The Hybrid Combinatorial Design-based Session Key Distribution Method for IoT Networks

Gundala Venkata Hindumathi[1], D. Lalitha Bhaskari[2]
Department of Computer Science & Engineering, JNTUK, Kakinada, India[1]
Department of Computer Science and Systems Engineering[2]
Andhra University College of Engineering (A),Andhra University, Visakhapatnam, India[2]

Abstract—**Internet of Things (IoT) is currently being used in a range of applications as cutting-edge technology. IoT is a technological platform that connects the physical and digital worlds, allowing us to use things remotely. Various sensor-connected nodes serve as objects that communicate with one another over the internet. Hence security-related problems are more likely to arise in IoT networks. However, due to resource constraints such as power and memory capacity, complex security algorithms cannot be implemented in IoT networks. One of the security measures for IoT networks is to implement the lightweight key distribution algorithm. The lightweight key management process is essential for IoT networks to share the key securely. We presented the new key-distribution approach based on the hybrid combinatorial design that implements lightweight algorithms and describes the analysis functions. The comparison to existing hybrid combinatorial works shows better connectivity, resilience, and scalability.**

Keywords—Key distribution; hybrid combinatorial design; IoT networks; resource constraint nodes; symmetric key generation

I. Introduction

The Internet of Things (IoT) is a system that allows multiple sensor nodes and wireless nodes to communicate without the need for human involvement. The term "things" in the Internet of Things refers to physical objects such as sensor nodes that monitor or access data from other networked devices. In the research aspect, IoT has been becoming a much-desired area. The security of each node's data is the primary issue in today's rapidly growing IoT networks.The security services are like confidentiality, authentication, and integrity of the data. Cryptographic algorithms and keys are required for encryption, and effective key management is essential for this process to work appropriately. Ineffective key management can make even the strong algorithms useless for any type of network. IoT networks also need to have strong key management procedures.

Even though key management is essential for IoT networks, using conventional key management methods demands more memory. Due to resource-constrained nodes' memory and battery limits, the IoT network requires a lightweight solution.Thus, we discussed about lightweight approaches that are already in use for key management.Basic methods to generate and distribute the keys to nodes in the network are symmetric keys and public keys. Even though the public key approach is widely used for key distribution, it could not be used often in IoT networks since it requires more memory and processing resources to run the code, and in many applications, these approaches are also costly. Hence, the Majority of IoT networks are using symmetric key distribution methods, which require only one key to share as mentioned by Alagheband et al. [1].

There are two methods for sharing keys amongst connected nodes: decentralized and centralized approaches. In the decentralized process, Nodes in the network can share their secret keys directly with one another to provide secure communication. Every node should hold private keys that are unique for communicating with each node in the network. Those private keys are exclusive to committed pairs only. However, as IoT networks grow, devices will be unable to keep as many secret keys in memory due to the restricted memory space of IoT nodes.

Another option for resolving this problem is to use a trustworthy centralized device to distribute private keys to all nodes in the network. Key Distribution Center (KDC) is an example of providing centralized service. Kouicem et al. [2] presented that the KDC is a mechanism that distributes keys to all the users in a network sharing sensitive or confidential information. When two nodes in a network need a connection, they request the KDC to generate a unique session key that end users can use as a secret key for communication. So, the nodes can share the data with other nodes connected to the network using Key Predistributions or KDC.

As a result, using a KDC with symmetric key distribution is the best way to distribute the key to all nodes. One of the best symmetric key generation approaches is combinatorial block designs.It uses a simple calculation to compute the blocks for different nodes. Many Authors have been working on this for determining the keys for multiple nodes.In the introduction, we covered the fundamental ideas of combinatorial block design, how the authors expanded these ideas to implement keys for every node, and a brief discussion on our approach.

Stinson et al.[3] used Balanced Incomplete Block Design (BIBD) which is one of the combinatorial designs to generate the blocks for sharing the keys securely with other nodes. When it is impossible to incorporate all treatments or factor combinations for every block, then BIBD is utilized here.

Assume there are b blocks, each with k keys, and v total number keys can be used, each key replicated r times. Thus,

$$br = vk$$

And also assume that the blocks (b) are just partially complete by confining with the following conditions.

1) $k < v$
2) In any block, the same key doesn't appear more than once.

$\lambda_{ij:}$ i and j are two different keys from the 'v', it gives the occurrences among the blocks.

Example 1: $v = 6, b = 4, k = 3, r = 2$

v={1,2,3,4,5,6}, b=no.of blocks, k=keys in each block,r= each key repitations in blocks.

So the Blocks are

b1:{1,2,3} ,b2:{1,4,5}, b3:{2,4,6}, b4:{3,5,6}

λ_{14}=1, λ_{46}=0 (It gives the pair occurrences in blocks.

In a Balanced Incomplete Block Design: $\lambda(v-1) = b(k-1)$.

Symmetric BIBD:

A BIBD is said to be Symmetric BIBD when $b = v; k = , r, \lambda = 1$

Example 2:

Consider (v,b,k,r,λ)=(7,7,3,3,1) because v=b;k=r

V(keys)={1,2,3,4,5,6,7}

b1:{1,2,3}

b2:{1,4,5}

b3:{1,6,7}

b4:{2,4,6}

b5:{2,5,7}

b6:{3,4,7}

b7:{3,5,6}

Another combinatorial method is the finite projection plane. A Finite Projection plane consists of P points and set of subsets of P called lines. A prime integer q (>=2) and that has four properties.

1) Every line should be having exactly q+1 points
2) Every point occurs on exactly q+1 lines
3) Exactly $q^2 + q + 1$ points used
4) Exactly $q^2 + q + 1$ lines used; then that can be called Symmetric Design with $(q^2 + q + 1, q + 1, 1)$ given by Stinson et al. [4].

Already existing key predistribution methods are mainly followed by three procedures.

1) Probabilistic: Keys are chosen randomly from the pool and assigned to the nodes.
2) Deterministic: Based on pre-defined procedures select the keys and assign them to the nodes.
3) Hybrid Approach: The combination of both approaches is mentioned above.

