Department of Creative Informatics
Graduate School of Information Science and Technology
THE UNIVERSITY OF TOKYO

Master's Thesis

A Distributed Content Subscription Mechanism with Revision Discovery to Decouple Content Sharing Platform and Creator ID

リビジョン発見を備えた分散型コンテンツ購読メカニズムによる
コンテンツ共有プラットフォームとクリエイター ID の分離

Zhihai Zhu
朱 志海

Supervisor: Professor Hiroshi Esak

January 2025

Abstract

This thesis introduces an innovative distributed content subscription mechanism designed to address the challenges of platform independence, anonymity, and censorship resistance for content creators and their audiences. By enhancing the Kademlia Distributed Hash Table (DHT) protocol with revision numbers and republication timestamps, this mechanism enables subscribers to discover content updates through heuristic revision queries while maintaining resilience in dynamic network environments.

The proposed mechanism leverages public key cryptography to ensure content authenticity and creator identification, thus establishing a secure and trust-enhanced environment for content sharing. Integration with peer-to-peer protocols, such as BitTorrent, further ensures scalable and efficient content distribution. This system is particularly valuable for creators operating under strict surveillance or content control, offering them the freedom to publish and share without reliance on centralized platforms.

Through a combination of theoretical analysis and experimental validation, the mechanism's effectiveness has been demonstrated. Simulations involving a network of 1000 nodes reveal the system's ability to maintain content availability and facilitate update discovery, even in the absence of the original content creator. These results highlight the potential of the system to provide a robust alternative to existing centralized and decentralized solutions.

The research contributes to the field of distributed systems by addressing the pressing need for decentralized content sharing mechanisms that prioritize user autonomy, privacy, and resilience. By incorporating advancements in cryptographic verification, DHT protocol extensions, and peer-to-peer integration, the proposed solution offers a comprehensive framework for secure, scalable, and censorship-resistant communication.

Future work for this study encompasses several promising directions, including the implementation of a Peer Exchange (PEX)-like mechanism to enhance update dissemination efficiency by leveraging the shared interests of peers already engaged in downloading and seeding content. Further evaluation of the mechanism under real-world networking conditions, such as geographically distributed Kubernetes deployments, could assess its robustness in scenarios involving packet loss, NAT traversal, and other complexities. Optimizing system configurations, such as Linux kernel parameters, could also enable efficient resource utilization for large-scale deployments. Additionally, comprehensive threat analyses, including resilience against DDoS attacks, and empirical testing of mitigation strategies, are vital for strengthening security.

概要

この論文では、コンテンツ制作者とその視聴者のために、プラットフォーム非依存性、匿名性、検閲耐性という課題に対処するための革新的な分散型コンテンツ購読メカニズムを提案しています。このメカニズムは、Kademlia 分散ハッシュテーブル（DHT）プロトコルを改良し、リビジョン番号や再公開タイムスタンプを導入することで、動的なネットワーク環境でも堅牢性を維持しながら、ヒューリスティックなリビジョンクエリを通じて購読者がコンテンツの更新を発見できるようにします。

提案されたメカニズムは公開鍵暗号を活用し、コンテンツの真正性と制作者の識別を保証します。これにより、コンテンツ共有においてセキュアで信頼性の高い環境を構築します。また、BitTorrent などのピアツーピア（P2P）プロトコルとの統合により、スケーラブルで効率的なコンテンツ配信を実現します。このシステムは特に、厳しい監視やコンテンツ管理下で活動する制作者にとって、中央集権的なプラットフォームに依存せずにコンテンツを公開・共有する自由を提供します。

理論的分析と実験的検証を組み合わせることで、このメカニズムの有効性が示されています。1000 ノードのネットワークを用いたシミュレーションでは、元のコンテンツ制作者が不在の場合でも、コンテンツの可用性を維持し、更新の発見を促進できることが確認されました。これらの結果は、このシステムが既存の中央集権型および分散型ソリューションに対する堅牢な代替案を提供する可能性を示しています。

本研究は、ユーザーの自律性、プライバシー、耐障害性を優先する分散型コンテンツ共有メカニズムの必要性に応えることで、分散システム分野に貢献します。暗号化による検証、DHT プロトコルの拡張、P2P 統合の進展を組み込むことで、提案されたソリューションは、安全でスケーラブルかつ検閲耐性のある通信のための包括的なフレームワークを提供します。

将来の研究では、ピア間の共有された関心を活用してコンテンツの更新配信効率を向上させる Peer Exchange（PEX）に似たメカニズムの実装が有望とされています。また、パケット損失や NAT 越えなどの複雑な状況を伴う地理的に分散した Kubernetes 展開など、実世界のネットワーク条件下でのメカニズムの堅牢性を評価することも課題とされています。さらに、Linux カーネルパラメータなどのシステム構成の最適化により、大規模な展開におけるリソース利用効率を向上させることができます。加えて、DDoS 攻撃に対する耐性や緩和戦略の実証的な検証を含む包括的な脅威分析も、セキュリティ強化において重要です。

Contents

Chapter 1 Introduction 1

Chapter 2 Distributed Content Sharing and Its Various Concerns 4
2.1 Peer-to-Peer Data sharing 4
2.2 Distributed Hash Table 5
2.3 Blockchain based Decentralized Platforms 6
2.4 Public Key Cryptography 7
2.5 Federated Social Platforms 9

Chapter 3 Distributed Publish and Subscribe System 11
3.1 Design Priciple and Consideration 12
3.2 Protocol Definition 13
3.3 Clients Runtime Behavior Definition 17

Chapter 4 Concept Verification and Experiments 19
4.1 Proof of Concept Implementation 19
4.2 Experiments and Results 19

Chapter 5 Comparison and Threat Analysis 22
5.1 Comparison with Previous Works 22
5.2 Threat and Security Analysis 23
5.3 Future Work Directions 25

Chapter 6 Conclusion 27

Publications and Research Activities 28

References 29

Chapter 1

Introduction

Social networks and user-generated content (UGC) platforms have emerged as foundational elements of modern digital ecosystems, profoundly shaping the ways in which individuals, communities, and organizations interact and communicate in the contemporary online environment [1]. These platforms, which encompass a broad range of applications including social networking sites (e.g., Facebook, LinkedIn), media-sharing platforms (e.g., YouTube, Instagram, TikTok), and collaborative content repositories (e.g., Wikipedia, Reddit), facilitate the creation, dissemination, and consumption of content by users on an unprecedented scale. Moreover, they have become powerful tools for identity expression and personal branding, allowing users to curate their digital personas and interact with a global audience. The accessibility of these platforms has lowered barriers to entry for creators, providing opportunities for individuals from diverse backgrounds to share their ideas, stories, and artistic expressions.

However, the proliferation of these platforms also presents significant challenges [2]. Issues such as data privacy, misinformation, algorithmic bias, and the potential for echo chambers and polarization have sparked critical debates about the ethical and regulatory frameworks governing their operation. As social networks and UGC platforms continue to evolve, addressing these challenges will be crucial to ensuring that they contribute positively to society while minimizing their potential harms. As companies operating social and user-generated content (UGC) platforms, or even countries continue to expand and exert greater control over data collection, recommendation algorithm manipulation and content moderation, concerns are growing regarding the diminishing internet freedom, as well as threats to privacy, anonymity, and internet autonomy of individual users and content creators.

Since the early days of the internet, there has been a persistent and growing effort to harness decentralized communication technologies to reduce users' and content creators' reliance on centralized social media platforms and the control they exert. These initiatives aim to address critical issues such as privacy concerns, censorship, and the monopolization of digital spaces by offering alternative systems that promote greater autonomy and user empowerment. Notable examples include federated protocols such as Mastodon [3], Matrix [4], BlueSky [5, 6], which enable distributed social networking and communication without centralized oversight. Additionally, peer-to-peer data exchange protocols like BitTorrent [7, 8], Emule [9], Usenet [10, 11], and IPFS [12, 13] provide decentralized methods of sharing and storing information, reducing the risks of single points of failure and enhancing content resilience. Blockchain-based platforms, such as Steemit [14, 15, 16], further extend this decentralization by integrating immutable ledgers for content ownership and reward systems, fostering transparency and incentivizing participation. These technologies not only offer users and creators compelling alternatives to traditional platforms, but also exert pressure on mainstream social media companies to innovate, improve privacy protections,

and adopt more user-centric policies. By challenging centralized control, decentralized technologies represent a significant step toward an internet ecosystem that prioritizes freedom, diversity, and inclusivity, paving the way for a more equitable digital future.

However, these systems also face critical limitations that hinder their ability to cater to diverse user needs effectively. For example, federated platforms like Mastodon and Matrix often suffer from fragmentation, where communities are isolated across independent servers, making cross-network communication and content discovery less seamless than on centralized platforms. Furthermore, while these technologies promise greater freedom of speech, they are not immune to external pressures. For instance, activists operating under repressive regimes may find their activities constrained by local laws targeting decentralized systems or by the need for technical expertise to navigate these platforms securely. Independent media creators in conflict zones may also struggle with decentralized platforms' limited reach and visibility, reducing their ability to amplify critical messages effectively. These limitations underscore the need for ongoing innovation to improve usability, scalability, and accessibility while preserving the core principles of decentralization. In the context of today' s content consumers favoring high-density information that demands greater transmission efficiency, coupled with increasingly pervasive internet surveillance and the erosion of creative and distributive freedom for content creators, there is a heightened need for privacy, freedom, autonomy, and data-sharing efficiency on the content sharing technologies.

