Design and Implementation of a Most Secure Cryptographic Scheme for Lightweight Environment using Elliptic Curve and Trigonohash Technique

BhaskarPrakash Kosta¹, Dr. Pasala Sanyasi Naidu²

Computer Science and Engineering Department
GIT, GITAM Deemed to be University
Vishakhapatnam, AP, India

Abstract—The Internet of Things (IoT) is a rising development and is an organization of all gadgets that can be gotten to through the web. As a central advancement of the IoT, wireless sensor networks (WSN) can be used to accumulate the vital environment parameters for express applications. In light of the resource limitation of sensor devices and the open idea of remote channel, security has become an enormous test in WSN. Validation as an essential security service can be used to guarantee the authenticity of data access in WSN. The proposed three factor system using one way hash function is depicted by low computational cost, and limit overhead, while achieving all other benefits. Keys are made from secret key for meeting for improving the security. We differentiated the arrangement's security and execution with some lightweight plans. As shown by the examination, the proposed plan can give more prominent security incorporates low overhead of correspondence. Encryption and unscrambling is done using numerical thoughts and by using the possibility of hash function. Mathematical thoughts are lightweight and update the security up by a staggering degree by diminishing the chances of cryptanalysis. When contrasted with different calculations, the proposed calculation gives better execution results.

Keywords—Internet of Things (IoT); authentication; one way hash function; lightweight environment; secret key

I. INTRODUCTION

As far off correspondence headway has made, brilliant world individuals can work by utilizing impending Internet of Thing (IoT) paradigm[1]. The fundamental contemplated the Internet of things, or IoT, is a strategy of interrelated getting ready contraptions, mechanical and automated machines, things, creatures or individuals that are given great identifiers (UIDs) and the capacity to move information over an association without expecting that human should human or human-to-PC facilitated exertion. Numerous splendid application can be made by using the data delivered by radio-frequency identification (RFID) and remote sensor organization. A comparable idea can be applied in various mechanical field[2] where canny application can be made using IoT devices. For example in healthcare[3] some distinguishing devices or sensors are embedded or attached to customer which assemble some basic information about customer, for instance, ECG, pulse, temperature and blood oxygen. these fundamental information is then shipped off affirmed clinical master with the help of a section encouraged by the affiliation who runs the center, taking into account which the treatment is overseen without the actual presence of clinical master (This technique can be used to perceive COVID-19 patient among the people who are in separate so the spread of sickness can be avoided).

Using WSN biological data can be accumulated. WSN are regularly involved customers, passages (gateway server) and sensors centers. By and large sensor centers have limited energy, computational force and limit power moreover it has been seen that the energy usage of sensor center depends upon correspondence detachment [4][28], the energy usage for sending or tolerating a message of l-bits over or from a distance (d) are assessed by Eqs. 1 and 2, independently. Here the free space model is used if (d) isn’t by and large a cutoff d0; something different, multi-way model is used.

\[
E_r(l, d) = \begin{cases} 
 lE_{elec} + lE_{amp}d^2 & \text{for } d < d_0 \\
 lE_{elec} + lE_{amp}d^4 & \text{for } d \geq d_0 
\end{cases}
\]

where \(E_{elec}\) is the energy required by the electronic circuit, \(E_{amp}\) are the energy required by the amplifier in free space and multi-path model, respectively. One of the plan for correspondence in WSN is showed up in Fig. 1.

In Fig. 1, the customer can start the correspondence by sending login message to the gateway (the customer is currently enrolled with the portal the passage (server) moreover keeps up its correspondence with different sensors), as of now the server examinations the sales of customer and makes correspondence with remarkable sensor to make the correspondence possible among customer and sensor. Using this philosophy the power use of sensor center point can be decreased and the life of WSN can be extended. Considering limited security highlight of remote channel and confined resources of IoT devices, IoT faces veritable security and assurance challenges [5].

Fig. 1. Architecture for Communication.
Das in 2009 Das arranged a check scheme [6] for WSN, this arrangement was found having issue with inside attack, impersonation attack and server(gateway) center bypassing attack. In 2012 A.K Das [7] arranged powerful customer approval for progressive WSN using light weight calculation, for instance, hash work and symmetric cryptography, in any case it moreover had issues, for instance, inside attack, server primary key revelation attack. In 2013, Xue et al. [9] proposed a lightweight client validation for WSN, and just hash activities are used to ensure the computational effectiveness. They asserted that their plan is secure against identity/secret key speculating assault and taken smart card assault. Be that as it may, their plan was discovered uncertain against stolen smart card assault, insider assault, following assault, and personality speculating assault [10].

As [8] reveals that most symmetric cryptography and hash based protocols are unreliable to user anonymity and smart card security breach assault, thus public key technique can be used like elliptic curve cryptography (ECC) which outfit same security as RSA with more diminutive key size. In 2011, Yeh et al. [11] considered affirmation convention for WSN reliant on ECC. This show had the issue customer namelessness (secrecy) and undetectability. Thereafter, Shi and Gong [12] presented an improved ECC based check convention for WSN. Regardless, these two shows can't give the features of customer namelessness (secrecy) and unrecognizability. In 2016, Jiang et al. [13] proposed an ECC based untraceable affirmation convention for WSN yet had some security defect as pointed by Li et al. [14].

In rest of this paper oversee proposed three-factor customer approval convention using one way hash function and contrasting encryption using one way hash function and disentangling calculation are similarly discussed and security discussion which are showed up in Sections II and III. Section IV oversees comparison of our show with other related show and discussion of security blemishes of Chang and Le's arrangement, Xiong- Li et al. scheme.

II. PROPOSED NONYMITY AUTHENTICATION PROTOCOL FOR IIOT

In this part, we work with a three-factor validation protocol for modern web of things to accomplish client secrecy and oppose cell phone misfortune assault [15]. Here, we receive the fuzzy extractor [16] to handle biometrics data. More noteworthy degrees of safety are accomplished by biometric secret key, like fingerprints, retina examines. Notwithstanding, two biometric perusing are infrequently indistinguishable, despite the fact that they are probably going to be close. Notation used in full text is shown in Table I. Our plan contains four stages, for example initiation, registration, authentication and key agreement, and password change. We delineate these stages as follows.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(U_m, ID_m, PW_m)</td>
<td>The Mobile Device(User), and the identification and secret Key</td>
</tr>
<tr>
<td>GN, X(\text{GN})</td>
<td>The gateway Server and its server secret key</td>
</tr>
<tr>
<td>S(_m), ID(_m)</td>
<td>The sensor hub and its identification</td>
</tr>
<tr>
<td>xx, XX</td>
<td>Private key and public key Gateway server</td>
</tr>
<tr>
<td>bm, Mm</td>
<td>Private key and public key of Mobile device(User)</td>
</tr>
<tr>
<td>X(_{\text{GN}-m})</td>
<td>Shared secret key between Gateway Server and Sensor hub</td>
</tr>
<tr>
<td>PP</td>
<td>A point on the elliptic curve</td>
</tr>
<tr>
<td>H(1(\cdot), @, @)</td>
<td>The hash, XOR, and XNOR operation</td>
</tr>
<tr>
<td>A</td>
<td>An Attacker</td>
</tr>
</tbody>
</table>

A. Initiation Phase

To introduce the framework, Gateway Server picks a single direction hash work H\(1(\cdot)\) which it imparts to every one of the clients. Additionally, it chooses an elliptic curve E dependent on a limited field \(F_p\). From that point forward, Gateway server picks a subgroup GG of E with request huge prime \(n\), and the generator is point PP. At that point Gateway worker produces a private key xx and figures the relating public key XX, where \(xx \in Z^*n\) and \(XX = xxPP\). Finally, GWN stores xx furtively and distributes the boundaries \(\{E(F_p), GG, PP, XX\}\).

1) The Gateway Server (GN)

a) Gateway server chooses a secret key \(X_{\text{GN}}\) and one way hash function H\(1(\cdot)\) which it shares with all the users through secured channel.

b) GN chooses an elliptic curve E dependent on a limited field \(F_p\). From that point forward, GN picks a subgroup GG of \(E(F_p)\) with request huge prime \(n\), and the generator is point PP.

c) GN produces a private key xx and figures the comparing public key XX, where \(xx \in Z^*n\) and \(XX = xxPP\).

d) GWN stores xx covertly and distributes the boundaries \(\{E(F_p), GG, PP, XX\}\).

e) Sensing Device (Ss) Registration Phase.

