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

Zhihai Zhu
Graduate School of Information Science and Technology, The University of Tokyo
zhihai@hongo.wide.ad.jp

Ye Tao
Graduate School of Information Science and Technology, The University of Tokyo
tydus@hongo.wide.ad.jp

Manabu Tsukada
Graduate School of Information Science and Technology, The University of Tokyo
mtsukada@g.ecc.u-tokyo.ac.jp

Hiroshi Esaki
Graduate School of Information Science and Technology, The University of Tokyo
hiroshi@wide.ad.jp

***Abstract*—This paper proposes a distributed content subscription mechanism that enables content creators to share updates with their audience while maintaining platform independence and anonymity. The mechanism extends the Kademlia distributed hash table (DHT) protocol by incorporating revision numbers and republication timestamps into the DHT key computation, allowing subscribers to discover content updates through heuristic revision queries. It leverages public key cryptography for creator identification and content authenticity, while integrating with established peer-to-peer protocols like BitTorrent for efficient content distribution. Preliminary testing with 200 simulated nodes demonstrates the mechanism's ability to maintain content availability and update discovery even when content creators are offline. This approach particularly benefits creators operating under strict content controls or surveillance, offering them greater creative freedom and distribution autonomy compared to existing centralized and decentralized solutions.**

***Index Terms*—decentralization, DHT, subscription, content sharing, asymmetric cryptography**

I. Introduction

Social networks and user-generated content (UGC) platforms are central to modern online life [1]. However, growing control over data collection, recommendation algorithms, and content moderation by corporations and governments raises concerns about diminishing internet freedom and threats to user privacy, anonymity, and autonomy [2]. Decentralized technologies, such as Mastodon [3], Matrix [4], BlueSky [5], [6], IPFS [7], [8], and Steemit [9]–[11], aim to address these issues by reducing dependence on centralized platforms. While these tools offer valuable alternatives, they often fall short of meeting the diverse needs of all user groups, particularly in contexts requiring high transmission efficiency and more speech freedom [12], [13]. This is critical for groups like independent media creators in conflicting districts and activists silenced by political constraints.

This study proposes a fully distributed content sharing and subscription mechanism that emphasizing platform independence, anonymity, high availability, and efficient transmission to address the above challenges. Leveraging digital signatures as creator identities, the mechanism extends the Kademlia protocol with revision numbers and timestamps for decentralized content subscription. Subscribers discover updates via concurrent queries, while content distribution integrates with protocols like BitTorrent [14], [15] for efficient delivery. This approach aims to surpass existing technologies in meeting specialized modern day decentralized content distribution needs.

The rest of the paper is organized as follows. Section II introduces the related work. Section III shows the detailed description of the designed protocol, and the reasoning behind each design choice. Section IV recalls several related distributed solutions and makes comparison with our design. Section V discussed several potential attack models to consider, like Sybil Attack, Protocol detection. Finally, section VI concludes the paper, mentioning the advantage of the proposed method and suggested future work direction about conducting more rigorous emulation and analysis of DoS attacks and mitigations.

II. Related Works

A. Public Key Cryptography

Public key cryptography [16] uses a pair of linked keys—a public key, shared openly, and a private key, kept confidential—for secure communication and authentication. Messages encrypted with the public key can only be decrypted by the private key, and vice versa, ensuring message integrity and authenticity. Digital signatures use the private key to create a unique signature for data, verified with the public key to confirm the signer's identity and detect any tampering. This study leverages these properties to verify content creator identities and ensure the authenticity of updates, with challenge-response authentication for validating DHT peer identities.

B. Distributed Hash Table

A distributed hash table (DHT) is a decentralized structure for efficient key-value lookups across a network. Among DHT designs, Kademlia [17], which uses XOR metrics for routing and redundancy to handle failures, is widely used, including in BitTorrent's mainline DHT [18]. While BitTorrent's DHT optimizes file sharing by mapping torrent info hashes to peer lists, its restrictions limit extensibility. This study focuses on extending Kademlia for enhanced performance and flexibility, for use in a subscription mechanism.

