IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 6 December 2023, accepted 3 January 2024, date of publication 8 January 2024,
date of current version 12 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3350780

RESEARCH ARTICLE

Efficient and Secure Group Key Management Scheme Based on Factorial Trees for Dynamic IoT Settings

K. SUDHEERADH, N. NEHA JAHNAVI, PRAMOD N. CHINE, AND GAURAV S. KASBEKAR, (Member, IEEE)
Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai 400076, India

Corresponding author: Gaurav S. Kasbekar (gskasbekar@ee.iitb.ac.in)

The work of Gaurav S. Kasbekar was supported in part by the NSF AI Institute for Future Edge Networks and Distributed Intelligence (AI-EDGE) Institute.

ABSTRACT The Internet of Things (IoT) extends Internet connectivity to resource-constrained devices such as sensors and actuators. It is prone to several security threats and attacks, due to which defence mechanisms such as encryption, message authentication codes, authentication, etc., need to be employed. Several IoT scenarios require secure communication to and data acquisition from multiple devices, which constitute a ''group''. Also, in dynamic IoT scenarios, devices join or leave a group from time to time, due to which the group keys, which are used for multicasting information within the group, need to be efficiently updated. We propose a novel scheme, based on factorial trees and the Chinese Remainder Theorem, for efficient Group Key Management. Our proposed scheme prevents malicious users from accessing information from a group and efficiently updates the group keys when devices join or leave a group, while guaranteeing forward and backward secrecy. We evaluate the performance of the proposed scheme via extensive mathematical analysis and numerical computations, and show that it outperforms schemes proposed in prior work in terms of the communication and computation costs incurred by IoT devices.

INDEX TERMS Authentication, Chinese remainder theorem, factorial tree, group key management, Internet of Things (IoT), key distribution center.

I. INTRODUCTION

The Internet of Things (IoT) is the extension of Internet connectivity to resource-constrained devices (e.g., sensors, actuators) and everyday objects [1]. IoT devices have low computational and storage power; hence, they cannot run the protocols traditionally used to achieve secure communications in the Internet. Therefore, we need to design lightweight and efficient protocols for the security of IoT devices [2]. In a network with IoT devices, the devices are often divided into *groups* of various sizes based on their use cases, hardware capabilities, etc. In a group where several users need access to the same encrypted data, a single shared key, known as *group key*, is used to encrypt and decrypt the data. Group Key Management (GKM) refers to the handling, revocation, updation and distribution of cryptographic keys to members of various groups in a communication network [3]. GKM involves the following actions:

Key generation: A cryptographic key (group key) is generated for a group, which is used to encrypt and decrypt the data communicated to and from the members of the group.

Key distribution: The generated key is securely distributed to all the members of the group who need access to the encrypted data, possibly using a Key Distribution Center (KDC) [2].

Key update: When a new member joins the group or an existing member leaves the group, the group key must be updated to prevent unauthorized access to the group communication.

For secure communication, a GKM system must ensure the confidentiality and integrity [4] of messages exchanged among group members. Confidentiality is achieved through encryption, which ensures that messages are readable only by the intended recipients [5]. Integrity is ensured through digital

The associate editor coordinating the review of this manuscript and approving it for publication was Eyhab Al-Masri.

signatures or message authentication codes (MAC), which allow recipients to verify the origin of messages [4]. In a dynamic IoT setting, devices join or leave a group from time to time, due to which the group keys need to be efficiently updated whenever a device joins or leaves, while maintaining forward and backward secrecy [6].

There is an extensive existing research literature on GKM schemes for traditional devices such as desktops and laptops as well as IoT devices (see Section II for a review). However, most of these GKM schemes perform inefficiently in dynamic IoT environments. Also, many of these schemes use asymmetric cryptography, which is computationally expensive and limits the scalability of the system. Several of the schemes proposed in prior work also have a high computational and communication cost, and hence are not suitable for IoT devices, which are resource-constrained. In contrast, we present a scheme in this paper, which performs efficiently in dynamic environments, while guaranteeing backward and forward secrecy, uses only symmetric key cryptography, and has much lower computational and communication costs than schemes proposed in prior work.

In this paper, we propose an efficient and secure GKM scheme for dynamic IoT settings based on factorial trees and the Chinese Remainder Theorem (CRT) [7]. Our scheme authenticates an IoT device before sharing information (keys) with it. The authentication is done using a cryptographic hash and nonces [7]. Our proposed scheme efficiently updates keys using hash updating or by leveraging the CRT when a device joins or leaves a group, while guaranteeing backward and forward secrecy. Also, we have implemented a 2-level authentication system in our protocol, which provides an additional layer of security against malicious users. This helps to prevent unauthorized access and ensures that only legitimate devices are able to access the system. By using factorial trees, we have reduced the number of keys that need to be stored and transmitted, which in turn reduces the amount of computation, storage, and communication required; this is particularly beneficial in resource-constrained environments [8], [9]. Our proposed protocol ensures secure group authentication by implementing checks on both the server and device side. It is also resistant to Man-in-the-Middle attacks [7] as information sent to the group is encrypted with the group key. We evaluate the performance of the proposed scheme via extensive mathematical analysis and numerical computations, and show that it outperforms state-of-the-art schemes proposed in prior work in terms of the communication as well as computation costs incurred by IoT devices.

The rest of the paper is organized as follows. We provide a review of related prior research literature in Section II. We review some relevant background concepts, viz., the CRT and factorial trees, in Section III. In Section IV, we present our system model and problem formulation. We describe our proposed scheme in Section V. We provide a security analysis of our proposed scheme in Section VI and an analysis of its storage, computation, and communication costs in Section VII. In Section VIII, we compare the performance of our proposed scheme with that of schemes proposed in prior work via numerical computations. Finally, we provide conclusions and directions for future research in Section IX.

II. RELATED WORK

The development of security measures for the IoT has attracted a lot of research interest. The survey [4] looks at potential IoT use cases and reasons why attackers may choose to target this novel paradigm. It provides a comprehensive list of edge-side IoT vulnerabilities and countermeasures. The authors also discuss numerous threats, attacks, and viable defences. The authors of [10] conducted an analysis of security and privacy issues in IoT systems and applications. The work also examined IoT attack classification techniques and current security mechanisms. Confidentiality, integrity, availability, authenticity, and privacy are only a few of the security concerns that the IoT faces, which are presented in a taxonomy in the study [11]. Subsequently, the authors review the IoT security protocols and products that are now available, including for key management, access control, authentication, and secure communication. In [11], [12], and [13], surveys of several IoT enabling technologies, such as RFID, MQTT, and CoAP, are provided. In order to provide low cost, low power, and low data rate wireless communication among resource-constrained devices, the IEEE standard 802.15.4 provides specifications for the physical and medium access control (PHY and MAC) layers [14]. In order to facilitate the functioning of the IoT, the Internet Engineering Task Force (IETF) has additionally developed protocols for a number of layers [15]. One of them is IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [16]. In order to send IPv6 packets from the Internet over resource-constrained networks using 802.15.4, the 6LoWPAN adaptation layer performs header compression as well as fragments packets into sizes that are compatible with constrained networks and reassembles them at the receiver [16]. Another IETF standard for routing over resource-constrained networks is the Routing Protocol for Low Power and Lossy Networks (RPL) [15]. To increase the size of the permitted application-layer payload, Datagram Transport Layer Security (DTLS) header compression and TLS-DTLS mapping [10] have also been proposed. Recent research has focused on developing secure protocols for various IoT standards [11]. The authors of [17] develop a secure and efficient Elliptic-Curve Qu–Vanstone (ECQV) implicit certificate issuance protocol specifically designed for IoT applications, integrate it with the secure join protocol of IEEE 802.15.4, and highlight the protocol's security features and efficiency.

Multifactor authentication for IoT scenarios has been thoroughly investigated over the past decade. For remote users to securely connect with a specific sensor node, a novel two-factor authentication approach is proposed in [18]. Wireless sensor networks (WSNs) constitute an integral part of the IoT. For WSNs to securely connect to the

Internet, the authors of [19] present a 3-factor authentication and key establishment strategy based on biohashing and the Rabin Cryptosystem that is effective and lightweight. In [20] and [21], authentication and key agreement protocols for WSNs have been designed using elliptic curve cryptography (ECC). In [22], a discussion on how to use the master and initial keys that are originally provided to devices and users at configuration time is provided. A key derivation function (HKDF) based on the hashed message authentication code (HMAC) is used to provide the authentication, key exchange, confidentiality, and message integrity elements of the protocol. In [23], the authors introduce a provably secure and lightweight authenticated key agreement protocol for the modern health industry. The protocol focuses on establishing secure keys for resource-constrained healthcare devices. A secure, anonymity-preserving, and lightweight authentication and key agreement scheme for home automation IoT networks is proposed in [24].

