

## **Secure group key agreement protocol with elliptic curve secret sharing for authentication in distributed environments**

Raju M. Sairise <sup>†</sup>

*Yadavrao Tasgaonkar College of Engineering and Management*

*Karjat*

*Raigad*

*Maharashtra*

*India*

Suresh Limkar <sup>\*</sup>

*Department of Artificial Intelligence & Data Science*

*AISSMS Institute of Information Technology*

*Pune*

*Maharashtra*

*India*

Sarika T. Deokate <sup>§</sup>

*Pimpri Chinchwad College of Engineering Nigdi*

*Pune*

*Maharashtra*

*India*

Shrinivas T. Shirkande <sup>‡</sup>

*Department of Computer and Incharge Principal*

*S. B. Patil College of Engineering Indapur*

*Pune*

*Maharashtra*

*India*

---

<sup>†</sup> E-mail: [rsairise566@gmail.com](mailto:rsairise566@gmail.com)

<sup>\*</sup> E-mail: [sureshlimkar@gmail.com](mailto:sureshlimkar@gmail.com) (Corresponding Author)

<sup>§</sup> E-mail: [sarikajaankar@gmail.com](mailto:sarikajaankar@gmail.com)

<sup>‡</sup> E-mail: [shri.shirkande8@gmail.com](mailto:shri.shirkande8@gmail.com)

Rupali Atul Mahajan<sup>@</sup>

Department of Computer Engineering  
Vishwakarma Institute of Information Technology Pune  
Pune  
Maharashtra  
India

Anil Kumar<sup>#</sup>

Department of Artificial Intelligence & Data Science  
Poornima Institute of Engineering & Technology  
Jaipur  
Rajasthan  
India

---

## Abstract

Secure communication is crucial in today's increasingly interconnected world, especially in distributed contexts where several parties must cooperate while maintaining confidentiality and authenticity. In order to overcome these difficulties, this work introduces a novel Secure Group Key Agreement Protocol (SGKAP) strengthened with Elliptic Curve Secret Sharing (ECSS). The proposed SGKAP uses the strength of elliptic curve cryptography to provide a shared secret key across participants, assuring data transmission confidentiality. Contrary to conventional methods, SGKAP integrates ECSS as a second layer of security to verify participant identity, thus boosting the protocol's resistance to malevolent adversaries. To create a shared group key, members exchange their elliptic curve public keys at the start of the protocol. During this stage, ECSS is used, and participants create distinctive authentication tokens using their personal secrets. Participants exchange and verify these tokens among themselves, ensuring that only those with the proper permissions have access to the group key. In addition to guaranteeing data confidentiality, SGKAP also verifies the group members' participation. Furthermore, by mandating secret-sharing-based authentication, ECSS strengthens the protocol's resistance to insider attacks. We carried out a comprehensive set of simulations and experiments in various distributed contexts to assess the performance and security of SGKAP with ECSS.

---

**Subject Classification:** Primary 68T07 Secondary 68M25.

**Keywords:** Key agreement protocol, Elliptic curve, Authentication, Security, Cryptography.

## I. Introduction

The way individuals cooperate remotely has been revolutionised by the introduction of new networking technology and the accelerating speed

---

<sup>@</sup> E-mail: [rupali.mahajan@viit.ac.in](mailto:rupali.mahajan@viit.ac.in)

<sup>#</sup> E-mail: [anil.vashu@gmail.com](mailto:anil.vashu@gmail.com)

of internet connectivity. This has led to the development of numerous group applications that enable users, who are frequently spread out geographically, to collaborate easily. The significance of secure group communication in these collaborative settings, however, cannot be emphasised. A Key Generation Centre (KGC), a dependable third party, is frequently involved in existing group key establishment methods [1] [2]. In many situations, group key establishment techniques pose severe security risks and a single point of failure due to their reliance on a reliable Key Generation Centre (KGC). The security of the established group key can be jeopardised by a malicious KGC, which could result in data breaches and unauthorised access to sensitive information [3].

