International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-6, March 2020

A Lightweight Cryptographic Algorithm using Pseudo Stream Cipher and Trigonometric Technique with Dynamic Key

Bhaskar Prakash Kosta, Pasala Sanyasi Naidu

Abstract: ***IoT (Internet of things) equipments is used heavily, but they are poor in security issue and security can be pierced. Also the message that the IoT device transmit is the main cause for their security lapse. that the IoT devices send may lead to the breach of users' privacy. To make the communication secure between IoT nodes and servers, a lightweight cryptographic algorithm using pseudo stream and trigonometric function with dynamic key is proposed. This algorithm works in different phases, the first is mutual authentication followed by key synchronization and then a trigonometry function for encryption and decryption, with updation of key after specific time period. The algorithm assures that the IoT node will not be overloaded and the security is enhanced by reducing the chance of cryptanalysis. The mutual authentication, session key synchronization and updation of session key are completed through several encrypted communication. Here, the key length and update cycle are variable to prevent attack. We compare the security and performance for mutual authentication, with some light weight authentication scheme and performance of encryption algorithm are compared to other algorithm like Hill Cipher, RC4, RSA. According to analysis the proposed mutual authentication and secret key for session synchronization can provide more security features with low over head of communication which is suitable for protect communication security of IoT with limited resource and power. The encryption decryption algorithm provides better performance. Trigonometric concept is used in the design of encryption decryption algorithm.***

Keywords: lightweight cryptographic scheme, internet of things, pseudo stream cipher.

I. INTRODUCTION

Tremendous assortment of IoT gadgets is there and they have part of utilization in day by day life, industry creation which pulled in numerous specialists to do their examination in the field of IoT. An IoT gadget does numerous things however when the information moves or dwells in hubs (IoT or other hub) its security is confronting serious difficulties [1]. Those IoT hubs which have constrained processing assets and low force. Security for information which it needs to transmit is issue and an outsider can undoubtedly break the security as it needs compelling assurance [2]. For the most part IoT gadgets are utilized to gather information for instance, the temperature sensors will record the present condition temperature or internal heat level. A few information which the IoT gadget gather may prompt the infringement of client security thought there are numerous cryptography calculation yet they are not reasonable for IoT hubs as a result of asset and force restriction additionally many light weight calculation are accessible however they require equivalent computation capacity on the two sides of correspondence. For some IoT gadgets vitality and processing assets are the restriction and bringing down this overhead is significant. Stream cipher use XOR activity to perform encryption and unscrambling, give better correspondence security however key administration gives all the more overhead (increasingly number of emphasis on the two sides)

We propose a mechanism to simplify key management. A lightweight cryptographic algorithm using pseudo stream cipher and trigonometric function with dynamic key is proposed which performs mutual authentication , session key synchronization and data encryption using trigonometric function with the minimum overhead of nodes . This scheme includes three phases i.e. mutual authentication phase, session key synchronization and data encryption phase and session key update phase. Here the server generates session key and synchronizes the key with IoT node by three times communication, which reduces the overhead of IoT nodes. Further for mutual authentication session key synchronization and data encryption and for session key update here two level key forms is used which is composed of a fixed confidential key and a dynamic session key. The confidentiality key is designed mainly for authenticating source for receiver and receiver for source and also transmitting session key to node from server i.e. authentication information is encrypted and decrypted using this key. It is stored locally on both server and node. Session key are generated for data encryption and decryption which changes periodically and used by trigonometric function (a new mathematical technique to prevent cryptanalysis). The dynamic session key's length and the update session key period can vary according to the need of specific application. The scheme is flexible and suitable for IoT device security protection.

The rest of paper is organized as follows. Section II puts light on different lightweight authentication and cryptographic schemes. Section III describes the proposed scheme and Section IV analyzes and reviews the security and performance of this scheme. Section V ends the paper.

Revised Manuscript Received on March 26, 2020.

Bhaskar Prakash Kosta, (Corresponding Author) Research Scholar, Department of CSE, GIT, GITAM Deemed to be University, Visakhapatnam, (A.P), India**.** E-mail: prakash09m@gmail.com

Dr. Pasala Sanyasi Naidu, Assoc Professor, Department of CSE, GIT, GITAM Deemed to be University, Visakhapatnam , (A.P), India.

II. VERIFICATION (LIGHTWEIGHT) AND CRYPTOGRAPHIC SCHEMES

To give security to the information which IoT needs to transmit numerous lightweight verification and cryptographic plans are proposed.