For group authentication and key management, researchers have put forth a number of methods, including effective Group Key Establishment (GKE) and GKM schemes. Schemes for GKE have been designed for resource-constrained network architectures by applying ECC in [25] and [26]. The authors of [8] and [9] discuss several key graph types that are utilised for the user-device structure and how they affect communication and processing costs. The authors of [27] advocate a one-way function as a structure for GKM rather than Logical Key Hierarchy (LKH) or tree-based structures since less bits are required. The authors of [28] investigate the applications and use cases of GKM in environments with limited resources. They also provide a comprehensive overview of existing protocols and algorithms in this context. In [29], a group key agreement protocol based on Elliptic Curve Diffie-Hellman (ECDH) and short signature schemes, aiming to establish secure and efficient group keys for multiple participants, is introduced. In [30], a GKM scheme is proposed for multicast systems where users join or depart groups on a dynamic basis. In [31], the authors provide a GKM technique for secure multicast communication that significantly reduces computation overhead. In [32], an efficient mechanism for group association and data sharing in IoT systems using edge computing is provided.

In [33], the authors proposed a GKM scheme for efficient urban computing with highly mobile IoT nodes based on blockchain and Unmanned Aerial Vehicle (UAV) technology. They have shown a considerable decrease in the number of rekeying messages using UAVs. In [34], the authors develop and analyze a secure, lightweight, and efficient method for establishing group keys in scenarios involving the Internet of Health Things (IoHT). In [35], a GKM scheme for smart grids, which leverages the security and efficiency of blockchain technology, is proposed. In [36], an innovative group communication framework aimed at enhancing security in the IoT is provided. It introduces an architecture that integrates secure communication protocols and access control mechanisms.

The authors of [37] proposed a flexible and effective GKM protocol for use in dynamic IoT contexts that maintains forward and backward secrecy and guards against collusion attacks. By utilising the LKH approach to reduce communication overhead caused by node entry and exit, the authors of [2] designed a two-tier GKM strategy. In both [2] and [37], the network is divided into User (Subscriber) Groups and Device Groups. In each User Group (respectively, Device Group), the users (respectively, devices) are arranged as in LKH in a binary tree structure. Our proposed scheme replaces the binary tree in [2] and [37] with a factorial tree; this leads to lower computational and communication costs under our proposed scheme than under the schemes proposed in [2] and [37], as is demonstrated via numerical computations in Section VIII.

Most of the GKM schemes in the above works perform inefficiently in dynamic IoT environments. Also, many of these schemes use asymmetric cryptography, which is computationally expensive and limits the scalability of the system. Several of the schemes proposed in the above works also have a high computational and communication cost, and hence are not suitable for IoT devices, which are resource-constrained. In contrast, we present a scheme in this paper, which performs efficiently in dynamic environments, while guaranteeing backward and forward secrecy, uses only symmetric key cryptography, and has much lower computational and communication costs than schemes proposed in prior work.

III. BACKGROUND

This section provides a review of some of the key concepts used in the proposed GKM scheme: the CRT and factorial trees.

A. CHINESE REMAINDER THEOREM (CRT)

Let $a_1, \ldots, a_n$ be co-prime numbers greater than 1 and $r_1, \ldots, r_n$ be integers such that $0 \leq r_i < a_i, \forall i \in \{1, \ldots, n\}$. The CRT [7] states that we can find a number X that satisfies the following congruencies:

$$\begin{aligned} X &\equiv r_1 \pmod{a_1}, \\ X &\equiv r_2 \pmod{a_2}, \\ &\vdots \\ X &\equiv r_n \pmod{a_n}, \end{aligned}$$

where $X \equiv r_i \pmod{a_i}$ means that r_i is the remainder when X is divided by a_i. By the CRT, there exists a unique solution X between 0 and $R = \prod_{i=1}^{n} a_i$, and it is given by:

$$X = \sum_{i=1}^{n} r_i A_i A_i',$$

where $A_i = (\prod_{j=1}^{n} a_j)/a_i$ and A_i' is the multiplicative inverse of A_i modulo a_i, i.e., $A_i A_i' \equiv 1 \pmod{a_i}$.

B. FACTORIAL TREE

Nodes are arranged in a *factorial tree* structure in our proposed scheme. In the factorial tree used in our proposed scheme, the IoT devices are at the leaf nodes, while the internal nodes have some internal keys associated with them. Each level l of the factorial tree contains $l!$ nodes. Each node at level l has $l+1$ child nodes. An example of a factorial tree is shown in Fig. 1, where each level l contains $l!$ nodes and the leaf nodes in level 4 contain IoT devices.

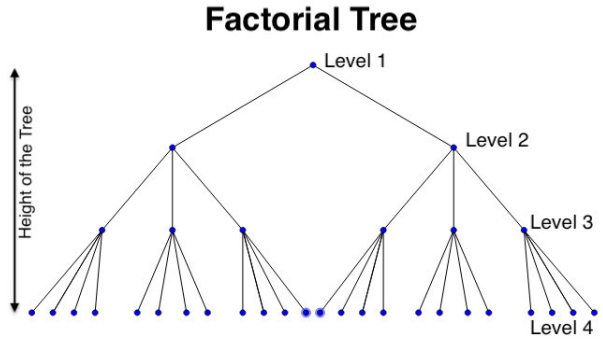


FIGURE 1. The figure shows a factorial tree.

IV. MODEL AND PROBLEM FORMULATION

We describe the system model and problem formulation in this section.

A. SYSTEM MODEL

The system model consists of a KDC, which is a centralized server that manages the keys for the different devices, and IoT devices, which are divided into groups of various sizes based on their use cases or hardware capabilities. In the rest of the paper, we focus on an arbitrarily selected group, which consists of n IoT devices. Throughout, we refer to the KDC and the IoT devices in the group as the server and nodes, respectively. All the nodes in a group are arranged in a factorial tree, and each group has a different factorial tree. Each IoT device (node) in the group occupies a leaf node in the factorial tree. All the nodes of the factorial tree, including non-leaf internal nodes and leaf nodes containing the IoT devices, are associated with a large prime number each, as explained later. These large prime numbers are generated using an algorithm such as the Sieve of Eratosthenes or the Sieve of Atkin [38], [39]. In the key renewal process described in Section V-D, which is used when a node leaves the group, we use these generated prime numbers of the internal nodes in the factorial tree to communicate the new group key to the non-leaving members.

B. PROBLEM FORMULATION

Our objective is to design a GKM scheme that securely distributes a group key to each node of the group, after authenticating the node, which can be used to securely multicast information to the group members. Also, the GKM scheme must securely distribute other cryptographic keys to different nodes, as needed, in order to efficiently handle dynamics in the network. In particular, the GKM scheme must enable the efficient updation or revocation of the cryptographic keys assigned to different nodes when changes take place in the network (a node joins or leaves the group) such that forward and backward secrecy is maintained at all times. Recall that by forward (respectively, backward) secrecy, we mean that a leaving (respectively, joining) node is prevented from decrypting messages exchanged after (respectively, before) it leaves (respectively, joins) the group [37]. Also, we seek to design a GKM scheme that minimizes the computational, communication, and storage costs incurred to the nodes in the network when changes take place in the network. We do this by utilizing a factorial tree and the CRT.

Similar to most prior works in GKM, we make the following assumptions. First, all network entities use the same cipher suite and keys are sufficiently large [37]. Second, the system is reactive to tampering; therefore, any node capture or compromise will be detected in practically small time, resulting in appropriate revocation and updation of node key material. Different approaches such as mobility based [40] and control theoretic [41] modelling have been presented in prior work to detect and revoke compromised nodes. Finally, we assume that the message integrity of all communication exchanged during the network operation is protected using standard mechanisms (e.g., message authentication codes) [11].

V. PROPOSED SCHEME

In this section, we describe our proposed scheme in detail.

A. INITIALISATION AND AUTHENTICATION

Initially, when the server has been set up, a unique identity I_S is assigned to the server. When IoT devices (referred to as nodes from here on) are added to a group, the server assigns each node i an identity I_i, a secret key K_i, and an initial nonce $\tilde{n}_i$. This secret key and nonce are known only to the server and node i. The server identity I_S is known only to the server and the initialized nodes. After the server assigns the secret key, nonce, and identity to each node, it builds the factorial tree of GKM and authenticates with all the nodes as described in Fig. 2. After the successful authentication of an initialized node i with the server, the server sends to node i a set of prime numbers P_i, which are associated with the internal nodes along the path from the root node to the leaf node corresponding to node i in the factorial tree. These prime numbers will be utilized later during rekeying, which is explained in Section V-D, when a node leaves the group. Once the initialization and authentication phase of a group is complete, each internal node of the factorial tree of the group has a prime number associated with it, and each leaf node has a secret key and a prime number associated with it.