Establishing a secure group key in a dispersed setting without a KGC becomes a very difficult undertaking. Traditional cryptographic techniques sometimes rely on centralised bodies, such as the KGC, to safely generate and distribute keys. Alternative methods are needed to establish group keys in situations when a central authority is not possible or acceptable. These difficulties are exacerbated in distributed contexts with limited resources, such as the Internet of Things (IoT) [4]. It might be challenging to carry out intricate cryptographic operations or handle huge key sizes on lightweight devices since they sometimes have constrained CPU resources. Therefore, creative solutions that strike a balance between security and resource efficiency are required to achieve safe group communication in IoT and other environments with limited resources.

The establishment of a strong and private group key is the basis of secure communication in such group applications where participants share similar goals and interests [5]. Many group key establishment procedures have historically relied on the participation of reliable third parties, frequently referred to as Key Generation Centres (KGCs), to distribute session keys among group members. By incorporating Elliptic Curve Secret Sharing (ECSS) into the key establishment process, the proposed Secure Group Key Agreement Protocol (SGKAP) presents a novel technique [6]. SGKAP provides improved security against frequent threats including eavesdropping and man-in-the-middle attacks by utilising elliptic curves [7]. This consideration is essential because contemporary distributed systems, such as the Internet of Things (IoT), frequently include lightweight devices with constrained processing power. SGKAP offers a potential approach for securing communication in IoT networks, cloud computing, and other collaborative applications by optimising computational performance and key size.

Contribution of paper is given as:

- The method makes a significant contribution by integrating ECSS for enhanced security.
- The protocol's applicability for resource-constrained distributed contexts, such the Internet of Things (IoT), is another important contribution.
- The protocol is in line with the needs of contemporary IoT networks and other contexts with similar resource constraints thanks to this efficiency, which is essential for guaranteeing that secure group communication can be maintained without placing an unnecessary computational burden on the participating devices.

## II. Review of Literature

Group communications are common in many contexts, including corporate networks, secure IoT installations, and military operations. In these situations, it is crucial to utilise a shared encryption key to safeguard the integrity and confidentiality of the sent data [8]. A central Key Generation Centre (KGC), which produces and distributes group keys to all authorised users, is the traditional method for securing group communication. A malicious attack on the KGC can result in a serious security lapse because the attacker will have access to all group keys [9].

In secure group key agreement methods, achieving security in a dispersed setting without depending on a reliable central body like a KGC poses a considerable difficulty. To meet this difficulty, researchers have looked into a variety of cryptographic algorithms [10]. One significant strategy is the use of secret sharing systems, which divide up the group key among various users and make it mathematically difficult for any one user to piece together the entire key.

It is essential to ensure that group keys are created effectively with the least amount of computational overhead [11]. Researchers have concentrated on improving the computational components of group key agreement methods [12] for IoT in order to overcome these difficulties. This includes utilising effective key distribution strategies and compact cryptographic algorithms. Adversaries may try to undermine the protocol through aggressive attacks, such listening in on conversations or introducing harmful data, or passive assaults, like attempting to acquire private data without disrupting communication.

Public Key Infrastructure (PKI) is a system that enables secure key exchange by utilising digital certificates and a reputable certificate authority (CA) [13]. These methods, called password-based key exchange (PBKE), let two parties create a shared secret key based on a password. They are frequently employed when users have a shared secret but lack public/private key pairings [14]-[18].

### III. Elliptic Curve Algorithm

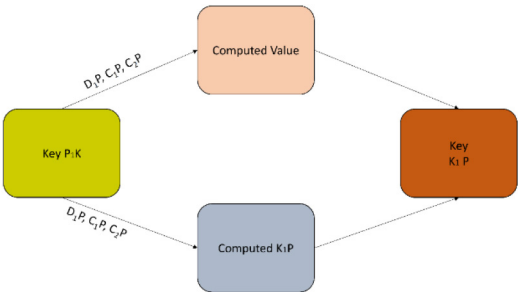


Figure 1

Framework of Elliptic Curve Secure Communication

Above Figure 1 Presents the Framework for Elliptic Curve Secure Communication. It is often known as ECs, are mathematical structures whose definitions can be found in equations of the type  $y^2 = x^3 + ax + b$ , where  $a$  and  $b$  are constants. These curves are ideal for use in cryptography applications because they have special algebraic characteristics. The fundamental operations of elliptic curve cryptography, point addition and point doubling, are two examples of such properties. Finding a third point that is also on the elliptic curve requires taking two points on the curve and adding them together. This process is crucial to key agreement protocols because it enables parties to securely combine their individual public keys to produce a shared secret key. Contrarily, point doubling entails taking a single point and adding it to itself to produce a brand-new point on the curve. These mathematical procedures may be computed quickly and serve as the foundation for safe key exchange.