All the sensing hub(Ss) in IoT are registered offline by the GN as follows.

Step SD1. For each device \(S_m\), the GN chooses a one way hash function H\(1(\cdot)\) which it shares with all the sensor nodes in secured manner and a unique identity ID\(_m\) and calculates the corresponding shared secret key \(X_{\text{ID}_m, \text{GN}} = (\text{IDs} \oplus X_{\text{GN}})\).
Step SD2. The GN pre-loads \{ ID_m, X_{\text{ID}_m-\text{GN}}, H1(\cdot) \} in the memory of S_s.

The Sensor Node(Ss)

The Server(GN)

Select IDs for Ss
compute \( X_{\text{ID}_m-\text{GN}} = X_{\text{GN}} \oplus \text{IDs} \)

\( ID_m, X_{\text{ID}_m-\text{GN}} \)

Store IDs, \( X_{\text{ID}_m-\text{GN}} \), \( H1(\cdot) \)

B. Registration Phase

To be associated with the framework as a legitimate client, the accompanying advances ought to be performed between client Um and GN, and afterward Um can get to the sensor information progressively by utilizing his/her cell phone.

Stage 1. A character IDm, a secret word PWm and an irregular number rm are picked by Um. Then, Um engraves the biometric data Bio_m on a cell phone with fuzzy extractor [16] and gets Gen(Bio_m) = (\( \sigma_m \), \( \tau_m \)). At that point Um ascertains the veiled secret key RPWm = PWm\( \oplus \) rm, (here \( \oplus \) stands for XOR operation) and submits \{ID_m, RPWm, \( \sigma_m \)\} to GN for enrollment.

Stage 2. While getting the enrollment demand, GN checks if IDm in the data set. In the event that indeed, Um requested to present another character. Something else, GN figures Am = H1(IDm\( \oplus \) X_{\text{GN}}) and Bm = Am\( \oplus \) (RPWm\( \oplus \) \( \sigma_m \)), and sends \{Bm, XX\} to Um through a reliable manner.

C. Authentication Phase

When Um wants to access the sensor data of Ss, the following authentication steps should be performed among Um, GN and Ss, and a session key is shared among these three parties at the end of the authentication phase.

Step 1. Um inputs IDm and PWm on the mobile device, and imprints the biometric information Bio'_m on the mobile device with fuzzy extractor. Then the mobile device calculates.

\[ \sigma_m = \text{Rep}(\text{Bio}'_m, \tau_m), \]
\[ \text{RPW}'m = (\text{PWm}\oplus \text{rm}), \]
\[ A'm = \text{Bm}\oplus (\text{RPW}'m \oplus \sigma_m). \]

checks A' \( m \) = Am. Imbalance implies in any event one of three variables is invalid, and the login demand is rejected by the cell phone. Or disaster will be imminent, the cell phone plays out the following stage. The cell phone delivers an arbitrary numbers bme \( Z'_n \) and calculates.

\[ \text{MM1} = \text{bmm}\oplus \text{PP}, \]
\[ \text{MM2} = \text{bmm}\oplus \text{XX}, \]
\[ \text{MM3} = \text{IDm}\oplus \text{MM2}, \]
\[ \text{MM4} = \text{IDs}\oplus \text{MM2}, \]
\[ \text{MM5} = H1(A'm) \]

Then the mobile device(User) submits the login request \{ MM1,MM3, MM4,MM5 \} to GN.

Stage 2. When receiving the request, GN first retrieves IDm, IDs and then computes A'm and then it calculates the hash value of A'm that is MM5.

\[ \text{MM2} = \text{xx}\oplus \text{MM1}, \]
\[ \text{ID}'m = \text{MM3} \oplus \text{MM2}, \]
\[ \text{ID}'s = \text{MM4} \oplus \text{MM2}, \]
\[ A''m = (\text{ID}'m \oplus X_{\text{GN}}), \]
\[ \text{MM5}' = H1(A'm) \]

GN server checks MM5' = MM5 If yes then the GN Server get a assurance that the user is genuine then GN server calculates the shared secret key X'_{\text{ID}_m-\text{GN}} between sensor node(Ss) (given by user) and itself then it calculates MM6, MM7 for performing authentication.

\[ X'_{\text{ID}_m-\text{GN}} = (\text{ID}'s \oplus X_{\text{GN}}), \]
\[ \text{MM6} = X'_{\text{ID}_m-\text{GN}} \oplus \text{ID}'m, \]
\[ \text{MM7}' = H1(X'_{\text{ID}_m-\text{GN}}) \]

At last, GWN Server sends \{ MM6,MM7 \} to sensor Node.

Step 3. After receiving the message from GN Server, Sensor node(Ss) first computes.

\[ \text{MM7}' = H1(X'_{\text{ID}_m-\text{GN}}) \]

Sensor node checks MM7' = MM7, if yes then sensor node(Ss) gets a assurance that the request came from correct server, then sensor node(Ss) calculates.

\[ \text{ID}'s = \text{MM6} \oplus X_{\text{ID}_m-\text{GN}} \]
\[ \text{MM8} = H1( X'_{\text{ID}_m-\text{GN}} \oplus \text{ID}'m) \]

The Sensor node also gets the information about the user who wants to communicate, after that it sends responses \{MM8\} to GN Server.

Step 4. After receiving the response from the sensor node(Ss) the GN Server retrieves computes.

\[ \text{MM8}' = H1(X'_{\text{ID}_m-\text{GN}} \oplus \text{ID}'m) \]

if (MM8' = MM8) , GN Server gets an assurance that sensor node(Ss) is genuine and generates the session key to be given to both mobile device(user) and sensor node(Ss):

\[ \text{SKS} = \text{ID}'m \oplus \text{IDs} \oplus X_{\text{GN}} \oplus A''m \oplus xx \]

\[ \text{MM9} = X'_{\text{ID}_m-\text{GN}} \oplus \text{SKS} \]
\[ \text{MM10} = \text{MM2} \oplus \text{SKS} \]
\[ \text{MM10} = H1(A''m \oplus \text{ID}'m) \]

Then GN Server sends MM9 to sensor node(Ss) and (MM10,MM10) to mobile device(user). Otherwise, server cuts off the communication.
Step 5. After receiving the message from GN Server, mobile device(User) computes $\text{MMM}'10$.

$\text{MMM}'10 = H_1(A'm \oplus \text{ID}_m)$

If $\text{MMM}'10 = \text{MMM}10$ then mobile device(user) gets a assurance that he is communicating with correct GN server then mobile device computes secret key($\text{SK}_m$).

$\text{SK}_m = \text{MM}10 \oplus \text{MM}2$

$\text{MM}11 = H_1(\text{MM}10)$

Then the mobile device(User) submits MM11 to GN Server.

Step 6. After receiving the message from GN Server, Sensor node(Ss) retrieves the secret key.

$\text{SK}_s = \text{MM}9 \oplus X_{\text{ID}_s - \text{GN}}$

$\text{MM}12 = H_1(\text{MM}9)$

Then the Sensor node submits MM12 to GWN Server.

Step 7. After receiving the message from mobile device(user) MM11 and Sensor node MM12, GN Server, computes the following.

$\text{MM}'11 = H_1(\text{MM}2 \oplus \text{SK}_s)$

$\text{MM}'12 = H_1(X_{\text{ID}_s - \text{GN}} \oplus \text{SK}_s)$

If $\text{MM}'11 = \text{MM}11$ and $\text{MM}'12 = \text{MM}12$, GN Server gets a assurance that Mobile Device(User) and sensor node(Ss) are is genuine and received the correct secret key and hence key synchronization ends.

If mutual validation and key exchange phase completes successfully, both mobile device(user) and Sensor hub(Ss) will transfer information by using the meeting key which will be computed by the GN server and given to both mobile device(user) and the sensor hub(Ss) in secured way. First source authentication is achieved, for this a one way hash function is used. The message digest is created for date($A_m$) which is available at both mobile device(user) and GN server, so this message digest will act as a authenticator. In the same way sensor node(Ss) and GN server mutual authenticate each other by creating a authenticator using hash function (HA-160) by using the common data $X_{\text{ID}_s - \text{GN}}$. The details of hash function (HA-160) is given below. After all the devices get assurance about other devices the GN server generates a secret key and passes it to both sensor hub(Ss) and mobile device(user) in a secured way.