RFID has been generally utilized in the field of programmed recognizable proof because of its favorable position, for example, non-contact, comfort, speed and unwavering quality yet its information or data that is moved is helpless against various assault. To give security to the information transmitted by RIFD, Chien (2007) accompanied ultra-lightweight convention called SASI [3]. The convention utilizes XOR, round move and other activity to perform shared validation. Be that as it may, the writing [4] brought up that the convention has an issue, for example, tag following and key spillage. In 2009, another ultra-lightweight convention Gossamer was proposed [5]. In 2012, Tian set forward a light weight convention utilizing change activity to disperse the request for bits [6]. In any case, they despite everything can't avoid replay, non synchronization or a few assaults [7][8]

To give security to the information moving in remote sensor networks, Wuu Yang right off the bat concocted a lightweight plan in 2007[9]. From that point forward, Binod Vaidya, thought of a strategy to give validation [10]. Be that as it may, later a few analysts discovered it has some security blemishes, for example, defenselessness to replay and disguise assault. Other than expanding calculation intricacy it is as yet powerless against replay and man in center assault.

Under the condition, we present a lightweight cryptographic plan. With the assistance of basic trigonometric activity, encryption and unscrambling are done which diminishes registering utilization and expands security. Right now new system is utilized to perform validation and meeting key synchronization through two encoded data trade in which overhead is decreased for IoT gadget. Likewise this plan can forestall assault adequately in light of solid verification and respectability on the information.

III. PROPOSED SCHEME DESCRIPTION

In this section, we will describe the lightweight cryptographic scheme using pseudo stream and trigonometric with dynamic key in detail. The proposed scheme has three phases as follows:

1) Mutual authentication phase.

2) Secret key for Session synchronization and data encryption phase.

3) Secret key for session update phase.

Before description, we firstly summarize the notations used throughout this paper in Table 1.

Table 1: Notation in this paper.

Symbol	Definition
$Node_a$	An IoT hub named $Node_a$
IDa	$Node_a$'s character
CKa	Confidentiality key of $Node_a$ for validation and encryption of SKS
IV	Verification Information
IIA	Information for character confirmation
EIIA	Encrypted information for identity authentication with CKa
SKS	The mystery key for meeting
ESKS	Encrypted mystery key for meeting with CKa
ISKSS	Information for mystery key for meeting synchronization
EISKSS	Encrypted data for mystery key for meeting synchronization with SKS
IUSKS	Information for update mystery key for next meeting
EIUSKS	Encrypted data for update mystery key for next meeting with SKS
Information	Data in plain content
CHK	Checking information for information acknowledgment
f (·)	The function haggled by $Node_a$ and server ahead of time, for example, the hash work
"$\oplus$"	The bitwise XOR operation

A. Mutual Authentication Phase

Fixed secrecy key is utilized right now validation. Accepting that every hub has an exceptional ID and a relating privacy key. The secrecy key of hub is just put away locally to guarantee its security. The hub stores the key and its own ID. The server stores the comparing rundown of the hub ID and the validation key. ID relates to the key individually. For instance, $Node_a$ has its extraordinary IDa and a relating privacy key CKa. Right now, sends its IDa to the server initially. The common confirmation stage is appeared in Figure1 and the means are portrayed as follows:

Step 1: $Node_a \rightarrow$ Server: {IDa}. The $Node_a$ submits its own IDa to the server.

Step 2: Server $\rightarrow Node_a$: {IV_1 }. After receiving the message, server finds CKa according to the IDa. Then server generates $Random_1$ and computes verification information:

$$IV_1 = CKa \oplus Random_1 \qquad (1)$$

Then, server sends IV_1 to $Node_a$.

Step 3: $Node_a \rightarrow$ Server: {IV_2^* , IV_3}. After receiving the message, $Node_a$ generates $Random_2$ and computes $Random_1^*$, IV_2^* and IV_3 by using the CKa as follows:

$$Random_1^* = IV_1 \oplus CKa$$

$$IV_2^* = CKa \oplus f_1(Random_1^*)$$

$$IV_3 = CKa \oplus Random_2 \qquad (2)$$

Then, the $Node_a$ sends IV_2^* , IV_3 to server.

Step 4: Server $\rightarrow Node_a$: { EIIA , IV_4^* }. After receiving the message, server computes:

$$IV_2 = CKa \oplus f_1 (Random_1) \qquad (3)$$

If $IV_2=IV_2^*$, server considers $Node_a$ has a valid identity and generates information for identity authentication IIA and computes EIIA, $Random_2^*$,VI_4^* as follows:

$$EIIA=CKa \oplus IIA$$

$$Random_2^* = IV_3 \oplus CKa$$

$$IV_4^* = CKa \oplus f_2 (Random_2^*) \qquad (4)$$

Then server sends EIIA, IV_4^* to $Node_a$. Otherwise, server cuts off the communication.