A key obstacle in elliptic curve cryptography is the (ECDLP). In the equation  $P = kG$ , where  $P$  is a known point on the curve,  $G$  is a known generator point, and  $k$  is the unknown scalar, the task is to discover the exponent (or scalar)  $k$ . Many elliptic curve cryptographic protocols'

security, including key agreement, depends on the ECDLP's computational difficulty, even for strong adversaries.

### Step 1: Initialization

Parties A and B agree on ECC parameters:

- EC equation:  $X^2 = Y^3 + cx + b$
- Base point G on the curve
- ECC prime modulus p
- ECC order n

### Step 2: Private Key Generation

Parties A and B each generate a private key:

A: a

B: b

### Step 3: Public Key Calculation

Parties K and L calculate their public keys on the elliptic curve:

- K computes:  $A = a * G$
- L computes:  $B = b * G$

### Step 4: Diffie-Hellman Key Exchange

Parties A and B perform a Diffie-Hellman key exchange using their ECC public keys:

A computes:  $s = a * B$

B computes:  $s = b * A$

### Key Exchange Authentication Algorithm:

#### Step 1: Initialization

- Parties A and B initialize their respective secrets:
- A: a (private key)
- B: b (private key)

**Step 2: Public Parameters**

- Parties exchange public parameters:
- A sends to B:  $A = g^a \bmod p$
- B sends to A:  $B = g^b \bmod p$

**Step 3: Key Calculation**

Parties calculate the shared secret key using received parameters:

- A computes:  $s = B^a \bmod p$
- B computes:  $s = A^b \bmod p$

**Step 4: Authentication Check**

Parties independently verify the authenticity of each other:

- A checks:  $B = g^b \bmod p$
- B checks:  $A = g^a \bmod p$

**Step 5: Shared Secret Key**

If authentication checks are successful, both parties have the same shared secret key,  $s$ , for secure communication.

**III. Methodology**

In this section, we provide two brand-new cryptographic techniques intended to improve group key management's security and effectiveness. ECSSS: ECSSS is a two-round approach designed to safely distribute and reconstruct a shared key made from an elliptic curve  $E$ , abbreviated as  $k1P$ . A user, let's call user  $U1$ , computes  $n$  shares of this key in the first round, with  $n-1$  shares being designated as public and 1 being retained privately. This private share is sent across a secure channel to  $n$  people in an encrypted form. Users  $U2$ ,  $U3$ ,  $U4$ , and so forth work together to recreate key  $k1P$  in the second round. Figure 1 provides an illustration of the ECSSS framework.

ADGKAP: Designed for group key agreement with authentication, ADGKAP is based on ECSSS. Three rounds make up this protocol. Each user  $U_i$  (where  $i$  is a positive integer between 1 and  $n$ ) computes  $n-1$  private shares and  $n-1$  public shares of a point  $kiP$  in the first round. Then, via a

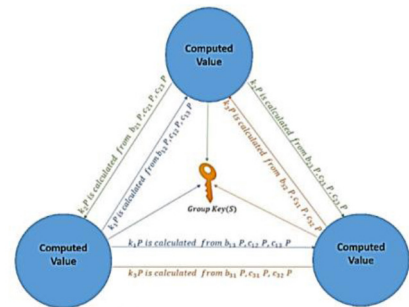


Figure 2

### System framework of Key Agreement Protocol

public channel, the encrypted private shares are sent. The received encrypted share and each user's private key are then used to rebuild each user's individual share. In the second round, with  $j$  ranging from 1 to  $n$ , and with the restriction that  $j$  not be equal to  $i$ , each user  $U_i$  reconstructs the key  $k_jP$  of another user  $U_j$ . Their respective share and  $n-1$  public shares are used to accomplish this. The group key  $S$ , which is effectively the sum of all rebuilt keys  $Pn k_jP$ , is separately recreated by each user in the last round. Additionally, each user in this round authenticates the group key by contrasting the hash of their own rebuilt key with the hashes of the keys of the other users. Figure 2 shows the Key Agreement Protocol structural layout.

