Private Virtual Tree Networks for Secure Multi-Tenant Environments Based on the VIRGO Overlay Network

Lican Huang [1]

Abstract

Hierarchical organization is a fundamental structure in real-world society, where authority and responsibility are delegated from managers to subordinates. The VIRGO network (Virtual Hierarchical Overlay Network for scalable grid computing) provides a scalable overlay for organizing distributed systems but lacks intrinsic security and privacy mechanisms. This paper proposes Private Virtual Tree Networks (PVTNs), a cryptographically enforced extension that leverages the VIRGO overlay to mirror real organizational hierarchies. In PVTNs, join requests are encrypted with the manager's public key to ensure confidentiality, while membership authorization is enforced through manager-signed delegation certificates. Public keys are treated as organizational secrets and are disclosed only within direct manager-member relationships, resulting in a private, non-enumerable virtual tree. Our work demonstrates, through the system model, protocols, security analysis, and design rationale, that PVTNs achieve scalability, dynamic management, and strong security guarantees without relying on global public key infrastructures.

Keywords— *Distributed Systems, Hierarchical Architecture, Access Control, Public Key Cryptography, Trust Delegation*

I. INTRODUCTION

DISTRIBUTED systems increasingly support collaboration across organizational boundaries while operating on shared public infrastructures. Most real-world organizations are inherently hierarchical, yet many existing distributed security models rely on flat or global trust assumptions that do not reflect these structures.

Modern distributed computing environments involve geographically dispersed participants, heterogeneous resources, and complex collaborative workflows. Technologies such as grid computing, cloud platforms, and peer-to-peer (P2P) systems have been widely adopted to support these environments. A central challenge in such systems is how to organize, manage, and secure large numbers of dynamic participants without relying on rigid centralized control.

The VIRGO network (Virtual Hierarchical Overlay Network for scalable grid computing) [5] addresses scalability and management by organizing nodes into a structured, dynamically evolving overlay hierarchy independent of physical network topology. VIRGO enables scalable management, dynamic reconfiguration, efficient service discovery, and delegation of responsibilities across virtual organizations. However, VIRGO assumes cooperative participants and does not explicitly provide cryptographic mechanisms for secure identity binding, private membership, delegation, or fine-grained authorization.

In open and federated distributed systems, the absence of strong cryptographic enforcement can lead to significant security and privacy concerns, particularly when multiple virtual organizations coexist on shared public infrastructures. To address these limitations, this paper introduces *Private Virtual Tree Networks (PVTNs)*, which leverage the VIRGO overlay to bind hierarchical structures to public-key–based trust delegation aligned with managerial roles.

The key idea behind PVTNs is to bind the hierarchical structure of the VIRGO overlay to cryptographic identities. Each node is identified by a public–private key pair, and each parent–child relationship is represented by a signed delegation certificate. As a result, virtual trees become private, verifiable trust structures that can safely operate over public networks without relying on global public key infrastructures.

The main contributions of this paper are as follows:

- The design of a public-key–based private virtual tree model built on the VIRGO overlay.

- Protocols for secure node joining, delegation, revocation.

- A decentralized authorization mechanism based on hierarchical trust chains.

- An analysis of security properties and applicability to distributed collaboration and computing environments.

II. RELATED WORKS

Distributed identity management, overlay networks, and secure authorization frameworks have been extensively studied. Foundational public key infrastructures and trust management systems include X.509 PKI [1], Kerberos [2], and PGP [3], which provide mechanisms for authentication and trust but often expose member identities to central authorities or require global verification. DNS security [4] further highlights the chal-

---
[1]Hangzhou Domain Zones Technology Co., Ltd., Hangzhou, China.
Corresponding Author: Lican Huang (e-mail:`lican.huang.hz@gmail.com`).

lenges of distributed trust. Grid computing and hierarchical overlay networks, including VIRGO [5]and The Grid [7], introduced scalable hierarchical approaches, while LDAP-based directory services [6] provide structured identity management.

A.   STRUCTURED OVERLAY NETWORKS

Structured overlay networks and distributed hash tables (DHTs) such as Pastry [19], Chord [23], and DHT performance frameworks [20] enable scalable decentralized routing. Secure routing in overlays [16] and self-certifying file systems [28] further illustrate mechanisms for secure distributed storage and identity verification. VIRGO [5] is a hierarchical overlay designed for scalable grid computing, highlighting how overlay networks can integrate structured authorization mechanisms.

B.   HIERARCHICAL   AUTHORIZATION SYSTEMS

Hierarchical authorization has been explored in PKI [1] and Kerberos [2]. While effective, these systems expose member identities to central authorities and require global verification. Role-based and delegation logic frameworks [11, 14] formalize hierarchical trust and delegation, while Blaze et al. [13] explore decentralized trust management. PVTNs extend hierarchical delegation into overlays, allowing nodes to **verify authorization without revealing private keys or full trust chains**.

C.   AUTHORIZATION AND TRUST MANAGEMENT

Access control in distributed and group environments has been explored extensively, including Role-Based Access Control (RBAC) models [24], key graph-based secure group communication [25], and surveys on key management for secure groups [26]. Logic-based approaches [27] provide formal foundations for hierarchical and distributed access control, complementing role-based and delegation logic frameworks. PVTNs leverage these models to enable privacy-preserving authorization in hierarchical overlays.

D.   DECENTRALIZED TRUST AND IDENTITY

Decentralized trust models such as PGP [3] and blockchain-based PKI [4, 10, 15] reduce reliance on central authorities. Certcoin [10], Blockstack [21], and Sovrin [22] demonstrate blockchain-based self-sovereign identity and global naming mechanisms. Privacy-preserving revocable identity management [9] and surveys of decentralized identity systems [8] highlight ongoing research in user-centric identity management. PVTNs improve upon these approaches by enabling **cryptographic proofs of authorization that preserve member privacy**, allowing hierarchical delegation without identity leakage.

E.   SECURITY IN HIERARCHICAL TREE NETWORKS

Hierarchical tree network security has been explored in distributed PKI trees, certificate revocation trees, and hierarchical access control overlays [11–13,16]. Key challenges include:

- **Delegation integrity**: preventing forged authorizations along tree paths.
- **Member privacy**: protecting node identities and positions in the hierarchy.
- **Proof verification**: allowing nodes to verify access without revealing full trust chains.

PVTNs enhance hierarchical structures with **private membership proofs, non-leakage of identities, and overlay-based verification**, achieving **secure, privacy-preserving hierarchical authorization**.

F.   DECENTRALIZED   IDENTITY   AND PRIVACY

Recent work in self-sovereign identity, verifiable credentials, and distributed authorization emphasizes **non-leakage of member identities** [9, 10, 15, 17, 18]. Mechanisms such as decentralized PKI, issuer-hiding DIDs, and zero-knowledge membership proofs allow participants to **prove authorization without revealing identifiers or trust paths**. PVTNs leverage these mechanisms for **local, unlinkable, non-revealing authorization proofs**.

G.   THREAT MODELS

Prior systems are vulnerable to adversaries attempting to:

- Infer **member identities** [3].
- Reconstruct **trust chains or overlay hierarchies** [13, 16].
- Forge or manipulate **authorization proofs** [11, 12].

PVTNs mitigate these threats via **hierarchical cryptographic delegation** and **zero-knowledge-style membership proofs** [9, 10], enabling **local verification without leaking sensitive information**.

H.   SUMMARY: DISTINCTIVE FEATURES OF PVTNS

PVTNs occupy a **distinct design point** by combining:

- **Hierarchical delegation** without central authorities [11, 12].
- **Private key visibility** and selective disclosure [10, 15].
- **Non-leakage of member identification**, protecting privacy under observation [9].

- **Overlay-based scalability** leveraging the VIRGO network [5, 16].

These features make PVTNs suitable for **secure, privacy-preserving authorization** in enterprise, consortium, and multi-tenant distributed systems, bridging the gaps left by PKI, Web-of-Trust, blockchain, and traditional overlay/grid infrastructures.

III. PRIVATE VIRTUAL TREE NETWORK ARCHITECTURE

A. ASSUMPTIONS

The following assumptions underpin the design and security analysis of the Private Virtual Tree Network (PVTN):

- **Cryptographic Identity and Key Privacy Assumption:** Each node independently generates a public/private key pair. There is no global certificate authority or publicly verifiable identity infrastructure. Trust is established exclusively through hierarchical delegation. Public keys are treated as organizational secrets rather than globally discoverable identifiers. Key visibility is strictly limited to direct manager–member relationships: a manager may disclose its public key only to its direct members, and a member may know only the public key of its direct manager. No external node can verify membership, traverse the virtual tree, or participate in the PVTN without explicit key disclosure by an authorized manager. This assumption ensures that the virtual tree remains private and non-enumerable.

- **Cryptographic Security Assumption:** Standard cryptographic primitives, including digital signature schemes, hash functions, and public-key encryption, are assumed to be secure against polynomial-time adversaries.

- **Overlay Reliability Assumption:** The underlying VIRGO overlay network reliably maintains its structured hierarchical topology and provides correct message routing and delivery. Participating nodes are assumed to follow protocol specifications at the overlay level.

B. SYSTEM MODEL AND NOTATION

The Private Virtual Tree Network (PVTN) leverages the VIRGO overlay network to organize nodes into a cryptographically protected hierarchical structure. Each node corresponds to a participant in the organization and is identified by a public/private key pair. Parent–child relationships are represented by signed delegation certificates, enabling verifiable trust chains without exposing the full organizational structure unnecessarily. The VIRGO overlay handles routing, message propagation, and structural maintenance, ensur-ing efficient and scalable network dynamics. The system consists of multiple *Private Virtual Tree Networks* (PVTNs) operating over a shared *VIRGO* structured overlay network. Each PVTN represents an independent organizational tenant and defines a hierarchical trust structure in the form of a rooted tree.