In response to the challenges identified with existing decentralized technologies, this study seeks to design and propose a novel content-sharing and subscription mechanism that addresses the limitations of current systems while adhering to the core principles of platform independence, strong anonymity, high availability, and efficient transmission. By focusing on these key attributes, the proposed mechanism aims to meet the evolving and increasingly demanding requirements of decentralized content distribution in the modern era, providing a robust alternative for users who prioritize privacy, resilience, and accessibility. Specifically, this mechanism is designed to operate independently of any central authority, ensuring greater user autonomy and mitigating the risks associated with centralized control. Strong anonymity measures are integrated to protect user identities, particularly for those operating in restrictive environments or under political constraints. High availability ensures consistent access to content, even in adverse network conditions, while efficient transmission addresses the scalability challenges faced by users behind underperforming network environments. To validate the effectiveness of this approach, the study conducts experiments and compares with previous related works, demonstrating that the proposed mechanism's improvement and superiority over existing technologies in addressing these specific critical issues. Through these analyses, this research contributes an advancement to the field of decentralized content distribution, offering a solution that aligns with modern demands and challenges.

The proposed content sharing and subscription mechanism builds upon the foundational principles of digital signatures and decentralized networking to provide a robust and secure framework for content distribution. Central to this approach is the use of private keys, generated and securely managed by content creators, which serve as unique identifiers to trace and authenticate all content originating from a specific individual. This not only ensures the integrity and provenance of the content but also eliminates the need for centralized identity verification systems. The mechanism extends the widely adopted Kademlia network protocol by introducing enhancements such as revision numbers and timestamps dual channel publishing and indexing, which are incorporated into the hashing and querying processes within the Distributed Hash Table (DHT) network. These modifications allow for dynamic tracking of content updates while maintaining the decentralized nature of the system. To further facilitate subscription functionality, subscribing clients

periodically perform heuristic version update queries and republish operations to ensure they always retrieve the latest revision of content from a given creator. Once retrieved, these updates integrate seamlessly with efficient decentralized content-sharing protocols like BitTorrent or IPFS, which handle the final stage of content distribution. This approach not only optimizes the availability and freshness of content but also leverages the scalability and resilience of existing decentralized protocols to create a comprehensive solution for decentralized content sharing and subscription. The design is strategically developed to be highly compatible with the widely used BitTorrent network infrastructure, ensuring seamless integration that enhances usability by leveraging the familiar, well-established framework of BitTorrent, while simultaneously lowering the entry barrier for new users by minimizing the learning curve for yet another client software and simplifying the process of getting started with decentralized content sharing, thus encouraging broader adoption and ensuring that both novice and experienced users can easily utilize the system without significant technical obstacles.

The rest of this thesis is organized as follows. Chapter 2 provides an overview of the development of distributed networking and social platform algorithms, highlighting their classical approaches as well as associated privacy and security concerns. Chapter 3 presents a detailed description of the proposed protocol, including the rationale behind key design decisions and the runtime dynamical requirements for participating clients. Chapter 4 outlines the proof-of-concept implementation and simulation experiments, utilizing containerization technologies and addressing scaling challenges. Chapter 5 compares the proposed design with related distributed solutions, examines potential attack models such as Sybil attacks and protocol detection, and outlines future research directions. Finally, Chapter 6 summarizes the findings, addressing pressing issues in privacy, security, and internet freedom, and highlights the advantages of the proposed method along with recommendations for future work.

Chapter 2

Distributed Content Sharing and Its Various Concerns

The emergence of distributed data sharing networks represents a paradigm shift in the way information is exchanged, stored, and managed. Unlike traditional centralized systems, these networks leverage decentralized architectures to eliminate single points of failure, improve scalability, and enhance user autonomy. Peer-to-peer (P2P) networks such as BitTorrent and IPFS exemplify this transformation, enabling users to directly share and retrieve data without relying on central servers.

Concurrently, distributed and federated social platforms have gained traction as privacy-respecting alternatives to conventional social media. Platforms like Mastodon and Diaspora operate on federated architectures, allowing users to connect across independently hosted servers while maintaining control over their data. These techniques emphasize interoperability, decentralization, and user sovereignty, fostering a more democratic approach to social networking.

Advancements in distributed ledger technologies, such as blockchain, further support the integrity and transparency of data sharing networks. By leveraging consensus algorithms and cryptographic techniques, these systems ensure tamper-proof data exchanges and robust security measures. This synergy between distributed systems and federated platforms has redefined content sharing paradigms, paving the way for innovative applications while addressing key challenges in data ownership and privacy.

However, these advancements come with an inherent set of security concerns that challenge the reliability, integrity, and privacy of shared information. This chapter explores the evolution of distributed content sharing systems and provides analysis of their associated security challenges.

In this chapter, we will provide a comprehensive overview of related prior works in the domains of distributed data sharing and social networking. We will critically analyze these contributions, highlighting their limitations and identifying the needs that the proposed mechanism seeks to address. Additionally, this chapter will detail the foundational technologies and theoretical underpinnings that inform and support the design and implementation of the proposed system. This discussion will lay the groundwork for understanding the contributions and innovations introduced in subsequent chapters.

2.1 Peer-to-Peer Data sharing

Peer-to-peer (P2P) data sharing technologies have revolutionized the landscape of distributed content distribution, offering an efficient and resilient framework for sharing files and information. Prominent P2P technologies include BitTorrent, IPFS, and eMule, each

of which embodies unique methodologies and architectures that cater to diverse use cases.

BitTorrent is among the most widely recognized P2P protocols, leveraging a swarming mechanism to distribute large files efficiently. In this protocol, files are split into smaller pieces, enabling users to download and upload simultaneously, thereby optimizing bandwidth usage and reducing the load on individual servers. BitTorrent relies on trackers to coordinate peers and facilitate the exchange of file pieces, though advancements such as Distributed Hash Tables (DHT) have reduced dependence on centralized trackers.

The InterPlanetary File System (IPFS) represents a modern evolution of P2P technologies, employing a content-addressed model to ensure data integrity and persistence. Unlike location-based addressing in traditional systems, IPFS uses unique cryptographic hashes to identify and retrieve data, promoting immutability and verifiability. This approach eliminates reliance on specific servers, making the system inherently resilient and decentralized.

eMule, an earlier P2P file-sharing application, utilized the eDonkey network to connect users for file exchange. Its hybrid architecture combined centralized servers for indexing with decentralized peer connections for data transfer. While less prevalent today, eMule highlighted early efforts to balance scalability with efficiency in P2P networks.

Despite their innovations, these P2P systems face notable limitations. BitTorrent often depends on centralized tracker and search engine sites for peer and torrent discovery, introducing potential vulnerabilities and points of failure. Similarly, IPFS, while decentralized, lacks a fully distributed searching and subscribing mechanism, making content discovery challenging in large-scale networks. These shortcomings underscore the need for advancements in decentralized indexing and search technologies to enhance the usability and resilience of P2P systems.

The limitations of these technologies highlight the broader challenges in designing fully autonomous and user-friendly distributed content sharing systems. Addressing these issues remains a critical focus for researchers and practitioners striving to improve the robustness and accessibility of P2P networks.

The primary issue this research seeks to address is precisely their limitation of lacking a fully distributed content sharing and subscribing mechanism. By resolving the distributed subscription problem and integrating with existing BitTorrent and IPFS infrastructures, we aim to provide content creators and their audiences with an efficient means to distribute and share content of interest using well-established technologies. Furthermore, this approach has the potential to operate under the protection of "Collateral Freedom" [17].

2.2 Distributed Hash Table

Distributed Hash Tables (DHTs) have emerged as a foundational technology in distributed systems, offering a scalable and decentralized approach to data lookup and storage. The development of DHTs dates back to early research efforts aimed at overcoming the limitations of centralized directory services, enabling efficient resource location in distributed networks. Various DHT designs exist, including Chord, Pastry, Tapestry, Kademlia, and its variant mainline-DHT utilized by the BitTorrent protocol.

One of the most influential DHT implementations, Kademlia, introduced in 2002, employs a novel XOR-based distance metric to organize nodes into a structured overlay network. This design enables efficient routing and lookup operations, with queries resolving in logarithmic time relative to the number of nodes. Kademlia's design also incorporates redundancy and fault tolerance, making it resilient to node failures and network partitions. Its versatility has led to its adoption in various applications, including eMule and

the Ethereum blockchain.

BitTorrent's Mainline DHT represents another significant advancement in DHT technology. Based on Kademlia, it serves as a decentralized tracker for BitTorrent networks, enabling peers to discover one another without relying on centralized servers. Mainline DHT utilizes compact routing tables and iterative lookups to optimize performance in dynamic and large-scale networks. Unlike traditional trackers, it provides a fully decentralized mechanism for managing peer discovery and metadata distribution.

While both Kademlia and Mainline DHT share foundational principles, they differ in their implementation details and application domains. Kademlia emphasizes general-purpose distributed lookups with a focus on flexibility and extensibility, while Mainline DHT is tailored for peer discovery use cases, prioritizing efficiency and simplicity. Additionally, Mainline DHT introduces mechanisms specific to BitTorrent, such as token-based security to prevent attacks like malicious brute force requests.

Despite their strengths, DHTs are not without limitations. Both Kademlia and Mainline DHT rely on structured overlays that require consistent maintenance of routing tables, which can introduce overhead in highly dynamic networks. Furthermore, DHT-based systems often lack robust mechanisms for secure data storage and access control, leaving them vulnerable to malicious actors, like Sybil Attack. These challenges underscore the need for continued research and innovation to enhance the scalability, security, and usability of DHTs in distributed systems.

This research primarily focuses on extending Kademlia's design and protocol, as earlier DHT designs exhibit inferior performance compared to Kademlia. Furthermore, the mainline-DHT enforces overly restrictive rules, which limit its potential for storing various types of data within the DHT and, consequently, its extensibility.