Fig. 2. Illustration of the Mutual Validation Stage and Key Synchronization Stage.
The Secret Key between User(Ui-Mobile) and Server(GN)
The Mobile device(User-Um)
private Key=bm
Public Key = MM1(MM1=bm*PP)
Secret key = bm*XX
= bm*xx*PP
The Server(GN)
private Key=xx
Public Key = XX(XX=xx*PP)
Secret Key =xx*MM1
= xx* bm * PP

The secret key for session synchronization is showed in
Fig. 2. Hash algorithm and data encryption phase is described.

III. HASH ALGORITHM(HA-160)
The proposed calculation is named as HA-160[24][25][26]. This assessment HA-160 (Hash Algorithm)
recognizes a message as responsibility with a most
uncommon length of under 2128 pieces and makes a 160-piece
message digest as yield. First the data message is isolated into
squares of 1024 pieces. This 1024 pieces is taken as
information, by then the information is diminished from
1024-digit squares to 512-cycle blocks. The Hash code
creation work perceives two wellsprings of data which are
1024 pieces square of the message and the instate MD
support (getting variable 160-bits). The cycle includes the
going with advances.
A. Attach Padding Pieces
The message is padded so its length is reliable to 896
modulo 1024 (length = 896 mod 1024). Padding is
continually done, whether or not the message is of needed
length. Subsequently, the amount of padding pieces is in the
extent of 1-1024. The padding contains alone 1 followed by
the significant number of 0's for example 10... 0.
B. Attach Length
A square of 128 pieces which contains the length of the
message (going before cushioning) is joined to the message.
This square is treated as an unsigned 128-digit number (most
basic byte first).
C. Initialize MD Pad
A 160-piece cushion is utilized to hold impermanent and
eventual outcome of the hash work. The assistance can be
tended to as five 32-digit registers (SS0, SS1, SS2, SS3 and
SS4). These registers are instated to the going with 32-bit
numbers (Hexadecimal qualities):
SS0 = 67 45 23 01
SS1 = efcdab 89
SS2 = 98 ba dc fe
SS3 = 10 32 54 76
SS4 = c3 d2 e1 f0
These attributes are same as the secret vector evaluations
of SHA-1 which are normalized by Federal Information
Processing Standards Publications (FIPS PUBS). These
qualities are dealt with in big endian plan, which is the
standard byte of a word in the low-address byte position.

D. Processes Message in 1024-Bitblocks
The message digest age strategy contains five sub
capacity. This part is named (HA-160) in Fig. 1 and its
reasoning is showed up in Fig. 2. Fig. 1 depicts the general
getting ready of message to make a information survey. The
consequence of the underlying two phases (after add padding
pieces and append size) yields a information that is a number
various of 1024-piece long. The all-inclusive information is
addressed as the gathering of 1024-piece blocks XX0,
XX1,XX2, XXL-1, so the total size of the all-inclusive
information is L × 1024 pieces (L = the amount of 1024 bit
hindiers), that is the outcome of various of 32-bit blocks. Here
K addresses the real length of the message in pieces, "IV" is
the fundamental vector which is used to present the five 32-
digit registers (SS0, SS1, SS2, SS3 and SS4). VCi, VCiv, VCq
and VCL-1 address instate MD(carry vector) which holds
widely appealing and inevitable result of the Hash work,
separately. Each round takes two data sources one 1024-cycle
blocks (XXq) of the message and a 160 piece pass
vector (VCq). Close to the completion of the L\textsuperscript{th} stage produces 160
piece message digest. From the start the given message is
disengaged into 1024-digit blocks, and each square is passed
to Hash Code making capacity (HA) as a commitment close
by the 160-piece vector(as shown in Fig. 3 and Fig. 4).

The Hash work (H\textsubscript{HA-160}) rationale is:
Remouldingtask(R\textsubscript{0i}) converts the given 1024-bit block
into adjusted 512-bit block.
Progression task(P\textsubscript{0i}) converts the given 160-piece initial
vector into extended 512-bit vector.
32-digit XOR operation(XOR\textsubscript{op}) performs XOR procedure
on each 32-pieces of altered 512-bit block and extended 512-bit
vector.
Partition and Modification operation ($P_{op}$) disconnects the 512-cycle block into 16 sub-square of 32-bits each and afterward each 32 digit block is isolated into four 8-bit block, after that it changes every eight bit block. This interaction is rehashed for every one of the 32 bit block.

New Point (on elliptic curve) estimation work ($NP_{fn}$) calculates new point on elliptic curve using 4 eight-cycle block (from each 32-bit block).

Where,

- $XXq =$ the $q$th 1024-cycle square of the information.
- $VCq =$ mooring variable arranged with the $q$th square of the information.
- $Rfn =$ Remoulding task.
- $P =$ (xxx1, yyy1) and $Q =$ (xxx2, yyy2).
- $Pfn =$ Progression task.
- XORop = XOR (Exclusive-OR) development performed on each 32-cycle square of the adjusted 512-digit block ($XXq$) and relating 32-pieces of the comprehensive 512-bit block ($VCq$).
- Pop = 512-digit square can be withdrawn into 16 sub-squares of 32-bits each and subsequently each 32 cycle block is confined into four 8-bit block, after that it changes each eight bit block.
- NPFn = initial 32-cycle square further parceled into 2 sub parts of 16-piece each.

1st sub part = 16 bit, apportioned into two sub part (8-bits, 8-bits) = (xxx1, yyy1).

2nd sub part = 16 bit, divided into two sub part (8-bits, 8-bits) = (xxx2, yyy2).

The characteristics (xxx1, yyy1, xxx2, and yyy2 as showed in Fig. 5) are changed over into entire numbers followed by processing another point on elliptic curve. The above interaction is revamped for staying fifteen 32-cycle block and the outcome is 16 sub square every one of 16 pieces. Presently every one of this 8 sub square every one of 16 digit is changed, the initial 8-cycle of first square is XOR with a 8-bit sub square which are each of the zeros, the outcome is put away in initial 8-bit sub square. The subsequent 8-digit sub square of first square is added with initial 8-bit sub square and result is put away in second 8-cycle sub square and the interaction is rehashed for every one of the 8-piece sub square. The outcome is arranged 256 cycle block. Presently neighboring sub square every 16 pieces are added which brings about a sub square of 32 digit and we get eight 32 piece sub square. At that point this eight 32-digit sub square are converted to four 32-bit sub block by added first and fifth, second and sixth, third and seventh and fourth and eight 32-digit sub square are added which brings about four 32-cycle sub square. By playing out some numerical procedure on over four processed 32 bit sub square and afterward XOR them brings about the fifth 32 cycle sub square. This five 32 piece sub square when concatenated adds up to 160 piece and when summed with introduced MD cradle shapes the hash code. This 160 piece hash code will be the contribution to next 1024 digit of message i.e it will go about as introduce MD cradle for next 1024 cycle of the info message. The last 160 piece code produced from last 1024 cycle of message will be the last message digest which will go about as confirmation code.
Fig. 6. Function of Sub-Work1.

Sub-work 2. The above result is again detached into sub square of 8 pieces and initial eight-cycle sub square and focus eight-bit sub square after bitwise complimenting(\(\sim\)) are XOR and the result is again bitwise complimented(\(\sim\)) and set aside in an alternate array(W2). The above cycle is reiterated for second 8-bit sub square and focus notwithstanding one 8-digit sub square and result is taken care of in second piece of disconnected array(W2). This is repeated for all extra 8-bit sub square. The result is 1024 cycle block is diminished to 512 digit block. The changed message as depicted under.

```cpp
int Mod_Inp(vector<int> mm, int nn)
{
    static int W[130]; //32 bits as a group
    int ii, jj = 0, pp, qq, kk1 = 0, TT = 0;
    pp = nn;
    nn *= 128;
    for (ii = nn; ii < nn + 128; ii++)
    {
        W[ii] = TT ^ mm[ii];
        TT++;
    }
    jj = ((ii + nn) / 2);
    qq = jj;
    for (i = nn; i < (qq + nn); i++)
    {
        W2[kk1] = ((~W[ii]) ^ (~W[jj]));
        kk1++;
    }
    jj++;
    return 0;
}
```

2) Progression Task \((P_{pr})\): The 160-piece Initial Vector (IV) is one of the commitments to the message digest creation work (HA-160). It will in general be stretched out to 512-bits by interfacing all hidden vector regards in indirect manner, which is called Concatenated Vector (VC). The association cycle is showed in Fig. 7.