For each node i, the following steps are performed in the initialization and authentication phase (see Fig. 2).

Step 1: The server first selects a node (say node i) and calculates the cryptographic hash of the node's identity I_i

Initialisation and Authentication

begin

for $i = 1 : n$ **do**
  Server $\rightarrow$ Node i: $E_{K_i}[H(I_i||\tilde{n}_i)]$
  if Node i: $H(I_i||\tilde{n}_i) \neq D_{K_i}[(E_{K_i}[H(I_i||\tilde{n}_i)])_{\text{from server}}]$ **then**
    Authentication Failed
  end if
  Node i $\rightarrow$ Server: $E_{K_i}[H(I_S||\tilde{n}_i)]$
  if Server: $H(I_S||\tilde{n}_i) \neq D_{K_i}[(E_{K_i}[H(I_S||\tilde{n}_i)])_{\text{from Node i}}]$ **then**
    Authentication Failed
  end if
  Server $\rightarrow$ Node i: $E_{K_i}[P_i]$
  Node i: $D_{K_i}[(E_{K_i}[P_i])_{\text{from server}}]$
  Node i $\rightarrow$ Server: $E_{K_i}[N_i||I_i]$
  Server: $D_{K_i}[(E_{K_i}[N_i]||I_i)_{\text{from Node i}}]$
end for

end

FIGURE 2. The figure shows the algorithm used for initialisation and authentication.

concatenated with the node's nonce $\tilde{n}_i$. This hash value is then encrypted with the corresponding secret key K_i. This encrypted message is then sent to the desired node (node i).

Step 2: Node i, upon receiving the encrypted message from the server, decrypts it using the secret key K_i and checks whether the decrypted message is the same as the computed cryptographic hash of the node's identity I_i concatenated with its nonce $\tilde{n}_i$. If the check is successful, the algorithm moves to the next step. If the check is unsuccessful, the authentication is considered failed, and it is assumed that a malicious user is trying to gain access to the node; no further action is required in this case.

Step 3: After a successful check in step 2, node i computes the cryptographic hash of the server's identity I_S concatenated with the node's nonce $\tilde{n}_i$, encrypts it with the secret key K_i and sends it to the server.

Step 4: The server decrypts the encrypted message received from node i using the secret key K_i. It then computes a cryptographic hash of its identity I_S concatenated with the node's nonce $\tilde{n}_i$. Similar to *Step 2*, the server checks whether the decrypted message is the same as the computed hash. If they are the same, the authentication is successful, and the algorithm moves to the next step. The decrypted message and computed cryptographic hash are not the same if a malicious node tries to communicate with the server. In this case, the authentication fails, and the server restarts the authentication process with the current node. If the authentication fails repeatedly, the server moves on to process the next node, and manual intervention is required for the current node.

Step 5: Once the authentication is successful, the server sends to node i a set of prime numbers P_i by encrypting them with the secret key K_i. This set P_i consists of the prime numbers associated with the internal nodes in the path from the root node to the leaf node corresponding to node i in the factorial tree.

Step 6: Node i, upon receiving the encrypted message, decrypts it using the secret key K_i. Upon decryption, the node obtains the set of prime numbers P_i. The node stores the values in set P_i, which will be utilized in the rekeying process when a node leaves the group, as will be explained in Section V-D.

Step 7: Node i then sends back to the server a nonce, N_i, concatenated with its id, I_i, encrypted with the key K_i. This nonce N_i is generated by node i and is shared only with the server.

Step 8: The server decrypts the received encrypted message from node i and stores the nonce N_i corresponding to node i.

The above steps are performed for all the nodes in the group. Upon successful initialization and authentication, the nonce generated by the server corresponding to each node i, $\tilde{n}_i$, is discarded on both the server side and the node i side as it is not used in the following steps. After these steps, each node i contains its identity I_i, server's identity I_S, set of prime numbers P_i and nonce generated by itself N_i. The server contains all the previously mentioned values corresponding to all the nodes. The N_i values received by the server from each node i are later used in creating the group key, as explained in Section V-B.

B. GROUP KEY CREATION AND DISTRIBUTION

After the authentication of each node in the group and initialization of the group, the server creates a unique group key, G_k, for the group. At this stage, the server contains a nonce (N_i) and a secret key (K_i) corresponding to each group node, which will be used to construct the group key (G_k). The server applies cryptographic hash to this nonce concatenated with the secret key of each node and XORs the values of all the hash outputs to generate the group key G_k (see Fig. 3). In this way, a unique group key G_k is obtained for each group. An interim group key, $G^*_{k,i}$, is sent to each node i instead of transmitting the actual group key, G_k, as explained in the following discussion, in order to achieve more secure communication of the group key.

Group Key Creation and Distribution

begin

Server: $G_k = H(N_1||K_1) \oplus H(N_2||K_2) \oplus \cdots \oplus H(N_n||K_n)$
for $i = 1 : n$ **do**
  Server: $MAC_i = H(G_k||I_i)$
  Server: $G^*_{k,i} = G_k \oplus H(N_i||K_i)$
  Server $\rightarrow$ Node i: $E_{K_i}[G^*_{k,i}||MAC_i]$
  Node i: $D_{K_i}(E_{K_i}[G^*_{k,i}||MAC_i]) = G^*_{k,i}||MAC_i$
  Node i: $G^*_{k,i} \oplus H(N_i||K_i) \implies G_k$
  if Node i: $H(G_k||I_i) \neq MAC_i$ **then**
    Authentication Failed
  end if
end for

end

FIGURE 3. The figure shows the algorithm used for group key creation and distribution.

The group key is shared with all the nodes by performing the following steps for each node i (Fig. 3):

Step 1: The server creates a message authentication code (MAC_i) by calculating the cryptographic hash of the group key, G_k, concatenated with the id, I_i, of node i. The server also calculates an interim group key, $G^*_{k,i}$, by XORing the

group key G_k with the hash of the nonce concatenated with the secret key, i.e., $G^*_{k,i} = G_k \oplus H(N_i||K_i)$.

Step 2: The server concatenates the interim group key and the MAC calculated in the previous step and encrypts the result using the secret key K_i, giving the following output: $E_{K_i}[G^*_{k,i}||MAC_i]$. This encrypted message is then sent to node i.

Step 3: Node i, upon receiving the message from the previous step, decrypts the encrypted message from the server using the secret key K_i. Upon decrypting the message, the node obtains the interim group key, $G^*_{k,i}$, and message authentication code, MAC_i. The node XORs this interim key with the cryptographic hash of its nonce concatenated with its secret key, $G^*_{k,i} \oplus H(N_i||K_i)$, to obtain the group key, G_k.

Step 4: The node then calculates the cryptographic hash of the group key concatenated with its id to get $H(G_k||I_i)$. It then checks whether this value is the same as MAC_i. If they are different, the key distribution is considered failed. If the key distribution fails, the node requests the server to retransmit the interim group key $G^*_{k,i}$. If the key distribution repeatedly fails, either the node is not initialized correctly, or it is malicious. Manual intervention is required in both cases. The key distribution is successful if the calculated value, $H(G_k||I_i)$, is the same as MAC_i, and the node stores G_k as its group key.

Note that in the above method, the server does not send the final group key, G_k, to any of the nodes. Only the interim group key, $G^*_{k,i}$, is sent to each node i, and only a legitimate node i can generate the group key, G_k, using its nonce, N_i, secret key, K_i, and id, I_i.

Adding new node to the group

begin

1: Server: $G_{k_{New}} = H(G_k)$
2: Server $\xrightarrow{\text{Broadcast to all the nodes of the group}} E_{G_k}[G_{k_{New}}]$
3: **for** $i = 1 : n$ **do**
4: Node i: $G_{k_{New}} = D_{G_k}[(E_{G_k}[G_{k_{New}}])_{\text{from server}}]$
5: **end for**
6: Server $\rightarrow$ New Node: $E_{K_i}[G_{k_{New}}]$
7: New Node: $D_{K_i}[(E_{K_i}[G_{k_{new}}])_{\text{from server}}]$

end

FIGURE 4. The figure shows the algorithm used to add a new node to a group.

