ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
School of Information Sciences and Technology
Department of Informatics

Navigating Security Challenges in Web3: From Ethereum Smart-Contracts to IPFS Content

A dissertation submitted in fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

by

Christos Karapapas

Athens
October 2025

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I feel truly fortunate and thankful to have had Professor George C. Polyzos as my advisor. A person of deep empathy and a genuine lead-by-example character, he was not only instrumental in the completion of this dissertation but also one of the most remarkable individuals I have had the privilege to meet. His guidance and feedback throughout my doctoral studies were always insightful, fair, and invaluable.

I would also like to thank Professor Vasilios A. Siris, Assistant Professor Spyridon Voulgaris, and Professor George Xylomenos, who were always willing to advise me, share their experience, and extend their genuine kindness throughout our collaboration. I would also like to express my gratitude to Associate Professor Constantinos Patsakis. His contribution in transforming my early ideas into mature, complete research works was fundamental, as was his overall support. I would also like to thank Professor Panagiotis Katerinis and Associate Professor Stavros Toumpis for the excellent collaboration we had over the years in T.A. duties.

As a new member of the MMlab, I was fortunate to meet and collaborate with a number of experienced researchers. I would like to thank Dr. Nikos Fotiou, whose professionalism and research insight have greatly influenced my work, as well as for our collaboration that extended from my master's studies through the completion of my doctoral research. I would also like to thank Dr. Livia Chatzieleftheriou, Dr. Merkouris Karaliopoulos, and Dr. Yannis Thomas. Although our collaboration was not extensive, their presence and contributions to the MMlab significantly elevated the quality of the lab—and, in turn, my own work. As a member of the MMlab, I also had the opportunity to collaborate with several younger researchers. A special acknowledgment goes to my colleague and friend, Dr. Iakovos Pittaras. Our collaboration, both within and outside the lab, was a constant source of support—free of any sense of competition and always driven by our shared goal of becoming better researchers and individuals. I would also like to thank Yannis Papageorgiou, Alexandros Antonov, and Vasilis Kallos. Their presence in the lab was always a pleasure, and I wish them the best of luck in all their future endeavors.

I am deeply grateful to my family for their unwavering support throughout these years—Nikos Karapapas, Maria Tzioti, and of course, my brother Kyriakos Karapapas, who was always there to listen, share my concerns, and offer advice. I am deeply grateful to my life companion, Vasso Lirantzi, for her understanding, empathy, and for grounding me whenever I lost my focus during this long journey.

I would also like to thank my friends, and especially Sotiris Polychronis, who—despite having no idea what I was talking about—patiently listened to all my excited ramblings about a new idea or experiment. Finally, I would like to thank *webphreaker* and *Nu11_Gr4m*—my younger selves who first explored code, curiosity, and creativity in the early digital playgrounds that shaped me. Without them, this dissertation would not exist.

VITA

2012	B. Sc. in Mathematics, University of Athens, Greece
2016	M. Sc. in Information Systems, Athens University of Economics and Business, Greece
2025	Ph. D. in Computer Science, Athens University of Economics and Business, Greece

PUBLICATIONS

Journal Publications

I. Pittaras, N. Fotiou, C. Karapapas, V. A. Siris, and G. C. Polyzos, "Secure smart contract-based digital twins for the Internet of Things," in *Elsevier Blockchain: Research and Applications*, vol. 5, no. 1, 2024

Refereed International Conference and Workshop Papers

C. Karapapas, I. Pittaras, G. C. Polyzos and C. Patsakis, "Hello, Won't You Tell Me Your Name?: Investigating Anonymity Abuse in IPFS," in *Proceedings of the 13th International Workshop on Cyber Crime (IWCC), in conjunction with the 19th International Conference on Availability, Reliability and Security (ARES)*, Ghent, Belgium, August 2025

T. Katsantas, Y. Thomas, C. Karapapas, G. Xylomenos, "Enhancing IPFS privacy through triple hashing," in *Proceedings of the 29th IEEE Symposium on Computers and Communications (ISCC)*, Paris, France, June 2024