### Algorithm:

#### Step 1: Initialization

Parties A and B agree on public parameters:

- A prime modulus  $p$  (a huge prime number)
- A original root  $g$  (a generator of the group modulo  $p$ )

#### Step 2: Private Key Generation

Parties A and B each generate a private key:

- A selects a secret integer  $a$
- B selects a secret integer  $b$

**Step 3: Public Key Calculation**

Parties K and L calculate their public keys using the parameters:

- K computes:  $K = g^a \bmod p$  (A's public key)
- L computes:  $L = g^b \bmod p$  (B's public key)

**Step 4: Key Exchange**

- Over the unsecure communication channel, Parties A and B exchange their public keys.

**Step 5: Shared Secret Key Calculation**

Parties A and B independently calculate the shared secret key ( $s$ ) using the received public keys and their private keys:

- A computes:  $s = B^a \bmod p$
- B computes:  $s = A^b \bmod p$

**Initialization:**

Public parameters:  $p$  (prime modulus),  $g$  (primitive root)

**Private keys:**  $a$  (L's private key),  $b$  (M's private key)

**Public Key Calculation:**

- L's public key:  $L = g^a \bmod p$
- M's public key:  $M = g^b \bmod p$

Key Exchange:

- L sends  $L$  to M.
- M sends  $M$  to L.

Shared Secret Key Calculation:

- L computes shared secret key:  $s = L^a \bmod p$
- M computes shared secret key:  $s = M^b \bmod p$

The shared secret key  $s$  calculated by both parties is identical, and it can be used as a cryptographic key for secure communication.

**IV. Result and Discussion**

A. *An attacker cannot rebuild the group key using  $n$  public shares:*

Let's assume that an attacker wishes to obtain  $kiP$  via  $n$  public shares. The attacker can create the following system of equations.  $kiP$  represents the target value that the attacker is trying to recover.

- $n$  is the number of shares.

- $ri1, ri2, \dots, ri(n-1)$  are coefficients associated with the shares.
- $zinP, zi(n+1)P, \dots, zi(n+i-2)P$  are points related to the shares.
- $ci1P, ci2P, \dots, ci(i-1)P$  are constants associated with the shares.

It seems like the attacker is trying to formulate a system of equations based on the relationships between the target value  $kiP$ , the coefficients, and the points associated with the shares.

| Steps | $U_1$                                                                                | $U_2$                                                                                | $U_3$                                                                                |
|-------|--------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| 1     | $h(S) = 34$<br>$C_1 = 34$ is public                                                  | $h(S) = 34$<br>$C_2 = 34$ is public                                                  | $h(S) = 34$<br>$C_3 = 34$ is public                                                  |
| 2     | $C_2' = 34$<br>$C_3' = 34$<br>$C_2' = C_2$<br>$C_3' = C_3$<br>Group key $S$ is Valid | $C_1' = 34$<br>$C_3' = 34$<br>$C_1' = C_1$<br>$C_3' = C_3$<br>Group key $S$ is Valid | $C_1' = 34$<br>$C_2' = 34$<br>$C_1' = C_1$<br>$C_2' = C_2$<br>Group key $S$ is Valid |