C. NODE ADDED TO GROUP

When a new node is added to a group, the group key, G_k, must be updated to maintain backward secrecy, i.e., the new node being added to the group should not be able to read the messages sent in the group before its addition to the group. Since the new node does not have the current group key, G_k, the newly updated group key, $G_{k_{New}}$, is sent to the existing group members by a broadcast message encrypted with the current group key G_k (see Fig. 4). One of the following two scenarios arises when a new node joins the group:

Scenario 1: There is an empty spot in the leaf node level of the factorial tree (see Fig. 5).

Scenario 2: The leaf node level of the factorial tree is full (see Fig. 6).

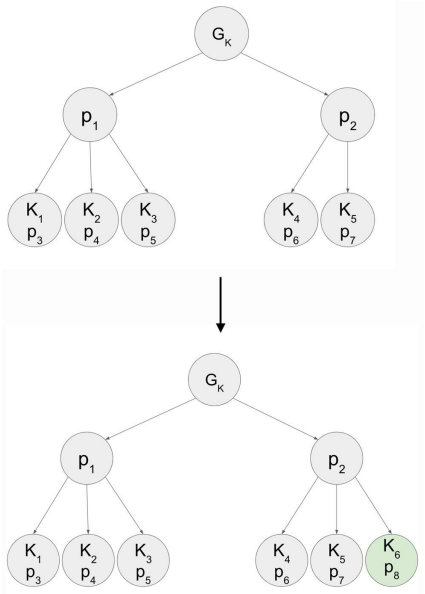


FIGURE 5. The figure shows the addition of a node to a group in scenario 1.

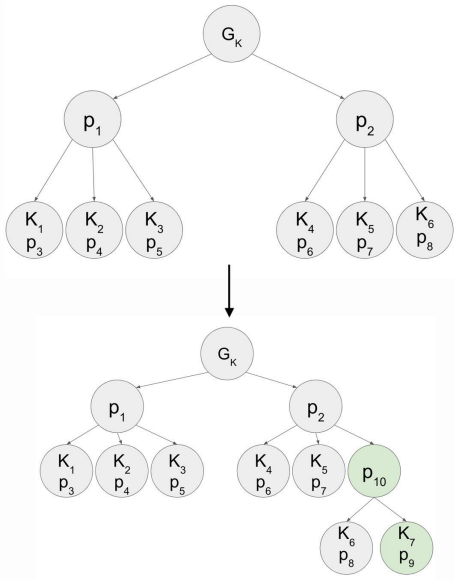


FIGURE 6. The figure shows the addition of a node to a group in scenario 2.

In both scenarios, the server computes the cryptographic hash of the current group key, G_k, to form the new group key, $G_{k_{new}}$ (see Fig. 4). It then broadcasts a message notifying the current group members to update the existing group key G_k to the new group key $G_{k_{new}}$. This broadcast message is encrypted with the current group key, G_k. The existing nodes then update the group key by decrypting the broadcast message and extracting the new group key $G_{k_{new}}$.

In scenario 1 (see Fig. 5), the new node is first initialized and authenticated as explained in Section V-A, and the new group key, $G_{k_{New}}$, encrypted with the new node's secret key

is sent to the new node by the server. The new node decrypts this message and obtains the new group key.

In scenario 2, the same steps as in scenario 1 are first performed. In addition, a new prime number corresponding to an internal node is to be sent to one of the existing leaf nodes. This new prime number, p, is sent to the existing leaf node as shown in Fig. 7. In the example shown in Fig. 6, the new prime number p_{10} is present at the internal node previously occupied by the existing node containing key K_6 and the prime number p_8 and is sent to the latter node. The new prime number, p, is shared with the new node in the node initialization and authentication step (see Section V-A).

Scenario when the factorial tree is full (scenario 2)

begin

1: Server $\rightarrow$ Existing Node: $E_{K_i}[p]$
2: Existing Node: $D_{K_i}[(E_{K_i}[p])_{\text{from server}}]$

end

FIGURE 7. The figure shows the additional steps used when the factorial tree is full (scenario 2).

D. NODE LEAVING GROUP

When a node leaves a group, the leaving node should not be able to read the group messages sent after it leaves, to maintain forward secrecy. Hence, the group key should be updated once a node leaves the group. The new group key cannot simply be the hash updated version of the current group key, and it cannot be distributed using the existing group key, since the leaving node has the existing group key. Some of the internal keys associated with the leaving node must also be updated.

The new group key is calculated by applying a cryptographic hash on the current group key concatenated with a nonce, N_S, selected by the server. The nonce N_S is efficiently distributed to all the nodes in the group except the leaving node with the help of the prime numbers in the internal and leaf nodes. This renewal of the group key and internal keys is done with the help of the CRT (see Section III-A), as explained below.

To compute the new group key, all non-leaving nodes must receive the nonce, N_S, and this nonce N_S must satisfy some conditions described later. The server first computes:

$$N_i = p_i - N_S, \tag{1}$$

where p_i (for $i = 1, \ldots, m$) are a set, say P^*, of prime numbers such that this set contains exactly one prime number either in an internal node or the leaf node on the path from the root to each non-leaving node. Also, P^* does not contain the prime number of any of the nodes on the path from the root to the leaf node corresponding to the leaving node. For example, in the upper figure in Fig. 9, the leaf node N_6 highlighted in red is the node that is leaving. In this case, P^* is chosen to be the set $\{p_1, p_6, p_7\}$ of prime numbers of the internal node and leaf nodes highlighted in yellow.

For each N_i obtained using (1) to be a positive integer, the nonce, N_S, selected by the server must satisfy the condition that $N_S < \min(p_i)$ for all $p_i \in P^*$. As mentioned in Section IV-A, the chosen prime numbers, p_i's, are large; hence, N_S can be chosen from a large range of numbers.

After calculating the values N_i, $i = 1, \ldots, m$, using (1), the server computes the unique solution X to the following set of equations using the CRT:

$$\begin{aligned} X &\equiv N_1 \pmod{p_1} \\ X &\equiv N_2 \pmod{p_2} \\ &\vdots \\ X &\equiv N_m \pmod{p_m} \\ &\text{where } p_1, \ldots, p_m \in P^*. \end{aligned}$$

The server then broadcasts the unique solution X to all the group nodes using the current group key G_k (see Fig. 8). Each non-leaving node then computes $N_i = X \bmod p_i$ using the prime number $p_i \in P^*$ on the path from the root to the non-leaving node. It then computes the nonce using $N_S = p_i - N_i$ (see (1)). Note that the node which is leaving is not able to compute N_i, and hence N_S, since the unique solution X is constructed using only the p_i values from the set P^*, which does not contain the prime number of any of the nodes on the path from the root to the leaf node corresponding to the leaving node. Finally, each non-leaving node computes the new group key by taking the cryptographic hash of the current group key concatenated with the nonce: $G_{k_{new}} = H(G_k||N_S)$. The above procedure is shown in Fig. 8.

Node leaving the group

begin

1: Server: $X \equiv N_i \pmod{p_i}$ (for $i = 1, \ldots, m$)
2: Server $\xrightarrow{\text{Broadcast to all the nodes in the group}}$ $E_{G_k}[X]$
3: **for** each non-leaving node j **do**
4: Node j: $D_{G_k}[(E_{G_k}[X])_{\text{from server}}] = X$
5: Node j: $N_i = X \bmod p_i$
6: Node j: $N_S = p_i - N_i$
7: Node j: $G_{k_{new}} = H(G_k||N_S)$
8: **end for**

end

FIGURE 8. The figure shows the rekeying algorithm used when a node leaves the group.

The leaving node also contains the prime numbers associated with the internal nodes along the path from the root node to the leaf node corresponding to the leaving node (e.g., the prime number p_2 in the upper figure in Fig. 9). These prime numbers must also be updated to prevent the leaving node from obtaining unauthorized access to the group communications in the future. The rekeying of the required prime numbers in the internal nodes is done similar to the way the unique solution X is communicated with the help of CRT. The difference is that in this case, instead of N_S, the new prime number p_{new} corresponding to each internal node along the path from the root to the leaf node corresponding to the

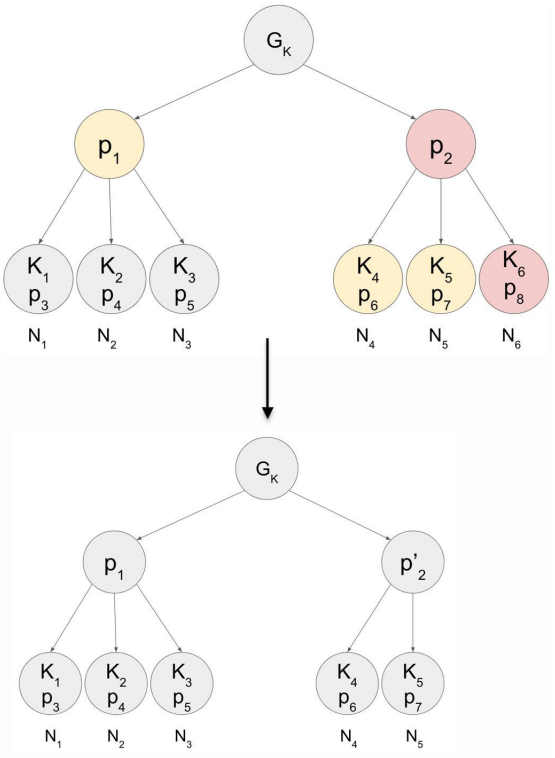


FIGURE 9. The figure shows an example in which a node leaves a group.

leaving node is shared. An example of such a prime number p_{new} is p'_2 in the lower figure in Fig. 9.