3) 32-bit XOR operation \((XOR_{32})\): XOR movement is done on starting 32-piece sub-block of modified 512-cycle square and expanded 512-piece square. For this initial four 8-digit block are extended and summed with the goal that it becomes 32 piece square. At that point this 32-piece square is XOR with initial 32-bit of extended or instate MD cradle of 512-piece square. This interaction is reiterated for rest of the pieces of adjusted 512-bit and extended 512-digit block. The outcome is 512-cycle block as demonstrated underneath.

```cpp
for (tt1 = 0; tt1 < 16; tt1++)
{
    if (tt1 == 0)
        WW3[tt1] = ((WW3[tt1]) ^ (EE));
    if (tt1 == 1)
        WW3[tt1] = ((WW3[tt1]) ^ (DD));
    if (tt1 == 2)
        WW3[tt1] = ((WW3[tt1]) ^ (CC));
    if (tt1 == 3)
        WW3[tt1] = ((WW3[tt1]) ^ (BB));
    if (tt1 == 4)
        WW3[tt1] = ((WW3[tt1]) ^ (AA));
    if (tt1 == 5)
        WW3[tt1] = ((WW3[tt1]) ^ (EE));
    if (tt1 == 6)
        WW3[tt1] = ((WW3[tt1]) ^ (DD));
    if (tt1 == 7)
        WW3[tt1] = ((WW3[tt1]) ^ (CC));
    if (tt1 == 8)
        WW3[tt1] = ((WW3[tt1]) ^ (BB));
    if (tt1 == 9)
        WW3[tt1] = ((WW3[tt1]) ^ (AA));
    if (tt1 == 10)
        WW3[tt1] = ((WW3[tt1]) ^ (EE));
    if (tt1 == 11)
        WW3[tt1] = ((WW3[tt1]) ^ (DD));
    if (tt1 == 12)
        WW3[tt1] = ((WW3[tt1]) ^ (CC));
    if (tt1 == 13)
        WW3[tt1] = ((WW3[tt1]) ^ (BB));
    if (tt1 == 14)
    
    WW3[tt1] = ((WW3[tt1]) ^ (BB));
}
```

Fig. 7. Formation of 160-Bit Block (IV') to 512-Bit Block (CV).
WW3[tt1] = (WW3[tt1]) ^ (AA);
if(tt1 == 15)
WW3[tt1] = (WW3[tt1]) ^ (EE);
}
}

4) Partition and Modification operation ($P_{op}$): Each 32-digit square of the above outcome is additionally allocated 2 sub-squares of 16-bits each. By then every 16-bit block is segregated into two 8-cycle sub square, all of these eight pieces are changed that is First the four pieces are taken from the 8-digit regard starting from least basic position then it is XOR with a variable1(mm) whose value is zero, the result is kept in a variable2(mm1) also the assessment of variable1(mm) is invigorated for instance it is given the assessment of variable2(mm1). Ensuing to completing the above advance again from 8-digit value(input), four pieces starting from most critical position is taken care of in variable3(mm2) and variable3 is changed by playing out a XOR with variable1(mm) . The data is reproduced for example 8-cycle regard is recomputed by taking care of the assessment of variable3(mm2)(four bits starting from least basic situation) in four pieces starting from most huge position and next four pieces of early on eight bit(input) communicating from least critical position gets the pieces from variable2(km1) (four pieces starting from least huge position) so this way the 8-digit of data is changed . A comparative procedure is embraced for remaining three 8-bit regards made from 32 cycle that is

BB1 = (WW3[tt] >> 8) & 0xff; → 2nd 8- piece number made from 32 bit number
AA2 = (WW3[tt] >> 16) & 0xff; → 3rd 8 cycle number made from 32 bit number
BB2 = (WW3[tt] >> 24) & 0xff; → 4th 8 digit number made from 32 cycle number
AA1 → y1

int Mod_Int2(int y1)
{
    int i, mm=0, mm1=0, mm2=0;
    mm1=y1 & 0xf;
    mm2=(y1 >> 4) & 0xf;
    mm1= mm1 ^ mm;
    mm= mm1;
    mm2= mm2 ^ mm;
    y1 = (mm2<< 4) + (mm1);
    return yy1;
}

A same strategy is adopted for remaining seven 32-digit blocks.

5) New Point(on elliptic curve) estimation task ($NP_{fn}$): Each 32-digit block is separated into two sub squares of same length. These sub squares go probably as two motivations behind an elliptic curve which are used in new point appraisal in elliptic curve as exhibited in Fig. 5. First by using two point slope(λ) of line is resolved the this slant value(M) is used in the calculation of new X and Y center point on elliptic curve. The resultant new point is 16 cycle regard i.e X center is 8 pieces and Y center point is 8 pieces so both taken together is 16 digit sub square. An amount of sixteen 16-cycle sub square is created after new point computation. As of now this new point are changed by performing XOR on new point. Each 8-digit sub square is XOR with its adjoining 8-cycle sub square beside beginning 8-bit sub square which is XOR with a 8 bit sub square, everything being zero. After above action connecting sub squares are incorporated an especially that it becomes 32 cycle sub square. Repeating this, achieves eight 32-digit sub square.

Incline of line in elliptic curve (for real) is determined as:

\[
slope(\lambda) = \frac{(yy2 - yy1)}{(xx2 - xx1)}
\]

where (xxx1,yyy1) and (xxx2,yyy2) are points on elliptic curve. New point on elliptic curve for reals are calculated using the following formula

\[
xx_3 (new point) = (slope(\lambda))^2 - xxx_1 - xxx_2
\]

and

\[
yy_3 (new point) = slope(\lambda) * (xxx_1 - xxx_2) - yyy_1
\]

the above calculation is for the case when xxx1 != xxx2 (this is assumed that xxx1 is not equal to xxx2). The logic of new point estimation is demonstrated below.

for ( tt = 0; tt< 16; tt++)
{
    AA1 = WW3[tt] & 0xff;
    AA1=(Mod_Int2(AA1));
    BB1 = (WW3[tt] >> 8) & 0xff;
    BB1=(Mod_Int2(BB1));
    AA2 = (WW3[tt] >> 16) & 0xff;
    AA2=(Mod_Int2(AA2));
    BB2 = (WW3[tt] >> 24) & 0xff;
    BB2=(Mod_Int2(BB2));
    MM=(BB2-BB1)/(AA2-AA1);
    AA3[iii]=(MM*MM)-AA1-AA2;
    AA3[iii]=AA3[iii]^TT1 ;
    BB3[iii]=MM*(AA1-AA2)-BB1;
    BB3[iii]=BB3[iii] ^ AA3[iii];
    TT1=BB3[iii];
}
that is \(AA3[i]\) is recreated from \(AA3[i]\), \(BB3[i]\), \(AA3[i+1]\) and \(BB3[i+1]\) i.e. here \(AA3[i]\) is left shifted by 24 bits, \(BB3[i]\) is left shifted by 16 bits, \(AA3[i+1]\) is left shifted by 8 bits and \(BB3[i+1]\) is left shifted by 0(zero) bits and then \(AA3[i]\), \(BB3[i]\), \(AA3[i+1]\) and \(BB3[i+1]\) are added to generate \(AA3[i]\). Similarly \(BB3[i]\) is recreated from \(AA3[i]\), \(BB3[i]\), \(AA3[i+1]\) and \(BB3[i+1]\) i.e. here \(BB3[i]\) is left shifted by 24 bits, \(AA3[i]\) is left shifted by 16 bits, \(BB3[i+1]\) is left shifted by 8 bits and \(AA3[i+1]\) is left shifted by 0(zero) bits and then \(AA3[i]\), \(BB3[i]\), \(AA3[i+1]\) and \(BB3[i+1]\) are added to generate \(BB3[i]\). For recalculating \(AA3[i+1]\) again \(AA3[i]\),
BB3[i], AA3[i+1] and BB3[i+1] are used i.e. here AA3[i+1] is left shifted by 24 bits, BB3[i+1] is left shifted by 16 bits, AA3[i] is left shifted by 8 bits and BB3[i] is left shifted by 0(zero) bits and then AA3[i], BB3[i], AA3[i+1] and BB3[i+1] are added to generate AA3[i+1]. Finally for recalculating BB3[i+1] again AA3[i], BB3[i], AA3[i+1] and BB3[i+1] are used i.e. here BB3[i+1] is left shifted by 24 bits, AA3[i+1] is left shifted by 16 bits, BB3[i] is left shifted by 8 bits and AA3[i] is left shifted by 0(zero) bits and then AA3[i], BB3[i], AA3[i+1] and BB3[i+1] are added to generate BB3[i+1]. The same process is repeated for all the value of ‘ii’. Here ‘ii’ is incremented by 2 when the statements inside looping structure are rotated using ii

6) Result: The 160-piece hash code is acquired by adding four 32 bit sub square i.e. first and fifth, second and sixth, third and seventh and fourth and eight 32-bit sub square are added which brings about four 32-digit sub square. Presently this four 32-cycle block got in above interaction are numerically controlled which brings about last 32-digit sub square as demonstrated beneath. The message digest (160 pieces) is gotten by connecting the five 32-cycle block got in above interaction.

\[
EE = \sim (AA) \land (BB >> 5) \land \sim (CC) \land \sim (DD << 5);
\]

160-bit Hash Code = AA || BB || CC || DD || EE

All hexadecimal numbers are copied into pass on vector (or instate MD cushion), which is the message blueprint of a given message(if the message size is 1024-piece square) in any case the above cycle is emphasized until the last 1024-digit square of the message.