Figure 3

## Key Round 3 for Authentication

$$kiP + ri1zi(K + J)P + \dots + ri(n-1)zi(K + I)P = ci(I + 1)R$$

$$kiP + ri1zi(2n-2)P + \dots + ri(n-1)zi(2n-2)P = cinR$$

There are an infinite number of solutions to the system of equations because there are only  $n - 1$  equations and  $n$  unknowns. Every answer has an equal  $1/\ln$  probability. Large  $l$  causes  $1/\ln$  to be insignificant. As a result, the attacker is unable to get key  $kiP$ , and as a result,  $S = \sum_{i=1}^n kiP$ . Figure 4 shows Key Round 2 for Reconstruction and Figure 5 Shows Secret Key Distribution for Round 1

| Steps | $U_1$                                                                                                                                                                                                                                                            | $U_2$                                                                                                                                                                                                                                                            | $U_3$                                                                                                                                                                                                                                                        |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1     | $M_2 = \begin{bmatrix} 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 2 & 4 \end{bmatrix}$<br>$M_2^{-1}(d_{21}P, c_{21}P, c_{23}P)^T$<br>$= \begin{bmatrix} (116, 97) \\ (203, 437) \\ (285, 426) \end{bmatrix} = \begin{bmatrix} k_2P \\ r_{21}P \\ r_{22}P \end{bmatrix}$      | $M_1 = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 6 & 36 \end{bmatrix}$<br>$M_1^{-1}(d_{12}P, c_{12}P, c_{13}P)^T$<br>$= \begin{bmatrix} (99, 290) \\ (140, 153) \\ (220, 253) \end{bmatrix} = \begin{bmatrix} k_1P \\ r_{11}P \\ r_{12}P \end{bmatrix}$      | $M_1 = \begin{bmatrix} 1 & 5 & 25 \\ 1 & 3 & 9 \\ 1 & 6 & 36 \end{bmatrix}$<br>$M_1^{-1}(d_{13}P, c_{12}P, c_{13}P)^T$<br>$= \begin{bmatrix} (99, 290) \\ (140, 153) \\ (220, 253) \end{bmatrix} = \begin{bmatrix} k_1P \\ r_{11}P \\ r_{12}P \end{bmatrix}$ |
| 2     | $M_3 = \begin{bmatrix} 1 & 8 & 64 \\ 1 & 9 & 81 \\ 1 & 10 & 100 \end{bmatrix}$<br>$M_3^{-1}(d_{31}P, c_{31}P, c_{32}P)^T$<br>$= \begin{bmatrix} (410, 375) \\ (121, 391) \\ (163, 367) \end{bmatrix} = \begin{bmatrix} k_3P \\ r_{31}P \\ r_{32}P \end{bmatrix}$ | $M_3 = \begin{bmatrix} 1 & 4 & 16 \\ 1 & 9 & 81 \\ 1 & 10 & 100 \end{bmatrix}$<br>$M_3^{-1}(d_{32}P, c_{31}P, c_{32}P)^T$<br>$= \begin{bmatrix} (410, 375) \\ (121, 391) \\ (163, 367) \end{bmatrix} = \begin{bmatrix} k_3P \\ r_{31}P \\ r_{32}P \end{bmatrix}$ | $M_2 = \begin{bmatrix} 1 & 7 & 49 \\ 1 & 4 & 16 \\ 1 & 2 & 4 \end{bmatrix}$<br>$M_2^{-1}(d_{23}P, c_{21}P, c_{23}P)^T$<br>$= \begin{bmatrix} (116, 97) \\ (203, 437) \\ (285, 426) \end{bmatrix} = \begin{bmatrix} k_2P \\ r_{21}P \\ r_{22}P \end{bmatrix}$ |
| 3     | Group key<br>$S = \sum_{j=1}^3 k_jP = (435, 7)$                                                                                                                                                                                                                  | Group key<br>$S = \sum_{j=1}^3 k_jP = (435, 7)$                                                                                                                                                                                                                  | Group key<br>$S = \sum_{j=1}^3 k_jP = (435, 7)$                                                                                                                                                                                                              |