2.3 Blockchain based Decentralized Platforms

Blockchain-based decentralized platforms have garnered significant attention in recent years due to their innovative approaches to consensus mechanisms and distributed ledger technologies. Central to this evolution are Proof of Work (PoW) and Proof of Stake (PoS), two predominant consensus algorithms that underpin a variety of blockchain networks. These mechanisms enable trustless and tamper-resistant systems, ensuring the integrity and reliability of decentralized platforms.

The classical Steemit platform exemplifies the application of blockchain in decentralized social media. Built on a delegated Proof of Stake (DPoS) blockchain, Steemit rewards users with cryptocurrency tokens for creating and curating content. This economic incentive model fosters active community participation while maintaining a transparent and immutable ledger of interactions. Steemit's design integrates blockchain-based voting and reputation systems to promote content quality and user accountability, setting a benchmark for decentralized social platforms. However, analysis reveals significant limitations in Steemit's decentralization and reward system. In the design of most blockchain based platforms, users are forced to store contents they are not interested, or even encrypted. The DPoS consensus protocol appears to fall short of its decentralization goals, as witness seats are likely controlled by a few large shareholders. And bots operate within a supply network facilitated by large shareholders, undermining the intended goal of rewarding high-quality content. Furthermore, the deep integration of social and value-transfer operations has fostered a marketplace where bots trade voting power for monetary gain. These findings underscore substantial misuse of the platform and highlight the inadequacies of current consensus protocols and reward systems, offering insights for improving blockchain-based social media platforms and designing more effective future systems.

LBRY, now rebranded as Odysee, represents another notable blockchain-based platform. Focused on decentralized content sharing, LBRY utilizes a blockchain to store metadata and manage content ownership. Its design allows creators to retain control over their intellectual property while facilitating direct monetization through cryptocurrency payments. By eliminating intermediaries, LBRY provides a more equitable revenue model for content creators and consumers.

Despite their promise, blockchain-based decentralized platforms face significant criticisms and challenges. Privacy concerns are prevalent, as blockchain's immutable nature often conflicts with the right to be forgotten and other privacy regulations. Security vulnerabilities, such as susceptibility to Sybil attacks and reliance on cryptographic primitives, pose additional risks. Moreover, issues related to internet autonomy arise when platforms are hosted on infrastructure that remains under centralized control, limiting their true decentralization.

Energy inefficiency is another prominent critique, particularly for Proof of Work systems, which require extensive computational resources to validate transactions. While Proof of Stake and other emerging consensus mechanisms aim to mitigate these environmental impacts, achieving a balance between security and sustainability remains an ongoing challenge.

The proliferation of automated bots also undermines the integrity of decentralized platforms. Bots can manipulate voting systems, content rankings, and economic incentives, distorting the intended democratic processes. Addressing these weaknesses necessitates advancements in consensus algorithms, enhanced security protocols, and mechanisms to ensure equitable participation and resource distribution.

In summary, blockchain-based decentralized platforms offer transformative potential but are accompanied by significant technical, environmental, and social challenges. As the field continues to evolve, resolving these issues will be critical to realizing the vision of truly decentralized and equitable digital ecosystems.

2.4 Public Key Cryptography

Public Key Cryptography (PKC) [18] represents a cornerstone of modern cryptographic protocols, enabling secure communication, authentication, and data integrity in open and untrusted environments. Unlike symmetric cryptography, which relies on a shared secret key, PKC employs a pair of mathematically linked keys: a public key, which can be widely disseminated, and a private key, which remains confidential to the owner. This dual-key mechanism underpins critical functionalities, including encryption, digital signing, and identity verification.

The security of PKC relies on the computational infeasibility of solving certain mathematical problems, such as integer factorization (in the case of RSA) or the discrete logarithm problem (used in Diffie-Hellman and elliptic curve cryptography). These hard problems ensure that even with access to a public key, an adversary cannot feasibly deduce the corresponding private key.

2.4.1 Digital Signing

One of the pivotal applications of Public Key Cryptography is digital signing, a process that guarantees the authenticity and integrity of digital messages or documents. A digital signature is generated using the signer's private key and can be verified by anyone with access to the corresponding public key.

To create a digital signature, the signer typically computes a hash of the message to

be signed. This hash is then encrypted using the signer's private key, producing the signature. Verification entails decrypting the signature with the public key to retrieve the original hash and comparing it with a newly computed hash of the received message. A match confirms that the message has not been tampered with and that it originates from the purported sender.

Digital signatures are indispensable in numerous domains, such as software distribution, electronic contracts, and secure email. They mitigate risks associated with message forgery and unauthorized alterations, thus bolstering trust in digital interactions.

2.4.2 Private Key as Identity

The use of a private key as a digital identity introduces a robust paradigm for secure verification. In this framework, the private key functions as a unique identifier for its owner, enabling identity verification without the need for traditional credentials like usernames and passwords. This approach has far-reaching implications for both security and user convenience.

The verification process typically involves the owner proving possession of the private key without revealing it. Techniques such as zero-knowledge proofs or cryptographic challenges can facilitate this. For instance, a verifier may generate a random challenge that the owner signs with their private key. By validating the signed challenge with the public key, the verifier can confirm the owner's identity while preserving the confidentiality of the private key.

This concept has been increasingly adopted in decentralized systems, such as blockchain networks and Web3 applications. In these contexts, private keys serve as self-sovereign identities, empowering users to control access to their assets and data without reliance on centralized authorities. Furthermore, private key-based identities align with principles of privacy and security by minimizing the exposure of sensitive information.

2.4.3 Security Considerations

While Public Key Cryptography provides robust mechanisms for secure communication and identity verification, its effectiveness hinges on the secure management of private keys. Compromised private keys can lead to unauthorized access, identity theft, and other severe consequences. Therefore, secure storage solutions, such as hardware security modules (HSMs) or trusted platform modules (TPMs), are essential.

Moreover, emerging threats, such as quantum computing, pose potential risks to the foundations of current PKC algorithms. Post-quantum cryptography seeks to address these challenges by developing quantum-resistant algorithms. These advancements will be critical for sustaining the security of digital signatures and private key-based identity verification in a post-quantum era.

In summary, Public Key Cryptography, with its foundational principles and applications, continues to shape the landscape of secure digital interactions. Its role in digital signing and private key-based identity verification exemplifies its versatility and significance in addressing modern security challenges.

The proposed mechanism will leverage these properties to provide and verify the content creator's identity, as well as the integrity and authenticity of newly published updates to the content.

2.5 Federated Social Platforms

In recent years, the landscape of social platforms has witnessed a significant transformation with the emergence of federated social platforms. Unlike traditional centralized platforms such as Facebook and Twitter, federated platforms are built on decentralized architectures, where multiple independent servers, or "instances," operate collaboratively while maintaining autonomy. This shift represents a response to growing concerns about centralized control, data privacy, and user autonomy.

Among the most prominent examples of federated platforms are Mastodon, Lemmy, LBRY/Odysee, and BlueSky. Mastodon, launched in 2016, has become synonymous with decentralized microblogging. Leveraging the ActivityPub protocol, Mastodon enables users to communicate seamlessly across independently hosted instances. Lemmy, on the other hand, focuses on decentralized link aggregation and discussion, drawing comparisons to centralized platforms like Reddit. Its architecture allows for community-driven moderation and content curation. LBRY, and its video-sharing derivative Odysee, emphasize decentralized content distribution using blockchain technology, providing creators with direct control over their work and monetization avenues. BlueSky, initially incubated by Twitter before becoming independent, introduces the AT Protocol, a novel approach to decentralization. BlueSky aims to combine the familiarity of traditional social media with the benefits of decentralized architecture, allowing users to choose algorithms and retain greater control over their data. Each of these platforms has contributed uniquely to the federated ecosystem, addressing specific use cases and user demands.

The rise of federated platforms can be attributed to several factors, including increasing public awareness of privacy issues, dissatisfaction with the monopolistic practices of traditional social media giants, and a growing interest in open-source and community-driven technologies. These platforms embody the principles of user empowerment, offering alternatives to centralized data collection and algorithmic control. However, despite their promise, federated platforms face significant challenges that hinder their widespread adoption and long-term viability.

One of the primary limitations of federated platforms is their tendency to gravitate towards centralized hubs. While the architecture promotes decentralization, popular instances often attract a disproportionate number of users, effectively replicating the centralization seen in traditional platforms. This phenomenon undermines the egalitarian ethos of federation and raises concerns about the resilience and sustainability of these networks.

Data privacy, a key selling point of federated platforms, is not without complications. While users may escape the pervasive surveillance of centralized platforms, their data remains vulnerable to administrators of individual instances. Malicious or negligent actors can still compromise user information, challenging the notion of absolute privacy in federated systems. Furthermore, the lack of standardization in privacy policies across instances exacerbates these concerns.

Moderation represents another critical issue for federated platforms. Decentralization shifts the responsibility for content moderation from a centralized authority to individual instance administrators. This approach, while empowering, often results in inconsistent enforcement of rules and standards. Smaller instances may lack the resources to effectively moderate content, while larger instances may adopt stringent policies that alienate users. Additionally, federated networks are not immune to harmful content, harassment, or coordinated misinformation campaigns, further complicating the moderation landscape.

This highlights the superiority of a fully distributed approach over federated solutions

in addressing these vulnerabilities and censorship.

Censorship and access restrictions also pose significant challenges. Despite the advantages of the federated client-server model like mitigation of single points of failure inherent in DNS solutions and resistance to TLS Client Hello SNI attacks, federated platforms are still vulnerable to external pressures. It remains vulnerable to batch IP blacklisting and TCP reset attacks, which can disrupt communication between instances or deny access to users in certain regions. Governments or powerful entities may exploit these vulnerabilities to suppress dissent or limit the reach of specific platforms. Furthermore, reliance on internet infrastructure controlled by centralized entities, such as domain registrars and hosting providers, exposes federated platforms to systemic risks.