The KDC implements key predistribution methods to get the keys for all nodes. The pre-key distribution can be acquired based on the key-Matrix approach by Chien et al. [5], So it helped share the key easily. Other pre-key distribution approaches are Blundo et al. [6] and Liu et al. [7], In these, Polynomial-based key pre-distribution was proposed for group key establishment. In Chan et al. [8], Two nodes having q keys should be linked, and the hash value of the q keys would be used for key verification that improved resilience from the attackers. Qian and Sun [9] presented the drawback of the above approach is that resilience increased but wouldn't guarantee to get the common key between two devices. Li et al. [10] was provided threshold value for random key pre-distribution in which each should communicate with its neighbor node with the same key. Catakoglu et al. [11] increased the resiliency of the previous system by adding numerous key rings.

Camtepe andYener [12], first time they presented the symmetric balanced incomplete design(SBIBD) for generating the keys for nodes in the network, however, the disadvantage is the scalability of the network with nodes. In comparison to prior techniques, Lee et al. [13] exhibited improved resilience.Ruj et al. [14] generated the pre-key distribution method using the partial BIBD technique, however, it did not share the keys with every node in the network. Ruj et al. [15], the same authors proposed a combinatorial strategy for improving BIBD and PBIBD resilience. Bechkit et al. [16] employed a new pre-key distribution design, a combinatorial-based way to determine the keys, which improved the scalability and connectivity.Bahrami et al. [17] presented great scalability of the network nodes by using residual key pre-distribution design for key pool generation.

Camtepe et al. [18] presented a combinatorial method for generating keys for network nodes that are connected. And they used SBIBD and GeneralizedQuadrangle (QD), which are the basictwo deterministic key pre-distribution designs. Complete connectivity between network nodes was the improvement of this algorithm. Also provided is the hybrid pre-key distribution method.Chakrabarti et al. [19] and Kavitha et al. [20] enhanced the scalability and connectivity of the previous approach.Dargahi et al. [21] enhanced the hybrid method to get the keys for almost all network nodes, but didn't get the exact number of keys to all network nodes. When compared to prior hybrid techniques, Akhbarifar et al. [22] used a hybrid strategy and provided improved connectivity and resilience. However, the unique keys were not generated for nodes in the network.

Despite the fact that combinatorial designs have been addressed extensively, not all linked network nodes are given the session keys. Every IoT network needs to be able to enable the construction of many nodes and should distribute a session and a unique key for every node.By supplying unique and dynamic keys for each node, we suggested a hybrid combinatorial method that resolves the problems discussed earlier. As a result, our system now supports network scalability.

Our entire method is detailed in a total of six sections:

- Secton 1 gives the introduction part of basic methods for Combinatorial block designs.
- Section 2 explains the existing hybrid approaches and their drawbacks in detail.
- Section 3 is our actual work to be implemented to generate the unique and session keys for every node.
- Section 4 gives the analysis of scalability, connectivity,resilience, and Memory utilization. And also provides the results analysis with graphs.
- Section 5 is a complete discussion.
- Section 6 is a conclusion.

II. Related Works on Hybrid Combinatorial Designs

Camtepe and Yener [12] proposed a first-time pre-key distribution strategy based on the SBIBD technique. It was the basic combinatorial design to get the keys for network nodes.

Assume there are b blocks, each with k keys, and v total number keys can be used, each key replicated r times.The following criteria were used to allocate keys to the nodes in the proposed algorithm.

$q^2 + q + 1 = length\ of\ keypool(v); here\ q\ is\ prime$

$q^2 + q + 1 = number\ blocks(b); here\ q\ is\ prime$

$q + 1 = \ keys\ assigned\ to\ the\ each\ block(k)$

$q + 1 = \ In\ the\ blocks, every\ key\ is\ repeated\ (r)$

The fundamental advantage of this approach is that it identifies the unique keys among the b nodes. This technique had good connectivity and resilience, but it lacks scalability. However, this strategy has the disadvantage of limiting the total number of blocks that meet the before-mentioned criteria. As a result, it was completely reliant on the q value. This approach could not identify the keys for all n nodes in the network; where N is the total number of network nodes, and that was not meet the above condition.Although this method cannot be applied to all of the network's nodes, it accurately delivers the keys for the limited number of nodes.

In Camtepe et al. [18] (HSYM), the previous approach was upgraded by including scalability and resilience properties. It was implemented using a hybrid technique that enhanced the number of nodes in the IoT networks. It could find the b blocks by using SBIBD and this method found the complimentary design for all symmetric blocks then chose q+1 keys and assigns them to the remaining nodes.The author's implementation is described in Algorithm1.The fact that more nodes have a chance of acquiring the same key reduces the probability of obtaining a key share, which is a drawback of this technique.

Algorithm 1: Hybrid Design of HSYM

```
Input(s): N (Total Number of nodes)
Output(s): K (Block size)
Begin
    1. Find largest prime power $q$ such that $k \leq K$;
    2. Generate base Symmetric
        • v objects $P = \{a_1, a_2, a_3, \ldots . a_v\}$
        • b blocks B= $\{B_1, B_2, B_3, \ldots . B_b\}$of size k;
    3. Generate Complementary Design of the base design:
       Blocks $\bar{B} = \{\bar{B}_1, \bar{B}_2, \bar{B}_3, \ldots, \bar{B}_b\}$where $\bar{B}_i = P - B_i$ and
       $|\bar{B}_i| = v - k for\ 1 \leq i \leq b$;
    4. Generate $N - b$ hybrid blocks
       $H = \{\bar{H}_1, \bar{H}_2, \bar{H}_3, \ldots, \bar{H}_{N-b}\}$ of size k. For ith block
       $H_i$where $1 \leq i \leq N - b$:
        • Randomly select a block in $\bar{B}$, say $\bar{B}_j$
        • Randomly select a k-subset $\gamma$ of the block $B_j$
          where $\gamma \notin H$,
        • Let $H_i = \gamma$ and $H = H \cup H_i$,
        • Use the variable $s_i$to hold index of the block
          $\bar{B}_j$from which the block $H_i$is obtained;
    5. Blocks of the Hybrid Design are $B \cup H \Rightarrow K$
End
```

Dargahi et al. [21] (MHS) proposed an enhancement version of the above hybrid approach. For b blocks, they also used the same BIBD method. For the remaining nodes in IoT networks, they used a different key pool. N-b times, they have chosen q+1 keys from the new key pool that were assigned to additional N-b nodes.The generation of the blocks is described in Algorithm 2.The authors have used more space in the node memory to store the extra keys and new key pool in the nodes, but we all know, that IoT devices have limited capacity.