Step 5: $Node_a$ receives EIIA and IV_4^* and get IIA by using the CKa as follows:

$$IIA = EIIA \oplus CKa$$
$$IV_4 = CKa \oplus f_2(Random_2) \quad (5)$$

If $IV_4 = IV_4^*$, $Node_i$ considers server's identity is valid and keep communication. Otherwise, $Node_i$ cuts off the communication.

The $Node_a$ — The Server

IDa — IDa →

Find CKa by using the IDa
Generate $Random_1$
$IV_1 = CKa \oplus Random_1$

← VI_1

$Random_1^* = IV_1 \oplus CKa$
$IV_2^* = CKa \oplus f1\ (Random_1^*)$
Generate $Random_2$
$IV_3 = CKa \oplus Random_2$

IV_2^*, IV_3 →

$IV_2 = CKa \oplus f_1(Random_1)$
Check $IV_2 ?= IV_2^*$
If $IV_2 = IV_2^*$, generate IIA
$EIIA = CKa \oplus IIA$
$Random_2^* = IV_3 \oplus CKa$
$IV_4^* = CKa \oplus f_2(Random_2^*)$

← EIIA, IV_4^*

$IIA = EIIA \oplus CKa$
$IV_4 = CKa \oplus f_2(Random_2)$
Check $IV_4 ?= IV_4^*$
If $IV_4 = IV_4^*$, keep communication

Figure 1. Representation of the shared verification stage

Now, shared verification among $Node_a$ and server is finished. Right now, and $Node_a$ just transmit IDa and check data in figure content. Secrecy key CKa - is just put away locally for unscrambling, which can diminish the chance of malevolent spying to take keys.

B. Secret Key for Session Synchronization and Data Encryption Phase

In the event that shared verification succeeds, server will create a mystery key for meeting and complete mystery key for meeting key synchronization with IoT hubs. The asset and force utilization of IoT hubs are low, yet the server's processing capacity is similarly acceptable. Besides, IoT hubs move information in portion by and large. The mystery key for meeting can be produced by the server and sent to the IoT terminal hardware in leisure time. Right now, load on the hub can be decreased. The mystery key for meeting synchronization is appeared in Figure 2 and information encryption stage is depicted later.

The Server — The $Node_a$

Generate SKS_1, $Random_3$
$ESKS_1 = CKa \oplus SKS_1$
$IV_5 = SKS_1 \oplus Random_3$

$ESKS_1$, IV_5 →

$SKS_1 = ESKS_1 \oplus CKa$
$Random_3^* = IV_5 \oplus SKS_1$
$IV_6^* = SKS_1 \oplus f_1(Random_3^*)$

← IV_6^*

$IV_6 = SKS_1 \oplus f_1(Random_3)$
Check $IV_6 ? = IV_6^*$
If $IV_6 = IV_6^*$ Generate **ISKSS**
$EISKSS = SKS_1 \oplus ISKSS$

EISKSS →

$SKSS = ESKSS \oplus SKS_1$

Figure 2. Representation of the meeting key synchronization stage

Stage 1: Server → $Node_a$: $\{ESKS_1, IV_5\}$. Subsequent to affirming the two characters, server creates another arbitrary $Random_3$ and a mystery key for meeting SKS_1. At that point server registers encoded meeting key $ESKS_1$ and confirmation data IV_5 as follows:

$$ESKS_1 = CKa \oplus SKS_1$$
$$IV_5 = SKS_1 \oplus Random_3 \quad (6)$$

After that, server sends $ESKS_1$, IV_5 to $Node_a$.

Stage 2: $Node_a$ → Server: $\{IV_6^*\}$. After receiving the message, $Node_a$ computes $Random_3^*$, SKS_1 and IV_6^* by using CKa and SKS_1 as follows:

$$SKS_1 = ESKS_1 \oplus CKa$$
$$Random_3^* = IV_5 \oplus SKS_1$$
$$IV_6^* = SKS_1 \oplus f_1\,(Random_3^*) \quad (7)$$

Then $Node_a$ sends IV_6^* to server.

Stage 3: Server → $Node_a$: {EISKSS}. After receiving the message, server computes:

$$IV_6 = SKS_1 \oplus f_1\,(Random_3) \quad (8)$$

If $IV_6 = IV_6^*$, server considers secret key for session synchronization is successful and generates information for secret key for session synchronization ISKSS. Then server computes:

$$EISKSS = SKS_1 \oplus ISKSS \quad (9)$$

After that, server sends EISKSS to $Node_a$.