Even with these limitations, federated social platforms continue to evolve, driven by active developer communities and a committed user base. Innovations in protocols, governance models, and moderation tools hold the potential to address some of the challenges outlined above. As these platforms mature, their ability to offer viable alternatives to centralized social media will depend on balancing decentralization with usability, security, and resilience.

Chapter 3

Distributed Publish and Subscribe System

The proposed mechanism is fundamentally rooted in three well-established distributed and cryptographic technologies: public key cryptography, the Kademlia network, and BitTorrent, as depicted in Fig. 3.1. These technologies provide a robust foundation for designing a secure, efficient, and scalable publish and subscribe system.

The subsequent sections present a comprehensive discussion on the rationale for incorporating these technologies into the system architecture, elaborating on their individual contributions and synergistic integration into the protocol.

Additionally, the structure and semantics of the message and remote procedure call (RPC) formats utilized within the protocol are meticulously defined, highlighting their role in enabling seamless communication and coordination among distributed entities. Furthermore, the permissible interactions facilitated by the protocol are elucidated, providing insights into its operational scope and constraints.

Finally, the dynamic behavior of the system during runtime is analyzed, offering a detailed characterization of its adaptive and responsive mechanisms under varying operational conditions. This holistic description underscores the interplay of cryptographic and distributed principles in shaping the proposed mechanism's functionality and resilience.

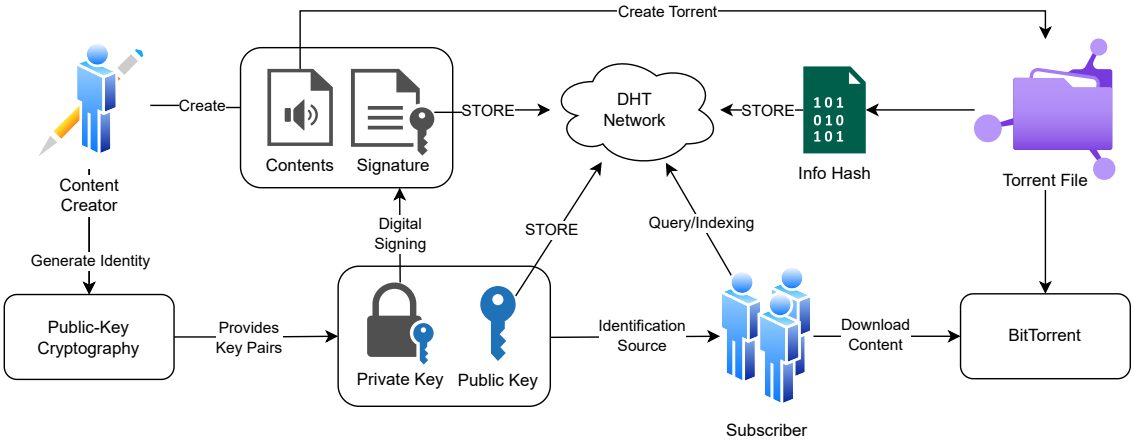

Fig. 3.1. A conceptual overview of the subscription mechanism and the foundational technologies.

3.1 Design Priciple and Consideration

3.1.1 Leveraging Asymmetric Cryptography

Our proposed mechanism will first leverage cryptographic principles, particularly the ability to verify private key ownership in public key cryptography [18]. This is achieved through challenge-response authentication, where any party with the knowledge of the public key can confirm ownership of the corresponding private key without accessing or knowing the private key's contents. Consequently, participants who complete this process can reliably associate the public key with its private key's owner.

Furthermore, the private key owner can generate a digital signature for theoretically any digital data using the private key. Other parties, using the corresponding public key, can then validate the authenticity and integrity of the data by verifying the signature.

This approach enables the unique identification of a content creator through their cryptographic key pair. It also ensures that new content can be authenticated as originating from the same creator.

The proposed mechanism will require all participating peers to generate their own cryptographical key pairs and save and reuse them through their client's online sessions. Content creators who use this mechanism to publish their contents would need to use their private key to sign a signature of the data and provide their public key along with the data and the signature when publishing to this extended DHT network.

Our protocol requires all participating peers to generate cryptographic key pairs and securely store their private keys for use across online sessions. This enables the derivation of a unique client identity by applying the SHA1 hash function to the public key, while also allowing the cryptographic proof of ownership of this identity to mitigate Sybil attacks against the DHT network. Content creators must also generate one or more distinct cryptographic key pairs, each serving as a unique identifier for a separate content creator identity, and use their private keys to sign their data. When publishing to the extended distributed hash table network, they must include the data, its digital signature, and the corresponding public key to enable verification by other participants.

3.1.2 Revamp Kademlia DHT's Key and Value

This research builds upon and extends the design and protocol of Kademlia which offers fast data indexing within a distributed network. In its original implementation, Kademlia does not impose a strict relationship between the key and the value stored in the Distributed Hash Table (DHT) [19]; yet, conventional designs typically map the SHA1 hash back to associated data.

Meanwhile, the protocol specifies that the data stored in the DHT to include metadata about an update to support public key authentication and version tracking, such as the creator's public key, data signature, a revision number, and the (re)publication date. This extension enables subscribers' clients to predict the key of subsequent publications in the DHT by defining the key as derived from the publisher's public key in combination with either the revision or the (re)publication date. Future keys can then be deduced with the knowledge of the current date and last known revision number.

By utilizing a set of modified remote procedure calls (RPCs) based on the original Kademlia protocol, content creators can publish new content with high reachability and anonymity. Meanwhile, subscribers can reliably and efficiently retrieve newly published content of interest.

This definition also introduces variability into the DHT key that responsible for a creator's latest update. Otherwise, all of a creator's publications and their subscribers' queries would target a specific region of the DHT network, leading to load-balancing issues, and overburden that network segment over time, particularly for the creators having large audiences. Additionally, this variability also helps to mitigate Sybil and black-hole attacks, which are further explored in Section 5.2.

For highly active publishers who release multiple updates within a short timeframe, relying solely on a date-specific hash to locate updates is insufficient. Introducing revision-specific hash could address this limitation, and by publishing the value object under both keys into the DHT network, we ensure that subscribers can reliably query either relative new version by date, or any the newer version from the creator by revision number.

The data payload is defined to be minimal but essential representative information, such as an info hash that uniquely corresponds to a torrent file, which, in turn, can be used to refer to and download a batch of files.

By incorporating DHT and BitTorrent, content creators can publish new content with high reachability and anonymity. Meanwhile, subscribers can reliably and efficiently retrieve newly published content of interest.

3.1.3 Incorporate BitTorrent Infrastructure

The well-established BitTorrent protocol, along with its mainline-DHT [20] design and metadata exchange extension, provides a highly efficient method for distributed data transmission. However, it lacks a fully distributed and censorship-resistant mechanism for subscribing to updates. This research proposes extending the original Kademlia network rather than modifying the mainline DHT design. As the mainline DHT is highly optimized to serve as a distributed peer tracker alternative for the BitTorrent protocol, which limits its ability to store arbitrary data in the DHT.

But the proposed method can still be seamlessly adopted and integrated into the design of the BitTorrent protocol and current popular BitTorrent clients. Even without immediate integration, the simulation implementation can leverage the existing BitTorrent infrastructure by using an "info hash" of a torrent file as the data payload. Updates retrieved using the proposed protocol can then be downloaded via magnet links, enabling a distributed and efficient transfer of torrent files and their associated data. This approach effectively extends the capacity of the proposed protocol to be able to represent any data in theory.

In scenarios where the risk of targeted attacks on the DHT network is significant, the data-storing nodes could also be designed to save the torrent metadata or even the data of the publication itself directly. This approach eliminates one layer of indirection, reducing the potential impact of malicious peer pollution and enhancing the resilience of the system against such attacks.

3.2 Protocol Definition

The protocol specifies that a key is derived as a specific hash of portions of the associated value, with the value encompassing all necessary information to ensure its authority, authenticity, and integrity can be verified. It also introduces extended functionality for the remote procedure calls (RPCs) in the original Kademlia network. These are illustrated in Fig. 3.2. Additionally, in Section 3.3 we provide detailed guidelines for the behavior requirements of both publishers' and subscribers' clients during the standard data exchange workflow, which ensures the stability and sustainability of the system.

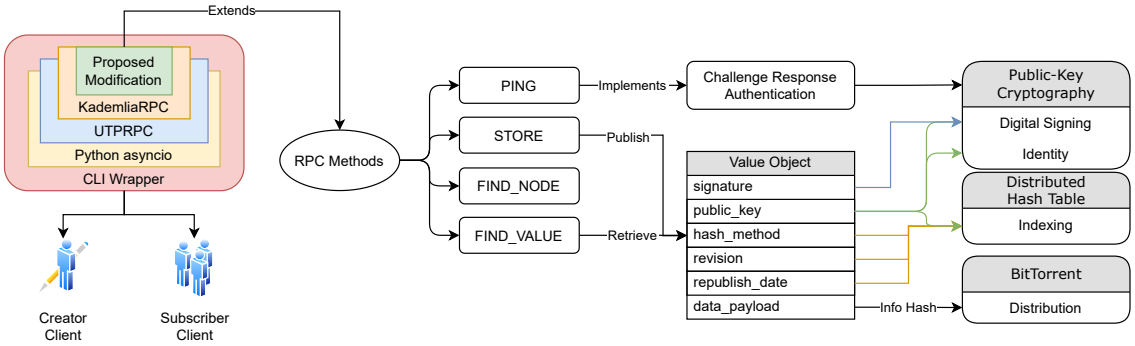

Fig. 3.2. The architecture overview of the proposed distributed subscription system's demonstration implementation.