Nodes and Keys Each node x independently generates and owns a public/private key pair (PK_x, SK_x). Private keys are never shared. Public keys serve as cryptographic identifiers and are scoped to direct manager-member only .

Node Types One PVTN has only one root, multi-layer managers, and leaf nodes. New joined node must be leaf node. Leaf node can be promoted as a manager only approved by its direct manager. That means the direct manager knows the types for all its members.

Managers and Delegation Each PVTN contains a distinguished root node R_i that acts as the trust anchor of tenant T_i. Authority is delegated along tree edges using signed delegation certificates. A delegation certificate issued by a parent node p to a child node c has the form:

$$Cert_{p \to c} = \text{Sign}_{SK_p}(PK_c, Role, Scope, Validity)$$

Only nodes holding valid delegation certificates are considered authorized managers or members.

VIRGO Overlay The VIRGO overlay provides scalable routing, message forwarding, and hierarchical dissemination. VIRGO does not act as a certificate authority and does not participate in cryptographic validation. All trust decisions are enforced exclusively within PVTNs.

Cryptographic Primitives The following cryptographic primitives are assumed:

- $Enc_{PK_x}(M)$: Public-key encryption of message M under public key PK_x.

- $Sign_{SK_x}(M)$: Digital signature on message M using private key SK_x.

- $H(\cdot)$: A collision-resistant cryptographic hash function.

Adversarial Model We assume a Dolev–Yao adversary with full control over the communication network. The adversary may intercept, replay, modify, or inject messages, but cannot break cryptographic primitives or forge signatures without the corresponding private keys. Compromise of a node reveals only its own private key and does not automatically compromise ancestor or sibling nodes.

Notation Summary

Symbol	Meaning
$PVTN_i$	Private Virtual Tree Network of tenant T_i
R_i	Root (trust anchor) of $PVTN_i$
(PK_x, SK_x)	Public/private key pair of node x
$Cert_{p \to c}$	Delegation certificate from p to c
$Enc_{PK_x}(\cdot)$	Encryption under public key PK_x
$Sign_{SK_x}(\cdot)$	Signature using private key SK_x
$H(\cdot)$	Cryptographic hash function

C. GLOBAL MESSAGE RULE

All protocol messages exchanged within or across Private Virtual Tree Networks (PVTNs) over the VIRGO overlay MUST comply with the following global communication rule.

Message Confidentiality Rule Every message M transmitted from a sender node x to an intended recipient node y is encrypted using the recipient's public key:

$$Msg_{x \to y} = Enc_{PK_y}(M)$$

This guarantees that only the holder of SK_y can decrypt and process the message, regardless of the routing path through VIRGO.

Message Authenticity Rule If a message carries authorization, delegation, revocation, or control information, the sender MUST sign the message payload before encryption:

$$Msg_{x \to y} = Enc_{PK_y}\big(Sign_{SK_x}(M)\big)$$

This ensures origin authenticity and integrity, while preventing intermediate nodes from observing signed content.

Join Request Rule Join requests initiated by a child node c toward a manager m follow the format:

$$JoinReq_c = Enc_{PK_m}(PK_c \parallel r \parallel JoinInfo)$$

where r is a freshly generated nonce. Only the intended manager can decrypt and evaluate the request.

Manager Response Rule All responses issued by managers, including delegation certificates, or approval decisions, MUST be encrypted using the child's public key:

$$Resp_{m \to c} = Enc_{PK_c}(Cert_c \parallel Meta)$$

This prevents disclosure of authorization artifacts to other nodes, including members of different tenants.

Upward Propagation Rule Any message propagated upward along the PVTN hierarchy (e.g., conflict checks, join-request hashes) MUST be encrypted under the public key of the immediate parent node. Messages are forwarded hop-by-hop until the tenant root is reached:

$$Msg_{x \to parent(x)} = Enc_{PK_{parent(x)}}(M)$$

Downward Broadcast Rule Messages broadcast downward from a manager or root (e.g., conflict results) are encrypted individually for each direct child node. Group or plaintext broadcast is prohibited:

$$Msg_{p \to c_i} = Enc_{PK_{c_i}}(M)$$

Cross-Tenant Restriction Nodes belonging to tenant T_i do not possess the public keys of managers in tenant T_j $(i \neq j)$. Consequently, they cannot decrypt, verify, or forge messages, certificates, or responses from other tenants.

Security Consequence Under this global rule, VIRGO functions purely as a routing substrate. Confidentiality, authenticity, freshness, and tenant isolation are enforced entirely by cryptographic means, independent of overlay behavior.

D. JOIN AND CONFLICT DETECTION PROTOCOL

Let a child node c request to join $PVTN_i$ under manager m.

Phase I — Join Request

1. The child generates (PK_c, SK_c) and a fresh nonce r.

2. The child sends a join request via VIRGO:

$$Enc_{PK_m}(JoinInfo \parallel PK_c \parallel r)$$

3. The requesting Manager decrypts the request using SK_m and verifies freshness of r.

Phase II — Tenant-Wide Conflict Detection
Let $h = H(PK_c)$.
Step 1: Upward Propagation

Upward Hash Propagation. The requesting manager encrypts and forwards h upward along the PVTN tree, one level at a time, until reaching the tenant root R_i:

$$Enc_{PK_{parent}}(h)$$

Each intermediate manager receives the message, and then decrypts it with its own private key, and encrypts it with its parent's public key, and forwards h only to its parent.
Step 2: Downward Broadcast

Downward Broadcast (Conflict Query). Start from root, the manager for its each member public key ℓ computes

$$\mathsf{Check}_\ell = \begin{cases} \mathsf{YES}, & \text{if } H(PK_\ell) = h \end{cases}$$

If any YES, stop, otherwise broadcasts h downward to all its manager members using hop-by-hop encrypted messages:

$$Enc_{PK_{child}}(h)$$

Step 3: Local Conflict Check (all members are leaves

The manager knows the public keys of all its members, for each member public key ℓ computes

$$\text{Check}_\ell = \begin{cases} \text{YES}, & \text{if } H(PK_\ell) = h \end{cases}$$

If any YES, then YES, otherwise NO

Step 4: Aggregation (Bottom-Up)

Hierarchical Aggregation Rule. Each manager waits for responses from all manager children and computes:

$$Resp_m = \begin{cases} \text{YES} & \text{if any child reports YES} \\ \text{NO} & \text{otherwise} \end{cases}$$

The result is forwarded upward encrypted under the parent's public key.

Step 5: Root Decision

At the root R_i:

$$\text{Decision} = \begin{cases} \text{YES} & \text{if any subtree reports conflict} \\ \text{NO} & \text{otherwise} \end{cases}$$

Phase III — Downward Decision Broadcast

Decision Dissemination. After collecting all upward responses, the root derives the final decision

$$D \in \{\text{APPROVE}, \text{REJECT}\}$$

for the join request identified by the hash

$$h = H(PK_c).$$

The root constructs a signed decision record

$$\text{Dec}_h^{(R_i)} = \text{Sign}_{SK_{R_i}} \left(h \parallel D \parallel t \parallel \text{Reason} \right),$$

where t denotes a timestamp and Reason optionally encodes conflict or policy-related information.

The decision record is disseminated downward along the PVTN tree. For each child node, a parent encrypts the message using the child's public key:

$$\text{Enc}_{PK_{\text{child}}} \left(\text{Dec}_h^{(parent)} \right).$$

Upon receipt, each intermediate node executes the following steps:

1. Decrypts the received message using its private key.

2. Verifies the signature of its direct parent .

3. Records the verified decision locally for auditing and replay protection.

4. Re-signs the decision using its own private key, producing

$$\text{Dec}_h^{(self)} = \text{Sign}_{SK_{\text{self}}} \left(h \parallel D \parallel t \parallel \text{Reason} \right).$$

5. Encrypts and forwards the newly signed decision to each of its children.

Phase IV — Join Authorization

Join Authorization. After completion of the conflict-detection and decision-dissemination phases, the manager responsible for the join request identified by $h = H(PK_c)$ receives the final decision record $\text{Dec}_h^{(m)}$, signed by its direct parent and transitively endorsed by the tenant root.

Upon receipt, the manager performs the following steps:

1. Decrypts the received decision message using its private key SK_m.

2. Verifies the signature of its parent and checks that the decision record corresponds to the hash h and a fresh timestamp t.

3. Confirms that the decision is APPROVE and that no conflict or policy violation has been reported.

Authorization Outcome.

- **Approval.** If the final decision is APPROVE, the manager issues a delegation certificate

$$Cert_c = \text{Sign}_{SK_m} \left(PK_c \parallel \text{Role} \parallel \text{Validity} \parallel r \right),$$

  binding the child's public key to the manager's authority. The certificate is transmitted to the joining node encrypted under the child's public key:

$$\text{Enc}_{PK_c}(Cert_c).$$

- **Rejection.** If the final decision is REJECT, the manager discards the join request and optionally returns a rejection notice

$$\text{Enc}_{PK_c} \left(\text{Reject} \parallel \text{Reason} \right),$$

  without issuing any delegation certificate.

This protocol enforces the tree invariant of PVTNs by ensuring that no public key may appear more than once within the same tenant. The upward propagation and downward broadcast of the request hash prevent cycles, duplicate membership, and cross-branch reattachment. Because only a hash of the public key is disseminated, no sensitive identity or topological information is leaked. Approval is granted only if global uniqueness within the tenant is confirmed.