C. Peer-to-Peer Data Sharing

BitTorrent [14], [15] and IPFS [7], [8] are P2P protocols designed for efficient content sharing. BitTorrent fragments files into pieces shared among peers, while IPFS uses a content-addressed system to locate data by its hash. Both protocols ensure data integrity and distribute loads effectively. However, they lack decentralized subscription mechanisms for tracking and subscribing to content updates without relying on centralized servers. This research aims to resolve this limitation, integrating subscription capabilities into existing infrastructures to support creators under censorship or platform control.

D. Blockchain-Based Platforms

Platforms like Steemit [9]–[11], BrokerChain [19], TRUCON [20] and Odysee/LBRY [21] use blockchain technology [11] for decentralized content sharing, integrating monetization through tokens rewarded for user engagement. However, these systems face decentralization challenges due to dominance by large shareholders and misuse of reward mechanisms. Issues like storing unwanted content and bot abuse further undermine their goals [13]. These shortcomings reveal the need for improved consensus protocols and better-designed systems for future blockchain-based platforms.

E. Federated Social Platforms

Mastodon [3], based on the ActivityPub protocol [5], enables federated social networking through interconnected servers, allowing user autonomy and localized content moderation. BlueSky [5], [6] is a similar technology aims to improve on Mastodon's data model. While federated models improve resilience and privacy compared to centralized platforms, they remain vulnerable to batch IP blacklisting and censorship tactics like SNI sniffing attacks [12]. This underscores the advantage of fully distributed systems over federated ones in addressing these vulnerabilities.

III. System Description

The proposed mechanism builds upon three well-established distributed and cryptographic technologies: public key cryptography, the Kademlia network, and BitTorrent, as demonstrated in Fig. 1. The subsequent section outlines the rationale for incorporating these technologies, details the methodology for their integration into the protocol, defines the message and remote procedure calls (RPC) format used in this protocol, and explains the interactions allowed by the protocol, see Fig. 2. Finally, it describes the dynamic behavior of the system during operation.

A. Design Priciple

1) Leveraging Asymmetric Cryptography: Our proposed mechanism will first leverage cryptographic principles, particularly the ability to verify private key ownership and data authenticity and integrity with digital signature [16].

Secure validation of peer identity is achieved through challenge-response authentication, where any party with the knowledge of the public key can confirm ownership of the corresponding private key. 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. This is to help mitigate Sybil attacks [22] against the DHT network.

Furthermore, a private key owner can generate a digital signature for theoretically any data using the private key. Other parties, using the corresponding public key, can then validate the integrity of the data by verifying the signature, and also identify that it comes from the desired content creator.

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) [17]; yet, conventional designs typically map the SHA1 hash back to associated data.

In this proposal, 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.

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 the "Threat Analysis" section.

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

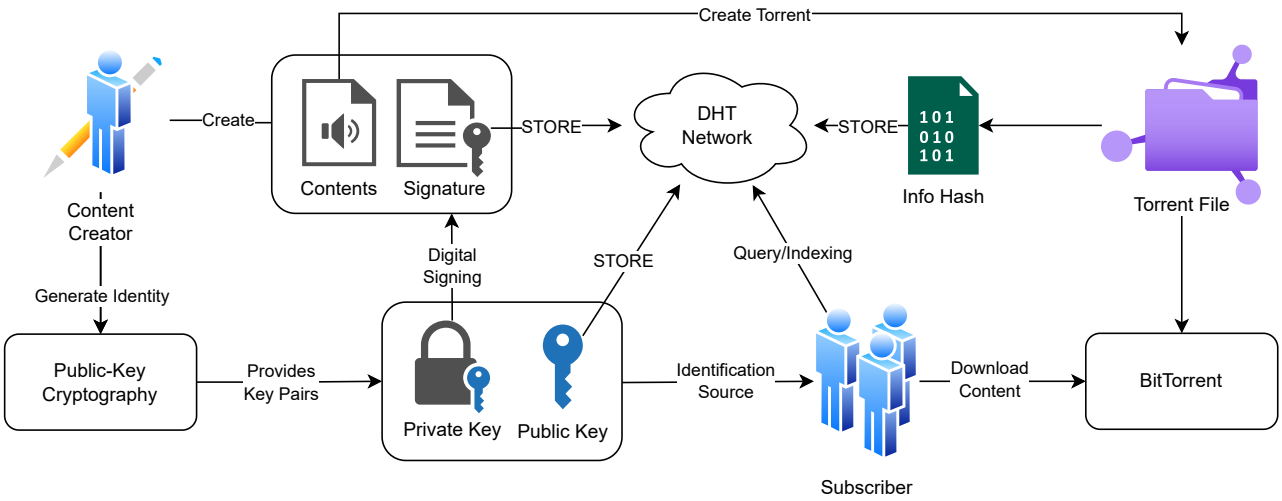

Fig. 1. A conceptual overview of the subscription mechanism and the foundational technologies.