3.2.1 Value Definition

While the original Kademlia network allows the storage of arbitrary binary data on the DHT, and the mainline DHT restricts storage to peer information only, neither is sufficient to support a subscription mechanism. Therefore, the protocol definition begins by defining the essential data format required for transportation within the extended DHT network design.

Any valid value object must include the following mandatory fields:

```
Value = {
  public_key     : Raw public key bytes
  revision       : Ever increasing integer
  republish_date: ISO format date string
  hash_method    : Bitfield indicates
                   the desired hashing
                   specification
  data_payload   : Raw data bytes, could
                   be torrent's info hash
  signature      : Signature over data
                   payload and revision
}
```

The signature must be generated exclusively by the content creator using the private key corresponding to the public key specified in the value object's `public_key` field. To create the signature, the content creator should concatenate the byte-encoded revision integer with the `data_payload` bytes and sign the resulting bytes using their private key. Including the revision in the signing input is critical to prevent unauthorized modifications, such as altering the revision number to overwrite data from the same creator with other revisions in the DHT.

Any values that cannot be verified using the public key, as the method specified above, will be deemed invalid. Consequently, `STORE` requests for such values should be rejected by all peers. Peers that consistently issue `STORE` requests with invalid values may be subject to blacklisting by compliant peers. However, the specific mechanism for blacklisting has not been prescribed at this stage.

It is important to note that the `republish_date` field is not included in the signed data. This exclusion is intentional, as the third-party peer republishing mechanism (detailed

later in Subsection 3.3.2) is designed to ensure that the content creator's data remains in the DHT network and continues to be consistently indexable, even if the creator is inactive for an extended period.

3.2.2 Key Definition

The key corresponding to a value object must be derived using the SHA1 hash function by hashing the `public_key` together with either the `revision` or the `republish_date`, as specified by the `hash_method` field. Currently, the `hash_method` field can only contain the either 0 or 1. A value of 0 indicates that the DHT key is derived by hashing the `public_key` concatenated with the `revision`, while a value of 1 indicates that the key is derived by hashing the `public_key` concatenated with the `republish_date`.

3.2.3 RPC Commands

The original Kademlia protocol defines four remote procedure call (RPC) commands. This research extends these commands to incorporate additional functionality and enhance security.

PING

The `PING` command probes a node to determine its availability, sending the sender's public key and a randomly generated nonce as a challenge for the receiving node. The responding node would then return its public key, a signature over the received nonce, and a new nonce it generates for its own challenge. The initiating node would subsequently sign the new nonce using its private key and send the signature within a second `PING` RPC message, which could also be combined with other operational RPC messages to optimize communication Round-Trip Time (RTT). All peers will be required to complete the challenge before being saved into other peers' routing table.

Through the use of peer IDs derived exclusively from each peer's public key and the implementation of challenge-response authentication to verify ownership of the claimed peer ID, we aim to mitigate Sybil attacks to some extends. This is further explained in Section 5.2.

Notably, incorporating this mechanism would not impact the performance of the existing design, as the querying node can verify the responding peer's identity upon receiving the first response. This is sufficient to complete the primary request without additional latency. The added final reply's latency only affects the updating of the querying node's information in the responding node's routing table, which is not latency-sensitive. Furthermore, since the original Kademlia protocol prioritizes established nodes over newly joined ones [19], the querying node typically does not need to be added to a fully bootstrapped node' s routing table.

STORE

The `STORE` command is extended in two ways:

On the Sender Side, the sending peer must correctly compute the DHT key hash as required by the "Key Definition" section to determine the appropriate locations within the DHT network for storing the value object. It is the sender's responsibility to construct a valid value object for initial publication or validate an existing value object before republishing it on behalf of the original content creator. Once the value object is prepared, the sender identifies the k-nodes it knows nearest to the DHT key and issues the `STORE` command to these nodes.

On the Receiver Side, upon receiving a `STORE` RPC request, the receiving peer must

validate the data payload and the revision field using the public key and the signature contained within the value object, as specified in the "Value Definition" section. The receiver then computes the DHT key hash for the incoming value and checks whether its internal storage already contains a value mapped to the same key.

If a collision occurs, the receiving peer verifies whether the `public_key` fields of the incoming and stored value objects match. Besides considering collision attack, in rare cases where two SHA1 hashes collide, the receiving peer can adapt its storage structure to hold multiple value objects from different creators mapped to the same DHT key. However, for simplicity, the demonstration implementation does not require this feature.

If the value is tagged to use revision-specific hash, a collision likely indicates identical values, and no further action is necessary. If the value uses date-specific hash and the incoming value shares all fields with the stored value except for the revision number, the receiver should replace the stored value if the incoming value is newer. This behavior supports the co-republish mechanism, discussed later in the "Subscribe Workflow" section.

In either cases, the receiver should refresh the value's time-to-live countdown and report a successful storage operation, even if the received value is older, as the report primarily confirms connectivity and functionality rather than the validity of the value.

Additionally, the use of digital signatures for verification eliminates the need for a temporary token mechanism, such as that defined in BitTorrent' s mainline-DHT protocol, to confirm the sender's legitimacy. Any verified value object should be stored.

FIND_NODE

The `FIND_NODE` command primarily operates as specified in the original Kademlia protocol. However, with the introduction of a peer ID validation mechanism, nodes should exchange and store each other's public key information instead of the randomly generated peer IDs used in the original design. To enhance security, the sender of must include a `PING` RPC for challenge-response authentication with every a `FIND_NODE` query message, accepting results only from nodes that pass validation. Additionally, the sender must respond to the recipient's challenge for its peer information to be inserted into the recipient's routing table. This approach ensures the authenticity of peer information across the DHT network.

FIND_VALUE

The `FIND_VALUE` command retains the functionality of searching by the DHT key hash but introduces additional parameters, including the creator's public key, the `hash_method`, and a minimum revision requirement. Receivers of this RPC are required to respond only with values that satisfy all specified criteria: matching public key, DHT key hash, `hash_method`, and a revision that meets or exceeds the minimum requirement. If no such value is known, the receiver must fall back to providing a corresponding `FIND_NODE` query's result.

Upon receiving a value reply, the RPC sender must verify that the returned value meets the query requirements and validate its signature before collecting the result. Peers returning non-compliant results should be deemed untrustworthy and subsequently removed from the querying peer's routing table. Furthermore, if the RPC sender collects multiple valid responses within a single round of query, which may occur when the `hash_method` specifies date-specific hashing, it should forward all valid values to the RPC command invoker.

3.3 Clients Runtime Behavior Definition

3.3.1 Publisher Workflow

To participate in the content distribution mechanism as a content creator and publisher, one should first generate two pairs of asymmetric cryptographic keys. The first key pair is designated for peer ID validation, while the second is used for signing content or, in the case of leveraging BitTorrent, for signing the `info_hash` of the torrent. And the creator could choose to distribute their signing key pair's public key with their other established identity if they wish to switch from platforms like SoundCloud, YouTube, Twitter, Blog, or torrent tracker sites. If they prefer this identity to stay completely anonymous, they can publish their public key along with the content's genre description on any anonymous forum or discussion border like 4chan.

Assuming the publisher and their audience use BitTorrent to exchange content from now on. The process begins with the publisher creating new content. Their client software should assist in generating a torrent file for the content, optionally including their previous torrent file history, and initiate seeding. Subsequently, the client software generates two DHT value objects, each containing the publisher' s public key, the publication revision (relative to their previous publication history), the production date, the `info_hash`, a signature signed with the signing private key, and the chosen `hash_method`. One value object should employ revision-specific hashing, while the other uses date-specific hashing. The client then publishes these value objects to the DHT network, enabling subscribers' clients to index the content. The `STORE` operation typically completes within a minute, similar to seeding with BitTorrent and mainline-DHT.

For content seeding, the speed depends on network bandwidth and the peer swarm. With an optimized "super-seeding" algorithm, the initial seeder typically needs to upload only twice the content size for other peers to become seeders and redistribute the content themselves. Subscribers who actively seed the content will then act as redistributors. If the publisher has limited time availability during initial seeding, established third-party services such as seed boxes or debrid services can assist by completing the initial seeding on behalf of them after the publisher uploads to these services. Additionally, a new category of services, similar to the Internet Archive, could be designed to automatically scrape and seed the publisher' s content, further supporting distribution. Such services are not essential to the proposed system's functionality, but they can also be valuable to mitigate DoS attacks.

When publishing new content, the publisher's client must repeat the process of creating a torrent file, generating updated value objects, and performing `STORE` and seeding operations. Only the `revision`, `republish_date`, `data_payload`, and `signature` require updates in this process.

3.3.2 Subscriber Workflow

To participate in the system as a subscriber, one should first observe some content creators of interest, and save their public keys in the client software. The client will then generate an asymmetric cryptographic key pair for peer ID validation and begin estimating the DHT hash corresponding to the creator's most recent publication, and initiate queries.

Revision discovery relies more on locating valid value objects in the DHT by recent dates and date-specific hash. The client computes date-specific key hashes using the creator's public key and recent several days' date, as described in the Subsection 3.2.2. Typi-

cally, a three-day window suffices, given that Kademlia's nodes discards <key-value>pairs (re)published more than 24 hours earlier.

After identifying a relatively up-to-date value object and the revision number the creator is recently at, the client will then query the DHT network using revision-specific hashes to progressively discover the latest version. When searching for updates, the client should search for several subsequent revisions simultaneously, as the creator's client may encounter network or operational issues that result in skipped revisions by accident. To not stress the network with too frequent queries, the client should not automatically query for updates way too often. It's recommended to only fetch updates on the first login per day to conduct the daily co-republish procedure, and let the user initiate update check manually when needed. Another possible measure to balance the stress being considered is if the publisher is overly popular, the protocol could add one extra field to the value object to be hashed together for each hash to act like a mirror. Currently this is considered unnecessary as mainline-DHT could handle popular torrent swarm's query just fine.