E.   HIERARCHICAL   LEAF   UPGRADE PROTOCOL

E.1.   ACTORS

- Leaf L

- Direct parent / manager P_0

- Immediate upper layer P_1 (knows PK_{P_0})

- Higher layers $P_2, \ldots, R$

- Root R

E.2.   NOTATION

- $H(ID_L)$: hashed leaf identifier
- SK_X, PK_X: private/public key of node X
- $\text{Sign}_{SK}(M)$: digital signature of message M
- $\text{Verify}_{PK}(M, \sigma)$: signature verification
- T: timestamp
- Nonce: random nonce to prevent replay attacks
- PolicyFlags: policy decisions (approve/deny) from each layer

E.3.   PROTOCOL STEPS

Step 0: Leaf Request (Optional)

$$\text{LeafHash} = H(ID_L)$$
$$\text{Req}_L = \{\text{LeafHash}, \text{DesiredRole}, T, \text{Nonce}\}_{SK_L}$$

Note: The leaf cannot self-upgrade; this is optional informational request.

Step 1: Direct Parent Signs Upgrade Request

$$\text{Req}_{P_0} = \{\text{LeafHash}, \text{DesiredRole}, T, \text{Nonce}\}_{SK_{P_0}}$$

Only P_0 can sign; confers authority to upgrade the leaf.

Step 2: Immediate Upper Layer Verification

1. Verify signature using PK_{P_0}:

$$\text{if Verify}_{PK_{P_0}}(\text{Req}_{P_0}) = \text{False} \Rightarrow \text{Deny}$$

2. Check that sender P_0 is a manager (not a leaf):

$$\text{if Role}(P_0) \neq \text{Manager} \Rightarrow \text{Deny}$$

3. Apply policy checks (tree depth, size, quota):

$$\text{if policy violation} \Rightarrow \text{Deny}$$

4. If denied, propagate Deny downward to P_0 and L; do not forward upstream.

Step 3: Forward Request Upstream

if Deny = False:

$$\text{forward } \{\text{Req}_{P_0}, \text{PolicyFlags}\}$$
$$\text{to higher layers } P_2, \ldots, R$$

Step 4: Root Approval

$$\text{Decision}_R = \{\text{LeafHash},$$
$$\text{DesiredRole},$$
$$T,$$
$$\text{Nonce},$$
$$\text{Approved/Deny}\}_{SK_R}$$

Root applies global policy checks and decides approval or denial.

Step 4b: Downward Propagation of Decision

- If request was denied by any intermediate layer, the downward decision begins from that layer.
- Otherwise, the root propagates its decision downward through the hierarchy to P_0 and L.

Step 5: Direct Parent Issues Formal Upgrade Certificate

$$\text{if Decision} = \text{Approved:} \quad \text{Cert}_L = \{\text{LeafHash},$$
$$\text{NewRole},$$
$$T,$$
$$\text{Nonce}\}_{SK_{P_0}}$$

Leaf L receives Cert$_L$ and upgrades role. If denied, no certificate is issued.

Step 6: Audit Trail(optional)

- Log all requests, signature verification results, policy flags, root or intermediate approvals/denials, and certificate issuance.

E.4.   PROTOCOL FLOW DIAGRAM

Figure 1 illustrates the hierarchical leaf upgrade protocol. The leaf node L sends a certificate request to its direct parent P_0 (manager). The request is then propagated upward through the hierarchy to P_1 and intermediate nodes $P_2 \ldots P_n$, ultimately reaching the root. Each intermediate node verifies the public key and role of the requester, enforces local policy constraints, and forwards approvals upward (if denials stop and backwards). Once the root approves the request, the decision is propagated back down through the hierarchy to the parent, which issues the certificate Cert$_L$ to the leaf. This flow ensures that leaves cannot self-promote and that all upgrades are authorized and validated at multiple layers.

E.5.   KEY PRINCIPLES

1. Leaf cannot self-upgrade.
2. Direct parent (P_0) is the sole issuer of the upgrade certificate.
3. Immediate upper layer verifies signature and manager role.
4. Intermediate layers and root can veto via policy.
5. Decision propagates downward starting from deny layer or root.
6. Audit trail maintained at all layers.
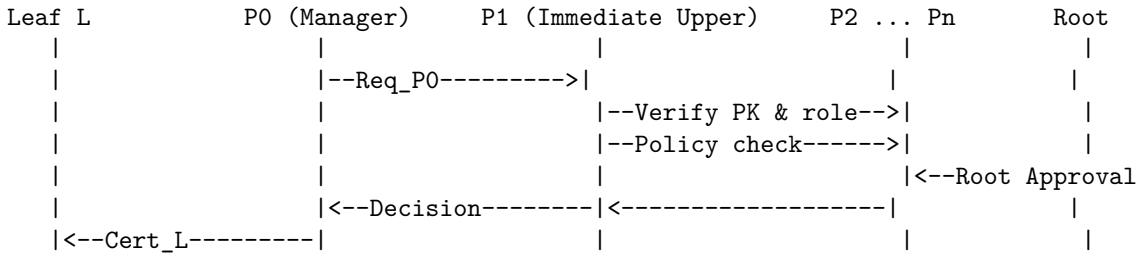7. Leaf ID is protected via hash (optionally salted).

```
Leaf L              P0 (Manager)      P1 (Immediate Upper)      P2 ... Pn        Root
  |                    |                    |                       |               |
  |                    |--Req_P0--------->|                       |               |
  |                    |                    |--Verify PK & role-->|               |
  |                    |                    |--Policy check------>|               |
  |                    |                    |                       |<--Root Approval
  |                    |<--Decision-------|<------------------|               |
  |<--Cert_L---------|                    |                       |               |
```

Figure 1: Hierarchical leaf upgrade flow diagram .

The hierarchical upgrade protocol inherently prevents protocol violations by restricting upgrade authority to the direct parent of a leaf, while all upper layers act only as policy arbiters rather than issuers. A leaf cannot self-upgrade because any valid upgrade request must carry a signature produced by its direct parent, whose role (manager vs. leaf) is already known and verifiable by its immediate upper layer. Although higher layers and the root do not possess the public key of the issuing parent, they can still determine that the request originates from a legitimate manager—rather than a leaf—because each layer verifies the identity and role of its direct child before forwarding the request upstream. To further strengthen this guarantee and prevent forged or misattributed delegation, a zero-knowledge proof (ZKP) can be attached to the parent-signed request, allowing the parent to prove possession of a valid manager credential and parent–child relationship with the leaf without revealing its private key or full identity. This ZKP enables upper layers to validate authorization correctness even in the absence of direct key knowledge, while preserving privacy and maintaining strict hierarchical control. Consequently, any node that disobeys the protocol—by attempting self-upgrade, unauthorized signing, or role forgery—is cryptographically excluded from successful promotion.

F. ACTION CERTIFICATE ISSUANCE AND POLICY ENFORCEMENT PROTOCOL

This protocol governs the authorization of actions for any node (leaf or manager) through hierarchical delegation, combining parent-issued certificates, upward policy evaluation, and downward enforcement decisions.

1. **Certificate Request Initiation (Node → Parent):**
   A node N submits a certificate request containing $H(ID_N)$, the requested role or action scope, and freshness values (T, Nonce) to its direct parent P_0. The node **cannot contact upper layers directly** and **cannot self-issue certificates**, because certificate validity depends entirely on the parent signature.

2. **Local Validation by Direct Parent:**
   The parent P_0 verifies that N is its immediate child and that the requested scope is within its delegated

authority. If the request violates local policy, it is rejected locally and not propagated.

3. **Certificate Proposal Construction (Parent):**
   If local validation succeeds, P_0 constructs a certificate proposal

   $$\mathcal{C}_0 = \{H(ID_N), \text{Scope}, T, \text{Nonce}\}$$

   and signs it with SK_{P_0}. At this stage, the certificate is not yet finalized.

4. **Upper-Layer Endorsement (Upward Propagation):**
   The signed proposal $\mathcal{C}_0$ is forwarded to the immediate upper layer P_1. Node P_1 verifies that:

   (a) P_0 is its legitimate child,
   (b) P_0 holds a manager role (not a leaf),
   (c) P_0's delegation rights are active.

   If valid, P_1 issues a signed endorsement

   $$\mathcal{E}_1 = \{\text{Role}_{P_0}, T, \text{Nonce}\}_{SK_{P_1}}$$

   Higher layers repeat check child's legitimate and may enforce policy strategies, such as subtree limits, risk evaluation, or global constraints.

5. **Root or Upper-Layer Policy Decision:**
   Any upper layer, including the root, may issue a signed denial decision based on policy considerations, even if all certificates and endorsements are valid. If no denial occurs, an approval decision is produced at the highest required layer.

6. **Downward Decision Propagation:**
   The signed approval or denial decision is propagated downward. Intermediate layers record the decision but do not modify the certificate itself.

7. **Certificate Finalization or Abortion (Parent):**
   Upon receiving the decision:

   - **If denied**, the parent P_0 aborts issuance; no certificate is delivered to N.
   - **If approved**, P_0 finalizes the certificate by assembling

   $$\text{Cert}_N = \left(\mathcal{C}_0, \{\mathcal{E}_1, \mathcal{E}_2, \dots\}\right)_{SK_{P_0}}$$

   and delivers it to node N.

Protocol Properties:

- **Universal coverage:** applies to any node, whether leaf or manager.

- **No self-authorization:** only direct parents issue certificates.

- **Issuer legitimacy:** upper-layer endorsements prove the issuing parent is authorized.

- **Policy sovereignty:** upper layers may refuse actions independently.

- **Hierarchical enforcement:** denials propagate downward; certificates propagate upward.