VI. SECURITY ANALYSIS

In this section, we explain how our proposed scheme achieves security using various mechanisms.

1) Our protocol ensures secure group authentication since in the initialization and authentication stage (see Section V-A), we use the nonce $\tilde{n}_i$ and the secret key K_i for authentication purposes. Since only the server and the desired node (node i) have this nonce and secret key, a malicious node cannot access the system.

2) We add an additional layer of security during authentication by implementing checks on both the server side and node side. These checks are performed in steps 3 and 7 of Fig. 2; both the server and the node check the hash of identity concatenated with nonce number, $h(I_i||n_i)$ and $h(I_s||n_i)$, respectively. If they do not match, then authentication is considered as failed.

3) The group key is constructed by the server using the nonce values N_i received from all the nodes and the secret keys K_i as explained in Section V-B. We ensure easy and secure sharing of the group key with all the nodes of the group. In particular, the server sends an interim group key $(G^*_{k,i})$ instead of sending the actual group key. Even if a malicious node gets access to an interim group key $G^*_{k,i}$, it will not be able to obtain the actual group key G_k from it since it does not have the nonce N_i and secret key K_i.

4) Our protocol is resistant to Man-in-the-Middle attacks since information sent to the group is encrypted with the group key. Also, all communication is done using symmetric key encryption, which is less expensive than public key encryption.

5) The proposed scheme achieves backward secrecy by hash updating of the group key by the server when a new node joins the group. The new group key is sent to the new node using encryption; hence, security is maintained. Also, since a cryptographic hash function is a one-way function, the new node cannot obtain the old group key using the new group key.

6) When a node leaves a group, the proposed scheme achieves forward secrecy by leveraging the CRT and the prime numbers from the internal and leaf nodes of the factorial tree to efficiently and securely communicate the nonce N_S, using which the new group key is computed, to the non-leaving nodes, while keeping it secret from the leaving node (see Section V-D).

VII. PERFORMANCE ANALYSIS

In this section, we analyze the storage, communication, and computation costs of our proposed scheme.

TABLE 1. Height of factorial tree for a given number of nodes.

n	t (height of tree)
1	1
2	2
3 - 6	3
7 - 24	4
25 - 120	5
121 - 720	6
721 - 5039	7

A. STORAGE COST

Table 1 shows the height of the factorial tree as a function of the number of nodes, n, in the group. For a given number of nodes, n, in the first column of Table 1, the height of the corresponding factorial tree, t, is shown in the second column. The height of the factorial tree is obtained by solving the following equation:

$$\min_t t! \geq n.$$

1) STORAGE COST OF NODE

Each node in the group stores the prime number values of the nodes that lie on the path from the root node to itself. If the height of the factorial tree is t, each node stores $t-1$ prime number values. Apart from these prime numbers, each node also stores the group key, G_k, nonce, N_i, and its own secret key, K_i. Therefore, the total number of values stored by the node is given by $t+2$. Hence, the storage complexity for each node is of the order $O(t)$. Fig. 10 shows the storage cost of a node versus the total number of nodes in the group.

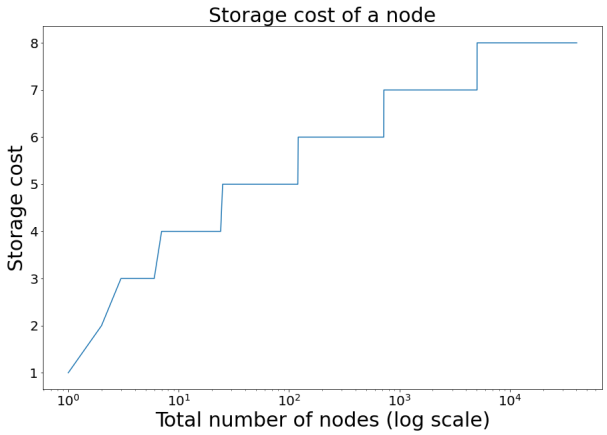


FIGURE 10. The figure shows the storage cost of a node versus the total number of nodes in the group.

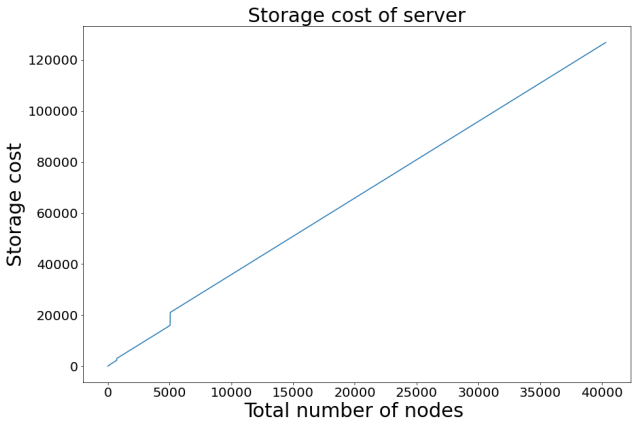


FIGURE 11. The figure shows the storage cost of the server versus the number of nodes in the group.

2) STORAGE COST OF SERVER

The server must store the prime number values from all the internal nodes as well as the leaf nodes of the group. When a factorial tree of height t is full, the leaf node level of the tree contains $t!$ nodes. When the factorial tree is not full, the number of nodes in the leaf node level, n_t, will be less than $t!$, i.e., $0 < n_t < t!$, and the tree will be full up to level $t-1$, i.e., level $t-1$ will have $(t-1)!$ nodes. Therefore, the total number of prime numbers to be stored by the server is divided into two parts:

Prime numbers stored in nodes up to level $t-1$:

For a factorial tree of height t, there are $l!$ internal nodes in level l of the tree. Hence, the number of prime numbers stored up to level $t-1$, where t is the height of the tree, is given by:

$$(\sum_{l=1}^{t-1} l!) - 1$$

The -1 term exists because of the fact that the root node does not have any prime number associated with it.

Prime numbers stored at level t:

Upon inspection of factorial trees containing different numbers of nodes, it can be deduced that the number of nodes in level t, n_t, is given by:

$$n_t = n - (t-1)! + \lceil \{n - (t-1)!\}/(t-1) \rceil.$$

The total number of prime numbers to be stored by the server is the sum of the above two parts, i.e.:

$$n_t + (\sum_{l=1}^{t-1} l!) - 1$$

Apart from these prime values, the server also stores n secret key values, n nonce values, and a group key. Hence, the total storage cost of the server is given by:

$$\text{Storage cost: } n_t + 2 \cdot n + \sum_{l=1}^{t-1} l!.$$

Fig. 11 shows the storage cost of the server versus the number of nodes in the group.

B. COMMUNICATION AND COMPUTATION COST

In this subsection, considering a group containing n nodes, we analyze the communication and computation costs for different stages of the proposed scheme.

1) GROUP AUTHENTICATION AND INITIALIZATION

During this stage, the server computes $2n$ cryptographic hashes, $2n$ encryptions, and $2n$ decryptions. For each node, the server sends two messages and hence sends $2n$ messages overall. Hence, the communication and computation costs are both of the order $O(n)$.

Each node computes 2 cryptographic hashes, 2 encryptions, 1 decryption, sends 2 messages to the server, and receives 2 messages from the server. Hence, the communication and computation costs for a node are both of the order $O(1)$

2) GROUP KEY CREATION AND DISTRIBUTION

The server performs n hash computations and $n-1$ XOR operations in the group key formation stage. It then computes 1 hash function, 1 XOR operation, and 1 encryption per node to compute the message authentication code, interim group key, and encrypted message, respectively. Overall, the server computes $2n$ hash functions and performs $2n-1$ XOR operations and n encryptions. The server then sends one message to each node, and hence, n messages overall. Thus, for this stage of the algorithm, the communication and computation costs of the server are both of the order $O(n)$.