Stage 4: $Node_a$ receives EISKSS and gets information for secret key for session synchronization ISKSS with session key SKS_1:

$$ISKSS = EISKSS \oplus SKS_1 \quad (10)$$

At this moment, secret key for session synchronization is completed.

(i) Encryption Algorithm

Stage 1: Assign all the letter sets for example beginning to end with any very much characterized number like 1,2,3,4 … and so on, it will be spared haphazardly at the two parts of the bargains.

Stage 2: In trigonometry activity we require the estimation of π .At the hour of encryption and decoding this estimation of pi will be required. The genuine estimation of π=3.141592 , however right now $Node_a$ will take the estimation of π be present meeting key SKS1 . For instance let the estimation of SKS1 be 5.137201.

Stage 3: Take the info message . For instance "a ab abc abcd" . Here each letters in order will be encoded by calling the trigonometric capacity Cos (x) , here x= a = 1.0 as arbitrarily characterized in stage 1. For next letter set x = b =2.0.

These qualities will go into the Cos (x) , and gives the figure content , here Cos (1.0) = 0.999593 , the standard estimation of Cos (1.0) = 0.54030 . As we have changed the estimation of π = SKSi = 5.137201; the outcome has been consequently changed and giving the figure content.

The general condition of Rule of ascertaining Cos (x) strategy is

Cos (x*pi/180.0) =y and CHK = CKa ⊕ y1

Here y will be considered as figure content

y1 = Cos(x1*pi/180.0) - - (1)

CHK1 = CKa ⊕ y1

y2 = Cos (x2*pi/180.0) - - (2)

Presently on the off chance that y1, y2, y3 and encryption calculation are known to the aggressors then likewise they won't have the option to rupture the data. Since the estimation of isn't general like π=3.141592 instead of it is 5.137201(SKS1) and can be allocated with some other worth.

Table 2: Demonstration of Encryption Process

Plain Text	Key (pi)	Cos(x **pi*/180.0)/y CipherText
a		0.999593
Space		0.611181
a		0.999593
b		0.998372
Space	5.137201	0.611181
a		0.999593
b		0.998372
c		0.996337

(ii) Decryption Process

At the time of Decryption process , we again require changed value of π . As a cipher text the value of **y** will go into the a$Cos(x)$ function for decryption process.

This is defined as x = $aCos(y)$ * 180.0 / pi and $CHK^* = CK_a$ ⊕ x where **x** is original data

$x1 = aCos(y1) * pi/180.0$ -------------------(1)

$CHK_1^* = CK_a \oplus x1$

Just if CHK* = CHK , the receiver(server) will keep correspondence. Something else, receiver(server) imagines that the information has been noxious altered. CHK is utilized to secure information trustworthiness. In the event that CHK*=CHK , here at any rate key won't be broken by the unapproved client. It takes care of the issue of Cryptanalysis. Here key is π . The estimation of π (going about as key) is known to approve client as it were.

Table 3: Demonstration of Decryption Process

Ciphertext	Key(pi)	aCos(y)*180/pi	Plaintext X
0.999593		1.0	a
0.611181		32.0	Space
0.999593		1.0	a
0.998372	5.137201	2.0	b
0.611181		32.0	Space
0.999593		1.0	a
0.998372		2.0	b
0.996337		3.0	c

(iii) Pseudo stream cipher:

Right now, set forward another cipher instrument called pseudo stream cipher. Conventional stream figure is quick and proficient however it experiences an issue of key trade. In pseudo stream cipher, there is no compelling reason to ensure synchronization of key generation. It gives a component that the key can be created by a side of the correspondence, for example, the server. At that point the key is synchronized through the encryption of an arbitrary and confirmation for the cipher content . Pseudo stream cipher rearranges key synchronization and lessens the overhead of IoT hubs.

C. Secret Key for Session Update Phase

So as to improve the security of correspondence, the plan utilizes a variable meeting key. The meeting key update period and key length can be balanced by various applications. On the off chance that the busybody needs vindictive spying, the meeting key must be broken progressively.

At the point when the common information correspondence time equivalents to a meeting key update period, server creates data for update mystery key for straightaway (new) meeting (IUSKS) and encodes it:

$$EIUSKS = CKa \oplus IUSKS \quad (11)$$

At that point server sends EIUSKS(encrypted data for update mystery key for next(new) meeting) to the hub. In the wake of accepting this data, hub quits sending information and sits tight for another meeting key. From that point onward, server creates another meeting key (for example second meeting so SKS2) and scrambles it by the classification key CKa:

$$ESKS_2 = CKa \oplus SKS_2 \quad (12)$$

What's more, server produces another arbitrary $Random_4$ and processes the new check data VI_7:

$$IV_7 = SKS_2 \oplus Random_4 \quad (13)$$

Server sends {ESKS2, IV_7} to $Node_a$. The ensuing procedure is equivalent to the meeting key synchronization process.