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.

B. Architecture

1) Cryptographic Keys and Signing: Our protocol requires all participating peers to generate cryptographic key pairs and securely store their private keys for use across online sessions. 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 DHT network, they must include the data, its digital signature, and the corresponding public key to enable verification by other participants.

2) Extending Kademlia Network: In the proposed design, we modify the original Kademlia network's key and value definition, along with the RPC commands, for predictable creator's content update indexing in a distributed manner. We also introduced challenge response authentication to provide peer ID validation against Sybil [22] and black hole attacks.

3) Incorporate BitTorrent: The well-established BitTorrent protocol, along with its mainline-DHT [18] design and metadata exchange extension, provides a highly efficient method for distributed data transmission. However, it lacks a fully distributed mechanism for subscribing to updates. By providing a trustable content subscription method, our proposed system can utilize BT's ability to achieve fast content distribution.

C. Protocol Definition

The protocol specifies that a key is derived as a hash of specific 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 RPC from the original Kademlia network, and provides guidelines for the behavior client software's workflow.

1) Value Definition: 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 by the content creator's private key. The creator should concatenate the revision integer with the `data_payload` and sign the result. Including the revision is critical to prevent unauthorized revision number modified values being used to overwrite data from the same creator with other revisions in the DHT.

Any values that cannot be verified using the public key, will be deemed invalid. `STORE` requests for such values should be rejected. Peers that consistently issue invalid `STORE` requests may subject to blacklisting.

It is important that the `republish_date` field is not included in the signed data to support co-seeder republish-

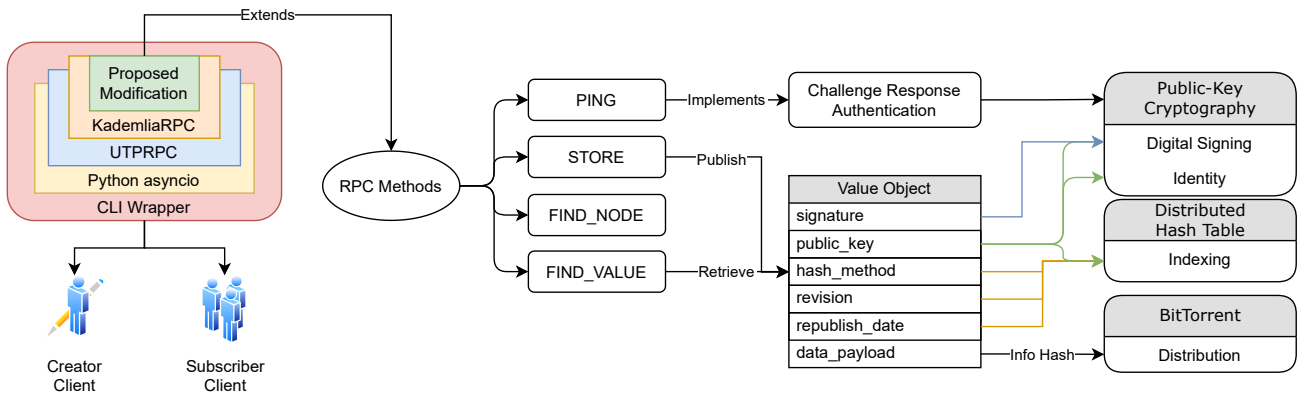

Fig. 2. The architecture overview of the proposed distributed subscription system's demonstration implementation.

ing mechanism, detailed later in the "Subscribe Workflow" section.

2) Key Definition: The DHT key of 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`. A `hash_method` value of 0 indicates that the DHT key should be derived by hashing the `public_key` concatenated with the `revision`, while a value of 1 indicates the `public_key` to be concatenated with the `republish_date`.