Figure 4

## Key Round 2 for Reconstruction

| Steps | User $U_1$                                                                                                                                                                                          | User $U_2$                                                                                                                                                                                          | User $U_3$                                                                                                                                                                                            |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1     | $U_1$ chooses $4 \times 3$ matrix<br>$A_1 = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 5 & 25 \\ 1 & 3 & 9 \\ 1 & 6 & 36 \end{bmatrix}$<br>$A_1$ is public                                                    | $U_2$ chooses $4 \times 3$ matrix<br>$A_2 = \begin{bmatrix} 1 & 3 & 9 \\ 1 & 7 & 49 \\ 1 & 4 & 16 \\ 1 & 2 & 4 \end{bmatrix}$<br>$A_2$ is public                                                    | $U_3$ chooses $4 \times 3$ matrix<br>$A_3 = \begin{bmatrix} 1 & 8 & 64 \\ 1 & 4 & 16 \\ 1 & 9 & 81 \\ 1 & 10 & 100 \end{bmatrix}$<br>$A_3$ is public                                                  |
| 2     | $U_1$ chooses randomly<br>$k_1 = 14, r_{11} = 32, r_{12} = 27$                                                                                                                                      | $U_2$ chooses randomly<br>$k_2 = 19, r_{21} = 21, r_{22} = 35$                                                                                                                                      | $U_3$ chooses randomly<br>$k_3 = 48, r_{31} = 61, r_{32} = 12$                                                                                                                                        |
| 3     | $U_1$ computes<br>$\begin{bmatrix} d_{12} \\ d_{13} \\ e_{12} \\ e_{13} \end{bmatrix} = A_1 \begin{bmatrix} 14 \\ 32 \\ 27 \end{bmatrix} = \begin{bmatrix} 186 \\ 849 \\ 353 \\ 1178 \end{bmatrix}$ | $U_2$ computes<br>$\begin{bmatrix} d_{21} \\ d_{23} \\ e_{21} \\ e_{23} \end{bmatrix} = A_2 \begin{bmatrix} 19 \\ 21 \\ 35 \end{bmatrix} = \begin{bmatrix} 397 \\ 1881 \\ 663 \\ 201 \end{bmatrix}$ | $U_3$ computes<br>$\begin{bmatrix} d_{31} \\ d_{32} \\ e_{31} \\ e_{32} \end{bmatrix} = A_3 \begin{bmatrix} 48 \\ 61 \\ 12 \end{bmatrix} = \begin{bmatrix} 1304 \\ 484 \\ 1569 \\ 1858 \end{bmatrix}$ |
| 4     | $U_1$ computes Key<br>$k_1 P = (99, 290)$                                                                                                                                                           | $U_2$ computes Key<br>$k_2 P = (116, 97)$                                                                                                                                                           | $U_3$ computes Key<br>$k_3 P = (410, 375)$                                                                                                                                                            |

**Figure 5**  
**Secret Key distribution for Round 1**

### Numerical Analysis 1:

**Initialization:** This stage involves putting up the starting options and parameters necessary for your numerical issue.

**Example:**  $0 \times 0$  is used to set the initial value for an iterative procedure.

**Evaluation of the Function:** Consider the mathematical function or equation you plan to solve or analyse.

**Example:**  $2 - 4 = 0$   $f(x) = x^2 - 4 = 0$

For iterative techniques, iteration The values should be changed over time to gradually approach the answer.

Calculate the error or the discrepancy between the current approximation and the correct answer.

**Example:** Convergence Criteria (for iterative techniques):  $E = x_{n+1} - x_n$  Determine if the error is within a tolerable range.

**Example:** The iteration should be stopped if  $E$ , where stands for a low tolerance value.

**Final conclusion:** The conclusion that is reached at the end of an iterative process is an approximation or solution.

**Example:** final  $= x$  final  $= n+1$

### B. Key Reconstruction:

Each user  $U_i, i = 2, 3, \dots, n$ , can reconstruct the key  $k_1 P$  by computing  $A_1$

$-1 (d1P, c1P, c2P, \dots, c(n-1)P)T$ , this result is the same as  $(k1P, r1P, r2P, \dots, r(n-1)P)T$ .

Figure 3 describes ECSSS as follows: Suppose there are three users in the scheme. User U1 chooses a key  $k1P$ , and computes shares: one private share  $d1P$  and public shares  $c12P$  and  $c13P$ . User U1 sends private share to U2, U3 through a secure channel. Finally, U2, U3, reconstruct the key  $k1P$  from  $d1P$  and  $c12P$  and  $c13P$ .