- **Misbehavior containment:** violations are detected and stopped at the earliest honest layer.

G. HIERARCHICAL DELEGATE-BASED CERTIFICATE VALIDATION FOR NODE ACTIONS

A node X receives an action request from N along with a parent-issued certificate. Since X cannot verify the parent directly, it forwards the request to the gateway(root can be gateway, but usaully not). The gateway then coordinates a hierarchical validation: downward propagation for local parent verification, followed by upward aggregation of approvals, and finally a gateway-signed confirmation back to X.

G.1. PROTOCOL STEPS

Step 0: Action request submission

$$N \to X : \text{Action}, \text{Cert}_N$$

The certificate Cert_N includes:

- The hash of the node identity $H(ID_N)$.

- Authorized role or action scope.

- Freshness values: nonce and timestamp to prevent replay attacks.

- Endorsements from upper layers $\{\mathcal{E}_1, \mathcal{E}_2, \dots\}$ proving issuer legitimacy.

Step 1: Forward to gateway

$$X \to \text{Gateway} : \text{Action}, \text{Cert}_N$$

Step 2: Downward hierarchical validation The gateway forwards the request downward through the hierarchy to the grandparents and parent nodes of N:

$$\text{Gateway} \to P_{\text{grandparent}} \to P_0 : \text{ValidateRequest}(\text{Cert}_N)$$

Grandparent P_1 checks the parent P_0:

- Verify that P_0 is a legitimate member of the tenant

- Check that P_0 has delegation rights to issue certificates

- Verify the integrity of P_0's certificate for N

- Ensure that cert is not self-issue certificates by P_0

- Optionally check subtree policy constraints (tree size, safety)

Parent P_0 checks the child N**:

- Verify that N's certificate request is valid (nonce, timestamp)

- Check that N is within the authorized role/scope

Step 3: Upward approval aggregation Each node checks:

- That its child is a legitimate manager/leaf

- Delegation rights are active

- Local policy compliance (tree size, safety, quotas, etc.)

- check the integrated $\mathcal{E}_{n-1}, \mathcal{E}_n, \mathcal{E}_{n+1}$ in Action certificate. Suppose this node is N. Node N knows the public keys of its parent and its children.

Each node returns a signed approval or denial to its immediate upper node:

$$P_0 \to P_{\text{grandparent}} \to \text{Root} : \text{Approval/Denial}$$

The root collects all upward decisions.

Step 4: Root downwards the decisions to gateway

Step 5: Gateway issues final approval If all checks pass, the gateway issues a signed certificate to X:

$$\text{Cert}_N^{\text{Gateway}} = \text{Sign}_{SK_{\text{Gateway}}}\Big(H(ID_N),$$
$$\text{Role}, T, \text{Nonce}, \text{DelegationProof}\Big)$$

Step 5: X validates and executes action Upon receiving the root-signed certificate, X verifies:

- Gateway signature validity

- Freshness and integrity

- Compliance with authorized scope and role

- Optional local policy constraints

If all checks succeed, X executes the action; otherwise, it is rejected.

G.2. ADVANTAGES OF THIS HIERARCHI-CAL VALIDATION PROTOCOL

Prevents unauthorized self-issuance

Nodes cannot bypass the hierarchy or issue their own certificates.

Only the parent and higher-layer nodes can validate and approve actions.

Decouples trust from local knowledge

The receiving node X doesn't need to know the parent's public key.

Trust is established through hierarchical delegation proofs and final root signature.

Ensures hierarchical policy enforcement

Intermediate parents can enforce subtree-specific rules (e.g., quotas, tree depth, safety).

Policies can be applied both downward and upward, allowing dynamic control.

Gateway-signed certificates act as a single source of truth.

Any node can verify that an action is authorized according to the full tenant hierarchy.

Supports privacy and minimal disclosure

Node identities can remain hidden via hash commitments.

Intermediate approvals provide validity without exposing sensitive identities unnecessarily.

Resilience and auditability

All approvals/denials are logged at intermediate nodes and root.

Misbehavior (e.g., a compromised parent) can be detected through inconsistencies in approval chains.

Scalable and flexible

Can handle deep hierarchies with multiple intermediate layers.

Upward and downward flows allow both small and large trees to be validated efficiently.

Summary:

This protocol combines hierarchical delegation, policy checks, and gateway-signed certificates, making it secure, auditable, and privacy-preserving, while preventing leaves or lower-layer nodes from misbehaving or bypassing authorization.

H. PRIVATE TREE CONSTRUCTION

Each node in the PVTN independently generates its own public/private key pair. The root node represents the top-level organizational authority and initializes the cryptographic trust anchor for the network. Authority is delegated strictly downward through signed delegation certificates issued by managers, and private keys are never shared.

Public keys are treated as organizational secrets rather than globally visible identifiers. Key visibility is restricted to direct manager–member relationships. No external node can participate in the tree or verify membership without explicit key disclosure by an authorized manager. This design ensures that the PVTN remains private, non-enumerable, and resistant to external observation or infiltration.

The private virtual tree is constructed incrementally as follows:

1. The organizational root generates a public/private key pair and establishes the initial trust anchor of the PVTN.

2. Each manager securely distributes its public key to prospective members or child nodes using trusted organizational channels (e.g., authenticated internal systems or out-of-band communication).

3. A node requests membership by submitting a join request encrypted with the public key of its designated parent or manager. The request includes the node's public key and organizational identity attributes (e.g., role, department, or node type).

4. Upon receiving the join request, the parent or manager invokes the join and conflict detection protocol described above. If the final decision is AP-PROVE, the manager issues a signed delegation certificate and admits the requester as a *leaf node* in the PVTN. If the decision is REJECT, the join request is denied.

5. Leaf nodes possess valid membership certificates but do not issue further delegations unless explicitly upgraded to managerial roles by their parent nodes.

6. Authorized Leaf members may be upgraded to managers using upgraded protocol; they can subsequently act as managers for subordinate nodes by distributing their public keys secretly and issuing delegation certificates for their members.

H.1. MEMBERSHIP JOIN

New members can request to join a PVTN under four scenarios:

1. **Member does not know the manager's ID/IP:** The join request is propagated through the VIRGO hierarchical overlay using n-tuple nodes in upper layers, ensuring delivery while limiting network-wide broadcast.

2. **Member knows the manager's public IP:** The request is sent directly to the manager, minimizing latency.

3. **Member knows the manager's local/private IP:** The request is propagated through the VIRGO hierarchical overlay or gateway nodes to reach the manager securely.

4. **Member knows the manager's VIRGO network ID:** Using the structured overlay protocol, the member routes the request from the root along the hierarchical tree path corresponding to the manager's ID.

Managers verify requests and issue signed delegation certificates. Membership within a domain is largely

static; revocation occurs only when a member leaves the organization voluntarily or due to termination or policy violations.

H.2.   DELEGATION MODELS

PVTNs support two delegation approaches:

- **Hierarchical delegation:** Members only know their immediate parent/manager. Keys and certificates above the parent remain hidden, preserving secrecy of the organizational structure and supporting high-security scenarios such as military networks.

- **Full-path certificates:** Delegation certificates contain the full path from root to the member, enabling independent verification without relying on the manager. This reduces bottlenecks, improves auditability, and supports distributed verification.

H.3.   MEMBERSHIP REVOCATION

Membership revocation occurs under the following circumstances:

- **Voluntary leave:** Members who resign or leave the organization have their certificates invalidated.

- **Termination or policy violation:** Managers revoke delegation certificates of members who no longer comply with organizational policies.

Revocation affects only the relevant subtree, minimizing disruption to unrelated members. Delegation certificates and key isolation ensure that revoked members cannot access PVTN resources or impersonate other participants.

H.4.   OVERLAY AND NETWORK DYNAMICS

The VIRGO overlay ensures that messages, delegation certificates, and revocation notices are efficiently propagated. Cryptographic verification prevents tampering by malicious or non-compliant nodes. Overlay-level redundancy and multiple verification paths enhance resilience, while behavioral monitoring and auditing by managers help maintain operational integrity.

I.   ORGANIZATIONAL MAPPING AND SOCIAL-DRIVEN HIERARCHY

In real-world society, most organizations naturally follow a hierarchical tree structure: an organization has a top-level authority, multiple levels of managers, and finally individual members at the leaves. Each manager is responsible for a subset of subordinates, and authority is delegated downward level by level.

Private Virtual Tree Networks (PVTNs) as Figure 2 shown are explicitly designed to reflect *social and organizational hierarchies* observed in real-world institutions. In most organizations, authority, responsibility, and trust are delegated top-down: from a root authority to managers, and from managers to individual members. PVTNs directly encode this social structure as a cryptographically verifiable tree, where each node represents an organizational entity and each edge represents a delegated trust relationship.

A PVTN consists of:

- a **root authority**, corresponding to the top-level organizational owner;

- **intermediate manager nodes**, representing departments, teams, or supervisors;

- **leaf members**, representing individual employees, services, or devices.

Each manager controls a subtree and is responsible for admitting, authorizing, and revoking its direct subordinates. Authority is not global or flat; instead, it follows real managerial responsibility boundaries, reducing administrative complexity and limiting the impact of compromise.

In contrast, the underlying VIRGO network is a *structural hierarchical peer-to-peer overlay*. VIRGO organizes nodes according to overlay identifiers and routing efficiency, independently of social roles, administrative domains, or trust semantics(although VIRGO can organize nodes according to social roles, here we use it scalability). Its hierarchy is designed for scalable lookup, routing, and fault isolation rather than organizational governance.