Algorithm 2: Hybrid Design of MHS

```
Input(s): N (Total Number of nodes)
Output(s): M Blocks
Begin
    1. Find the largest prime number $q$ Where $q^2 + q + 1 < N$
    2. Generate the first symmetric $(q^2 + q + 1, q + 1,1)$-
       BIBD with the following key pool:
        • $KP_1 = \{K_1, K_2, \ldots, K_V\}$Containing
          v objects,
    3. Generate b blocks $B = \{B_1, B_2, \ldots, B_b\}$ from $KP_1$;
    4. Choose a number d where $0 < d \leq q^2 + q + 1$;
    5. Generate the second symmetric $(q^2 + q + 1, q + 1,1)$-
       BIBD with the following key pool:
        • $KP_2 = \{K'_1, K'_2, K'_3 \ldots, K'_V\}$ Containing v
          objects
        • $KP_2$ is generated in a way that d keys differ
          from $KP_1$ and other keys are the same,
    6. Generate b blocks $M = \{M_1, M_2, \ldots, M_b\}$ from $KP_2$;
    7. Assign b blocks from B to b nodes $(b < N)$;
    8. Choose$(N - b)$blocks from $M$ in a random manner and
       assign them to $N - b$ remaining nodes
End
```

Akhbarifar et al. [22] (MHSYM) proposed a new enhancement of the previous proposes. They identified two random blocks in b, combined their blocks data, then extracted the q+1 keys from it. Then allocated each of the remaining nodes with a random selection of q+1 keys.Algorithm 3's detailed explanation of the entire process. The key share probability was increased as compared to Camptepe [18] method, but there is no guarantee that at least one common key would be allocated among the blocks. The complete procedure explained in Algorithm 3.

Algorithm 3: Hybrid Design of MHSYM

```
Input(s): N (Total Number of nodes)
Output(s): M Blocks
Begin
    1. Find the largest prime number $q$
    2. Where $q^2 + q + 1 < N$
    3. Generate the first symmetric $(q^2 + q + 1, q + 1, 1)$-BIBD with the following key pool:
        • $KP_1 = \{K_1, K_2, \dots, K_V\}$ containing $v$ objects,
    4. Generate $b$ blocks $B = \{B_1, B_2, \dots, B_b\}$ from $KP_1$ and assign them to $b$ nodes;
    5. Choose two blocks among $b$ blocks randomly;
    6. Merging two blocks to construct new key-pool $M$;
    7. Select $(N - b)$ blocks among $q + 1$ subsets of $M$ and assign them to $N - b$ remaining nodes.
End
```

As mentioned above, the Procedures to apply the hybrid combinatorial design won't generate keys for all blocks of nodes.Our method uses a limited memory source to provide session keys for all blocks of nodes.The proposed work covers related algorithms and also provides examples for key generation.

III. Our Proposed Work

The Symmetric BIBD (SBIBD) allows multiple users in the same network to share the same keys without causing any problems. The IoT Architecture has not been supporting for huge capacity of memory inbuilt and high processing devices. Because of the above-mentioned reasons, the IoT node connected to the network is unable to remember all of the keys required for communication with other nodes in the network.

The SBIBD allows for the storage of the smallest amount of keys on the devices themselves, however, scalability is an issue here. If the network grows larger, nodes will be unable to store numerous keys in the tiny size memory. So, we are providing a new solution to this problem, a centralized system called Dynamic Key Generation and Distribution Center (DGDC). And the whole design that we suggest is depicted in Fig. 1.

In the context of IoT, we describe the symmetric key authentication and key management system based on BIBD. In this paper, we present a technique for exchanging the secret key that uses for providing the different security levels to assure scalability and confidentiality. We propose a technique for key agreement between two IoT devices that have never been in contact before, based on trusting the centralized server or using a proxy-based approach.

Fig. 1, describes the overall architecture that we have implemented to generate the session key and distribute it to the host which is requestedto the centralized server. The diagram itself is made up of three different blocks: DGDC, Initiate System (A) which starts to set up the communication connection, and Destination System (B) which accepts data from User (A) after receiving the Session key from DGDC.

The connected systems first exchanged their symmetric key to communicate with the centralized block, which is DGDC. Before implementing this architecture, the symmetric keys (secret keys for authentication) should be shared with DGDC so that other systems already connected to the network can communicate with it. Hence, this step is really important for our design because it is also providing authentication.Key generation and Key Distribution are the two main components of DGDC's actual work.

For Generating the keys, DGDC always works on the below-mentioned algorithms to implement the symmetric keys for all connected nodes. The previous algorithms mentioned in the related works are not implementing unique keys for all connected nodes.It is a pioneering building component for dynamic key implementation and distribution, increasing data security by often changing node keys.

In the DGDC, Data generation block contains all of the modules that have been proposed to create dynamic and unique keys for data transactions carried out by connected nodes. The modules are:

1) SBIBD,

2) Building the remaining nodes,

3) Computing the unique keys for each node in the network using a hybrid combinatorial design approach,

4) Reconstructing the outgoing blocks of nodes to protect the keys that have been compromised.

To create a complete table with unique keys for every node, DGDC executes each module in the order that they are presented.Once the table has been built, DGDC verifies requests using secret keys before sending the session key to the requested nodes.