IV. ENCRYPTION ALGORITHM

Step 1: In trigonometry operation, for encryption we use a trigonometric formula made up of Cos function i.e. \(Cos(x \times pi / 180.0) = y\) then Sin function i.e. \(Sin(y \times pi / 180.0) = y1\), here we require the value of pi for Cos and Sin function.

At the time of encryption and decryption this value of pi will be needed. The actual value of \(\pi\) is 3.141592, but in this case the Sensor Nodeand mobile device(user) will take the value of \(\pi\) the current key generated from the secret key for session SK_m or SK_s. For example, let the value of key be 77.

Here is the method how the key for encryption or decryption is generated from secret key for session(SK_i).

1) First the secret key is stored in a variable and then this variable is rotated ten times, for each rotation it checks whether the rotation is even or odd. If even, the variable is right shifted by one and the shifted value is stored in a new array(xx), also the variable gets the current value of array x and the array is incremented. If odd, the variable is left shifted by one and the shifted value is stored in a new array(xx), also the variable gets the current value of array x and array is incremented. This process is repeated for a array of size ten, so at the end of first step we get ten different values generated from secret key. The process is shown below:

\[
y = sks
jj = 1;
for(ii = 0; ii < 10; ii++)
\]

2) Now the new array is analyzed that is from secret key ten different keys are generated in stage1, then these ten keys are XOR and the result of XOR operation is kept in a variable(t). After computing the value of t, the new array is modified that is array xx first position will now hold the value of array xx second position and array xx second position will now hold the value of array xx third position and the process continues till ninth position tenth array xx position will hold the value of variable t. After completion of above step one position of array xx(tenth position) is altered, the same process is repeated for all the position(ninth to one position) of array xx. So after stage2 the ten different key generated in step one will be holding all new values.

\[
jj = 1;
for(jj = 0; jj < 10; jj++)
\]

www.ijacsa.thesai.org
3) The ten key generated in stage2 are reduced to five that is first key is XOR with tenth key and the result is stored in a separate array(kk1) in first position after a left shift by zero, then second key XOR with ninth key and result is stored in array(kk1) second position after a left shift by one and the process continues till stage2 ten keys are reduced to five. The process is shown below:

```
jj=10;
for(ii=0;ii<jj;ii++)
{
    kk1[ii]=(kk1[ii] + kk1[jj])<<ii;
    jj=jj-1;
}
```

4) The ten keys of stage2 are reduced to five in stage3 then in stage4 this five keys are XOR and the result is stored in a variable (kkmm). the output of stage4 will be the input to stage5.

```
kkm=0
for(ii=0;ii<5;ii++)
    kkmm=(kkmm^kk1[ii]) ;
```

5) Finally the result of stage4 is reduced to a number less than 10 by applying modular operation. This result as shown in Table II will be acting a key.

Shared secret key for session = 77

<table>
<thead>
<tr>
<th>TABLE II. GENERATED KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>KK1</td>
</tr>
<tr>
<td>KK2</td>
</tr>
<tr>
<td>KK3</td>
</tr>
<tr>
<td>KK4</td>
</tr>
<tr>
<td>KK5</td>
</tr>
<tr>
<td>KK6</td>
</tr>
<tr>
<td>KK7</td>
</tr>
<tr>
<td>KK8</td>
</tr>
<tr>
<td>KK9</td>
</tr>
<tr>
<td>KK10</td>
</tr>
</tbody>
</table>

Final value of Key =3.000000

Step 2: Take the input message, for example “24, 32, 56, 65, 73”. Here every element of input message will be encrypted by calling the trigonohash technique. The input(xx) is taken from input string to be encrypted, first the input is converted into radians by multiplying xx with pi/180.0 and the generated value is given to Cos function. The output of Cos function will be the input to Sin function. The output of Sin function is again manipulated which forms the final cipher text to be transmitted to destination. So the process of encryption can be described as four step process as illustrated below:

```
for(ii=0;ii<nx;ii++) // nx→ Number of input data
{
    yy11[ii]=cos(sz[i]*key/180.0); // sz[i] →ith Data
    yy12[ii]=sin(yy11[ii]*key/180.0);
    ss1=int(yy12[ii]);
    frac1= yy12[ii]-ss1;
    ss2 = Mod_Int3(ss1);
    yy12[ii]=ss2;
    ab1[ii]= hash1(sz[i]);
}
```

```
int Mod_Int3(int xx)
{
    int static r1;
    int AA1,BB1,AA2,BB2;
    AA1 =xx & 0xff;
    BB1 = (xx >> 8) & 0xff;
    AA2 = (xx >> 16) & 0xff;
    BB2 = (xx >> 24) & 0xff;
    r1 = (AA1 << 24) + (BB1 << 16) + (AA2 << 8) + (BB2);
    return r1;
}
```

These values(xx in radian) will go into the Cos (xx) , and the output of Sin (yy) after manipulating it with Mod_Int3 function form the cipher text. As we have changed the value of π = SK1 = 1.137201(for example) the result will be automatically changed as seen in the cipher text.

The general methodology for calculating cipher text is a four step process.

1) The first input is given to Cos(input*key/180.0) as Cos function takes input in radian form, the input is multiplied with π/180 here πis assigned the value of generated key.

2) The output of Cos function will be the input to Sin function i.e. the out will be given to Sin(cos output*key/180.0), as Sin also takes radian value the same procedure is adopted as above for generating output.

3) The output of Sin function is manipulated using Mod_Int3 function which rearranges the bytes in the input i.e in Sin function output least significant byte is placed most significant bytes place i.e interchanged then (least significant -1) byte is interchanged with (most significant -1) this is repeated for all the bytes /2 in the output. The output of
Mod\_Int3 is the final cipher text which is transmitted to destination along with the hash value(HA-160) of input.

Now if yy12[ii] and encryption algorithm are known to the attackers then also they will not be able to breach the information. Because the value of $\pi$ is not general like $\pi = 3.141592$ rather than it is 1.137201 (SK, or generated key) Experimental results are shown in Table III.

<table>
<thead>
<tr>
<th>Input Message</th>
<th>Key (pi)</th>
<th>CipherText</th>
<th>Hash value of Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>0.015350</td>
<td>05c8778d2274ca4940aca313c32a7</td>
<td>1ebf99b5971</td>
</tr>
<tr>
<td>32</td>
<td>0.014351</td>
<td>82817f1da244ca39d0dca32eb77</td>
<td>1eaf4654610d</td>
</tr>
<tr>
<td>56</td>
<td>0.009919</td>
<td>2814db78a2388a55c37d1a4ab26a</td>
<td>decbebe7bd68</td>
</tr>
<tr>
<td>65</td>
<td>0.007806</td>
<td>4f15be822088a4553ad1a5a323a</td>
<td>ddbb12e8bd8</td>
</tr>
<tr>
<td>73</td>
<td>0.005780</td>
<td>e514db78a2388a55c37d1a4ab26a</td>
<td>decbae87bd68</td>
</tr>
</tbody>
</table>

V. DECRYPTION PROCESS

Take the input message, for example “0.000256, 0.000239, 0.000165, 0.000130, 0.000096”. Here every element of input message will be decrypted by calling the trigonohash technique. The input(xx) is taken from input string to be decrypted, first in the input, the effect of Mod\_Int3 function of encryption module is nullified by arranging the bytes of input in original form by using Mod\_Int3 function of decryption module. This output is given to $\sin$ function and the output of $\sin$ function will go as input to $\cos$ function. The output of $\cos$ function is given to hash function(HA-160). The calculated hash code is matched with the received hash code, if matched then the output of $\cos$ function is accepted as the message transmitted by sender else rejected. So the process of decryption can be described as four step process as illustrated below:

```c
for(ii=0;ii<nx;ii++) // nx -> Number of input data
{
    ss1=int(yy11[ii]);
    frac1=yy11[ii]-ss1;
    ss2=Mod\_Int3(ss1);
    yy11[ii]=ss2;
    yy11[i]=yy11[i]+frac1;
    kk2[i]=asin(yy11[i])*(180.0/key));
    kk3[i]=(acos(kk2[i])*(180.0/key));
    qq1=int(round(kk3[ii]));
    ab2[i]=hash1[qq1];
}
int Mod\_Int3(int xx)
```

These values(xx) after rearrangement with Mod\_Int3 function will go into the $\sin$ (xx), and the output of $\cos$ (yy) after calculating the message digest of output and comparing it with received message digest, if the comparison is positive will form the plain text or message to be accepted by receiver. As we have changed the value of $\pi = SK_i = 1.137201$(for example), the result will be automatically changed as seen in the plain text.