By querying value objects updates, the client could keep up with new contents as they were published. If BitTorrent is used, subscribers can leech content using the corresponding `info_hash`. They can also contribute to content distribution by seeding for extended periods.

To ensure data remains available in the DHT for as long as needed, the system relies on key-value republishing. Like in the original Kademlia design, peers storing a key-value pair republish the data every hour. And peers which were just storing the data for caching purpose or overtimes becomes not the k-nearest nodes responsible for the data may discard it after 24 hours or even shorter, depending on their ID's distance to the key [19]. They are also allowed to discard any value object older than 7 days. Consequently, the initial publisher is supposed to republish their data within a 24-hour window to maintain its availability. In the proposed system, the detail of the republish procedure should be the same as described in section 2.5 of the original Kademlia paper.

A special case arises when a creator cannot publish a value object with an updated `republish-date` for a given day. To maintain query availability, every subscriber is required to query for the new date's value object on a new day during its hourly routing table refresh routine. If no result is found or if the result is outdated, the subscriber should update the `republish-date` of the latest known value object to the current date and republish it on behalf of the creator. This approach ensures data availability as long as at least averagely one active subscriber is online intermittently.

Chapter 4

Concept Verification and Experiments

4.1 Proof of Concept Implementation

A simulation-based proof-of-concept implementation was developed using Python, leveraging the asynchronous event-driven capabilities of the `asyncio` library to facilitate efficient concurrent network messaging and remote procedure calls (RPCs). The implementation began with the construction of a UDP-based RPC protocol, built on the foundation of `asyncio.DatagramProtocol`. This protocol defined and registered callbacks for handling incoming datagrams, which were subsequently parsed into objects to invoke the corresponding RPC methods.

The protocol was then extended to incorporate the four RPC methods outlined in the proposal, enabling core system interactions and facilitating data exchange between components. To support these interactions, the implementation included a mapping data structure with timeout-based eviction, ensuring efficient resource management and dynamic adaptation to changing network conditions. Furthermore, two network crawlers were implemented to represent node and value iterative query strategies, reflecting the proposed system's operational mechanisms. A high-level interface class was also developed to streamline tasks such as bootstrapping, control, and data processing.

These components collectively formed a demonstration library, which served as the foundation for designing three distinct client programs: an intermediate-node client, a publisher client, and a subscriber client. These client programs were tailored to interact with the library and demonstrate the system' s functionality. To facilitate experimentation, these programs were bind-mounted within Podman container images pre-configured with the necessary Python dependencies and operating system environments. The experiment control script orchestrated the deployment and execution of these containers, ensuring a reproducible and isolated testing environment for validating the proof-of-concept implementation.

4.2 Experiments and Results

Due to computing resource and cost limitation, we firstly run the simulation in a lab wise shared server rather than public cloud services like Amazon AWS or Cloudflare Workers. Although, for more serious and realistic experiment results, deploying and testing on VPS swarms or FaaS platforms could be a good future work direction.

With a single server as the test bed, the best choice is to use a container swarm rather than launching multiple virtual machines. Containers are a virtual environment created with Linux namespace technologies and can provide highly efficient networking emulation, dependency management, and runtime environment isolation.

Using containers can easy the network configuration, as containers will be auto assigned

with IP address in the associated network, and can also be easily manually specified. Due to user permission limitation and protection of other user's research related data on the shared environment, we choose to avoid Docker rootful engine as running Docker rootful containers requires root permission and the programs in the container basically always runs with root privileges. Instead, we chose a rootless alternative container engine implementation, Podman 5.

At first, for testing, one intermediate-node client is first run as the rendezvous node. Then 200 intermediate-node clients instances were deployed as in the DHT network, each hosted in a separate container with a unique IPv4 address from the same user-defined network. All of them are configured to connect to the rendezvous node for the initial bootstrap. But then we discover that due to Linux's ARP Table garbage collection kernel parameter's default limitation, these nodes' inter connection along with other user's programs running on the same OS produced a lot of ARP entries, and when it exceeds the default limitation of 20000 entries, containers won't be able to communicate with new neighbors, thus resulting bootstrap or functional failure.

Then during debugging of the scaling process, we also run into other similar limitations, like Podman's user key storage limitation, Docker's Go sub process count limitations, Linux bridge's max simultaneous connectable ports (thus containers) limits. And most of them requires complex troubleshooting and recompiling of large scale projects like Linux Kernel and Docker Engine. Due to the theoretical property of this research that it extends and bases on the underlying Kademlia protocol's solid implementation, it is not very persuasive to say there are huge difference to conduct an extremely large scale network simulation comparing to a 1000 nodes experiment, as 1000 level scale is sufficient to prove the effectiveness of the design, but going up will only means to test the design of the original Kademlia algorithm, which has been tested thoroughly during these years.

The final design of the experiment goes as: First run one intermediate-node client as the rendezvous node. Then deploy 1000 intermediate-node clients instances into the DHT network, each assigned a dedicated IP in a continuous range from the same user-defined network, hosted in a separate container, and bootstrapped with the previous node's IP to even the distribution of the stress on nodes.

Then four additional client containers were bootstrapped and configured with specific roles: The first acted as the content creator's client, periodically publishing new `Value` objects to the DHT network indexed both by revision and date. The second as the co-seeder's client, deployed immediately after the publisher client, and periodically tries to fetch newer updates produced by the publisher client from the DHT network, while also periodically republish the latest version of they own. The third is deployed halfway though the experiment, to test if a client joined midway during the publication process could rely on the date based key hashing and indexing to quickly find the latest update of the timing. Finally, the fourth is deployed after the publisher client quit, plus 4 loops of the publication interval, to test if the other subscribers could successfully help to co-republish the known latest version, even after the original creator's client had gone offline for an extended period.

At first, we set the emulated day length to be 20 seconds, which means there will be one (re)publish operation per 20 seconds, and the emulated experiment period to 20 days (revisions). Because without traffic control (TC), containers in the same container networks are basically running on localhost quality network, everything went very fast and smooth. After debugging, all the clients' publish, republish, retrieves successfully as expected.

With introducing of TC, we could emulate network conditions like latency and packet loss on container's virtual network adapters. With the packet loss set to 1% everywhere, the system still works, but rarely some clients starts to complain about not receiving

replies for outgoing RPCs. With the packet loss set to 10%, the system could barely function at all, as nodes will ditch out other nodes' information from their routing table after not receiving replies. With the latency set to 10ms, the clients reports `FIND_VALUE` RPC with barely unobservable efficiency differences, as `FIND_VALUE` short circuit during each batch of query iteration when a `VALUE` is found. But `FIND_NODE` as long as `STORE`, which completely relies on `FIND_NODE` to fully discover the nodes closest to the DHT key, can be observed for an average of 2s duration between starting and finishing. When setting latency to 20ms, `FIND_VALUE` is still returning at a relatively fast speed at about 5s, but `FIND_NODE` and `STORE` runs at over 20s, which break an emulated day's length, thus breaking the assumption of the system that a DHT `STORE` could be finished reasonably within a whole day's span, and thus breaks the function of the system. On 150ms delay setting, the nodes could barely finish the publishing and discovery of publications within the experiment time.

Chapter 5

Comparison and Threat Analysis

5.1 Comparison with Previous Works

Properties	BT, IPFS	Mastodon	Steemit	YouTube	Proposal
Speech Freedom	High	Medium	High	Low	High
Censorship Resilience	High	Medium	Medium	Low	High
Transport Efficiency	High	Medium	Low	High	High
Anonymity	High	Medium	High	Low	High
Subscription	No	Yes	Yes	Yes	Yes

Table 5.1. Comparison of different distributed solutions across various requirements.

Several prior attempts have addressed the challenges of decentralized social networking and data sharing. However, during the conceptual discussions of this research, questions frequently arose about how our approach differs in its objectives and methods. This section outlines the specific needs our proposal seeks to address, highlights key dimensions for evaluating the effectiveness of decentralized approaches, summarizes the core characteristics of other well-known methods, and presents a comparative analysis to clarify the advantages of our proposal in addressing the identified issues.

Our primary focus is on individuals such as independent media creators operating in conflict zones, democratic activists, journalists, and advocates for privacy and freedom from mass surveillance. These individuals often face significant challenges in achieving internet autonomy and freedom, especially when their content involves controversial opinions or the disclosure of sensitive information suppressed by powerful entities, including state actors. Their audiences, usually also subject to internet censorship, require reliable and efficient data-sharing mechanisms, the ability to discover continuous updates, and identity anonymity.

Existing P2P sharing protocols, such as BitTorrent [7, 8] and IPFS [12, 13], exhibit limitations due to the absence of a distributed subscription mechanism. These systems rely on tracker sites to provide updated metadata, which are vulnerable to shutdowns, often for legal or copyright-related reasons [21].

Blockchain-based solutions like Steemit [14, 15, 16] present another set of challenges. Users have limited control over the data they are required to store, and while proof-of-work mechanisms are resource-intensive, proof-of-stake systems tend to promote centralization. These systems are also vulnerable to manipulation by large-scale miners and bots, undermining their utility for the target audience [22].

Federated solutions, such as Mastodon [3] and BlueSky [5, 6], are more effective in providing user interactivity and group moderation. However, these features are unnecessary —or even counterproductive—for the target users in our scenario. Federated systems are

highly susceptible to IP blacklisting, DNS pollution, and TLS SNI sniffing attacks, particularly when servers are self-hosted or rely on CDNs. Furthermore, their client-server architecture limits connection speeds due to a lack of concurrency. While emerging platforms like BlueSky aim to return data autonomy to users by their novel data structural design, more established systems like Mastodon have been criticized for their dependence on server administrators having too much user profile managing power.