Here, the architecture also proposed by us gives more security levels to the data because the session keys are not known by each individual connected system in the network. If an attacker compromises one of the systems, the attackers are unable to identify the session keys from the compromised system as it never stores any keys in their systems.

In particular, eight steps must be completed to observe the workings of our model. They are mentioned below in the Fig. 1. The model can generate and distribute the session key for communication between the request systems based on the mentioned processes. One of the most essential features of the proposed approach is the ability to dynamically alter the session keys of each system.

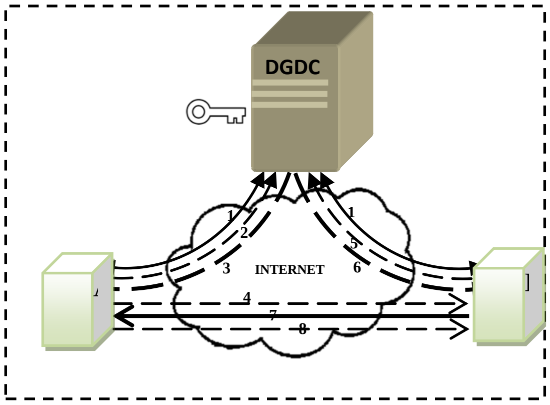

Fig. 1. Proposed Architecture.

1) Both users A and B identified their symmetric keys and exchanged them with DGDC for authentication purposes.

2) User A requests a session key from DGDC to communicate with User B.

3) By using a symmetric key, DGDC completes the authentication process and obtains the common key of both parties. And sends it to User A.

4) Using the Session key provided by the DGDC, User A transfers the data to User B.

5) Using its symmetric key, User B requests the session key from DGDC.

6) Like Step3, DGDC finishes its authentication process and provides the session key (which is already shared with A) to B.

7) User B uses the session key to decrypt the data provided by User A and provides the acknowledgment in an encrypted format.

8) The communication between User A and User B begins with the use of the same session key.

In the Proposed Work, The main required module is DGDC, it is generating the session keys dynamically and distributes them to the systems. First, we have to complete the code for generating the SBIBD with restricted blocks provided in Algorithm 4. The session keys for all nodes in the network could not be generated through the SBIBD procedure.

Where N is the number of network nodes used in communication. Calculate N>=q2+q+1, where q is the largest prime integer that may be used to solve the preceding equation; the result is v and b. Here, The 'N' and the 'v ' may not be the same. That is, the SBIBD algorithm was unable to determinethe keys for each node in the N network. SBIBD can be generated with v blocks and q+1 keys, which are represented by the k in each block.

The input for Algorithm 4 is N which is the number of nodes that need to be connected to the network, where v, k, and r are generated by the above Algorithm 4. The maximum number of nodes (blocks) in a network for generating session keys in SBIBD is represented by b. However, Algorithm 4 provides limited session keys for a few numbers of network nodes, therefore we are improvising by using other Algorithms 5, 6, and 7.

Algorithm 4: Design of SBIBD

```
Input(s): N (Total Number of nodes)
Output(s): B
Begin
   1. Choose the maximum prime number q to compute the below
      equation
                        $q^2 + q + 1 \leq N$
   2. Using the previous equation, generate inputs for producing
      the blocks.
      $v = q^2 + q + 1$; where v is the size of the key pool
      $b = q^2 + q + 1$; b is the number of blocks
      $k = q + 1$ ; k is the number of keys allotted to each block
      $\gamma = 1$; $\gamma$ denotes, In SBIBD, each node has only one
      shared key to communicate to other nodes in the B.
   3. Construct blocks B using Symmetric BIBD design
                        $B = SBIBD(v, b, k, \gamma)$.
             Then assign the blocks in $B = \{B_1, B_2, .., B_b\}$
End
```

Algorithm 5 completes the generation of remaining blocks of the network nodes. Algorithm4 computes the ' B ' number of blocks, while Algorithm5 will handle the rest.c=N-b; c is the number of blocks to be calculated, where N network nodes and b have already been given in Algorithm 4. Algorithm 5 determines which of the c number blocks should be assigned to the network's other nodes. In Algorithm 5, the R represents the remaining nodes of the IoT network.Select the keys from the key pool, and then place them as keys to generate the blocks by the requirements of Algorithm 5.

Algorithm 5: Design for remaining nodes(R)

```
Input(s): c,v,k
Output(s): R
Begin
  1. Construct the (v,N-b,k,r,γ); here v is the key pool, N-b
     blocks need to construct, k keys for each node, r
     repetitions among the blocks, γ = 2 or more; means each
     block in N-b should share two or more keys among the
     q+1 keys.
  2. As a result, each key from the key pool can only be used
     at most 3q times in the construction of N-b blocks.
  3. Then return R blocks from this Algorithm
                  $R = \{B_{N-b,} B_{N-b+1,} \dots, B_N\}$
End
```

The final blocks are represented by $H = B \cup R$ which is input for Algorithm 6 and also computed the key pair values for all resource-constrained nodes.