At this stage, each node performs 1 decryption, 1 XOR operation, 1 cryptographic hash, and does not send any messages, leading to $O(1)$ computation cost and no communication cost.

3) NODE ADDED TO GROUP

In this part of the algorithm, the server computes 1 hash function, performs 1 encryption, and sends 1 broadcast message to all group members and 1 direct message to the new node, leading to computation and communication costs of the order $O(1)$.

Each node performs one decryption to obtain the new group key and does not send any messages.

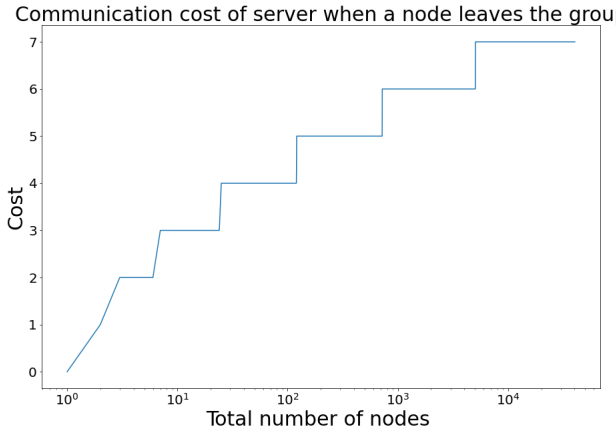


FIGURE 12. The figure shows the communication cost of the server when a node leaves the group versus the total number of nodes in the group.

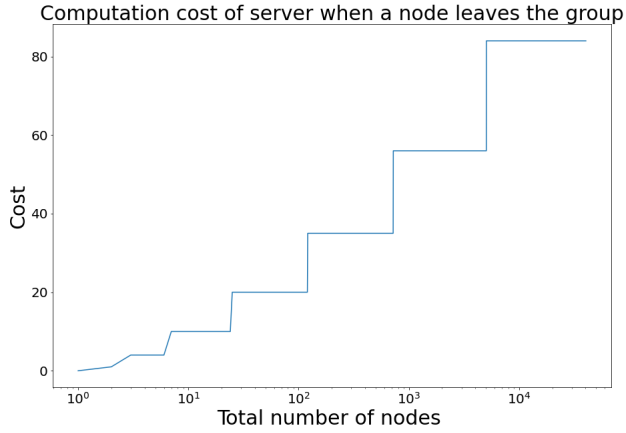


FIGURE 13. The figure shows the computation cost of the server when a node leaves the group versus the total number of nodes in the group.

4) NODE LEAVING GROUP

The server computes 1 unique solution, X, for sharing the nonce N_S with the non-leaving nodes. This unique solution X is computed with the help of $1 + 2 + \ldots + (t - 1)$ prime numbers, where t is the height of the factorial tree. As an example, in Fig. 9, the height of the tree is 3, and the leaving node (marked red) leaves from level $t = 3$. To construct the unique solution X in this case, we need prime numbers from the non-leaving part (marked yellow), which make up the set P^*. The number of such prime numbers (corresponding to nodes marked yellow) in this example is 0 from level 1, 1 from level 2, and 2 from level 3, i.e., $0 + 1 + 2$; by extrapolating this logic, the number of prime numbers required to construct X for a general case of a factorial tree of height t is given by $1+2+\ldots+(t-1)$. The computation cost required to construct this unique solution is of the order $O(t^2)$.

It then performs 1 encryption and sends 1 broadcast message. This leads to communication and computation costs of the order $O(1)$ and $O(t^2)$, respectively. This unique solution construction and communication is also to be done for $t - 2$ prime number values, which also need to be rekeyed (see Section V-D). This leads to the total communication and computation costs of the order $O(t)$ and $O(t^3)$, respectively. Figs. 12 and 13 show the communication and computation costs of the server, respectively, versus the total number of nodes in the group.

In this phase, to receive and process the unique solution, X, each node performs 1 decryption, 1 modulus operation, 1 subtraction, computes 1 cryptographic hash function, and does not send any messages as can be seen from Fig. 8. This process needs to be done once for sharing the value of N_S and $t - 2$ times for rekeying the internal prime number values (see Section V-D). Consequently, this results in a node computation cost of the order $O(t)$ and incurs no communication cost for any node.

VIII. NUMERICAL RESULTS

In this section, we compare the performance of our proposed scheme with those proposed in the prior works *Groupit* [2] and *Kabra* et al. [37]. Both these works use the Logical Key Hierarchy (LKH) model proposed in [8]. In the LKH model, the nodes/users are arranged in a binary tree structure for efficient and scalable rekeying. We improve upon this model by using a factorial tree instead of a binary tree in the internal structure, which results in better computation and communication costs.

The study conducted in [42] compared the computational complexity of the Advanced Encryption Standard (AES) symmetric key encryption algorithm, the SHA-256 cryptographic hash function, and Elliptical Curve Cryptography (ECC), which is used in asymmetric encryption. Their results indicated that ECC requires significantly more computational resources than AES, while AES is more computationally demanding than SHA-256.

As stated in [42], we assume that hashing (Hash) through SHA-256 requires $T_0 = 460$ ns, encryption (Enc) or decryption (Dec) using AES-256 with a block size of 64 takes $T_1 = 800$ ns, which is equal to $1.74 \times T_0$, and asymmetric decryption (AsyDec) using ECC-224 takes $T_2 = 114000$ ns, which is equal to 247.83 T_0.

As mentioned in Section II, *Groupit* [2] and *Kabra* et al. [37] have a two-tier GKM scheme wherein one of the tiers has the Users/Subscribers (respectively, Devices) of a User Group (respectively, Device Group) occupy the leaf nodes of a binary tree structure. This arrangement is shown in Fig. 14 and corresponds to the inner structure of the two-tier GKM scheme. In the two-tier GKM scheme, the other tier, viz., the outer structure, consists of an LKH arrangement of the previously mentioned User/Subscriber Groups (SGs) or Device Groups (DGs), i.e., the SGs (respectively, DGs) themselves occupy the leaf nodes of a binary tree in the outer tree structure as shown in Fig. 15.

Our proposed scheme replaces the binary tree in Fig. 14 with a factorial tree. The Devices (respectively, Users) in the

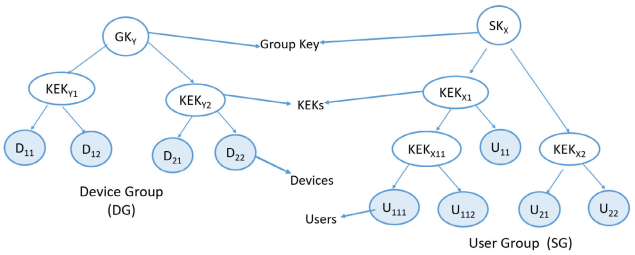


FIGURE 14. The figure shows the inner structure of the two-tier GKM scheme proposed in [2] and [37]. Each node other than the root and leaves is associated with a key called the Key Encryption Key (KEK).

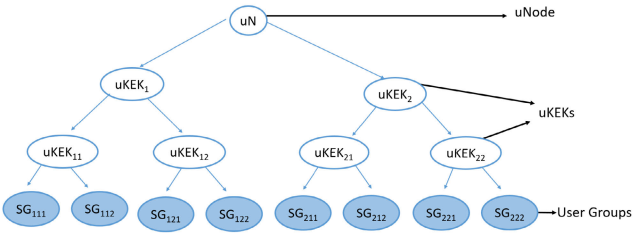


FIGURE 15. The figure shows the outer structure of the two-tier GKM scheme proposed in [2] and [37]. Each node other than the root and leaves is associated with a KEK.

DG (respectively, SG) mentioned in [2] and [37] are replaced by nodes in our scheme.

To compare the performance of our proposed scheme with those in [2] and [37], we assume that our model has an outer structure similar to the one in Fig. 15 and an inner structure as described in the previous sections. We perform numerical computations on a system with one SG and one DG– similar results hold in the case where there are multiple SGs and DGs. In the following discussion, we consider the DG to be comprised of 50 devices and the SG to be comprised of M users. To maintain consistency in notation with respect to previous sections, the users in the SG are referred to as nodes in subsequent discussions.