Then, we offer a verified distributed group key agreement technique based on the Elliptic Curve Secret Sharing Scheme (ECSSS) that distributes shares via open channels and validates the group key.

The goal of the strategy is to use a public channel to create a key group  $S$  with  $n$  users,  $U1$ ,  $U2$ , and  $Un$ . These three directions are the three axes of this technique: authentication, key restoration, and enclosed dissemination.

First Round: Secret Key Distribution:

First Round: Secret Key Distribution Algorithm

### Step 1: Initialization

- Initialize parameters and security settings.
- Select an elliptic curve ( $E$ ) and a base point ( $P$ ) on the curve.
- Choose a prime modulus ( $p$ ) and an order ( $n$ ) for the curve group.
- Set the number of users ( $n$ ).

### Step 2: Private Key Generation

- For each user  $i$  ( $i = 1$  to  $n$ ):
- Generate a random private key  $k_i$  in the range  $[1, n-1]$ .
- Compute the corresponding public key  $k_iP$ , where  $P$  is the base point on the curve.

### Step 3: Public Share Calculation

- For each user  $i$  ( $i = 1$  to  $n$ ):
- Compute  $(n-1)$  public shares for the private key  $k_i$ .
- Each public share is computed using secret sharing methods like ECSSS.

**Step 4: Share Distribution**

- Securely transmit the private share to (n-1) other users using an authenticated and confidential channel.

**Step 5: Private Share Verification**

- Each user verifies the authenticity of the received private shares to ensure they have not been tampered with.

**Step 6: Key Reconstruction**

- Each user reconstructs their private key  $k_i$  using their private share and the received public shares.

**Step 7: Shared Secret Key Calculation**

- Once all users have reconstructed their private keys  $k_i$ , they can compute the shared secret key  $S$  as the sum of their individual private keys:  $S = k_1 + k_2 + \dots + k_n$ .

The Secure users are in the scheme. In round 1, user  $U_1$  chooses a key  $k_{1P}$  and computes encrypted shares,  $b_{12P}$  for  $U_2$ ,  $b_{13P}$  for  $U_3$  and also computes public shares  $c_{12P}$ ,  $c_{13P}$  and sends them to both users,  $U_2$ , and  $U_3$ . Similarly, users,  $U_2$  and  $U_3$  choose keys  $k_{2P}$  and  $k_{3P}$  respectively. User  $U_2$  computes encrypted shares  $b_{21P}$  for  $U_1$  and  $b_{23P}$  for  $U_3$  and  $U_3$  computes encrypted shares,  $b_{31P}$  for  $U_1$  and  $b_{32P}$  for  $U_2$ . User  $U_2$  computes public shares  $c_{21P}$ ,  $c_{23P}$  sends them to both users  $U_1$  and  $U_3$ . User  $U_3$  computes public shares  $c_{31P}$ ,  $c_{32P}$  sends them to both users  $U_1$  and  $U_2$ . In round 2: User  $U_1$  computes  $k_{2P}$  using  $b_{21P}$  and public shares  $c_{21P}$ ,  $c_{23P}$ . And also computes  $k_{3P}$  using  $b_{31P}$  and public shares  $c_{31P}$ ,  $c_{32P}$ . Similarly  $U_2$  computes  $k_{1P}$  and  $k_{3P}$ , and  $U_3$  computes  $k_{1P}$  and  $k_{2P}$ . Finally, users,  $U_1$ ,  $U_2$ , and  $U_3$  can compute the group key as  $S = k_{1P} + k_{2P} + k_{3P}$ .

**V. Conclusion**

Elliptic Curve Secret Sharing for Secure Group Key Agreement Protocol (ECSSS) for Authentication in Distributed Environments offers a novel solution to the serious security issues posed by group communication

in distributed environments. This system establishes reliable and effective group key agreements by utilising the strengths of elliptic curve cryptography, secret sharing, and authentication. This protocol's capacity to do away with the necessity of a reliable Key Generation Centre (KGC) in the group key establishment process is one of its major accomplishments. By eliminating the single point of failure, this not only improves security but also makes it well suited for contexts with limited resources, such the Internet of Things (IoT) and mobile devices. It is a significant contribution to the field of secure group communication because of the creative ways it uses elliptic curve cryptography, secret sharing, and authentication to build a solid basis for protecting communication in resource-constrained and decentralised ecosystems.