PVTNs are therefore *socially driven trust overlays* constructed on top of the VIRGO structural overlay. The VIRGO network provides efficient hierarchical routing and message propagation, while PVTNs impose organizational meaning, cryptographic authorization, and privacy constraints. Multiple independent PVTNs may coexist on the same global VIRGO overlay, enabling secure multi-tenant and consortium deployments without revealing organizational structure or membership relationships.

IV.   MULTI-PVTN COEXISTENCE AND ISOLATION

A.   MULTI-TENANT MODEL

A single VIRGO overlay may simultaneously support multiple Private Virtual Tree Networks. Each PVTN represents an independent organizational trust domain with its own root authority, delegation hierarchy, and key space. Although join requests and control messages traverse the same VIRGO substrate, cryptographic isolation ensures that authorization and membership remain strictly separated as Figure 3 shows.

The PVTN architecture operates over a shared VIRGO overlay network while supporting multiple independent organizational tenants. Each tenant T_i is represented by an isolated *Private Virtual Tree Network* $PVTN_i$, which defines its own trust domain, membership policy, and administrative hierarchy.
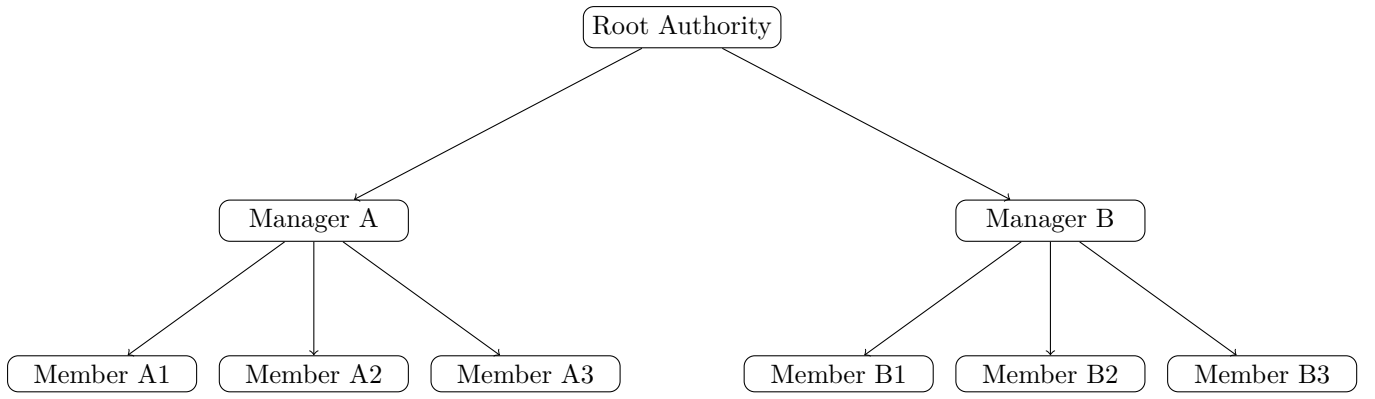
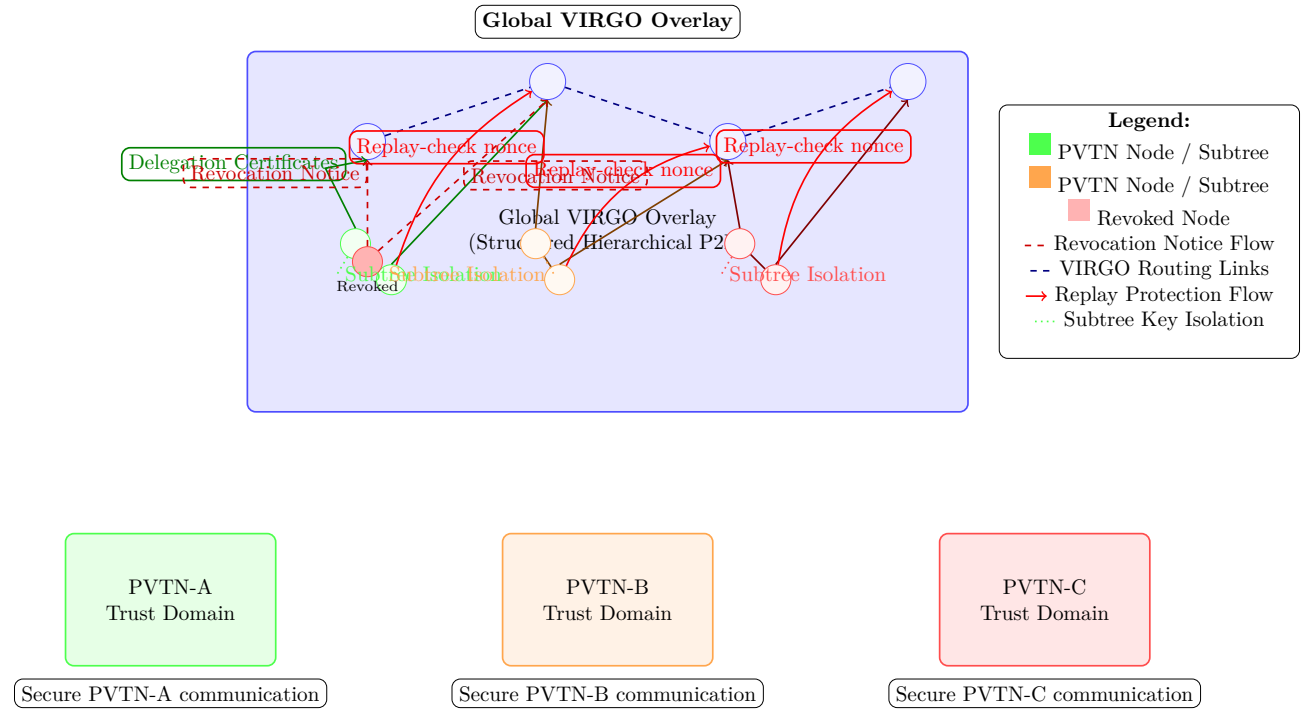Figure 2: Private Virtual Tree Network architecture with multiple members under each manager.

Figure 3: Multi-PVTN architecture over VIRGO overlay with color-coded legend. Revocation propagation, subtree key isolation, delegation certificates, and replay protection are illustrated. This figure clarifies trust, security, and multi-tenancy properties.

Let the root node of $PVTN_i$ be R_i, representing the highest authority of tenant T_i. The root generates a public/private key pair (PK_{R_i}, SK_{R_i}) that serves as the cryptographic trust anchor for the entire tenant. No global root authority or system-wide certificate authority exists.

- Each node $N_j \in PVTN_i$ independently generates its own key pair (PK_j, SK_j) and maintains exclusive control over its private key.

- Membership and authority relationships are expressed through delegation certificates $C_{parent \to child} = \text{Sign}SKparent(PK_{child}, scope, constraints)$.

- All delegation certificates are explicitly scoped to tenant T_i and are cryptographically invalid outside the issuing tenant.

- The VIRGO overlay provides scalable routing and hierarchical message dissemination but does *not* act as a trust authority and does not participate in cryptographic validation or authorization.

Each $PVTN_i$ therefore forms a cryptographically isolated trust tree that shares only the routing substrate with other tenants.

Tenant Separation Property For any two tenants T_i and T_k with $i \neq k$, there exists no implicit cryptographic linkage between $PVTN_i$ and $PVTN_k$. In the absence of explicit cross-tenant delegation, certificates, public keys, join requests, and membership information from one tenant cannot be verified, inferred, or enumerated by nodes in another tenant. Consequently, compromise, revocation, or structural changes within one tenant do not affect the security or integrity of other
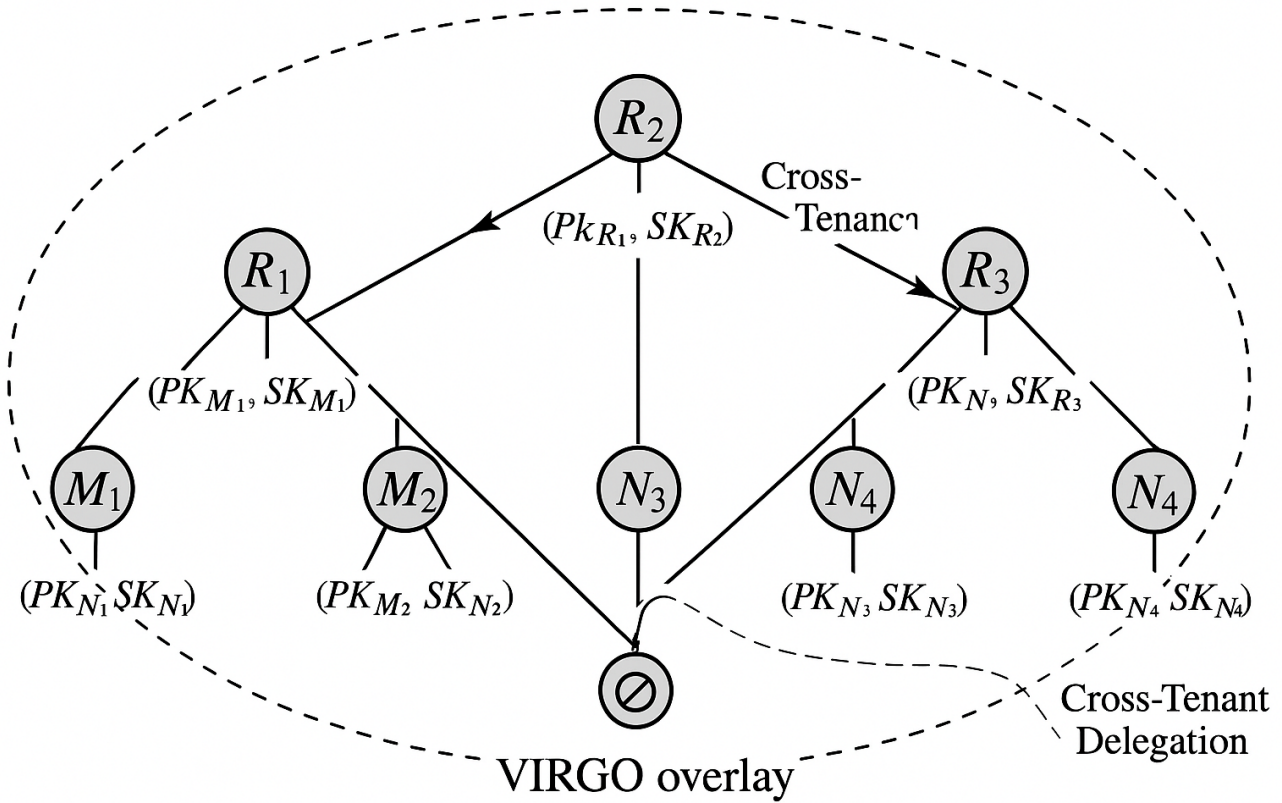
Figure 4: Cross-Tenant of PVTNs

tenants, even though all tenants share the same VIRGO overlay infrastructure.