Algorithm 6 is used to generate the v number of keys, however, the remaining keys were not able to be generated directly. The remaining c keys are found and perform an XOR operation on the common keys that existed between the two nodes. At the end of Algorithm6, be able to find the unique session keys between each node in the network. Here, Algorithm6 uses 32 bit (8 bytes) key for computation as the IoT devices could be handled easily with this length.

```
Algorithm 6: Hybrid Combinatorial Design with Unique keys
Input(s): N,v,b,K,B
Output(s): H(Total no.of blocks), x (The Session Key)
Begin
1.  Execute Algorithm1 to get the $q^2$+q+1 Symmetric block within N blocks.
        $v = q^2 + q + 1$ (Number of keys used)
        $Key = \{K_1, K_{2,}.., K_v\}$
        $b = q^2 + q + 1$ (Number of nodes generated with the length of k by Algorithm1)
        $B = \{B_1, B_2, .., B_b\}$
2.  Generate N-b blocks using Algorithm2.
        $R = \{B_{N-b,} B_{N-b+1,} ..., B_N\}$
        $Key = \{K_1, K_{2,}.., K_v\}$
3.  Hybrid Design's Blocks are $H = B \cup R$
4.  Choose any two blocks from N (BB, BR, and RR) blocks randomly and determine the common key(s) of these blocks that should store in $l$.
    Example: Here we have taken two blocks $B_1$, $B_{N-b}$.
        $l = B_1 \cap B_{N-b}$
    Get the common keys that are presented in both blocks.
5.  (i) If the length of the $l$ is one then directly take the key as the secret key for both blocks.
        $if\ length(l) == 1;$
        $then\ x = l\ and\ x\ as\ a\ secretkey$
    (ii) If the length of the $l$ is above one, take the last two keys from blocks and do the XOR operation among those keys.
        $iflength(l) \geq 2$
        $thenx = l[length-1] \oplus l[length-2]$
        $x\ is\ a\ secretkey$
    (iii) If the length of the $l$ is null, select the first key-value from each block and calculate XOR between those keys. For example
        $if\ length(l) == 0$
        $then\ x = first\ key(B_1) \oplus first\ key(B_{N-b})$
6.  $x$ is the final secret key that is given by the DGDC.
End
```

The blocks for nodes are generated by DGDC up through Algorithm 6, and those key values in blocks aresent to nodes during transaction time. Once generated, they can be used every time, so there are chances of keys being compromised. Thus, We have also implemented an Algorithm 7 to get a solution for compromised keys by an attacker. Algorithm 7 illustrates how we can avoid attacks by utilizing a technique that shuffles the keys in the blocks in a certain amount of time.

```
Algorithm 7: Reconstruction of H
Input(s): H (Total blocks with keys)
Output(s): H blocks
Begin
1.  For every, Threshold time(T) changes the key values of nodes
2.  Shuffle all blocks of the H and assign the values of the block to nodes
        $H = shuffle(H)$ for $\Delta$T
3.  And shuffle each block key value of the H to get the session key from Algorithm3.
        $Apply\ \forall Bin(H);\ like$
        $H(B_i) = shuffle(H(B_i))$ for $\Delta$T
End
```

The complete workflow illustrates the DGDCs in Fig. 2.

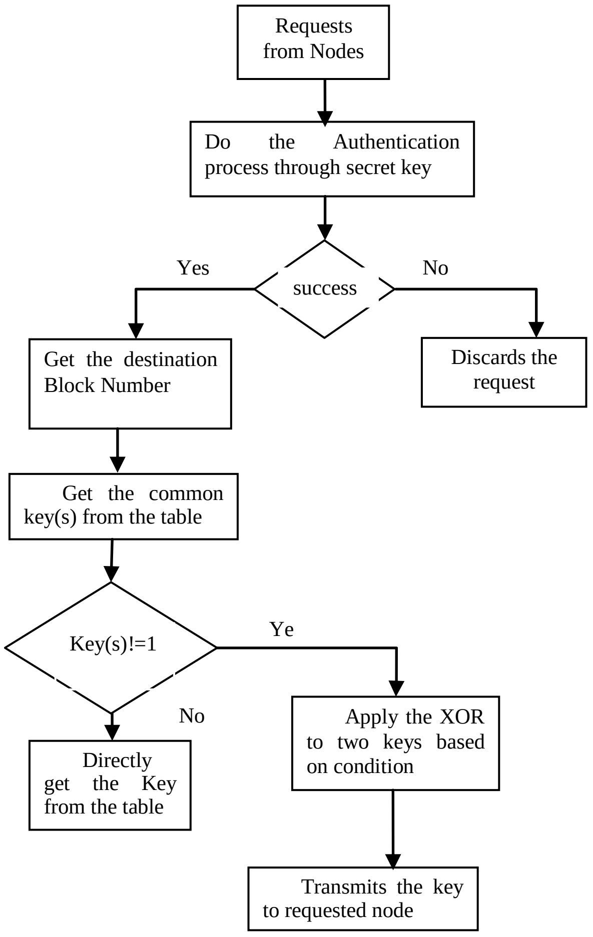
Requests from Nodes
Do the Authentication process through secret key
success
Yes
No
Get the destination Block Number
Discards the request
Get the common key(s) from the table
Key(s)!=1
Ye
No
Directly get the Key from the table
Apply the XOR to two keys based on condition
Transmits the key to requested node

Fig. 2. The Workflow of Dynamic Key Generation and Distribution Center (DGDC).

Example 3: In the Example, We are taking network size as 7 (Select maximum prime number that satisfies q=2, 22+2+1=7). So.

v= 2^2+2+1= 7, k=q+1=4

Such that the total number of blocks (B) designed by the SBIBD=7. Each DGDC module identifies the session key for every block by using Algorithm 4. And below Table I shows the key numbers for each block.

These are the seven keys stored in DGDC before starting the communication.

{1: 31037803, 2: 34051950, 3: 75095512, 4: 67731601, 5: 90790958, 6: 42721930, 7: 56819008}

By using the above network configuration the users can communicate with each other. For example, if User 1(B1) wants to transmit the data to User 6 (B6), DGDC identifies the common key between B1 and B6 i.e., 3. Then select the '3' Key value from Algorithm 3 that is already given above. Here the value is 75095512. DGDC transmits this Key to User 1 as well as User 6 when it sends a request for communication.

TABLE I. CONSTRUTION OF 7 NODES USING ALGORITHM1

User(s)	Block Number	Key number1	Key number2	Key number3
1	B1	1	2	3
2	B2	1	4	5
3	B3	1	6	7
4	B4	2	4	6
5	B5	2	5	7
6	B6	3	4	7
7	B7	3	5	6

Example 4: Here, We are taking 20 as the input, N (q=3, 32+3+1=13). We could not use the value for q is 2.

$v = q^2 + q + 1 = 13, k = q + 1 = 4$

These blocks are getting from the DGDC from Algorithm4. But the given N value is 20. So, we have to find out the other blocks by using Algorithm 5. Table II shows the key numbers up to block 13.