The general methodology for calculating plain text is a four step process:

1) The input to decryption algorithm is manipulated using Mod\_Int3 function which rearranges the bytes in the input i.e input least significant byte is placed most significant bytes that means interchanged then (least significant -1) byte is interchanged with (most significant -1) this is repeated for all the bytes /2 in the input. The output of Mod\_Int3 is given as input to next stage.

2) The output of previous stage acts as the input to $\sin$ function i.e. the out will be given to ($\sin(aTan output)$*key/180.0), as $\sin$ also takes radian value the same procedure is adopted as above for generating output.

3) The output of $\sin$ function will be the input to $\cos$ function i.e. the output will be given to ($\cos(aSin output)$*key/180.0), as $\sin$ function also takes radian value the same procedure is adopted as above for generating output. The output of $\cos$ is given to Hash function(HA-160) for generating message digest, then the generated message digest is compared with received message digest(from sender), if matched then only the output of $\cos$ is accepted as received message(plain text) otherwise it is rejected. Experimental results are shown in Table IV.

<table>
<thead>
<tr>
<th>Ciphertext</th>
<th>Key(pi)</th>
<th>Hash value check</th>
<th>Plain Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.015350</td>
<td>3.0000</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>0.014351</td>
<td>3.0000</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>0.009919</td>
<td>3.0000</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>0.007806</td>
<td>3.0000</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>0.005780</td>
<td>3.0000</td>
<td>73</td>
<td></td>
</tr>
</tbody>
</table>

www.ijacsa.thesai.org
VI. MYSTERY KEY FOR SESSION UPDATE PHASE AND USER PASSWORD UPDATE PHASE

Right when the data correspondence time among sender and beneficiary methodologies update period, GN Server makes information for update secret key for next session(DUMK) and encodes it and sends it to both User and Sensor Node.

\[
\text{EIDUMKS}_s = X \text{ IDs} - \text{GN} \oplus \text{DUMK} \\
\text{EIDUMKU}_m = \text{MM}2 \oplus \text{DUMK}
\]

At that point server sends EIDUMKSs (encrypted data for update mystery key for next(new) meeting) to the Sensor Node and EIDUMKUm (encrypted data for update mystery key for next(new) meeting) to the User. Subsequent to getting this data, Sensor Node quits sending information and hangs tight for another mystery key meeting. Subsequent to getting the solicitation both User and Sensor Node suspends their activity and User sends \{ IDm, RPWm, σm \} to GN Server and the cycle of enlistment and verification and meeting key synchronization start which brings about formation of new meeting key(= second meeting so SKS2) at both at mobile device(user) and Sensor Node(Ss) additionally at this junction, if client needs to hold can change his password(PWm).

The aggregate of the above are the nuances of the lightweight arrangement. The arrangement consolidates three phases. In like manner approval stage, we present a grouping key (xx1, bm, XGN and XIDm - GN) and pre realized a solitary bearing hash function(H1). It can comprehend shared approval just as hinder various attacks. Variable gathering key is proper for different applications and prevents attacks satisfactorily.

VII. DISCUSSION OF SECURITY FEATURES

In this part, we show our protocol achieves some excellent functional properties and can resist well-known attacks. Besides, the comparative analysis of our protocol and other relevant protocols are also given in this part.

A. Proper Mutual Affirmation

The versatile device(user) and GN Server can validate each other in light of the fact that toward the beginning of verification, mobile device(user) himself checks if he is certified by contrasting a worth he makes i.e Am = A'm. The client registers A'm esteem from the information which he supplies at the hour of verification. The Am is pre figured worth done at the hour of enrollment from the data(Bm) given by GN server. So whenever figured value(A'm) is equivalent to preexisted value(Am) which implies that client is certifiable and when GN server registers MM5 = MM5, GN server gets a confirmation that the client is real. Also the sensor hub get an affirmation from the GN server when it checks MM7 = MM7. The GN server gets an affirmation about the sensor hub when it figures MM8 and contrasts it and the got MM8 numa from the sensor hub i.e MM8 = MM8 gives a confirmation to the GN server about the sensor hub. The portable device(user) get an affirmation about GN server when he figures MMP'M10 and contrasts it and the got MMP'M10 numa from the GN server. The GN server and mobile device(user) has a pre shared one way hash work H1( ), just real GN Server and portable device(user) realizes the arranged one way hash work H1( ).

and the GN worker and sensor hub has a pre shared one way hash work H1( ), just real GN Server and node(Ss) realizes the arranged one way hash work H1( ).

B. Quickly Identification for Unapproved Login

Here the plan the Sensor Node information can be gotten to by versatile device(user) simply in the wake of passing shared confirmation and key synchronization stage and toward the beginning of the stage the client's login(identity ID),secret word and biometric needs to pass the check. Some unacceptable secret key checking instrument permits the cell phone to rapidly recognize and dismiss unapproved login brought about by wrong secret word, and the confirmation cycle can be found in the Step 1 of shared verification and key synchronization stage. Nonetheless, the protocol [17] had no such instrument to recognize wrong secret word.

C. Oppose Mobile Device Misfortune Attack

In the event that an aggressor A gets Um's cell phone ,A can recover the parameters \{ Bm, Gen(.),Rep(.), XX, rm \} in the cell phone by utilizing power examination attack[18]. Here Bm = Am (RPWm ⊕ rm), RPWm = (PWm ⊕ rm).

Am = Bm ⊕ (RPWm ⊕ rm ), and σm is an irregular number extricated from Um's biometric information . As Bm, RPWm and Am are figured from obscure arbitrary numbers RPWm, rm , σm and H1(one way hash work), the assaulter A will be unable to figure these boundaries precisely which may bring about finish of validation stage as the variable A'm at the hour of verification may not match so our protocol can dodge cell phone misfortune assault.

D. Customer Mystery and Intractability

Believe that a login demand message \{ MM3, MM4,MM5 \} is snooped by aggressor A, where MM3= IDm ⊕ MM2, MM4=IDs ⊕ MM2 and MM5= H1(A' m) where MM2= bm*XX is critical and this key is the result of private key of user(bm) utilizing elliptic curve idea and XX is the public key of GN server utilizing elliptic curve idea i.e. key is made by Diffie-Hellman key exchange idea, assailant A can't recover MM3,MM4 until it have the data about the private key of GN Server or private key of mobile device(user). MM5 is created from Am which is known to just mobile device(user) and GN server and from hash code the assaulter A can't recover Am. So by utilizing private key and public key (elliptic curve idea and Diffie-Hellman key exchange) , symmetric mystery key of GN server(XGN) and one way hash work H1( ), the proposed convention accomplishes the component of client mystery. Here after GN Server gets the login data it can recover the Um recognizable proof (IDm) as it has the key additionally in this protocol first the client personality check is done carefully toward the beginning of validation and key synchronization phase(Am=A'm). Likewise the component of login request message relies upon bm(user private key), in this way aggressor A can't follow User(as no fundamental data is passed straightforwardly) from public channel. In any case, the scheme [17] can't accomplish the capacity of intractability.

E. Sensor Hub Mystery

In this plan, Sensor Node's personality isn't sent out in the open channel as plaintext. At the point when Um needs to get
to the Sensor Node(Ss) information, Sn character is scrambled when Um send the solicitation to GN server for correspondence with Sensor Node Ss which GN Sensor can recover by decoding it with comparing secret key. Any aggressor without the information on relating secret word can't get Sensor Node's personality IDs and the plan accomplishes the component of sensor hub namelessness.