Membership Revocation Across Tenants Although PVTNs are designed for relatively stable organizational membership, revocation is required to reflect real-world administrative events such as role changes, policy violations, or credential compromise.

Revocation Procedure

The responsible manager removes the revoked member's credentials and metadata from its local storage.

If the revocation is triggered by node misuse or compromise, the revocation notice is propagated to all members through the hierarchical tree.

B. CROSS-TENANT AUTHORIZATION

While tenants are cryptographically isolated by default, PVTNs support controlled cross-tenant collaboration through explicit authorization as Figure 4 shows. Such collaboration is intentional, auditable, and narrowly scoped.

- Cross-tenant authorization is initiated only by tenant managers, typically at or near the root level.

- Managers exchange delegation certificates out-of-band, establishing limited trust relationships between specific nodes or roles across tenants.

- Delegation certificates explicitly encode scope, duration, and permitted actions, preventing uncontrolled trust propagation.

Nodes from tenant T_k seeking access to resources in tenant T_i must present valid delegation certificates issued by an authorized manager in $PVTN_i$. Such certificates do not merge the two trees; instead, they form controlled trust bridges between otherwise independent hierarchies.

This approach enables inter-organizational cooperation, consortium operation, and federated workflows without sacrificing tenant privacy, membership confidentiality, or administrative autonomy.

While tenants are cryptographically isolated by default, PVTNs support controlled cross-tenant collaboration through explicit authorization.

- Cross-tenant authorization is initiated exclusively by tenant managers, typically at or near the root level.

- Managers exchange delegation certificates out-of-band, establishing narrowly scoped trust relationships between specific nodes or roles across tenants.

- Delegation certificates explicitly encode scope, duration, and permitted actions, preventing uncontrolled trust propagation.

Such certificates do not merge trust trees or introduce transitive trust. Instead, they create auditable, limited

trust bridges between otherwise independent PVTNs, preserving tenant autonomy and isolation.

C. ISOLATION LEMMAS

Lemma 1 (PVTN Isolation). Let $PVTN_i$ and $PVTN_j$ be two distinct Private Virtual Tree Networks operating over the same VIRGO overlay. A node authorized in $PVTN_i$ cannot obtain valid membership, delegation certificates, or topology information of $PVTN_j$ without possession of a delegation certificate issued by an authorized manager in $PVTN_j$.

Proof (Sketch). Join requests for $PVTN_j$ are encrypted using public keys known only to authorized managers within $PVTN_j$. Delegation certificates are signed exclusively with the corresponding managers' private keys. Under the Dolev–Yao adversary model, a node outside $PVTN_j$ cannot decrypt join requests, forge certificates, or verify delegation chains without explicit managerial authorization. Since the VIRGO overlay provides routing only and does not participate in trust or identity validation, access to the overlay does not reveal PVTN membership or topology. Therefore, cryptographic isolation between $PVTN_i$ and $PVTN_j$ is preserved. □

Lemma 2 (Cross-Tenant Join Isolation). Let $PVTN_i$ and $PVTN_j$ be two distinct tenants with $i \neq j$. A node belonging to $PVTN_j$ cannot obtain a valid delegation certificate from $PVTN_i$ without possession of a manager public key of $PVTN_i$ explicitly disclosed through authorized delegation.

Proof (Sketch).

Join requests are encrypted using the public key of an authorized manager within the target PVTN, preventing disclosure or modification by external nodes. Each request includes a freshly generated nonce r, which is cryptographically bound to the issued delegation certificate.

Upon approval, the manager generates a delegation certificate signed with its private key, ensuring authenticity, integrity, and non-repudiation. The manager's response—containing the signed delegation certificate and associated metadata—is encrypted using the requester's public key, ensuring that only the requesting node can access the authorization outcome.

Crucially, members of other tenants do not possess the manager's public key of the target PVTN. As manager public keys are treated as organizational secrets and disclosed only within direct manager–member relationships, nodes belonging to other tenants cannot correctly encrypt join requests, decrypt authorization responses, or verify delegation certificates. Consequently, cross-tenant nodes cannot obtain valid membership certificates or infer internal membership structure.

Under the Dolev–Yao adversary model, an adversary lacking the manager's private key cannot forge valid delegation certificates, and an adversary lacking either the manager's public key or the requester's private key cannot participate in or observe the join protocol. The binding of the nonce r to both the encrypted join request and the signed certificate prevents replay of stale join messages. Together, these mechanisms ensure confidentiality, authenticity, freshness, and strict tenant isolation.

Lemma 3 (Loop Freedom). The conflict detection protocol terminates and is loop-free.

Proof (Sketch). Messages propagate strictly upward during aggregation and strictly downward during dissemination, following the PVTN tree structure. No node sends messages laterally or upward after receiving the final decision. Since the tree is finite, the protocol terminates.

V. THREAT MODEL AND SECURITY ANALYSIS

A. FORMAL THREAT ASSUMPTIONS

We adopt the classical Dolev–Yao adversarial model, under which the adversary has complete control over the communication network. Specifically, the adversary can eavesdrop on, intercept, delay, replay, drop, inject, and arbitrarily modify messages transmitted over the VIRGO overlay network. The adversary may also coordinate multiple malicious nodes and attempt to infer organizational structure through traffic analysis.

Cryptographic primitives are assumed to be ideal: the adversary cannot decrypt ciphertexts without possession of the corresponding private key, cannot forge digital signatures without access to the signing key, and cannot derive private keys from public keys. Hash functions are assumed to be collision resistant.

The adversary may compromise a subset of nodes, thereby learning their private keys and all locally stored certificates. However, compromise of a node does not automatically compromise its parent, siblings, or ancestors unless explicit delegation exists. The adversary may generate arbitrarily many identities (Sybil attempts), but cannot obtain valid delegation certificates without managerial authorization.

We assume the underlying VIRGO overlay network provides reliable message routing and hierarchical addressing. Attacks targeting pure availability (e.g., flooding or routing disruption) are considered orthogonal and are outside the scope of this work.

B. SECURITY GOALS

PVTNs are designed to satisfy the following security objectives:

- **Confidentiality**: Membership requests and internal control messages must not be readable by unauthorized parties.

- **Authentication**: All membership and delegation actions must be cryptographically verifiable.

- **Authorization**: Only explicitly authorized managers can admit or revoke members.

- **Sybil Resistance**: Identity creation alone must not grant network membership.

- **Compromise Containment**: Breaches must be limited to the affected subtree.

- **Non-enumerability**: External observers must not be able to enumerate members or organizational structure.

C.   SECURITY ANALYSIS

Confidentiality of Join Requests   Join requests are encrypted using the designated manager's public key. Since public keys are not globally visible and are only disclosed within direct manager–member relationships, unauthorized nodes cannot decrypt join requests or infer organizational membership. This protects against passive eavesdropping and traffic inspection attacks.

Authentication and Authorization   Authorization is enforced through manager-signed delegation certificates. Each certificate cryptographically binds a child's public key to a parent's authority. Verification requires a valid signature chain rooted at a trusted manager, preventing unauthorized membership even if network routing is compromised.

Replay Protection   Each join request includes a cryptographically random nonce or timestamp bound to the request contents. Managers maintain a local cache of recently seen nonces to prevent replay attacks. Replayed messages are rejected without affecting existing memberships.

Sybil Resistance   While an adversary may generate an arbitrary number of public/private key pairs, identities alone convey no trust in PVTNs. Membership is granted only through the explicit issuance of a delegation certificate by an authorized manager, reflecting human or organizational approval. Consequently, Sybil identities cannot infiltrate the PVTN without repeatedly obtaining valid managerial authorization. The absence of a global identity namespace and the use of manager-scoped delegation further prevent Sybil nodes from accumulating influence or bypassing access control.

Compromise Containment   If a node is compromised, the adversary gains access only to that node's private key and certificates. Because delegation is strictly hierarchical, the compromise affects only the node's subtree. Ancestors and sibling subtrees remain unaffected, preserving global security.

Non-enumerability and Privacy   Public keys and certificates are not published in global directories. External observers cannot enumerate nodes or infer tree structure, even with full network visibility. This property is critical for privacy-sensitive environments such as enterprises, research consortia, and military networks.

D.   THREAT MODEL SUMMARY

The table 1 summarizes how PVTNs enforce confidentiality, integrity, and authorization despite operating over a fully adversarial network. All security guarantees are cryptographically enforced at the application layer and do not rely on the correctness or trustworthiness of the underlying VIRGO overlay.