The below-mentioned keys are the basic keys that are stored in the DGDC and these keys are also used for calculating the other node keys by using Algorithm 6.

{1: 56940651, 2: 83179189, 3: 88850165, 4: 50901991, 5: 95809326, 6: 88046686, 7: 45506527 , 8: 42631960, 9: 36152950, 10: 31237906, 11: 91772959, 12: 87834612, 13: 13247806}

The Remaining nodes are: 20-13=7. Table III shows the key numbers of the remaining nodes.

TABLE II. CONSTRUCTION OF 13 NODES USING ALGORITHM1

User (s)	Block Number	Key Number1	Key Number2	Key Number3	Key number4
1	B1	1	2	3	4
2	B2	1	5	6	7
3	B3	1	8	9	10
4	B4	1	11	12	13
5	B5	2	5	8	11
6	B6	2	6	9	12
7	B7	2	7	10	13
8	B8	3	5	10	12
9	B9	3	6	8	13
10	B10	3	7	9	11
11	B11	4	5	9	13
12	B12	4	6	10	11
13	B13	4	7	8	12

TABLE III. CONSTRUCTION OF REMAINING 7 NODES USING ALGORITHM 2

User (s)	Block Number	Key number1	Key number2	Key number3	Key number4
14	B14	1	2	4	7
15	B15	1	2	4	10
16	B16	2	4	10	13
17	B17	2	4	9	10
18	B18	1	4	7	9
19	B19	4	7	10	13
20	B20	4	7	10	11

Algorithm 5 can generate multiple possibilities to build the tables to address the aforementioned problem.One of the solutions has mentioned in Table III. The DGDC can select any

But, here we can get the duplicate key numbers for identified blocks. We have implemented Algorithm 6 to calculate the accurate key for both parties. For Example, User 1 (B1) wants to send the data to User 17 (B17). So, DGDC needs to identify the key for them by using Algorithm6 itself.

The block key numbers are again mentioned here for reference.

B1-(1, 2, 3,4)

B17-(2,4,9,10)

Two common keys from the above blocks are 2 and 4. The keys values are taken from above dictionary for 2: 83179189 and 4: 50901991. After applying Algorithm 6,the output key-value is D3878818. So, DGDC transmits this common key to both users for further communication.

We shall receive new blocks for nodes after the same table with keys has been used for a time determined by the DGDC.

IV. ANALYSIS

The connectivity, scalability, resilience, and memory utilization of our model are all evaluated.

A. Scalability

The model can be scalable with the maximum number of nodes that were constructed for the IoT network. The model works with all keys in the keyring that correspond to the maximum number of IoT nodes that can be supported. The number of blocks generated with their keyrings determines the network's scalability. The scalability of a proposed approach is

$$q^2 + q + 1 + \binom{n^2 + 2qn + n}{q + 1}$$

here n is an integer value to get the next prime number and$(q^2 + q + 1)$ is identified by Algorithm1.

The following equation is for the calculation of the remaining nodes:

$$(q + n)^2 + (q + n) + 1 - (q^2 + q + 1) =$$

$$n^2 + 2qn + n$$

B. Connectivity

The probability of any two IoT nodes sharing only one communication key.

The main advantage of this model is to get the probability of key share at most 1 for maximum all cases.

$$p_{BB} = \frac{\binom{q^2+q+1}{2}}{\binom{N}{2}} = \frac{q^2+q+1(q^2+q)}{N(N-1)}$$

$$p_{BR} = \frac{\binom{q^2+q+1}{1}\binom{N-(q^2+q+1)}{1}}{\binom{N}{2}}$$

$$= \frac{2(q^2 + q + 1)\big(N - (q^2 + q + 1)\big)}{N(N-1)}$$

$$p_{RR} = \frac{\binom{N-(q^2+q+1)}{2}}{\binom{N}{2}}$$

$$= \frac{(N - (q^2 + q + 1)(N - (q^2 + q + 1) - 1)}{N(N-1)}$$

According to the proposed model $p_{BB} + p_{BR} + p_{RR} = 1$, because the connectivity should be 1 in all maximum cases in the proposed approach.

$$p_{keyshare} = p_{BB} + p_{BR} + p_{RR} = 1$$

C. Resilience

Resilience means reliability among the network nodes from the attacker. The capture attack is called by capturing and revealing the key values from the nodes. So, the links which are used by the attacked key that might be compromised then those links are at risk. The proposed approach employs a unique key to communicate across nodes. And at random times, it shuffles all key values of blocks and blocks values as well As a result, if an attacker captures a key, it will not be worked after the shuffle.

$$p(L|C_x) = \sum_{\forall i} p(l_i|l)(p(D_i|C_x)$$

Where L denotes the link, $C_x$is x nodes are captured, l_i is the secure link between devices that already shared the i^{th} key in the pool. $D_i$has identified the key pool that includes key $i$is compromised. In our proposed system, from Algorithm 3, each key appears in the B blocks.

$r = q + 1$. For R blocks, each key repetitions are, $r' = 3q$

$$p(l_i|l) = \frac{\binom{((q+1)+3q)}{2}}{\binom{q^2+q+1+\binom{n^2+2qn+n}{q+1}}{2}}$$

The probability of key$i$, appearing in one or more of the x compromised keyrings is:

$$p(D_i|C_x) = 1 - \frac{\binom{((q^2+q+1)+(N-(q^2+q+1)))-((q+1)-3q)}{x}}{\binom{q^2+q+1+\binom{n^2+2qn+n}{q+1}}{x}}$$

When x keyrings are captured, the probability of a link being compromised can be calculated as.

$$p(L|C_x) = \sum_{i=1}^{q^2+q+1} p(l_i|l)(p(D_i|C_x)$$

$$= q^2 + q + 1\,\frac{\binom{((q+1)+3q)}{2}}{\binom{q^2+q+1+\binom{n^2+2qn+n}{q+1}}{2}}\,p(D_i|C_x) \cong p(D_i|C_x)$$