F. Appropriate for Internet of Things Applications

In our plan, GN Server is a confidant in outsider for mobile device(user) and Sensor Node(Ss) and the verification and key synchronization stage is finished with the assistance of GN Server. With this the over-burden of both User and Sensor Node is diminished additionally it saves energy utilization of both portable device(user) and Sensor Node(Sn) (as the energy utilization of sensor hub is straightforwardly extent to the correspondence distance) also expands the life cycle of sensor nodes. Alongside when the common validation and key synchronization stage finishes, a meeting key SKS at both mobile device(user(Um)) and Sensor Node(Ss) is given by GN server, which can guarantee the resulting secure correspondence among Um and Ss. Consequently the portray design for our plan is reasonable for most IoT application. However, in some past related work [13], the client discuss straightforwardly with sensor hubs, it would lessen the existence pattern of sensor organizations and isn't reasonable for IoT applications.

G. Restrict Impersonation Attack

From the depiction of verification and meeting key Agreement stage, assailant A necessities the data of IDm and Am = (IDm ⊕ X_{GN}) to mimic as a real client to make a substantial login demand. However, from the description above, the proposed protocol provides the feature of user anonymity, and any gathering with the exception of GN can't get client's personality IDm. Likewise, Am must be recovered by Um from Bm utilizing PWm, rm and om or can be determined by GN server utilizing IDm and GNS(secret key of GN server) when he/she gets IDm. Therefore, aggressor A can't get Am without knowing required data, and our protocol can maintain a strategic distance from client impersonation assault. Additionally, to mirror as the GN worker, the private keys xx1 and GNS(secret key of server) is fundamental data for assailant A to produce substantial correspondence messages. Nonetheless, xx1(private key) and mystery key(X_{GN}) of GN server and known to GN server just, and the proposed convention can stay away from entryway impersonation assault.

H. Oppose Replay Assault

In this plan, the single direction hash work system is received to oppose replay assault. In various strides of confirmation and key synchronization period of the plan, the hash code is created by Um, GN Server and Sensor Node for processing correspondence messages. Since the hash code carefully follow avalanche effect little change in message will create gigantic change in hash code so it would be hard for an assailant A to figure the shared message on which the hash code are produced, so it is hard to perform replay assault. Consequently, our plan is secure from replay assault.

I. Simple Secret Key Change

In our plan, as the update time frame for key finishes, the GN worker starts a solicitation for change of mystery key for meeting. For this the GN server sends encoded message to User and sensor hub, so the cycle of progress of secret phrase begins. In the wake of accepting the solicitation both mobile device(user) and Sensor Node(Ss) suspends their activity and User sends \{ IDm , RPWm, om\} to GN Server and the cycle of enlistment and confirmation and meeting key synchronization start which brings about production of new meeting both at User and Sensor Node. At this intersection the mobile device (user) can likewise change his password (PWm).

<table>
<thead>
<tr>
<th>TABLE V. PARAMETERS VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters Values(in bits)</td>
</tr>
<tr>
<td>IDm - 80</td>
</tr>
<tr>
<td>IDs – 160</td>
</tr>
<tr>
<td>X_{GN}, X_{IDS,GN}PWm(All Password) - 160</td>
</tr>
<tr>
<td>bm,xx1,XX,MM1 - 320</td>
</tr>
<tr>
<td>Hash value – 160</td>
</tr>
</tbody>
</table>

Note: Values taken as per Xiong-Li[21]

VIII. PERFORMANCE ANALYSIS

A. Process Computation Time

We look at the time of the common attestation stage, which is executed using[19][20] in the three conventions (DES algorithm, Xiong-li scheme(2017) and our convention). Analysis, were led on a PC with windows7 32bit, 2.00 GHz with 2 GB of internal memory utilizing C/C++ language. The yield says that our plan accomplishes the best time execution of 31000 μs(microseconds) as shown in Fig. 8, DES algorithm 31000 μs(microseconds) and Xiong-Li(2017) common attestation time is 31000 μs(microseconds) , which is shown in the Fig. 9 beneath.

Our proposed plot is having less overhead, and can restrict various attacks. Researching Xiong-Li(2017)[21] basic affirmation in this arrangement it allows the sensor device to go about as server as it makes the mysterious expression and accommodates customer and does an incredible arrangement getting ready which is illicit of lightweight count rules. Also ChangLi(P1) plan like shared confirmation isn’t done suitably, faces stolen smart card attack, further more encounters tracking attack, not applicable to practical application, delayed and costly detection of wrong password input, lack of identity detection mechanism and unfriendly mystery key change. In our arrangement, we simply use XOR and XNOR action, one way Hash work and elliptic curve cryptography methodology for shared approval and meeting key synchronization. Moreover, session key is created by server and synchronized with IoT node and user (mobile device). Fig. 9 shows that our scheme takes minimum time for authentication. It is a nice technique to minimize the overhead of IoT hub.
B. Performance Analysis of Hash Algorithm (HA-160)

These three calculations HA-160, SHA1, and MD5 were tried for comparison dependent on the execution time necessities as shown in Fig. 10, Fig. 11 and Fig. 12. All the calculations have been actualized in C/C++ and run on windows 7 32-bit, CPU 2.00 GHz with 2 GB of internal memory. With the aftereffects of the analysis, it was discovered that SHA1 and MD5 requests more execution time than HA160 to create hash code.

The outcome in above Fig.11 shows time taken by three calculation (MD5, SHA1, and HA160(size of message taken by HA160 is double than SHA1)) for producing hash code for the message “The quick brown fox jumped on the lazy dog”.

Table VI shows the message digest conveyed by three calculation (HA160, SHA-1 and RIPEMD160) for the message "The quick brown fox jumped on the lazy dog". Table VII shows the adjustment in hash code when the message in Table VI is modified for example dog is dislodged with cog, HA-160 produces a hash code with change of paying little heed to hexadecimal attributes from e and f(starting from left showed up in solid), SHA-1 yield is a hash code with change of with or without hexadecimal attributes from 9 and 1(starting from left showed up in serious) and RIPEMD 160 produces a message digest with change of paying little mind to hexadecimal attributes from f (starting from left showed up in strong) and Table VIII shows the adjustment in hash code when the message in Table VI is adapted to occurrence dog is supplanted with eog, HA-160 produces a message digest with change of each hexadecimal quality , SHA-1 yield is a hash code with change of with or without hexadecimal attributes from f, 2, 4 and 3(starting from left showed up in unprecedented) and RIPEMD 160 produces a message digest with change of with or without hexadecimal qualities from 4 and 4(starting from left showed up in strong).So on an ordinary HA-160 shows better result when any character is changed. The result is better than SHA160 and RIPEMD160 as shown in Fig. 11.
Table VI. Result

| Algo/Input: “The quick brown fox jumped on the lazy dog” HA160: 5abc142 e821f1a2 49860829 85f46818 697e332 SHA1: 429146fd 2297adbf d4ab4b14 f2eb3b0d 6a74963 RIPEMD160: 39349b80 fc87e4c4 87edec87 4943f253 e6051126 |

Table VII. Result (in Input Dog Changed to COG)

| Algo/Input: “The quick brown fox jumped on the lazeeg” HA160: 5abc142 e821f1a2 49860829 85f46818 697e332 SHA1: 429146fd 2297adbf d4ab4b14 f2eb3b0d 6a74963 RIPEMD160: 39349b80 fc87e4c4 87edec87 4943f253 e6051126 |

Table VIII. Result (in Input Dog Changed to EOG)

| Algo/Input: “The quick brown fox jumped on the lazeeg” HA160: 5abc142 e821f1a2 49860829 85f46818 697e332 SHA1: 429146fd 2297adbf d4ab4b14 f2eb3b0d 6a74963 RIPEMD160: 39349b80 fc87e4c4 87edec87 4943f253 e6051126 |

Table IX. Communication Cost of Our Proposed Mechanism

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Communication Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Scheme</td>
<td>3136 bits</td>
</tr>
<tr>
<td>Xiong-Li[21]</td>
<td>2688 bits</td>
</tr>
<tr>
<td>Chang-Le’s[21]</td>
<td>2400 bits</td>
</tr>
</tbody>
</table>