E.   PERIODIC KEY ROTATION

To mitigate long-term key exposure, PVTNs support localized periodic key rotation. Each manager periodically generates a fresh public/private key pair and securely distributes the new public key to its direct subordinates. New delegation certificates are issued under the updated key, while older certificates naturally expire.

Key rotation is strictly local to the affected subtree and does not require global coordination or rekeying across tenants. This design minimizes operational disruption while significantly reducing the attack window associated with compromised long-term keys.

F.   SECURITY GUARANTEES

Under the stated assumptions, PVTNs provide:

- Cryptographically enforced hierarchical authorization

- Strong privacy through non-enumerable membership

- Robust resistance to Sybil and forgery attacks

- Damage containment under partial compromise

These properties collectively demonstrate that PVTNs offer a secure and scalable foundation for multi-tenant, hierarchy-driven distributed systems operating over public infrastructures.

VI.   DEPLOYMENT SCENARIOS

Consider a large enterprise or a multi-institution research consortium operating over a shared public infrastructure. The VIRGO overlay provides scalable routing and hierarchical organization across all participants. Each department, subsidiary, or partner organization establishes its own Private Virtual Tree Network (PVTN), rooted at its respective authority or manager.

In an enterprise setting, employees join by submitting encrypted requests through the VIRGO overlay. Only designated managers can decrypt and approve membership, issuing delegation certificates scoped to organizational roles. Departments remain cryptographically isolated, ensuring that internal organizational structures and membership remain private from external observers.

In a research consortium, each institution operates its own PVTN over the shared VIRGO substrate. Cross-consortium collaboration is enabled through controlled delegation between managers without exposing global

Table 1: Threat Model and Mitigations in Private Virtual Tree Networks (PVTNs)

Threat	Description	Mitigation
Eavesdropping	Interception of join, delegation, or control messages over the public VIRGO overlay	Join requests encrypted with manager public keys; no plaintext exposure
Replay Attack	Reuse of previously valid join or authorization messages to gain unauthorized access	Nonce or timestamp binding; replay detection at manager nodes
Forgery	Creation of fake membership, delegation, or revocation certificates	Delegation certificates signed with manager private keys; signature verification
Sybil Attack	Generation of multiple fake identities to gain influence or unauthorized access	Manager-controlled authorization; identities require valid delegation certificates
Node Compromise	Exposure of a node's private key due to compromise or insider threat	Cryptographic isolation at subtree level; no implicit trust propagation
Key Exposure	Long-term leakage of cryptographic keys through operational or cryptanalytic means	Periodic key rotation and certificate expiration within subtrees
Unauthorized Cross-Tenant Access	Attempts by nodes from other tenants to join or enumerate a PVTN	Explicit cross-tenant delegation only; default cryptographic isolation
Malicious Overlay Behavior	Non-compliant or malicious VIRGO nodes attempting message manipulation	End-to-end cryptographic verification independent of overlay routing

membership or relying on a central authority. When a member leaves or a credential is compromised, revocation affects only the relevant subtree, minimizing operational disruption.

This deployment model supports secure collaboration, delegated administration, and privacy-preserving multi-tenancy, all without reliance on global certificate authorities or centralized identity providers.

A. GATEWAY-MEDIATED HIERARCHICAL VERIFICATION FOR STORAGE ACCESS

In this scenario, a child node presents a certificate issued by a manager to a storage node. The storage knows only the gateway's public key. To verify the certificate, the gateway coordinates hierarchical validation: the request is propagated downward to grandparents and parents, who check certificate validity and delegation, and then approvals propagate back up to the gateway. Both child and manager identities remain hidden to preserve privacy.

A.1. PROTOCOL STEPS

Step 0: Child submits request

$$\text{Child} \rightarrow \text{Storage} : \text{Cert}_{m \rightarrow c}$$

The certificate includes:

- A cryptographic commitment to the child identity Commit(child_id).

- Permissions for storage access.

- Nonce and validity period to prevent replay.

- Endorsements from upper layers $\{\mathcal{E}_1, \mathcal{E}_2, \dots\}$ proving issuer legitimacy.

- Zero-knowledge proof $\text{ZKP}_{\text{manager}}$ showing the manager belongs to the tenant.

Step 1: Storage forwards certificate to gateway

$$\text{Storage} \rightarrow \text{Gateway} : \text{Cert}_{m \rightarrow c}, \text{Storage\_ID}, \text{context}$$

Step 2: Gateway upward to hierarchy The gateway forwards the request upwards to the root if needed (or coordinates directly with upper layers) and initiates downward verification to grandparents and parents:

$$\text{Gateway} \rightarrow P_{\text{grandparent}} \rightarrow P_0 : \text{ValidateRequest}(\text{Cert}_{m \rightarrow c})$$

Step 3: Grandparent and parent verification Each intermediate node performs checks:

- **Grandparent** P_1:

  - Verify P_0 is a legitimate member of the tenant.

  - Check delegation rights of P_0 to issue certificates.

  - Ensure that cert is not self-issue certificates by P_0.

  - Optionally enforce policy constraints (subtree size, safety).

- **Parent** P_0:

  - Verify the child N's certificate for integrity and freshness (nonce, timestamp).

- Check that N is authorized for the requested action.

- generate a random number , forward with valid flag to gateway.

- in the same time , sign the random number with public key of N , and send to N.

Step 4: Approval propagation upward Validations propagate upward to the gateway as approvals or denials:

$$P_0 \rightarrow P_{\text{grandparent}} \rightarrow \text{Gateway : Approval / Denial}$$

Step 5: Gateway issues signed proof for storage The gateway generates a signed proof that storage can verify using its public key:

$$\text{SignedProof} = \text{Sign}_{SK_{\text{Gateway}}} \Big(H(\text{Cert}_{m \rightarrow c}),$$
$$\text{DelegationProof},$$
$$\text{Storage\_ID},$$
$$\text{Nonce},$$
$$\text{ValidityPeriod},$$
$$\text{Random Number by } P_0$$
$$\Big)$$

Step 6: Storage verification and access grant Storage verifies the gateway signature. If valid, It asks N to send the random number received. Then it compares the number from N and the number in cert from gateway. If same, then proof N is real by using its own private key to decrypt message to get the number. and is in the same Tenant. Then it grants access according to the approved permissions. Storage does not need to know the public keys of any intermediate nodes.

A.2. SECURITY AND PRIVACY PROPERTIES

- **Child identity privacy**: Cryptographic commitment in the certificate.

- **Manager identity privacy**: Zero-knowledge proof in the certificate; opaque delegation proofs.

- **Hierarchical validation**: Grandparent and parent nodes validate certificate and delegation before approval.

- **Gateway trust anchor**: Storage trusts only the gateway's signature.

- **Proof authenticity**: Manager signs certificate hash; parents/grandparents approve validity.

- **Replay protection**: Nonce and validity period prevent certificate reuse.

- **Access control**: Storage grants access solely based on gateway-verified proof.

A.3. USEFULNESS AND PRACTICAL BENEFITS

The gateway-mediated hierarchical verification protocol provides significant advantages in distributed systems. Storage nodes do not need to know the public keys of intermediate nodes; they rely solely on the gateway's signature, which greatly simplifies key management in large hierarchies. Delegation rules, subtree policies, and safety constraints are enforced by grandparents and parents, ensuring that only authorized nodes can issue certificates or perform actions. Privacy is preserved through zero-knowledge proofs and cryptographic commitments, hiding the identities of both children and managers while still enabling secure verification.

The protocol mitigates misbehavior by preventing leaves or compromised parents from bypassing hierarchical checks, as approvals must propagate through multiple layers before a gateway-issued certificate is trusted. It is scalable because initial certificate validation is delegated to intermediate nodes, and broadcasts are limited to relevant branches, making it suitable for large-scale systems. Upper layers can flexibly enforce policies such as maximum tree depth, subtree size limits, temporal restrictions, or risk-based constraints without exposing internal node identities to storage. Additionally, storage nodes can execute actions solely based on the gateway-signed certificate, allowing lightweight operation without maintaining complex trust relationships. Hierarchical delegation proofs also provide auditability, enabling accountability while maintaining privacy.

This approach is particularly useful for multi-tenant cloud storage, where access policies must be enforced without revealing the internal structure or member identities of a tenant. It is also well-suited for IoT networks with lightweight devices that cannot store all public keys of hierarchical managers, as well as collaborative platforms where enterprises with hierarchical structures need to delegate permissions securely. Finally, the protocol ensures secure content distribution, as certificates issued through hierarchical delegation guarantee that only authorized nodes can access or modify content while minimizing trust exposure.

VII. DISCUSSION

By aligning cryptographic trust with managerial authority, PVTNs reflect real operational practices and reduce administrative complexity while preserving scalability. Unlike traditional PKI-based or flat trust models, PVTNs restrict key visibility and authorization to direct manager–member relationships, preventing global enumeration of participants and mitigating the risk of unauthorized access.

The hierarchical structure of PVTNs, supported by the VIRGO overlay, enables efficient routing, delegation, and dynamic membership management. Subtree-based revocation ensures that security incidents, such as compromised credentials or departing members, affect only the relevant portion of the network, minimiz-

ing operational disruption. This approach also facilitates fine-grained access control within multi-tenant or consortium environments without relying on centralized authorities.

Moreover, PVTNs provide a flexible framework for cross-organizational collaboration. Controlled delegation between managers allows secure, auditable interactions between separate virtual trees while preserving privacy and isolation. By combining overlay-based scalability with cryptographic enforcement, PVTNs offer a practical solution for secure, dynamic, and privacy-preserving distributed systems, bridging the gap between theoretical trust models and real-world hierarchical organizations.