Our proposed system increases resilience when compared to previous models. The other systems probabilities of resilience are:

In the [18] model:$p(L|C_x) = 1 - \frac{\binom{q^2}{x}}{\binom{q^2+q+1}{x}}$

In the [21] model:

$$p(L|C_x) = 1 - \frac{\binom{2q^2}{x} + 2\binom{q^2}{x}}{\binom{2q^2+2q+2}{x}}$$

In the [22] model:

$$p(L|C_x) = 1 - \frac{\binom{((q^2+q+1)+(N-(q^2+q+1)))-((q+1)+\binom{2q}{q}))}{x}}{\binom{q^2+q+1+\binom{2q+1}{q+1}}{x}}$$

Resilience values are provided for 500, 800, and 1700 nodes in Tables IV, V, and VI, respectively. Fig. 3, 4, and 5 show the graphs for the corresponding tables with various nodes. Different methods for hybrid combinatorial design are provided in tables and figures, and it is demonstrated that our approach produces the best results when compared to other ways.

TABLE IV. RESILIENCE VALUES FOR 500 NODES

N	Compromized nodes x	q value	HSYM	MHS	MHSYM	Our Approach
500	20	19	0.669	0.664	0.644	0.639
	40		0.897	0.89	0.87	0.852
	60		0.97	0.965	0.942	0.939
	80		0.992	0.98	0.975	0.971
	100		0.998	0.996	0.986	0.984
	120		0.999	0.999	0.992	0.99
	140		0.999	0.999	0.999	0.998

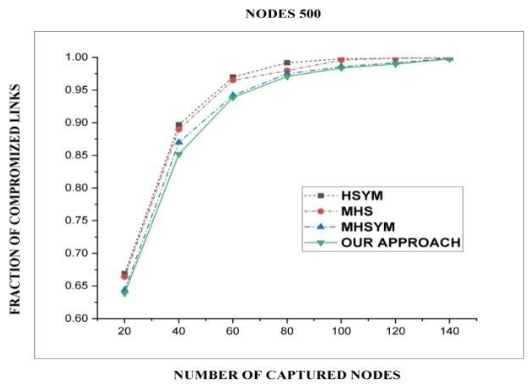

Fig. 3. Resilience Simulation Results of Our Approach Versus HSYM[18], MHS[21], and MHSYM[22] for the 500 Nodes.

TABLE V. RESILIENCE VALUES FOR 800 NODES

N	Compromized nodes x	q value	HSYM	MHS	MHSYM	Our Approach
800	40	23	0.84	0.83	0.81	0.8
	60		0.94	0.93	0.93	0.91
	80		0.97	0.97	0.96	0.95
	100		0.994	0.99	0.99	0.98
	120		0.997	0.996	0.99	0.99
	140		0.999	0.998	0.99	0.99
	160		0.999	0.999	0.99	0.99

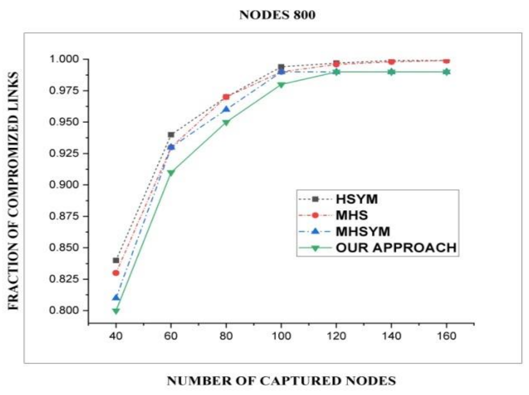

Fig. 4. Simulation Resilience Results of Our Approach Versus HSYM[18], MHS[21], and MHSYM[22] for the 800 Nodes.

TABLE VI. RESILIENCE VALUES FOR 1700 NODES

N	Compromized nodes x	q value	HSYM	MHS	MHSYM	Our Approach
1700	80	37	0.89	0.94	0.87	0.86
	100		0.94	0.9904	0.91	0.9
	120		0.967	0.996	0.95	0.93
	140		0.982	0.998	0.97	0.96
	160		0.9904	0.999	0.99	0.98
	180		0.994	0.999	0.99	0.98
	200		0.997	0.999	0.99	0.98

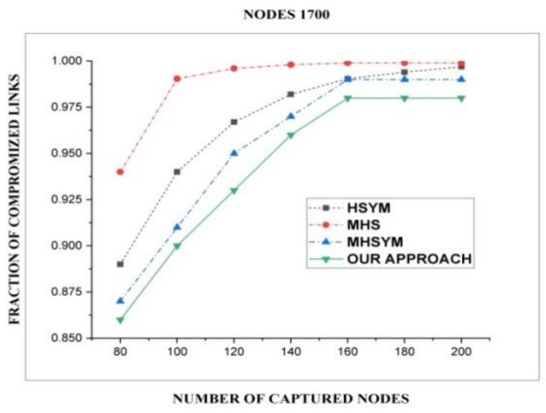

Fig. 5. Resilience Simulationresults of Our Approach Versus HSYM[18], MHS[21], and MHSYM[22] for the 1700 Nodes.

The above graphs and tables prove that our system greatly reduces the probability of compromized network links.Each node receives a different key for its links, and they all also get dynamic keys.

D. Memory Utilization

Here, DGDC is proposed as a centralized key distributor in the proposed system. So, there is no pressure on any network node to maintain all keys in the memory. The IoT node should store only one key that is applied to get the session key from DGDC.

As a result, We can declare that our proposed strategy improves node capture resilience with a combinatorial design.The notations and descriptions of the different parameters used in the article are given in Table VII.