C. Karapapas, G. Xylomenos, and G. C. Polyzos, "Enhancing IPFS Bitswap," in *Proceedings of the International Conference on Information and Communication Technology for Intelligent Systems(ICTIS)*, Las Vegas, USA, May 2024

C. Karapapas, G. C. Polyzos, and C. Patsakis, "What's inside a node? Malicious IPFS nodes under the magnifying glass," in *Proceedings of the 38th International Conference on ICT Systems Security and Privacy Protection (IFIP SEC)*, Poznan, Poland, June 2023

C. Karapapas, G. Syros, I. Pittaras, and G. C. Polyzos, "Decentralized NFT-based Evolvable Games," in *Proceedings of the 4th Conference on Blockchain Research and Applications for Innovative Networks and Services (BRAINS)*, Paris, France, September 2022

I. Pittaras, N. Fotiou, C. Karapapas, V. A. Siris, and G. C. Polyzos, " Secure, Mass Web of Things Actuation Using Smart Contract-based Digital Twins," in *Proceedings of the 27th IEEE Symposium on Computers and Communications (ISCC)*, Rhodes, Greece, June 2022

C. Karapapas, I. Pittaras, and G. C. Polyzos, "Fully Decentralized Trading Games with Evolvable Characters using NFTs and IPFS," in *Proceedings of the Workshop on Decentralizing the Internet with IPFS and Filecoin (DI2F)*, Espoo, Finland, June 2021

C. Karapapas, I. Pittaras, N. Fotiou, and G. C. Polyzos, "Ransomware as a Service using Smart Contracts and IPFS," in *Proceedings of the 2nd IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, Toronto, Canada, May 2020

ABSTRACT OF THE DISSERTATION

Navigating Security Challenges in Web3:
From Ethereum Smart-Contracts to IPFS Content

by

Christos Karapapas

Doctor of Philosophy in Computer Science

Athens University of Economics and Business, Athens, 2025

Professor George C. Polyzos, Chair

Web3 is not merely a technological evolution but a radical shift with deep philosophical implications. It envisions an Internet where users are at the center, yet without a centralized authority. Its goal is to ensure that users have full ownership of the data they generate, as well as the value—whether economic or informational—that arises from it.

This paradigm shift finds applications in various fields, such as Decentralized Storage, which enables secure, distributed, and censorship-resistant data storage solutions; Non-Fungible Tokens (NFTs), which guarantee ownership and authenticity of digital assets; and Decentralized Gaming, which leverages blockchain

technology to enable player-owned economies, provable asset scarcity, and transparent game mechanics.

These, along with numerous other innovations, are shaping the next era of the Internet. As expected, Web3 has drawn the attention of researchers who are striving to establish it from the ground up using emerging, still-maturing technologies. Naturally, this has also caught the eye of malicious actors, who take advantage of the novelty and complexity of these interconnected systems for their own gain. The challenge now is to ensure that security advances in parallel with the growth of the Web3 ecosystem, preventing it from becoming an unstable or hostile environment.

The contribution of this dissertation to this effort is multifaceted. First, we study the literature on Ethereum blockchain, NFTs, and the Interplanetary File System (IPFS), which serves as a cornerstone of the Web3 data storage layer. Our goal is to identify vulnerabilities and analyze whether – and to what extent – malicious activity exists. Next, from the perspective of malicious actors, we anticipate how they might exploit these technologies. We document potential attack vectors, making them easier to detect and mitigate. Finally, we propose design improvements that enhance the availability and scalability of key Web3 application-layer services, laying the groundwork for more scalable, resilient, and future-proof decentralized applications.

Chapter 1

Introduction

The Web3 concept has attracted significant attention in recent years. One of the main attractions of Web3 is an emphasis on returning control of data to their owners, empowering them to determine who can access their data and enabling monetization of the information they generate. Central to achieving these objectives are Distributed Ledger Technologies (DLTs) along with token based economics. Web3 also envisions decentralized services that cater to the Internet of Things (IoT) era, integrating cryptocurrencies to facilitate peer-to-peer (P2P) financial interactions and digital value exchange. Web3 is often conceptualized as comprising different stacks, each consisting of various protocols that collaborate to deliver services to users. These protocols cover areas such as data storage, name resolution, Decentralized Identifiers (DIDs), and, at a higher level, services like social media, gaming, and marketplaces. The growing adoption of Web3 is reflected across multiple domains, including NFT marketplaces, gaming, the Metaverse, lending, and investment platforms. The total number of users engaging with Web3 services already exceeds two million [1], underscoring its increasing impact on digital economies and decentralized infrastructures. Looking ahead, recent projections estimate that Web3 will reach approximately one billion users by 2027 [2], while its market size is expected to grow from $7.23 billion in 2025 to over $42 billion by 2030.[1]

