no computational power consumption issue on the client side sensor nodes.

As per the result extracted from the implementation here we can see that right now only 16-18 digits of data can be decrypted. As it may be required that sometimes large amount of data or longer data must be sent from one sensor node to another sensor node in this kind of condition it is necessary that we have to modify the code so that the requirement can be fulfilled. As this algorithm is available it can be easily hard coded to the sensor node at chip level. As this whole process is making communication between two sensor devices it is necessary that communication medium should also be reliable for which separate mechanism should be deployed. Furthermore for raising the security for such devices this algorithm can be infused with any other lightweight encryption algorithm. If possible than this solution can be tested on various other platforms.

7. CONCLUSION & FUTURE WORK

The simple conclusions are: yes, TEA is easy to implement, fast, efficient. If implemented and applied properly, TEA can be an excellent choice, particularly for encryption and decryption of small, short-term data in resource constrained devices. I have personally seen simple XOR algorithms used for "encryption" in scenarios where the programmers "do not have sufficient time, assets, or necessity for elaborate encryption algorithms." Tiny Encryption Algorithm would undoubtedly be a better substitute.

8. LIMITATIONS

This Research work has some limitations like when applying this project or deploying it, it is necessary that both the systems must be connected to the server for effective communication. As the project is still in its early phase there is a limitation on how long or lengthy messages can be send. Particularly as this Application is developed taking Android Platform in consideration some time might be given for developing it for other platforms. Last but not the least this is not currently deployed hard-coded as still some changes required but this can also be done easily.

REFERENCES

[1] vikram reddy andem, a cryptanalysis of the tiny encryption algorithm, year 2003.
[2] john a. stankovic, wireless sensor network, published in the year june 19, 2006.
[3] derekwilliams, the tiny encrytion algorithm, published in the year april 26, 2008.
[4] scheier, bruce. a self-study course in block-cipher cryptanalysis. cryptologia, vol. 24(1). january 2000.
[5] steil, michael. 17 mistakes microsoft made in the xbox security system. october, 2005.
[6] steil, michael. the hidden boot code of the xbox. xbox-linux. august, 2005.
[7] a. cerpa, j.wong, l. kuang, m. potkonjak, and d. estrin, statistical model of lossy links in wireless sensor networks, ipsn, april 2005.
[8] I. elson, l. girod, and d. estrin, fine-grained network time synchronization using reference broad- casts, osdi, december 2002.
[9] G. ganeriwal, r. kumar, and m. srivastava, timing-sync protocol for sensor networks, acmsensys, november 2003.