TABLE VII. NOTATIONS OF PARAMETERS

Data related to implementing the Combinatorial designs	Parameter Notation
Blocks (nodes) connected to the IoT network	N
Blocks are generated by SBIBD	B
Remaining Blocks	R
Blocks are generated by HBIBD	H
Number of keys used in each block	k
Key Pool	v
Keys each replicated in the blocks	r
Number of keys intersecting any two blocks	γ

V. DISCUSSION

There is a demand for network security research that is essential due to the upsurge of online transactions. Every user in the transactions believes that the data will be secure and unaltered during transmission. To make secure data and provide reliable keys, a lot of algorithms can be used to provide confidentiality for the data and key-management techniques. In the present work, we are discussing a simple key management algorithm with less time and space complexity compared to the relevant studies on key management algorithms using combinatorial design. We observed that if the network has more than 800 nodes, the comprised links are reduced when compared to existing techniques. We also mentioned the relevant graphs of resilence in Fig. 3, Fig. 4, and Fig. 5 for various nodes.

VI. Conclusion and Future Work

In this paper, we present a new hybrid combinatorial key distribution scheme for IoT networks that improves the key share probability, scalability, and resilience against capture attacks. In comparison to the other three hybrid methods, our experimental outcomes were better. For all connected nodes, our suggested approach provides the key sharing probability with 1. Every link in the established IoT network can use the same unique key. This paper also provides low resilience values against capture attacks when compared to other schemes. We will also extend this work to reduce the resilience of specific attacks like a man in the middle, Denial of service. We would like to implement it in real networks for better analysis.

References

[1] Alagheband, Mahdi R., and Mohammad Reza Aref. "Dynamic and secure key management model for hierarchical heterogeneous sensor networks." IET Information Security 6.4 (2012): 271-280.

[2] Kouicem, Djamel Eddine, AbdelmadjidBouabdallah, and Hicham Lakhlef. "Internet of things security: A top-down survey." Computer Networks 141 (2018): 199-221.

[3] Stinson, Douglas R., and Scott A. Vanstone. "A combinatorial approach to threshold schemes." SIAM Journal on Discrete Mathematics 1.2 (1988): 230-236.

[4] Stinson, Douglas R. "Combinatorial designs: constructions and analysis." ACM SIGACT News 39.4 (2008): 17-21.

[5] Chien, Hung Yu, Rung-Ching Chen, and Annie Shen. "Efficient key pre-distribution for sensor nodes with strong connectivity and low storage space" 22nd International Conference on Advanced Information Networking and Applications (aina 2008). IEEE, 2008.

[6] Blundo, Carlo, et al. "Perfectly secure key distribution for dynamic conferences" Information and Computation 146.1 (1998): 1-23.

[7] Liu, Donggang, Peng Ning, and Kun Sun. "Efficient self-healing group key distribution with revocation capabilit." Proceedings of the 10th ACM conference on Computer and communications security. 2003.

[8] Chan, Haowen, Adrian Perrig, and Dawn Song. "Random key predistribution schemes for sensor network" 2003 Symposium on Security and Privacy, 2003. IEEE, 2003.

[9] Qian, Sun. "A novel key pre-distribution for wireless sensor networks" Physics Procedia 25 (2012): 2183-2189.

[10] Li, Wei-Shuo, et al. "Threshold behavior of multi-path random key pre-distribution for sparse wireless sensor networks." Mathematical and Computer Modelling 57.11-12 (2013): 2776-2787.

[11] Catakoglu, Onur, and Albert Levi. "Uneven key pre-distribution scheme for multi-phase wireless sensor networks." Information Sciences and Systems 2013. Springer, Cham, 2013. 359-368.

[12] Camtepe S, Yener B "Key distribution mechanisms for wirelesssensor networks: a survey" Rensselaer Polytechnic Institute,Troy,New York, Technical Report,2005, 05-07.

[13] Lee, Jooyoung, and Douglas R. Stinson. "A combinatorial approach to key predistribution for distributed sensor networks" IEEE Wireless Communications and Networking Conference, 2005. Vol. 2.

[14] Ruj, Sushmita, and Bimal Roy. "Key pre-distribution using partially balanced designs in wireless sensor networks" International Journal of High Performance Computing and Networking 7.1 (2011): 19-28.

[15] Ruj, Sushmita, Amiya Nayak, and Ivan Stojmenovic. "Pairwise and triple key distribution in wireless sensor networks with applications" IEEE Transactions on Computers 62.11 (2012): 2224-2237.

[16] Bechkit, Walid, et al. "A highly scalable key pre-distribution scheme for wireless sensor networks" IEEE transactions on wireless communications 12.2 (2013): 948-959.

[17] Bahrami, PoonehNikkhah, et al. "A hierarchical key pre-distribution scheme for fog network." Concurrency and Computation: Practice and Experience 31.22 (2019): e4776.

[18] Camtepe, Seyit A., and BlentYener. "Combinatorial design of key distribution mechanisms for wireless sensor networks" IEEE/ACM Transactions on networking 15.2 (2007): 346-358.

[19] Chakrabarti, Dibyendu, Subhamoy Maitra, and Bimal Roy. "A key pre-distribution scheme for wireless sensor networks: merging blocks in combinatorial design" International Journal of Information Security 5.2 (2006): 105-114.

[20] Kavitha, T., and D. Sridharan. "Hybrid design of scalable key distribution for wireless sensor networks" International Journal of Engineering and Technology 2.2 (2010): 136.

[21] Dargahi, Tooska, Hamid HS Javadi, and Mehdi Hosseinzadeh. "Application-specific hybrid symmetric design of key pre-distribution for wireless sensor networks" Security and Communication Networks 8.8 (2015): 1561-1574.

[22] Akhbarifar, Samira, et al. "Hybrid key pre-distribution scheme based on symmetric design" Iranian Journal of Science and Technology, Transactions A: Science 43.5 (2019): 2399-2406.

Authors' Profile

G.V.Hindumathi is currently pursuingPh.D. in Jawaharlal Nehru TechnologicalUniversity, Kakinada, India. She is specialized in Internet of Things andNetwork Security. Her research topic is onSecurity issues on Internet of Things.

Dr D.Lalitha Bhaskari works asProfessor in Andhra University,Visakhapatnam, and Andhra Pradesh. Her areas of expertise include: Deep Learning,Network Security, and Image Processing.And she got Young scientist award from byIEI.