Blockchain was born in 2008 with the release of the Bitcoin whitepaper [3]

[1] `https://www.mordorintelligence.com/industry-reports/web-3-blockchain-market`

and serves as the foundation of the Web3 paradigm. It is a ledger of blocks cryptographically linked together. While its primary application has been in cryptocurrency technology, in recent years, blockchain has expanded into various other fields, including voting systems, supply chain management, the Internet of Things (IoT), and even healthcare. Furthermore, cryptocurrencies have evolved technologically, integrating with smart contracts to facilitate smoother interactions between these emerging technologies. As of January 2025, there are over 10,000 different cryptocurrencies,[2] reflecting their growing popularity and increasing integration into everyday applications. While the vast number of cryptocurrencies could be perceived as a challenge rather than a weakness, as it creates a highly fragmented market and increases the difficulty for users to identify reliable assets, it does not hinder the dominance of major cryptocurrencies. In fact, the top 20 cryptocurrencies account for nearly 90% of the total market capitalization, demonstrating their strong establishment in the financial ecosystem, with the total crypto market cap reaching $2.8 trillion.[2]

Data storage is another crucial component of the Web3 ecosystem, enabling decentralized, secure, and tamper-resistant storage solutions through technologies such as InterPlanetary File System (IPFS), Filecoin, Storj, SIA, and others. Among these, IPFS stands out as one of the most significant protocols in decentralized storage. Developed by Protocol Labs[3] as an open-source project, it has gained considerable attention in recent years. Notably, in January 2024, the Filecoin Foundation, in collaboration with Lockheed Martin, successfully deployed IPFS in space by transmitting data to and from an orbiting satellite using a space-adapted version of the protocol.[4] Beyond high-profile experiments, IPFS is seeing widespread adoption, with more than 3 million web client accesses and over 300,000 unique nodes serving content in the P2P network every week [4]. Furthermore, the growing number of research papers with "IPFS" in their titles highlights its increasing prominence among researchers, with the Semantic Scholar

[2] https://www.statista.com/statistics/863917/number-crypto-coins-tokens/

[3] https://www.protocol.ai/

[4] https://www.lockheedmartin.com/en-us/news/features/2024/smartsat-equipped-satellite-uploads-new-mission-on-orbit.html

tool returning more than 800 results for such publications over the past two years.

These advancements highlight the rapid evolution of Web3 technologies and their growing influence across multiple industries. However, despite the potential benefits, Web3 still faces significant scalability, security, and adoption challenges. As decentralized ecosystems expand, new vulnerabilities emerge, raising concerns about data integrity, privacy, and regulatory compliance. Addressing these challenges is crucial to ensuring a stable and sustainable Web3 infrastructure. Given these complexities, the following section delves into the motivation behind this research, exploring the key challenges that Web3 must overcome to fulfill its vision of a decentralized and user-centric digital world.

1.1 Motivation for the dissertation

Web3 comes with very ambitious promises for the technological future of the Internet, which explains its mass adoption and the growing interest from both the public and researchers. However, like every coin has two sides, Web3 has characteristics that, if misused, can become disadvantages. One of the most critical challenges Web3 faces is the lack of oversight and regulatory authority, a factor that has already led to significant financial losses in cases of fraud and unregulated protocols. The absence of regulatory frameworks makes it difficult to prevent illicit activities, increasing the risks for users. Additionally, most Web3 components rely on peer-to-peer (P2P) networks, which, while promoting decentralization, also create fertile ground for privacy leakage. The anonymity promised by Web3, though intended to ensure fairness and equality, has also been leveraged for illicit activities, such as money laundering, fraud, and darknet transactions. This raises concerns about the balance between privacy and security in decentralized environments. From a technical perspective, the immaturity of many Web3 technologies makes them vulnerable to frequent security breaches. With most blockchain-based applications still in experimental stages, attacks exploiting smart contract bugs and consensus mechanism weaknesses are increasingly common. Even Web3's open-source philosophy, meant to foster transparency and community engagement, has

been turned against itself. Malicious actors have leveraged publicly available code to orchestrate attacks and exploit vulnerabilities, raising the question of whether complete transparency can coexist with security in a decentralized ecosystem. This paradox underscores the ongoing debate between openness and protection in Web3 development.