The entirety of the above are the subtleties of the lightweight plan. The plan incorporates three stages. In common verification stage, we present a classification key CKa and arranged functions f1 (·) and f2 (·). It can understand shared verification as well as forestall numerous assaults. In key synchronization stage, pseudo stream cipher was proposed to rearrange key synchronization. Variable meeting key is reasonable for various applications and forestalls assaults adequately.

IV. SECURITY AND PERFORMANCE ANALYSIS

A. Security Analysis

Right now, condense security investigation of our proposed plot as follows:

1) Mutual confirmation: The hub and the server can validate one another, in light of the fact that solitary the authentic server has the classification key CKi as indicated by its IDi. Also, just the veritable hub and server knows the arranged capacities f1 (·) and f2 (·). From Step 4 in the common confirmation stage portrayed in Section III, by checking whether $VI_2 = VI_2^*$, the server can confirm the authentic of the hub. Additionally from stage 5 in the shared validation stage, the hub can confirm the authenticity of the server dependent on checking whether $VI_4 = VI_4^*$.
2) Key obscurity: The keys in our plan are never transmitted in plain content , they are constantly transmitted in figure message, regardless of the secrecy key CKa or meeting key SKSi. Also, the meeting key SKSi is refreshing. This can forestall the spillage of keys to pernicious aggressors.
3) Secret key for Session synchronization and refreshing: The plan utilizes a variable meeting key to improve the security of correspondence. As per various applications, standard for key refreshing changes. For delicate or high-traffic information, we can build the update recurrence and the key length. Be that as it may, if the information is coldhearted or low-traffic, the key update recurrence and the key length can be decreased.
4) Resistance to disguise assault: To effectively finish a disguise assault, the aggressor must pass confirmation in the common validation stage and to have the option to decipher the check message accurately. The privacy key CKa of $Node_a$ is never moved in plaintext in the confirmation stage. In addition, it's difficult for assailant to get the arranged function f1(·) and f2(·). Over all how all the letters in order are interpreted in Encryption and Decryption in addition to the last check by CHK makes it the most troublesome undertaking for assailant.
5) Resistance to replay assault: when all is said in done, utilizing timestamps is a decent method to forestall replay assaults. Be that as it may, the calculation and examination of timestamps additionally welcomes overhead on figuring asset. Right now, and the server arranged two capacities ahead of time. In view of classification key CKa , meeting key SKSi , arranged capacities f1(·) and f2(·), interpretation for various letter set in encryption and decoding capacity and last check by CHK makes replay assault doesn't work in shared verification stage , mystery key synchronization for meeting and information encryption stage.
6) Resistance to man-in-the-center assault: This plan can adequately forestall man-in-the-center assaults due to solid validation of personality, interpretation for various letter set in encryption and unscrambling capacity and solid trustworthiness on the information. Any adjustment on the qualities in common verification stage will cause the coming up short checking of VI_2 or VI_4. Plus, regardless of the server and the hub will check the information uprightness by checking the CHK information. In the event that the information is altered, the server or the hub will dismiss this correspondence.
7) Non Repudation: The Sender and the collector won't have the option to deny the message transmission.

Security properties of the proposed conspire, contrasted and related works are outlined in Table 4.

Table 4: Security comparison of our scheme and related schemes

Items	Our Scheme	SASI	Goassamer convention
Common authentication	Yes	Yes	Yes
Key anonymity	Yes	Yes	Yes
Meeting key synchronization	Yes	No	No

Protection from disguise attack	Yes	No	Yes
Protection from replay attack	Yes	No	No
Protection from man-in-center attack	Yes	No	No

B. Execution Analysis

We compare the time of the common validation stage, which is actualized [13] [14] in the three conventions (SASI, Our plan and Gossamer convention). Right now, plans are directed on a work areas with windows 32bit, CPU 2.00 GHz with 2 GB of RAM. Trial results are the average time of three examinations for three conventions(protocol) (likewise the outcomes are shown and the analysis are finished by considering two labels/IoT gadget are joined to server in addition to in each of the three cases 40 bytes are considered for keys in three cases. Here transmission times of information are not taken into consideration only processing time and displaying the result time is taken into account) are shown in Figure 3. After analyzing Figure 3, we notice that our plan accomplishes the best time execution of 16000 ms and SASI likewise has great execution of 31000 ms. Goassamer convention has the biggest time utilization of 62000 ms, which is around multiple times as long as our own.