The model assumes that the VIRGO overlay operates reliably and that participating nodes follow protocol specifications. Malicious or non-compliant nodes may attempt to disrupt routing or authorization processes. To mitigate such risks, PVTNs rely on cryptographic verification of all membership operations and delegation certificates, allowing invalid or tampered messages to be detected and discarded. In addition, overlay-level redundancy and multiple independent verification paths provided by VIRGO improve tolerance to node failures and misbehavior. Behavioral monitoring and auditing by managers or designated supervisory nodes can further identify protocol violations and trigger the isolation or revocation of misbehaving participants.

A.   LIMITATIONS

While PVTNs provide strong security guarantees and scalable management, several limitations remain.

First, cryptographic operations associated with new member joins introduce additional computational and communication overhead. Although join events are typically infrequent, the cost may become noticeable in highly dynamic environments. When a joining node does not know the identifier or network location of the responsible manager, the join request is routed through the VIRGO hierarchical overlay using n-tuple nodes in upper layers. This hierarchical propagation ensures that the request eventually reaches the appropriate manager without resorting to network-wide broadcast; however, it may increase message size and join latency, particularly in large-scale deployments.

Second, there are malware or virus-infected nodes, and delayed detection of subtle protocol deviations, particularly in highly dynamic or large-scale environments. While cryptographic verification can prevent unauthorized actions, compromised nodes may still behave correctly at the protocol level while attempting to disrupt availability or leak information through side channels.

Addressing such threats may require stronger adversarial models, enhanced behavioral monitoring, intrusion or anomaly detection mechanisms, and additional fault-tolerance techniques. These extensions, along with defenses against large-scale coordinated malware propagation, are left for future work.

VIII.   CONCLUSION

PVTNs leverage the VIRGO overlay network to construct private, cryptographically enforced hierarchical trust domains over open and untrusted infrastructures. By encrypting join requests using manager public keys and enforcing authorization through signed delegation certificates, PVTNs provide fine-grained, verifiable control over membership and access while preserving scalability. In addition, the design supports public key hiding, ensuring that node public key is disclosed only to authorized parent and its children , thereby reducing information leakage and limiting adversarial reconnaissance.

Authorization decisions are governed by action certificates that explicitly bind permitted operations to delegated identities. Each action certificate is cryptographically verifiable and validated along the delegation chain before execution, ensuring that all actions conform to current authorization policies. Certificate validity checks, including signature verification, freshness, and revocation status, are enforced at each decision point, preventing the misuse of stale or forged credentials.

The proposed architecture integrates the scalability and routing efficiency of VIRGO with strong security guarantees derived from public-key cryptography and explicit delegation chains. Unlike flat PKI-based or fully decentralized trust models, PVTNs achieve a balance between administrative control and distributed operation, making them suitable for enterprise, consortium, and multi-tenant environments that require both autonomy and accountability.

Importantly, PVTNs are not visible to non-member nodes. Membership information, public keys, and internal topology are disclosed only to authorized participants, preventing external entities from discovering or enumerating private virtual trees. Moreover, PVTNs provide a flexible framework for cross-organizational collaboration: controlled delegation between managers enables secure and auditable interactions across separate virtual trees while preserving privacy and isolation. By combining overlay-based scalability with cryptographic enforcement, PVTNs offer a practical solution for secure, dynamic, and privacy-preserving distributed systems, effectively bridging the gap between theoretical trust models and real-world hierarchical organizations.

Future work will focus on a comprehensive performance evaluation of PVTNs under realistic workloads, including join latency, latency of certificate issued and check , and cryptographic overhead. Optimizing cryptographic operations and message aggregation for large-scale deployments remains an important direction. Additionally, integrating PVTNs with existing middleware and cross-tree federation mechanisms will expand applicability to multi-organization ecosystems. Research into resilience against active adversaries and potential extensions to support attribute-based or policy-driven access control is also planned.

IX.   DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE MANUSCRIPT PREPARATION PROCESS

Statement: During the preparation of this work the authors used ChatGPT in order to prepare and writing the draft paper. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

REFERENCES

[1] D. Cooper, et al., "Internet X.509 Public Key Infrastructure Certificate and CRL Profile," *IETF RFC 5280*, 2008. `https://www.rfc-editor.org/info/rfc5280`

[2] B. Neuman and T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 33–38, Sep. 1994.

[3] P. Zimmermann, *The Official PGP User's Guide*, MIT Press, Cambridge, MA, USA, 1995, ISBN: 978-0-262-74017-3.

[4] A. Hubert and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers," *IETF RFC 5452*, Jan. 2009.

[5] L. Huang, "VIRGO: Virtual Hierarchical Overlay Network for Scalable Grid Computing," In: P. M. A. Sloot, A. G. Hoekstra, T. Priol, A. Reinefeld, M. Bubak (eds.), *Advances in Grid Computing - EGC 2005*, Lecture Notes in Computer Science, vol. 3470, Springer, Berlin, Heidelberg, 2005. DOI: `https://doi.org/10.1007/11508380_93`

[6] T. Howes, M. Smith, and G. Good, *Understanding and Deploying LDAP Directory Services*, Addison-Wesley, 2003.

[7] I. Foster and C. Kesselman, Eds., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999.

[8] A. Goel and Y. Rahulamathavan, "A Comparative Survey of Centralised and Decentralised Identity Management Systems: Analysing Scalability, Security, and Feasibility," *Future Internet*, vol. 17, no. 1, art. 1, 2025, doi:10.3390/fi17010001.

[9] Fang, J., Feng, T., Guo, X. et al. "Privacy-enhanced distributed revocable identity management scheme based self-sovereign identity." *J Cloud Comp* 13, 154 (2024). https://doi.org/10.1186/s13677-024-00715-8

[10] C.Fromknecht ,D. Velicanu, and S. Yakoubov , "A Decentralized Public Key Infrastructure with Identity Retention," *IACR ePrint Archive*, 2014. `https://eprint.iacr.org/2014/803.pdf`

[11] N. Li, B. Grosof, and J. Feigenbaum, "Delegation Logic for Distributed Authorization," *ACM Transactions on Information and System Security (TISSEC)*, Volume 6, Issue 1 Pages 128 - 171, 2003. `https://doi.org/10.1145/605434.605438`

[12] N. Li and J. C. Mitchell, "RT: A Role-based Trust-management Framework," in *Proc. 3rd DARPA Information Survivability Conference and Exposition (DISCEX III)*, Washington, DC, USA, 2003, vol. 1, pp. 201–212, doi:10.1109/DISCEX.2003.1194885.

[13] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management," in *Proc. 17th IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 1996, `https://ieeexplore.ieee.org/document/502679`

[14] P. Chapin, C.Skalka, and X. Wang "Authorization in Trust Management: Features and Foundations," *ACM Computing Surveys*, vol. 40, no. 3, art.9, pp. 1-48, 2008.

[15] R. Halder, D. Shin , and D. Roy, "Blockchain-Based Decentralized PKI Using Web of Trust," *Journal of Cybersecurity and Privacy* , 4(2), 196-222, 2024. `https://www.mdpi.com/2624-800X/4/2/10`

[16] M. Castro, et al., "Secure Routing for Structured Peer-to-Peer Overlay Networks," in *5th Symposium on Operating System Design and Implementation (OSDI)*, 2002.

[17] B.Li, M. Ma. "An Advanced Hierarchical Identity-Based Security Mechanism by Blockchain in Named Data Networking". *J Netw Syst Manage* 31, 13 (2023). `https://doi.org/10.1007/s10922-022-09689-x`

[18] A. Mohammad, "Distributed Authentication and Authorization Models in Cloud Computing Systems: A Literature Review" *J. Cybersecur. Priv.* 2022, 2(1), 107-123; `https://doi.org/10.3390/jcp2010008`.

[19] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems". In: Guerraoui, R. (eds) Middleware 2001. *Middleware* 2001. *Lecture Notes in Computer Science*, vol 2218. Springer, Berlin, Heidelberg. `https://doi.org/10.1007/3-540-45518-3_18`

[20] J. Li, et al., "A Performance vs. Cost Framework for Evaluating DHT Design Tradeoffs under Churn,"in *Proc. IEEE INFOCOM*, Miami, FL, USA, Mar. 2005, `DOI:10.1109/INFCOM.2005.1497894`

[21] M. Ali et al., "Blockstack: A Global Naming and Storage System Secured by Blockchains," *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, Denver, CO, pp.181–194, 2016.

[22] F. Wang and P. De Filippi, "Self-sovereign identity in a globalized world: Credentials-based identity systems as a driver for economic inclusion," *Frontiers in Blockchain*, vol. 2, art. 00028, 2020, `doi:10.3389/fbloc.2019.00028.`

[23] I. Stoica et al., "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," *IEEE/ACM Trans. Netw.*, 11 (2003), pp. 17-32.

[24] R. Sandhu et al., "Role-Based Access Control Models," *Computer*, 29(2),pp. 38–47, 1996.

[25] C. Wong et al., "Secure Group Communications Using Key Graphs," *IEEE/ACM Transactions on Networking*, 8(1), pp.16–30, 2000. `DOI:10.1109/90.836475`

[26] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication," *ACM Computing Surveys (CSUR)*, 35.3 (2003): 309-329.

[27] M. Abadi, "Logic in access control," *18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings.*, Ottawa, ON, Canada, 2003, pp. 228-233,`doi:10.1109/LICS.2003.1210062.`

[28] Mazières and D. D. Folkman. "Self-certifying file system". Diss. Massachusetts Institute of Technology, 2000