For convenience, we introduce the integer-valued inverse of the factorial function, $\alpha(n)$, which computes the height of the factorial tree (t) for a given number of nodes (n):

$$\alpha(n) = x \text{ where } x \text{ is the solution of } \min_x x! \geq n.$$

A. COMMUNICATION COST COMPARISON

We first calculate the number of messages exchanged when a node leaves. The numbers of messages exchanged under different algorithms are given by:

Groupit:
1 Broadcast + $(\lceil \log_2(M) \rceil + \lceil \log_2(50) \rceil + 50)$ multicasts
Kabra et al.:
1 Broadcast + $(\lceil \log_2(M) \rceil + 2)$ multicasts
Our proposed algorithm:
1 Broadcast + $(\alpha(n) + 2)$ multicasts

Both the prior works use binary tree structures and hence involve the $\lceil \log_2(M) \rceil$ term in the rekeying steps. In contrast, the rekeying step in our proposed algorithm varies as $\alpha(n)$. Since $\alpha(n)$ increases slower than $\lceil \log_2(M) \rceil$, our proposed algorithm has lower communication costs than those of prior works, as can be seen from the plot in Fig. 16, which compares the communication costs of different schemes when a node leaves.

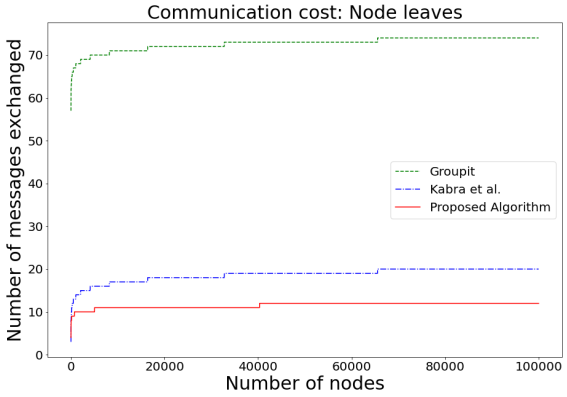


FIGURE 16. The figure shows the communication costs under GroupIt, Kabra et al. and our proposed algorithm versus the number of nodes for the case when a node leaves.

B. COMPUTATION COST COMPARISON

In this subsection, we compare the computation costs under different schemes when a node leaves or joins the group for the system described earlier.

1) USER JOINS

When a node joins a group, for the system described earlier consisting of M nodes and 50 devices, the computation cost for the Groupit scheme is almost the same as for the one in Kabra et al. We compare the computation performance of those two schemes and that of our proposed algorithm. When a new node joins, the total computation costs incurred by all the nodes under different schemes are given by:

Groupit:
2M Hash + $\log_2(M)$ Dec $\implies$ $2MT_0 + \log_2(M) \cdot 1.74\ T_0$
Kabra et al.:
2M Hash + $\log_2(M)$ Dec + 1 Dec
$\implies$ $2MT_0 + (\log_2(M) + 1) \cdot 1.74\ T_0$
Our proposed algorithm:
M Dec $\implies$ M$\cdot$ $1.74\ T_0$

Fig. 17 shows that our proposed algorithm has lower computation costs than those under the schemes proposed in prior works. This is due to the absence of the $\log_2(M)$ term and also because just one decryption per node is required in our proposed algorithm compared to 2 hash function computations per node under the schemes in prior works. Compared to Fig. 16, the difference between the computation costs of our proposed scheme and those of prior works is less prominent because of the linear dependence on M under all the three schemes.

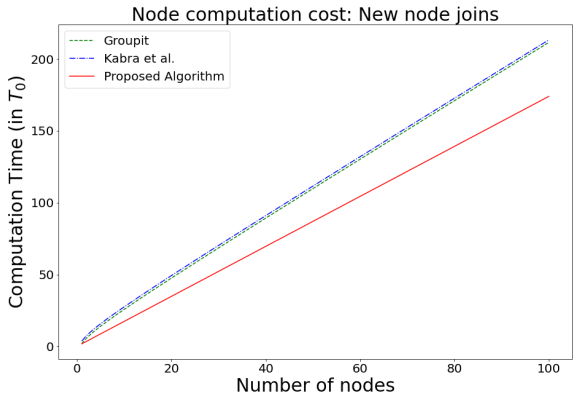


FIGURE 17. The figure shows the total computational cost for all nodes under GroupIt, Kabra et al. and our proposed algorithm versus the number of nodes for the case when a new node joins.

2) USER LEAVES

Finally, we compare the total computation costs incurred by all the nodes when a node leaves the group. The costs are as follows:

Groupit:

$$2\text{M Hash} + \log_2(M)\,\text{Dec} \implies 2MT_0 + \log_2(M) \cdot 1.74\,T_0$$

Kabra et al.:

$$2\text{M Hash} + \log_2(M)\,\text{Dec} + 1\,\text{Dec} \implies 2MT_0 + (\log_2(M) + 1) \cdot 1.74\,T_0$$

Our proposed algorithm:

$$\text{M Dec} \implies \text{M} \cdot 1.74\,T_0$$

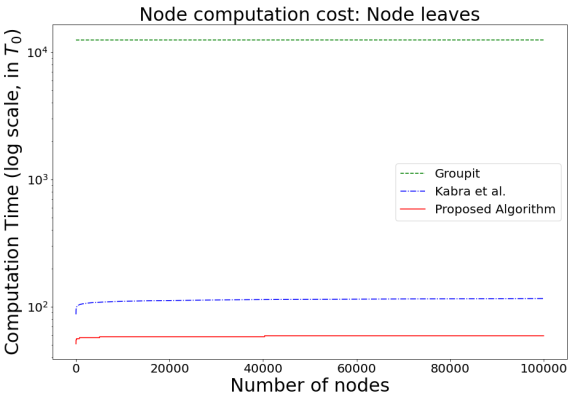


FIGURE 18. The figure shows the total computation cost for all nodes under Groupit, Kabra et al. and our proposed scheme versus the number of nodes for the case when a node leaves.

Fig. 18 shows a comparison of the total computation costs of all the nodes when a node leaves under the three schemes. Since the schemes in prior work use a binary tree structure that grows exponentially with the base of 2, the computation cost is the inverse of the exponential function, which leads to a logarithmic term in their respective expressions. In contrast, in our proposed algorithm, we use a factorial tree. Hence, the computation cost is proportional to the inverse of the factorial function (denoted earlier by $\alpha(n)$). Since the factorial function increases faster than the exponential function, the inverse of the factorial function ($\alpha(n)$) increases slowly compared to the inverse of the exponential function. Due to this reason, our proposed algorithm has lower computation cost, as can be seen from Fig. 18.

IX. CONCLUSION AND FUTURE WORK

We proposed a new scheme, based on factorial trees and the CRT, for GKM in dynamic IoT scenarios. We evaluated the performance of our proposed scheme via extensive mathematical analysis and numerical computations. Using numerical computations, we showed that our proposed scheme outperforms GKM schemes from prior work in terms of communication and computation costs. Our protocol arranges the nodes in each group into a factorial tree structure, and an interesting direction for future work is to extend the idea of factorial trees to arrange the groups themselves into an outer factorial tree structure, i.e., nodes in each group are arranged into a factorial tree structure, and different groups are in turn arranged into another factorial tree structure. This arrangement may lead to further improvement in the communication and computation costs.

REFERENCES

[1] R. H. Weber, "Internet of Things—New security and privacy challenges," *Comput. Law Secur. Rev.*, vol. 26, no. 1, pp. 23–30, Jan. 2010.
[2] Y.-H. Kung and H.-C. Hsiao, "GroupIt: Lightweight group key management for dynamic IoT environments," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 5155–5165, Dec. 2018.
[3] Y. Challal and H. Seba, "Group key management protocols: A novel taxonomy," *Int. J. Comput. Inf. Eng.*, vol. 2, no. 10, pp. 3620–3633, 2008.
[4] A. Mosenia and N. K. Jha, "A comprehensive study of security of Internet-of-Things," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 4, pp. 586–602, Oct. 2017.
[5] S. Chandra, S. Paira, S. S. Alam, and G. Sanyal, "A comparative survey of symmetric and asymmetric key cryptography," in *Proc. Int. Conf. Electron., Commun. Comput. Eng. (ICECCE)*, Nov. 2014, pp. 83–93.
[6] R. Vaid and V. Katiyar, "Security issues and remidies in wireless sensor Networks- a survey," *Int. J. Comput. Appl.*, vol. 79, no. 4, pp. 31–39, Oct. 2013.
[7] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th ed. London, U.K.: Pearson, 2017.
[8] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 68–79, 1998.
[9] M. A. Mughal, P. Shi, A. Ullah, K. Mahmood, M. Abid, and X. Luo, "Logical tree based secure rekeying management for smart devices groups in IoT enabled WSN," *IEEE Access*, vol. 7, pp. 76699–76711, 2019.
[10] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in Internet-of-Things," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1250–1258, Oct. 2017.
[11] J. Granjal, E. Monteiro, and J. Sá Silva, "Security for the Internet of Things: A survey of existing protocols and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1294–1312, 3rd Quart., 2015.
[12] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.
[13] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, Jan. 2015.
[14] J. A. Gutierrez, M. Naeve, E. Callaway, M. Bourgeois, V. Mitter, and B. Heile, "IEEE 802.15.4: A developing standard for low-power low-cost wireless personal area networks," *IEEE Netw.*, vol. 15, no. 5, pp. 12–19, Sep. 2001.
[15] Z. Sheng, S. Yang, Y. Yu, A. V. Vasilakos, J. A. Mccann, and K. K. Leung, "A survey on the IETF protocol suite for the Internet of Things: Standards, challenges, and opportunities," *IEEE Wireless Commun.*, vol. 20, no. 6, pp. 91–98, Dec. 2013.