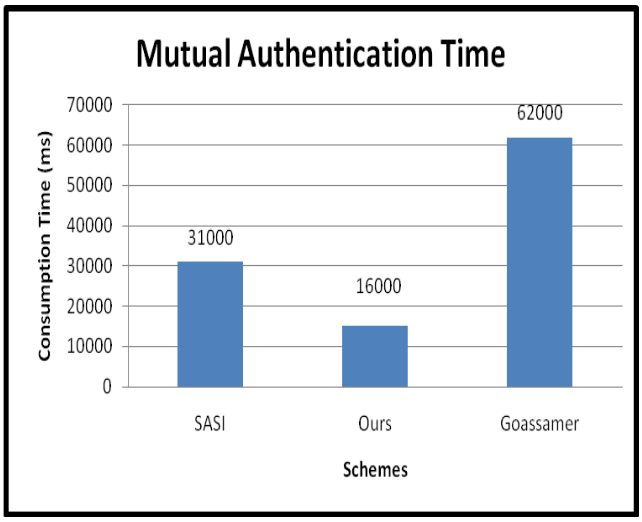


Figure 3. Correlation of common verification time among SASI, Our plan and Goassamer convention.

Our proposed plot is less overhead, and can oppose different assaults. SASI utilizes bitwise XOR, addition mod 2^m and turn to(rotation) acknowledge common verification. They cost low utilizations yet give constrained security highlights. Goassamer utilizes bitwise XOR, addition mod 2^m, revolution(rotation) and mixbits capacity to acknowledge common confirmation. Contrasted and the SASI and Goassamer, our plan takes less overhead and gives greater security highlights. In our plan, we just use XOR activity and functions(secret) to acknowledge common verification and meeting key synchronization. Also, meeting key is produced by server and synchronized with IoT hub. It is a decent method to lessen the overhead of IoT hubs.

The examination of the proposed trigonometry calculation for encryption and unscrambling has been done and exhibited in Fig2 and Fig3. The Encryption and Decryption Algorithm was coded in C Language[13][14]. It was aggregated with MinGW-GCC 4.8.1, on the Core 2 Duo Processor, 2.00 GHz under windows 7 OS. The investigation parameters are plain content size (in Bytes) and time taken in encryption and unscrambling (in microseconds).

Table 5: Execution times for encryption algorithm on Core(TM) 2 Duo, 2.00 GHz

Size(In Byte)	Time(Microsecond)
6	0
11	0
15	16000
19	31000
25	31000
35	46000
50	78000

Table 6: Execution times for decryption algorithm on Core(TM) 2 Duo, 2.00 GHz.