In comparison, while many decentralized methods tackle specific shortcomings of centralized platforms like YouTube and Twitter, our proposal is designed to overcome the limitations of these approaches while providing an effective solution to the unique challenges faced by the target audience.

5.2 Threat and Security Analysis

5.2.1 Sybil Attack

As DHTs have emerged as a fundamental component in decentralized systems, enabling efficient storage and retrieval of data without reliance on a central authority, their decentralized nature, however, renders them susceptible to various security threats, among which the Sybil attack [23] is particularly pernicious. This subsection provides an overview of the Sybil attack and its implications for DHT-based systems, elucidating the mechanisms through which adversaries exploit these vulnerabilities and outlining the countermeasures proposed in the literature.

The term "Sybil attack" refers to a security threat in which a single adversary subverts the trust model of a network by creating multiple false identities, or "Sybil nodes." These malicious nodes collaborate to disrupt the normal functioning of the system, manipulating data routing, storage, or decision-making processes. In DHTs, where peer-to-peer interactions rely on the assumption of trustworthiness among nodes, the proliferation of Sybil nodes undermines the integrity of the network. By gaining a disproportionate influence, the attacker can manipulate data placement, intercept communications, or launch denial-of-service attacks. These attacks pose significant challenges for applications relying on DHTs, including blockchain systems, file-sharing networks, and content delivery platforms.

Various strategies have been proposed to mitigate the threat of Sybil attacks in DHT-based systems. These countermeasures can be broadly categorized into identity management, reputation systems, and resource-based approaches:

Identity Management: Centralized or decentralized identity verification schemes aim to prevent the creation of multiple false identities. Certificate authorities, for instance, can issue cryptographic credentials to authenticate nodes, while decentralized methods such as Web of Trust rely on peer endorsements.

Reputation Systems: By tracking the behavior of nodes over time, reputation systems penalize malicious activity, thereby incentivizing honest participation. Metrics such as query response accuracy or routing reliability are commonly employed to assess trustworthiness.

Resource-Based Defenses: These approaches require nodes to demonstrate the possession of scarce resources, such as computational power (proof-of-work) or storage capacity. By imposing a cost on identity creation, these mechanisms limit the feasibility of launching large-scale Sybil attacks.

For the scenario of the task this research aim to solve, the Sybil attack can be combined with a black hole attack that when the adversary targets a specific content creator's the DHT key, and controls a large amount of clients with peer ID crafted very close to the

target DHT key, they can effectively control the vicinity of that DHT key in the network and incoming `STORE` and `FIND_VALUE` requests. Although the adversary can't forge fake publications from the content creator as digital signing is protecting against this type of attack, they still can simply behave as black holes. They could only be replying to `PING` and `FIND_NODE` to advertise more malicious nodes, while ignoring all the storing requests and never replying any actual value objects.

Introducing variance in the creator's responding DHT key is helpful to mitigate this problem, as the original Kademlia's design prefers long live nodes over newcomers, and regularly changing the DHT key makes it harder for the attackers to infiltrating the network.

To truly facing the threat of this type of attacks, we can adopt a challenge-response authentication mechanism with using peer's public key and SHA1 hash to identify a peer's ID, thus significantly raises the difficulty of work for the adversary to forge any arbitrary peer ID.

5.2.2 DoS Attack

Another particularly hard to mitigate attack is the Sybil + DDoS Attack, as more distributed system design has to rely on personal computing nodes with relatively weaker computing resources, comparing to an actual experienced adversary's large scale botnet.

When introducing extra cryptographical verification steps into the distributed system, the designer need to pay extra care to the cost of work ratio between the normal serving nodes and a possibly malicious service consuming nodes. Ideally, The serving nodes should always pay much lesser effort during a verification process.

In the case of the proposed protocol. We specifically choose Ed25519 as the public key cryptography algorithm, which is very fast for verifying. Although in the future development and research of improvement, we could possibly make the querying node pay more effort by imposing more challenging tasks to help mitigate DDoS attacks.

Although this protocol can also be co-hosted by third party companies operates like Real Debrid, to provide "mirrors" on CDNs for the publishers using this protocol. But this goes back to the centralized aspect of the internet, and detailed discussion will not be expanded here.

5.2.3 Load Balancing

The concern of load balance and hotspot stressing in a DHT network can be addressed by introducing variability, for example, an extra "variance" field which also participates in the DHT key hashing procedure of the same data to the next version of protocol definition. By distributing the responsibility of storing data across multiple areas within the DHT, this approach reduces the likelihood of localized congestion. However, this strategy introduces additional complexity for subscribing peers, as it becomes more challenging to discover the updated keys corresponding to subsequent revisions.

In the current design, by dynamically altering the hash keys for each data revision or periodically updating them, it is already helping to prevent the consistent publications of a popular content creator from overloading a specific segment of the DHT network. Moreover, by leveraging the inherent caching mechanism of the Kademlia protocol for frequently queried keys ensures that the overall network swarm maintains robustness against potential stress caused by high demand.

This dynamic key variation mechanism also offers partial mitigation against Sybil attacks. By requiring adversaries to predict and mimic a much larger and dynamic portion of nodes in the DHT network with identifiers that are closest to the dynamically changing

hash keys of a creator's latest and future publications, the effort and resources needed to launch an effective Sybil attack are significantly increased. Although this approach alone does not provide complete protection, it imposes a considerable barrier compared to fixed hash key schemes. Adversaries must now continuously adapt to a broader range of potential DHT nodes, and the node routing table stability and old live nodes preference principle from the Kademlia network design also helps to make it more challenging to compromise the system.

5.2.4 Protocol detection and anti-censorship

This method doesn't rely on DNS name resolution to discovery other nodes and join the DHT network, and there are no fixed roles of servers. There exist no tough technique requirement for hosting a server, like owning domain, set up name resolution, owning fixed globally routable IPs. So it won't be affected by DNS pollution, IP blacklist, SNI sniffing attack, Domain ownership take down, or ISP termination of services. The most effective way to disturb the normal function of this system is to deploy deep UDP datagram identifying by analyze the datagram and test the by the characteristic of the datagrams of this protocol, and apply datagram filtering. Current BitTorrent and DHT implementations also has the same limitation, which makes them detectable and filterable by user ISPs.

This research relies on the fact that BitTorrent is often even more widely used in countries that conducts internet censorship, because the residents in these countries often need to retrieve large amount of data (often games, software, and videos with tools like BitTorrent, or otherwise they can't find and buy them in the country's market legally.

The massive use of technologies like BitTorrent, etc., hopefully would create collateral freedom [17] for our protocol as we can deeply integrate into BitTorrent protocol and clients and maintain our freedom by making censorship conductor worry more about blocking BitTorrent as a whole or take the computational resource to detect and filter the traffic of this protocol.

This could possibly be further mitigated by disguising the traffic as BT's encrypted download traffic. Although, the implementation and experiments will be left as one part of the future work.

In the meanwhile, users can still use any proxy and anonymity tools like tor or i2p or others they have on their hand to proxy just tiny bit of the traffic issued by this protocol's clients, and get back to exchange data with BT efficiently after they get updated metadata of the new content. And the publisher still benefit greatly from having a more free way to spread out their contents.

5.3 Future Work Directions

This work opens several promising avenues for future exploration and improvement. One potential direction involves the implementation of a mechanism analogous to BitTorrent's Peer Exchange (PEX) extension. Given that subscribers to a specific content creator are already engaged in downloading and seeding the same torrents, they inherently share a heightened interest in updates from the publisher. This mutual interest significantly increases the probability that these peers are aware of new content or updates earlier than randomly selected nodes or nodes encountered during traditional update propagation procedures. Leveraging this property could enhance the efficiency of content dissemination within the network.

Additionally, the proposed mechanism warrants further evaluation under more realistic networking conditions. This includes deploying testing clients on geographically

distributed Kubernetes swarms hosted on rental virtual private servers (VPS) to observe performance in real-world scenarios. These environments introduce challenges such as packet loss, firewall restrictions, NAT traversal, and other networking complexities. By conducting such tests, the robustness and adaptability of the system under diverse and dynamic conditions can be rigorously assessed. Furthermore, efforts could be directed towards optimizing the configuration of Linux kernel parameters and container engine settings. This optimization would facilitate more efficient resource utilization, enabling the hosting of larger-scale container swarms on single machines without significant degradation in performance.

Another important direction involves conducting a more comprehensive threat analysis to assess the network's resilience against malicious activities. For instance, the system's ability to withstand distributed denial-of-service (DDoS) attacks targeting a specific content creator's identity should be thoroughly examined. Detailed simulations should be carried out to test and demonstrate the effectiveness of proposed mitigation strategies, with an emphasis on collecting and analyzing empirical data to validate the results.

Lastly, distributed systems, including the one proposed in this work, face inherent security challenges. Addressing these challenges demands the exploration of advanced methodologies and innovative solutions. Future research could focus on broadening the scope of knowledge in this domain and developing more efficient and reliable mechanisms to mitigate potential vulnerabilities, thereby enhancing the overall security and robustness of the system.

By pursuing these directions, this work can be further refined, contributing to the broader understanding and advancement of distributed systems and peer-to-peer networks.

Chapter 6

Conclusion

This research introduces a novel distributed content subscription mechanism, which addresses the limitations of current decentralized content-sharing systems by combining the principles of public key cryptography, peer-to-peer networking, and distributed hash tables. This mechanism demonstrates an effective advancement in mitigating issues related to platform dependency, censorship, and anonymity while enhancing content availability and efficient transmission.

The contributions of this work are threefold. First, it extends the capabilities of the Kademlia DHT protocol by incorporating revision tracking and dynamic key variability, enabling robust content discovery even in the absence of the original publisher. Second, it effectively integrates with well-established peer-to-peer protocols, such as BitTorrent, ensuring scalability and compatibility, and collateral freedom with existing infrastructure while maintaining high transmission efficiency. Third, the system leverages public key cryptography to ensure the authenticity and integrity of content, providing a secure framework for creators and subscribers and mitigate attacks.