3) RPC Commands: `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 as the final response. All peers will be required to complete the challenge before being saved into other peers' routing table.

`STORE`: The `STORE` command is extended in two ways:

For the sending peer, it first compute the appropriate DHT key as required by the "Key Definition" part to determine the location within the DHT network for storing the value object. It must 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.

For the receiver, upon receiving a `STORE` request, it must validate the value object as specified in the "Value Definition" section, and reply invalid request with an error. It then computes the corresponding DHT key, and check for key collision in its internal storage. As much as we'd like to adopt more modern hash functions other than the already broken SHA1 [23], for demonstration purpose, we will stick to BitTorrent's current choice. If intentional collision attack is a threat, implementations can use securer hash functions or multimap as the data storage.

If a collision occurs, the receiver checks whether the `public_key` field matches any stored values. If the value uses date-specific hash and shares all fields except for the revision number and the data payload with the stored value, the receiver should replace the stored value if the incoming one is newer. This is to support the co-republish mechanism, discussed later in the "Subscribe Workflow" section.

The receiver should then 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.

`FIND_NODE`: Comparing to Kademlia protocol, nodes should exchange and store each other's public key instead of the randomly generated peer IDs with the `FIND_NODE` command. To enhance security, the sender of must send a `PING` RPC for peer ID validation prior to every `FIND_NODE` query if the queried node has not been validated, and only accept results when the validation succeeds. Additionally, the sender must respond to the recipient's validation challenge for its peer information to be recorded by the recipient.

`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 mush respond with values that satisfy all specified criteria. If no such value is known, the receiver must fall back to correspond as if it receives a `FIND_NODE` query.

Upon receiving a value reply, the RPC sender verifies that the returned value before collecting the result. If the 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 API invoker.

4) Publisher Workflow: The content creator should first generate two pairs of asymmetric cryptographic keys. The first pair is designated for peer ID validation, while the second is used for signing. The creator then distribute their signing public key to their audiences.

When the publisher creates new content, they should generate a torrent file for the content, and initiate seeding. Subsequently, their client software should generates two DHT value objects, with the info hash of the torrent to be the `data_payload` together with other metadata of this publication, and the only difference being `hash_method`. The client then publishes these value objects to the DHT network.

When publishing new content, the publisher's client must repeat the previous process but instead with the updates to the `revision`, `republish_date`, `data_payload`, and `signature`.

5) 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 "Key Definition" section. Typically, 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.

Upon receiving the latest update, subscribers can start downloading content identified by the `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, just like the original Kademlia design. Additionally, now peers are also allowed to discard value object older than 7 days.

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.

D. Concept Verification

A simulation was implemented in Python, leveraging the asynchronous event-driven capabilities of the `asyncio` library to enable efficient concurrent network messaging and remote procedure calls. Initially, a UDP-based RPC protocol was constructed atop the `asyncio.DatagramProtocol`. This protocol was then extended to include the four RPC methods suggested in the proposal. The implementation also included a mapping data structure with timeout-eviction, two network crawlers representing node and value iterative query strategies respectively, and a high-level interface class for control and data processing. These components collectively formed a demonstration client, which was subsequently encapsulated in a Docker image.

For testing, 200 client instances were deployed as intermediate nodes in the DHT network, each hosted in a separate container with a unique IPv4 address from the same user-defined network. Three additional client containers were configured with specific roles: one acted as the content creator's client, another as the co-seeder's client, and the third as a newly joining subscriber's client. The creator's client was scripted to sign and publish updates at a predefined time schedule before terminating. The co-seeder's client continuously queried for updates, performed republishing as specified, and downloaded and seeded the updates. Observations confirmed that the subscriber's client, launched later, successfully discovered and downloaded the latest published content, even after the original creator's client had gone offline for an extended period.

IV. Comparison

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 which attempts to address these challenges, 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 and IPFS [7], [8], exhibit limitations due to the absence of a distributed

TABLE I
Comparison of different distributed solutions across various requirements. H: High, M: Medium, L: Low, N: No, Y: Yes.