## References

- [1] J.-M. Bohli, "A framework for robust group key agreement," in Proc. Int. Conf. Comput. Sci. Appl. (ICCSA), in Lecture Notes in Computer Science, vol. 3982, pp. 355–364 (2006).
- [2] E. Bresson, O. Chevassut, and D. Pointcheval, "Provably-secure authenticated group Diffie–Hellman key exchange," ACM Trans. Inf. Syst. Secur., vol. 10, no. 3, pp. 255–264 (2007).
- [3] J. Katz and M. Yung, "Scalable protocols for authenticated group key exchange," J. Cryptol., vol. 20, no. 1, pp. 85–113 (Jan. 2007).
- [4] T. Brecher, E. Bresson, and M. Manulis, "Fully robust tree-Diffie–Hellman group key exchange," in Proc. 8th Int. Conf. Cryptol. Netw. Secur. (CANS), in Lecture Notes in Computer Science, vol. 5888, pp. 478–497 (2009).
- [5] S. Jarecki, J. Kim, and G. Tsudik, "Flexible robust group key agreement," IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5, pp. 879–886 (May 2011).
- [6] A. Joux, "A one round protocol for tripartite Diffie–Hellman," in Proc. ANTS, in Lecture Notes in Computer Science, vol. 1838, pp. 385–394 (2000).
- [7] Anil Kumar & Sandeep Kumar Sharma. Information cryptography using cellular automata and digital image processing, *Journal of Discrete Mathematical Sciences and Cryptography*, 25:4, 1105-1111 (2022), DOI: 10.1080/09720529.2022.2072437.

- [8] Sapkal, Vinod, Gupta, Praveen & Khan, Aboo Bakar. Deep learning based phishing website identification system using CNN-LSTM classifier, *Journal of Information and Optimization Sciences*, 44:3, 315–330 (2023), DOI: 10.47974/JIOS-1343
- [9] S. Ajani and M. Wanjari, “An Efficient Approach for Clustering Uncertain Data Mining Based on Hash Indexing and Voronoi Clustering,” 2013 5th International Conference and Computational Intelligence and Communication Networks, pp. 486–490 (2013), doi: 10.1109/CICN.2013.106.
- [10] K.-A. Shim, “A round-optimal three-party ID-based authenticated key agreement protocol,” *Inf. Sci.*, vol. 186, no. 1, pp. 239–248 (Mar. 2012).
- [11] Khetani, V. ., Gandhi, Y. ., Bhattacharya, S., Ajani, S. N. ., & Limkar, S. Cross-Domain Analysis of ML and DL: Evaluating their Impact in Diverse Domains. *International Journal of Intelligent Systems and Applications in Engineering*, 11(7s), 253–262 (2023). Retrieved from <https://ijisae.org/index.php/IJISAE/article/view/2951>.
- [12] X. Gu, Y. Zhao, and J. Yang, “Reducing rekeying time using an integrated group key agreement scheme,” *J. Commun. Netw.*, vol. 14, no. 4, pp. 418–428 (Aug. 2012).
- [13] D. Liu, D. Huang, P. Luo, and Y. Dai, “New schemes for sharing points on an elliptic curve,” *Comput. Math. Appl.*, vol. 56, no. 6, pp. 1556–1561 (Sep. 2008).
- [14] V. P. Binu and A. Sreekumar, “Threshold multi secret sharing using elliptic curve and pairing,” (Mar. 2016), arXiv:1603.09524.
- [15] Y. Liu, Q. Sun, Y. Wang, L. Zhu, and W. Ji, “Efficient group authentication in RFID using secret sharing scheme,” *Cluster Comput.*, vol. 22, no. S4, pp. 8605–8611 (Jul. 2019).