C. Communication Overhead

To investigate the correspondence overhead, we expect that the eight messages (i.e., Message1, Message2, Message3, Message4, Message5, Message6, Message7 and Message8) are communicated during the enrollment system and the confirmation method. All the more accurately, Message1 and Message2 are sent in the enrollment strategy and Message3, Message4, Message5, Message6, Message7 and Message8 are communicated in the verification technique. Message1 incorporates the sensor’s character IDm, RPWm(encrypted password) and uniform random number generated from Biometric information(σm) and Message-2 includes Bm((a value generate from GWN server from Am) and XX(public key of GWN Server) . In addition, Message3 includes MM1, MM3, MM4 and MM5 which are calculated as follows:

| MM1 | = | bm*PP |
| MM3 | = | MM2 ⊕ IDm |
| MM4 | = | MM2 ⊕ IDs |
| MM5 | = | H1(A’ m) |

Moreover, the parameters MM6 and MM7 in Message 4, are calculated as follows:

| MM6 | = | X'I IDG.N ⊕ ID’m |
| MM7 | = | H1(X’ IDG.N) |

then, the parameter MM8 in Message 5, is calculated as follows:

| MM8 | = | H1(X’ IDG.N ⊕ ID”m) |

also, the parameter MM9, MM10 and MM10 in Message 6, is calculated as follows:

| MM9 | = | X’ IDG.N ⊕ SKS |
| MM10 | = | MM2 ⊕ SKS |
| MM10 | = | H1(A”m ⊕ ID’m) |

Finally, the parameter MM11 in Message 7 and MM12 in Message 8 is calculated as follows:

| MM11 | = | H1(MM10) |
| MM12 | = | H1(MM9) |

Table V contains the setting of boundaries that we have accepted for assessing the correspondence overhead of the proposed component as shown in Table IX. Subsequently, the general transmission capacity overhead of the proposed component is determined as follows:

\[ bw = \sum \text{Message } i \]

\[ i = 1 \]

Message 1 = |IDm| + |RPWm| + |σm| (random number) = 400 bits

Message 2 = |Bm| + |XX| = 480 bits

Message 3 = |MM1| + |MM3| + |MM4| + |MM5| = 1120 bits

Message 4 = |MM6| + |MM7| = 320 bits

Message 5 = |MM8| = 160 bits

Message 6 = |MM9| + |MM10| = 480 bits

Message 7 = |MM11| = 160 bits

Message 8 = |MM12| = 160 bits

D. Computational Expense

To figure the computational expense of the proposed system, we have thought about the accompanying documentations: Th indicates the expense of single direction hash work, Tm signifies the expense of Multiplication activity, Ts means the expense of encryption and unscrambling activity. In view of the three parts (i.e. brilliant sensor, client and verification worker) which are utilized in the proposed instrument, the computational expense of every segment is introduced as follows:

1) Smart Sensor: The brilliant sensor performs calculations just in the verification method.

2) User: The client performs calculations in the enlistment and verification stage.

3) GN Server: The GN server performs calculation activities in enlistment and validation stage. The proposed component’s computational expense is delineated in Table X. Because of the way that the proposed verification system depends just on scalar duplication activity utilizing ECC, a symmetric encryption/decoding activity and hash activity.

We use the simulation result [23] where Tm(cost based only on scalar multiplication operation using ECC) = 1.226ms, Ts(cost based only on symmetric encryption/decryption operation) = 2.049μs and Th(cost based only on hash operation) = 2.580 μs.
TABLE X. COMPUTATIONAL COST OF OUR PROPOSED MECHANISM

<table>
<thead>
<tr>
<th>Scheme</th>
<th>User(Mobile device)</th>
<th>GWN Server</th>
<th>Smart Sensor</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Scheme</td>
<td>3<em>Th + 2</em>Tm</td>
<td>6*Th +</td>
<td>3*Th</td>
<td>2482.96 µs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1*Tm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xiong-Li[21]</td>
<td>7<em>Th+ 2</em>Tm +</td>
<td>8*Th +</td>
<td>5*Th</td>
<td>3743.412 µs</td>
</tr>
<tr>
<td></td>
<td>2Ts</td>
<td>1*Tm + 4Ts</td>
<td>+2*Ts</td>
<td></td>
</tr>
<tr>
<td>Chang-Le'[21]</td>
<td>6<em>Th +2</em>Tm</td>
<td>8*Th</td>
<td></td>
<td>4953.000 µs</td>
</tr>
</tbody>
</table>

Note: Considering the process computation time for mutual authentication, computation cost, communication cost and other indicators, our algorithm performs well in most of the cases. The most prominent advantage is its process computation time for mutual authentication and communication cost.

E. Encryption and Decryption Algorithm Analysis

The investigation of the proposed trigonohash calculation for encryption and unscrambling has been done and exhibited in Table III and Table IV. The Encryption and Decryption Algorithm was coded in C/C++ Language[19][20]. It was compiled with MinGW-GCC 4.8.1, on the Core 2 Duo Processor, 2.00 GHz under windows 7 OS(32 bit). The investigation boundaries are plain content size (in Bytes) and time taken in encryption and unscrambling (in µ seconds).

a) Comparative Performance Analysis: The proposed Trigonohash Algorithm(LW-algorithm) has been compared with other existing algorithms like RC4, Hill-Cipher[19][20], RSA, Present(Block Cipher with key 80 bits and plain content 64 bits or 8 bytes) [27] and ELSCA(LW-Stream Cipher) - (Variant-HassanNoura(2019(key consist of 4 location)) stream cipher)][22]. Likewise, the results are shown below. From the going with Fig. 13 and Table XI we can say that the run time unpredictability of trigonometric Algorithm is underneath than the other existed Algorithm plot.

TABLE XI. SAMPLE RESULT FROM DIFFERENT ALGORITHM.(RESULT IN MICROSECOND)

<table>
<thead>
<tr>
<th>No of Bytes</th>
<th>RC4 (stream cipher)</th>
<th>Hill-Cipher (Symmetric encryption)</th>
<th>RSA (public key encryption)</th>
<th>Present(LW-Block Cipher(64 bits))</th>
<th>ELSCA(LWStreamCipher-2017)</th>
<th>Trigonohash(LW-algorithm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>31000</td>
<td>16000</td>
<td>31000</td>
<td>47000</td>
<td>15000</td>
<td>00000</td>
</tr>
<tr>
<td>20</td>
<td>46000</td>
<td>46000</td>
<td>31000</td>
<td>78000</td>
<td>46000</td>
<td>16000</td>
</tr>
<tr>
<td>28</td>
<td>62000</td>
<td>63000</td>
<td>47000</td>
<td>125000</td>
<td>47000</td>
<td>31000</td>
</tr>
<tr>
<td>36</td>
<td>93000</td>
<td>78000</td>
<td>78000</td>
<td>156000</td>
<td>93000</td>
<td>62000</td>
</tr>
</tbody>
</table>

In this paper, we propose a strong and weighty cryptographic arrangement for Industrial Internet of Thing using one way hash work, elliptic curve cryptography and trigonohash(trigonometry and hash work) idea and endeavored to improved security and execution of a cryptographic arrangement for lightweight contraption using key generation. It gives keen and feasible instruments for basic affirmation, meeting key synchronization and meeting key update. They are proper for IoT hub with limited handling resources and force. First thing, we present the nuances of shared affirmation using elliptic curve cryptography and one way hash work and a way to deal with hinder replay attack without timestamp. Besides, again elliptic curve cryptography and one way hash idea is progressed to simply recognize meeting key synchronization lastly by using mathematical thoughts data encryption and unscrambling are done to diminish the overhead of IoT center points. In addition, we have taken a gander at the security and execution of our arrangement with some lightweight approval and cryptographic plans. The arrangement is lightweight anyway can thwart attacks satisfactorily, which is useful for guaranteeing the security of the correspondence between IoT center points, users(mobile gadget) and server.

The proposed figuring for encryption isn't simply giving the quick data encryption anyway it gives an unrivaled security diverged from other count through a most grounded key as well. The assessment of $\pi$ is changing each time which is created from the common key that makes the count adequate. The estimation moreover makes the cryptanalysis cycle complex. Since there are dark variables will be more and the amount of conditions will be less when stood out from other standard encryption computations. This calculation could be well valuable for online applications like secure web based talking, data transmission between resource constraint devices and health sector.
REFERENCES


[22] Hassan Noura and Ali Chehab* An Efficient and Secure Variant of RC4 Stream Cipher Scheme for Emerging Networks“ 978-1-5386-5670-0/18/$31.00 c 2019 IEEE.