Size(In Byte)	Time(Microsecond)
5	15000
8	31000
13	47000
18	47000
26	62000

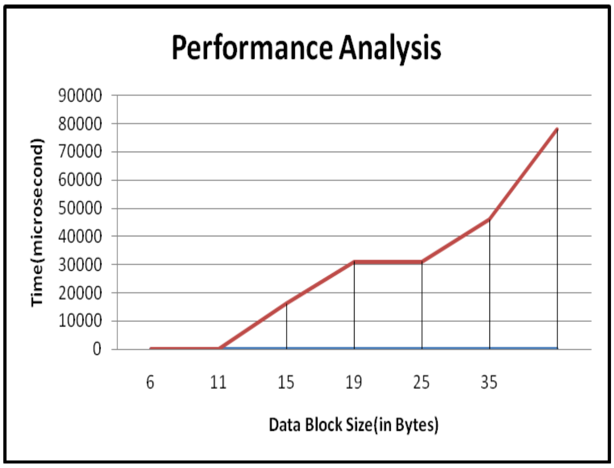


Figure 4: Performance analysis of encryption

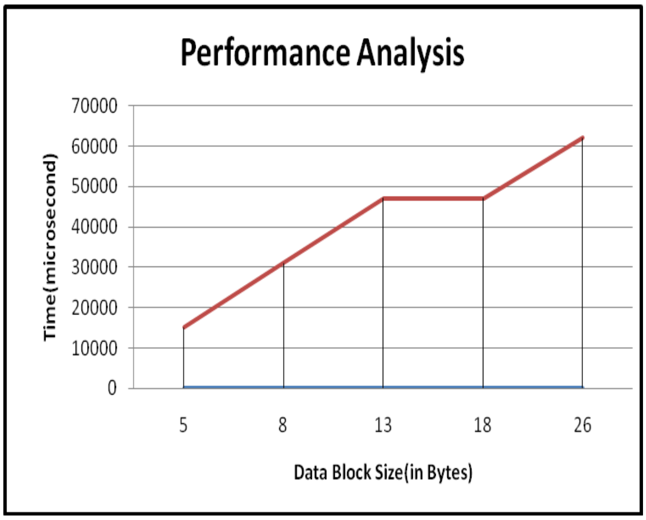


Figure 5: Performance analysis of Decryption

V. KNOWN PLAIN TEXT CRYPTANALYSIS ATTACK CHECKING ON TRIGNOMETRIC ALGORITHM

Assume here that for any plaintext the assailant has a figure content which is "0.527718 0.673219 0.642737" , now we should check the assaulter will have the option to break the plain content or not , and we accept that the aggressor is notable about the calculation. From the outset assailant expect that the plain content is x1 x2 x3 and all these are factors and those might be a genuine number . What's more, the estimation of π1 isn't known to him

$0.527718 = \text{Cos}(x1 * \pi / 180.0)$ ________________(i)

$0.673219 = \text{Cos}(x2 * \pi / 180.0)$ ________________(ii)

$0.642737 = \text{Cos}(x3 * \pi / 180.0)$ ________________(iii)

From equation (i)

$\text{Cos}^{-1}(0.527718) = x1 * \pi / 180.0$

$\pi = 10466.74826 / x1$ ________________(iv)

From equation (ii)

$\text{Cos}^{-1}(0.673219) = x2 * \pi / 180.0$

$\pi = 8583.120435 / x2$ ________________(v)

From equation (iii)

$\text{Cos}^{-1}(0.642737) = x3 * \pi / 180.0$

$\pi = 9000.681338 / x3$ ________________(vi)

From the above equation we can get that

$\pi = 10466.74826 / x1 = 8583.120435 / x2 = 9000.681338 / x3$

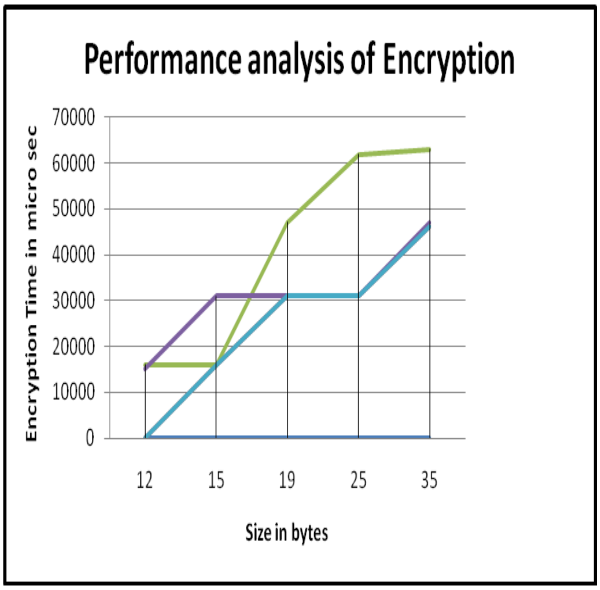
So now from the above condition the assailant won't have the option to get the estimation of x1, x2 , x3 . Since the estimation of π is distinctive instead of the regular worth. So it is demonstrated that the cryptanalysis assault is neglected to split the trigonometric figure.

VI. COMPARATIVE PERFORMANCE ANALYSIS

The proposed calculation has been contrasted and other existed calculations like RC4, Hill-Cipher and RSA[11][12]. What's more, the outcomes have appeared in the accompanying outline. From the accompanying figure we can say that the run time multifaceted nature of trigonometric Algorithm is beneath than the other existed Algorithm conspire.

Table 7: Comparison of execution times by proposed algorithm with other existed algorithm for encryption.

	Time in micro sec for selected algorithm ON Core(TM) 2 Duo, 2.00 GHZ			
No of Bytes	RC4	Hill-Cipher	RSA	Trigonometry
12	0	16000	15000	0
15	16000	16000	31000	16000
19	31000	47000	31000	31000
25	31000	62000	31000	31000
35	46000	63000	47000	46000



---------- Trigonometry
---------- RC4
---------- RSA
---------- Hill Cipher

Figure 6: Comparative analysis

VII. CONCLUSION

Right now, propose a lightweight cryptographic calculation utilizing pseudo stream cipher with dynamic key. It gives practical and attainable instruments for common confirmation, meeting key synchronization and meeting key update. They are appropriate for IoT hubs with constrained figuring assets and force. Right off the bat, we present the subtleties of shared validation and an approach to forestall replay assault without timestamp. Besides, pseudo stream figure is advanced to just acknowledge meeting key synchronization and information encryption to decrease the overhead of IoT hubs. What's more, we have contrasted the security and execution and some lightweight confirmation and cryptographic plans. The plan is lightweight yet can forestall assaults successfully, which is fit for ensuring the security of the correspondence between IoT hubs and servers.