Through theoretical analysis and simulation-based experimentation, the efficacy of the proposed mechanism was validated under a simulated network environment involving 1,000 nodes. The results underscore the system's ability to maintain content availability, and support decentralized subscription functionalities, thereby offering a viable alternative to existing centralized and federated content-sharing solutions.

This research emphasizes the critical need for user-centric design in addressing challenges such as privacy, freedom of expression, and internet autonomy. The mechanism's design aligns with contemporary demands for decentralized content distribution, reflecting an intersection of technological innovation and ethical considerations.

Future work could extend this framework by addressing specific challenges highlighted during the research. For example, incorporating a peer exchange-like mechanism to optimize update dissemination, conducting real-world evaluations under diverse networking conditions, and performing comprehensive threat analysis to further enhance security. Additionally, adapting the system for large-scale deployments and exploring compatibility with emerging distributed ledger technologies could broaden its applicability and impact.

In conclusion, this thesis contributes a step forward in the pursuit of a decentralized internet ecosystem that prioritizes resilience, scalability, and user empowerment. By addressing critical gaps in existing technologies, the proposed mechanism paves the way for a more equitable and secure digital environment, embodying the principles of freedom and innovation that underpin the vision of a truly decentralized web.

Publications and Research Activities

(1) Zhihai Zhu, Ye Tao, Manabu Tsukada, Hiroshi Esaki. "A Distributed Content Subscription Mechanism with Revision Discovery to Decouple Content Sharing Platform and Creator ID." In 2025 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 2025.
(2) Zhihai Zhu, Nayuta Yamane, Naoya Tezuka, Experimenting Modern Secure Mail Service and DNS solutions, WIDE Project 2024.
(3) Zhihai Zhu, Nayuta Yamane, Shared GPU computing resource management and experiments with container application of job scheduling, reproducible build and container technologies, WIDE Project 2024.

References

[1] KN Hampton. Social networking sites and our lives. *Pew Research Center*, 2011.

[2] Chi Zhang, Jinyuan Sun, Xiaoyan Zhu, and Yuguang Fang. Privacy and security for online social networks: challenges and opportunities. *IEEE network*, 24(4):13–18, 2010.

[3] Matteo Zignani, Sabrina Gaito, and Gian Paolo Rossi. Follow the "mastodon" : Structure and evolution of a decentralized online social network. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 12, pages 541–550, 2018.

[4] Guido Cornelis Schipper, Rudy Seelt, and Nhien-An Le-Khac. Forensic analysis of matrix protocol and riot. im application. *Forensic Science International: Digital Investigation*, 36:301118, 2021.

[5] Martin Kleppmann, Paul Frazee, Jake Gold, Jay Graber, Daniel Holmgren, Devin Ivy, Jeromy Johnson, Bryan Newbold, and Jaz Volpert. Bluesky and the at protocol: Usable decentralized social media. In *Proceedings of the ACM Conext-2024 Workshop on the Decentralization of the Internet*, pages 1–7, 2024.

[6] Leonhard Balduf, Saidu Sokoto, Onur Ascigil, Gareth Tyson, Björn Scheuermann, Maciej Korczyński, Ignacio Castro, and Michal Król. Looking at the blue skies of bluesky. In *Proceedings of the 2024 ACM on Internet Measurement Conference*, pages 76–91, 2024.

[7] Bram Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72. Berkeley, CA, USA, 2003.

[8] Johan Pouwelse, Paweł Garbacki, Dick Epema, and Henk Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *Peer-to-Peer Systems IV: 4th International Workshop, IPTPS 2005, Ithaca, NY, USA, February 24-25, 2005. Revised Selected Papers 4*, pages 205–216. Springer, 2005.

[9] Yoram Kulbak, Danny Bickson, et al. The emule protocol specification. *eMule project, http://sourceforge. net*, 2005.

[10] Nancy K Baym. From practice to culture on usenet. *The Sociological Review*, 42(S1):29–52, 1994.

[11] Michele Tepper. Usenet communities and the cultural politics of information. In *Internet culture*, pages 39–54. Routledge, 2013.

[12] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[13] Erik Daniel and Florian Tschorsch. Ipfs and friends: A qualitative comparison of next generation peer-to-peer data networks. *IEEE Communications Surveys & Tutorials*, 24(1):31–52, 2022.

[14] Zhiyong Liu, Yueping Li, Qingfei Min, and Mengting Chang. User incentive mechanism in blockchain-based online community: An empirical study of steemit. *Information & Management*, 59(7):103596, 2022.

[15] Barbara Guidi. When blockchain meets online social networks. *Pervasive and Mobile Computing*, 62:101131, 2020.

[16] Peilin Zheng, Zigui Jiang, Jiajing Wu, and Zibin Zheng. Blockchain-based decentral-

ized application: A survey. *IEEE Open Journal of the Computer Society*, 4:121–133, 2023.

[17] David Robinson, Harlan Yu, and Anne An. Collateral freedom: A snapshot of chinese internet users circumventing censorship. *Open Internet Tools Project Report*, 2013.

[18] Martin E Hellman. An overview of public key cryptography. *IEEE Communications Magazine*, 40(5):42–49, 2002.

[19] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *International workshop on peer-to-peer systems*, pages 53–65. Springer, 2002.

[20] Liang Wang and Jussi Kangasharju. Measuring large-scale distributed systems: case of bittorrent mainline dht. In *IEEE P2P 2013 Proceedings*, pages 1–10. IEEE, 2013.

[21] Dennis Trautwein, Aravindh Raman, Gareth Tyson, Ignacio Castro, Will Scott, Moritz Schubotz, Bela Gipp, and Yiannis Psaras. Design and evaluation of ipfs: a storage layer for the decentralized web. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 739–752, 2022.

[22] Chao Li and Balaji Palanisamy. Incentivized blockchain-based social media platforms: A case study of steemit. In *Proceedings of the 10th ACM conference on web science*, pages 145–154, 2019.

[23] Juan Pablo Timpanaro, Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Bittorrent's mainline dht security assessment. In *2011 4th IFIP International Conference on New Technologies, Mobility and Security*, pages 1–5. IEEE, 2011.

Acknowledgements

I am deeply grateful to my supervisor, Prof. Hiroshi Esaki, for his invaluable guidance and inspiration throughout my academic journey. He has been a beacon of wisdom and patience, teaching me that research should always be driven by the needs of society rather than limited by our existing knowledge. Under his mentorship, I ventured into the emerging fields of Decentralized Networking, Cybersecurity, and DevOps, where I explored topics such as p2p networking, network infrastructures, and wireless networking. Prof. Esaki' s vision encouraged me to tackle some of the most pressing challenges in these areas. His kindness, politeness, and unwavering support have profoundly shaped my experience. He generously provided numerous opportunities for his students to test their knowledge and skills, opening doors to possibilities I never imagined.

I owe a tremendous debt of gratitude to my mentor, Assoc. Prof. Hideya Ochiai, who has been an indispensable source of support since my first year at The University of Tokyo. His concern for my well-being, particularly during difficult times due to geopolitical challenges and personal health issues, has been both reassuring and motivating. He guided me toward discovering my research direction and making informed decisions, demonstrating not only his brilliance as a researcher but also his deep care as a teacher and friend. His commitment to sustainable development goals (SDGs) and anti-bias movements is truly inspiring. I am especially grateful for our candid and enriching conversations, often held in both Japanese and English, including memorable discussions during the summer seminar camp trip. These moments were pivotal in helping me adapt to Japanese customs and build meaningful connections within the community.

My heartfelt thanks also go to Assoc. Prof. Manabu Tsukada, who actively monitored my academic progress and provided unwavering support during my challenging times. His guidance, particularly when I faced difficulties in the final semester, helped me regain my focus and motivation. Through his seminars and laboratory, I learned invaluable methodologies for conducting academic research. His dedication to nurturing a laboratory environment that is inclusive, diverse, and resource-rich for international students is truly commendable. His open-mindedness and determination to provide equal opportunities have left a lasting impact on me, both as a student and as a researcher.

I would also like to express my sincere appreciation to Dr. Ye Tao, whose mentorship has been instrumental since the beginning of my first publication. Dr. Tao patiently spent countless hours discussing and teaching me the intricacies of academic research and writing, from identifying relevant literature to his clear and compelling explanatory figures and charts crafting technics. His feedback, always objective and constructive, helped me refine my ideas and verify their theoretical foundations. Additionally, he introduced me to the world of DevOps and the challenges of large-scale, server-dedicated tasks, broadening my perspective on research applications. I will forever cherish the memories of working with him and learning from his wisdom.

I extend my heartfelt thanks to all the members of Esaki, Ochiai, and Tsukada laboratories for fostering a stimulating and collaborative environment. I learned immensely

from our discussions and experiments together. I am especially grateful to Kengo Suzuki for his support in my Japanese student life, experiments, and job hunting, and to Naoya Tezuka for his guidance on paper writing, Japanese culture, and accessing past years' learning resources. My thanks also go to Nayuta Yamane for his technical expertise and his invaluable assistance with network-related experiments and operations.

My gratitude extends to my friends from around the world, whose inspiration and advice have been invaluable to my academic progress. I am particularly thankful to Yu Wang, a geologist and computer scientist from Utrecht University, for his encouragement to bring me back to academia and pursue professional research. His introductions to academic conventions and resources were indispensable.

Finally, I wish to convey my deepest love and appreciation to my parents and family members. Their constant strength and support, despite facing their own challenges, have been the foundation of my journey. I dedicate this thesis to them with all my heart.

January 2025