Properties	BT	Mastodon	Steemit	YouTube	Proposal
Freedom	H	M	H	L	H
Anti-Censorship	H	M	M	L	H
Transfer Speed	H	M	L	H	H
Anonymity	H	M	H	L	H
Subscribe	N	Y	Y	Y	Y

subscription mechanism. These systems rely on tracker sites to provide updated metadata, which are vulnerable to shutdowns, often for legal or copyright-related reasons [24].

Blockchain-based solutions like Steemit [9]–[11] 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 [13].

Federated solutions, such as Mastodon 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.

V. Threat Analysis

In this section, we will briefly discuss some security concerns of this mechanism.

A. Sybil Attack

The Sybil [25] and black hole attack threat is when the opponent targets a specific content creator's the DHT key, and fire up large amount of clients with peer ID crafted very close to the target DHT key, effectively controlling the vicinity of that DHT key in the network and incoming `STORE` and `FIND_VALUE` requests. The opponent can simply behave as black holes, 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.

And pairing challenge response authentication with using peer's public key and SHA1 hash to identify a peer's ID, the Sybil attack could be well mitigated.

B. Protocol detection and anti-censorship

This method avoids reliance on DNS name resolution and fixed server roles, eliminating vulnerabilities like DNS pollution, IP blacklisting, domain takedowns, or service terminations. While susceptible to deep packet inspection and filtering, similar to BitTorrent and DHT implementations, it leverages the widespread use of BitTorrent, particularly in censored regions where large data transfers are common. By integrating with BitTorrent protocols and clients, the system aims to achieve "collateral freedom" [26], making censorship more challenging without broader disruptions.

VI. Conclusion

This paper presents a distributed content subscription mechanism designed to empower content creators to share updates with their audience while ensuring platform independence and anonymity. This solution offers advantage for creators facing strict content controls or surveillance, providing enhanced creative freedom and autonomy compared to existing centralized or decentralized platforms.

Future work will involve testing the proposed mechanism in realistic network environments, and conducting rigorous threat analysis and revise the design of `PING` and `STORE` RPC against DoS attacks targeting content creators with e.g. client puzzle method, and validating the effectiveness of mitigation measures through simulations and data analysis.

References

[1] Keith N. Hampton, Lauren Sessions Goulet, Lee Rainie, and Kristen Purcell. *Social networking sites and our lives*, volume 1. Pew Internet & American Life Project, Washington, DC, 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. *arXiv preprint arXiv:2402.03239*, 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] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[8] 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.

[9] 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.

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

[11] Peilin Zheng, Zigui Jiang, Jiajing Wu, and Zibin Zheng. Blockchain-based decentralized application: A survey. *IEEE Open Journal of the Computer Society*, 4:121–133, 2023.

[12] Aravindh Raman, Sagar Joglekar, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. Challenges in the decentralised web: The mastodon case. In *Proceedings of the Internet Measurement Conference*, pages 217–229, 2019.

[13] 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.

[14] Bram Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer Systems*, volume 6, pages 68–72, 2003.

[15] 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 Berlin Heidelberg, 2005.

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

[17] 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, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[18] 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.

[19] Huawei Huang, Xiaowen Peng, Jianzhou Zhan, Shenyang Zhang, Yue Lin, Zibin Zheng, and Song Guo. Brokerchain: A cross-shard blockchain protocol for account/balance-based state sharding. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 1968–1977. IEEE, 2022.

[20] Mingyang Yuan, Yang Xu, Cheng Zhang, Yunlin Tan, Yichuan Wang, Ju Ren, and Yaoxue Zhang. Trucon: blockchain-based trusted data sharing with congestion control in internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 24(3):3489–3500, 2022.

[21] Jun Li, Alex Grintsvayg, Jeremy Kauffman, and Charles Fleming. Lbry: A blockchain-based decentralized digital content marketplace. In *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 42–51. IEEE, 2020.

[22] 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.

[23] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full sha-1. In *Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part I 37*, pages 570–596. Springer International Publishing, 2017.

[24] 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.

[25] Srivatsan Sridhar, Onur Ascigil, Navin Keizer, François Genon, Sébastien Pierre, Yiannis Psaras, Etienne Rivière, and Michał Król. Content censorship in the interplanetary file system. *arXiv preprint arXiv:2307.12212*, 2023.

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