The proposed calculation for encryption isn't just giving the expedient information encryption however it gives a superior security contrasted with other calculation through a most grounded key moreover. The estimation of π is changing each time that makes the calculation sufficient. The calculation additionally makes the cryptanalysis procedure complex. Since there are obscure factors will be more and the quantity of conditions will be less when contrasted with other standard encryption calculations. This calculation could be well utilitarian for online applications like chatting. The examination of Encryption chart mirrors the information block size and encryption time as a straight connection(linear relation), after the block size of 25Bytes.

REFERENCES

1. Huda Saadeh ,Wesam almobaideen ,Khair Eddin Sabri "Internet of Things: A Review to Support IoT Architecture's Design" IT-DREPS Conference, Amman, Jordan Dec 6-8, 2017.
2. Kai Zhao1 , Lina Ge1 "A Survey on the Internet of Things Security" 2013 Ninth International Conference on Computational Intelligence and Security
3. Chien H Y. SASI: A New Ultra lightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity [M]. IEEE Computer Society Press, 2007.
4. CAO T, BERTINO E, LEI H. Security Analysis of the SASI Protocol [J]. IEEE Transactions on Dependable and Secure Computing. 2009, 6(1), pp. 73-77.
5. Peris-Lopez P, Hernandez-Castro J C, Tapiador J M E, et al. Advances in Ultra lightweight Cryptography for Low-Cost RFID Tags: Gossamer Protocol[C]// Information Security Applications. Springer-Verlag, 2009, pp. 56-68.
6. TIAN Y, CHEN G L, LI J. A New Ultra-lightweight RFID Authentication Protocol with Permutation[J]. IEEE Communications Letters. 2012, 16(5), pp. 702-705.
7. SUN H M, TING W C, WANG K H. On the Security of Chien's Ultra-lightweight RFID Authentication Portocol[J]. IEEE Transactions on Dependable and Secure Computing. 2011, 8(2), pp. 315-317.
8. P. D'Arco and A. De Santis. On ultralightweight RFID authentication protocols. IEEE Trans. Dependable and Secure Computing, vol. 8, no. 4, pp. 548–563, July-Aug. 2011.
9. Huei-Ru Tseng, Rong-Hong Jan, and Wuu Yang "An Improved Dynamic User Authentication Scheme for Wireless Sensor Networks " 1930-529X/07/$25.00 © 2007 IEEE
10. Binod Vaidya, Dimitrios Makrakis, Hussein T. Mouftah "Improved Two-factor User Authentication in Wireless Sensor Networks " Second International Workshop on Network Assurance and Security Services in Ubiquitous Environments
11. O P Verma, Ritu Agarwal, Dhiraj Dafouti, Shobha Tyagi, "Peformance Analysis Of Data Encryption Algorithms",IEEE 2011.
12. Akash Kumar Mandal, Chandra Parakash, "Performance Evaluation of Cryptographic Algorithms: DES and AES", IEEE Conference on Electrical, Electronics and ComputerScience 2012.
13. 3GPP TS 35. 201 V14. 0. 0, Specificatio n o f the 3GPP Confidentiality and Integrity Algorithms; Document 1: f8 and f9 specifications, 2017.
14. 3GPP TS 35. 202 V14. 0. 0, Specificatio n o f the 3GPP Confidentiality and Integrity Algorithms; Document 2: Kasumi algorithm specifications, 2017.

AUTHORS PROFILE

Bhaskar Prakash Kosta received his MCA(NIT Calicut, India 1998) and ME CSE (Anna University, India 2007). He is currently is pursuing a PhD in Computer Science and Engineering . He has about 15 years of experience in Teaching. His research interests lie in computer network , databases ,network and information security and in the design and implementation of cryptographic algorithm for various devices

Dr. P Sanyasi Naidu His research area includes Applied Cryptography, Network and Cloud Security. He got prestigious "**Governor's National Award for Excellence in Research and Development**" with full honours and privileges in recognition of his remarkable achievements and outstanding contribution in the field of Research, Publications, Patents, Training, Mentoring, & Teaching in Engineering discipline. He is one of the top performer of the certification courses conducted by **IIT Bombay "Foundation Program in ICT for Education", "Pedagogy for Online and Blended Teaching-Learning Process", "Mentoring Educators in Education Technology".** He got 10 ELITE **NPTEL** certificates conducted by various **IIT**'s for successful completion and Mentoring of various advanced Computer Engineering Courses and **Mezzanine** Technologies