[16] L. F. Schrickte, C. Montez, R. d. Oliveira, and A. R. Pinto, "Integration of wireless sensor networks to the Internet of Things using a 6LoWPAN gateway," in *Proc. 3rd Brazilian Symp. Comput. Syst. Eng.*, Dec. 2013, pp. 119–124.

[17] C.-S. Park, "A secure and efficient ECQV implicit certificate issuance protocol for the Internet of Things applications," *IEEE Sensors J.*, vol. 17, no. 7, pp. 2215–2223, Apr. 2017.

[18] M. Turkanović, B. Brumen, and M. Hölbl, "A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion," *Ad Hoc Netw.*, vol. 20, pp. 96–112, Sep. 2014.

[19] Q. Jiang, S. Zeadally, J. Ma, and D. He, "Lightweight three-factor authentication and key agreement protocol for Internet-integrated wireless sensor networks," *IEEE Access*, vol. 5, pp. 3376–3392, 2017.

[20] C.-C. Chang and H.-D. Le, "A provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 1, pp. 357–366, Jan. 2016.

[21] A. K. Das, S. Kumari, V. Odelu, X. Li, F. Wu, and X. Huang, "Provably secure user authentication and key agreement scheme for wireless sensor networks," *Secur. Commun. Netw.*, vol. 9, no. 16, pp. 3670–3687, Nov. 2016.

[22] A. Bin-Rabiah, K. K. Ramakrishnan, E. Liri, and K. Kar, "A lightweight authentication and key exchange protocol for IoT," in *Proc. Workshop Decentralized IoT Secur. Standards*, 2018, pp. 1–6.

[23] M. Abdussami, R. Amin, and S. Vollala, "Provably secured lightweight authenticated key agreement protocol for modern health industry," *Ad Hoc Netw.*, vol. 141, Mar. 2023, Art. no. 103094.

[24] A. Gupta and G. S. Kasbekar, "Secure, anonymity-preserving and lightweight mutual authentication and key agreement protocol for home automation IoT networks," in *Proc. 14th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2022, pp. 375–383.

[25] P. Porambage, A. Braeken, C. Schmitt, A. Gurtov, M. Ylianttila, and B. Stiller, "Group key establishment for secure multicasting in IoT-enabled wireless sensor networks," in *Proc. IEEE 40th Conf. Local Comput. Netw. (LCN)*, Oct. 2015, pp. 482–485.

[26] N. Ferrari, T. Gebremichael, U. Jennehag, and M. Gidlund, "Lightweight group-key establishment protocol for IoT devices: Implementation and performance analyses," in *Proc. 5th Int. Conf. Internet Things, Syst., Manage. Secur.*, Oct. 2018, pp. 31–37.

[27] D. Balenson, D. McGrew, and A. Sherman, "Key management for large dynamic groups: One-way function trees and amortized initialization," Internet-Draft, Los Angeles, CA, USA, Tech. Rep. draft-balenson-groupkeymgmt-oft-00, 1999.

[28] F. Samiullah, S. Akeylek, M. Lee Gan, and Y. Aun, "Group key management in resource constraint environment: Applications and use cases," *Int. J. Adv. Natural Sci. Eng. Researches*, vol. 7, pp. 269–278, Apr. 2023.

[29] Z. Yang, Z. Wang, F. Qiu, and F. Li, "A group key agreement protocol based on ECDH and short signature," *J. Inf. Secur. Appl.*, vol. 72, Feb. 2023, Art. no. 103388.

[30] L. Veltri, S. Cirani, S. Busanelli, and G. Ferrari, "A novel batch-based group key management protocol applied to the Internet of Things," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2724–2737, Nov. 2013.

[31] P. Vijayakumar, S. Bose, and A. Kannan, "Chinese remainder theorem based centralised group key management for secure multicast communication," *IET Inf. Secur.*, vol. 8, no. 3, pp. 179–187, May 2014.

[32] H. Tan, "An efficient IoT group association and data sharing mechanism in edge computing paradigm," *Cyber Secur. Appl.*, vol. 1, Dec. 2023, Art. no. 100003.

[33] G. Heo, K. Chae, and I. Doh, "Hierarchical blockchain-based group and group key management scheme exploiting unmanned aerial vehicles for urban computing," *IEEE Access*, vol. 10, pp. 27990–28003, 2022.

[34] C. Trivedi and U. P. Rao, "Secrecy aware key management scheme for Internet of Healthcare Things," *J. Supercomput.*, vol. 79, no. 11, pp. 12492–12522, Jul. 2023.

[35] Z. Wang, R. Huo, and S. Wang, "A lightweight certificateless group key agreement method without pairing based on blockchain for smart grid," *Future Internet*, vol. 14, no. 4, p. 119, Apr. 2022.

[36] R. Prabha, M. Razmah, S. Senthilpandi, S. Suganthi, and S. Sridevi, "Design of a novel group communication framework to improve security in Internet of Things," in *Proc. 8th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, vol. 1, Mar. 2022, pp. 967–970.

[37] A. Kabra, S. Kumar, and G. Kasbekar, "Efficient, flexible and secure group key management protocol for dynamic IoT settings," *EAI Endorsed Trans. Internet Things*, vol. 7, no. 25, Apr. 2021, Art. no. 168862.

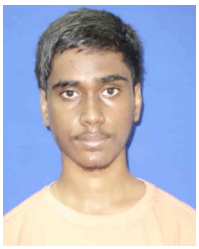
[38] M. E. O'neill, "The genuine sieve of eratosthenes," *J. Funct. Program.*, vol. 19, no. 1, pp. 95–106, Jan. 2009.

[39] N. Khairina, "The comparison of methods for generating prime numbers between the sieve of Eratosthenes, Atkins, and Sundaram," *SinkrOn*, vol. 3, no. 2, p. 293, Apr. 2019.

[40] M. Conti, R. Di Pietro, L. V. Mancini, and A. Mei, "Emergent properties: Detection of the node-capture attack in mobile wireless sensor networks," in *Proc. 1st ACM Conf. Wireless Netw. Secur.*, Mar. 2008, pp. 214–219.

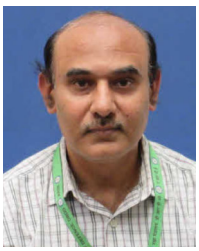
[41] T. Bonaci, L. Bushnell, and R. Poovendran, "Node capture attacks in wireless sensor networks: A system theoretic approach," in *Proc. 49th IEEE Conf. Decis. Control (CDC)*, Dec. 2010, pp. 6765–6772.

[42] A. de la Piedra, A. Braeken, and A. Touhafi, "A performance comparison study of ECC and AES in commercial and research sensor nodes," in *Proc. Eurocon*, Jul. 2013, pp. 347–354.



K. SUDHEERADH received the B.Tech. and M.Tech. degrees in electrical engineering from the Department of Electrical Engineering, IIT Bombay, in 2023. His research interests include network security and the Internet of Things.

N. NEHA JAHNAVI received the B.Tech. and M.Tech. degrees in electrical engineering from the Department of Electrical Engineering, IIT Bombay, in 2023. Her research interests include network security and the Internet of Things.



PRAMOD N. CHINE received the M.Tech. degree in electrical engineering from the Department of Electrical Engineering, IIT Bombay, in 2004, where he is currently pursuing the Ph.D. degree. His research interests include game theory and network security.

GAURAV S. KASBEKAR (Member, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology (IIT), Bombay, Mumbai, India, in 2004, the M.Tech. degree in electronics design and technology (EDT) from the Indian Institute of Science (IISc), Bengaluru, India, in 2006, and the Ph.D. degree in electrical engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2011. He is currently an Associate Professor with the Department of Electrical Engineering, IIT Bombay. His research interests include modeling, design, and analysis of wireless networks. He was a recipient of the CEDT Design Medal for being adjudged the best master's student in EDT with IISc.