Blockchain serves as the foundational technology of Web3 and is integral to the functionality of cryptocurrencies. It offers features such as transparency, decentralization—which mitigates single points of failure—data integrity safeguarded by cryptographic methods, and transaction anonymity. However, these attributes have also been exploited for malicious purposes. For instance, ransomware attackers commonly demand payments in cryptocurrencies to maintain their anonymity. The immaturity of certain blockchain technologies has led to significant security breaches; a notable example is "The DAO hack" in 2016, where vulnerabilities in smart contract code resulted in the theft of approximately $60 million worth of Ether.[5] Beyond financial transactions, blockchain has also been misused as a communication channel and data storage medium by malware authors [5]. These instances highlight the dual-edged nature of blockchain technology: while it provides robust solutions for secure and decentralized applications, it also presents new vectors for cyber threats that necessitate vigilant security measures.

Decentralized storage is also a key component of Web3, as it aims to replace or complement the functionality of major cloud services. So far, there have been many contenders for this role, with the InterPlanetary File System (IPFS) emerging as the most prominent solution. Its application spans various fields, with NFT storage being one of the most significant use cases, contributing to its widespread recognition and popularity. Due to its extensive daily usage, many traditional Web 2.0 companies have found ways to integrate or leverage decentralized storage solutions. However, as expected, this popularity has also attracted the attention of malicious actors. For instance, the Storm botnet has been reported to use the IPFS network for malicious purposes [6], and there have been multiple phishing attacks exploiting the system.[6] Furthermore, the P2P nature of IPFS makes it

[5] https://www.gemini.com/cryptopedia/the-dao-hack-makerdao

[6] https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/ipfs-the-

susceptible to privacy leakage. Numerous reports indicate that a user's identity or activity within the network can become exposed, raising concerns about data confidentiality [7]. Thus, it becomes evident that there is a pressing need to study how malicious actors exploit IPFS, develop countermeasures against its misuse, and enhance its privacy mechanisms to ensure more secure decentralized storage solutions. As decentralized storage solutions like IPFS continue to evolve, their integration with blockchain technology further complicates the security landscape. While blockchains ensure immutability and decentralized trust, they alone cannot handle large-scale data storage, leading to the growing reliance on external decentralized storage networks. However, this interdependence also introduces new security risks, reinforcing the need for a holistic approach to securing the Web3 ecosystem.

The application layer of Web3 has seen significant growth, with NFTs and decentralized gaming playing a major role in its expansion. These applications have also given rise to a new economic model known as play-to-earn (P2E), where players can generate real-world value through in-game assets and interactions. They leverage blockchain technology for transparency, immutability, and programmability, while also relying on decentralized storage solutions to ensure persistent and tamper-resistant data availability. The integration of these technologies enables Web3 to redefine digital ownership, virtual economies, and interactive experiences. In a gaming environment, where user interaction is fundamental, metrics such as response time play a crucial role in ensuring a seamless experience. Meanwhile, the increasing popularity of decentralized gaming puts pressure on developers to build scalable infrastructures that can accommodate growing demand.[7] Finally, since NFTs and digital ownership are integral to these ecosystems, ensuring long-term availability of assets remains a critical requirement. To achieve this, companies should avoid relying on centralized data storage solutions, which suffer from single points of failure, and instead adopt decentralized storage networks that enhance resilience and data permanence.

---

new-hotbed-of-phishing

[7] https://dappradar.com/blog/blockchain-gaming-reaches-new-record-4-2-million-daily-active-users/

Web3's rapid expansion presents both opportunities and challenges, as its decentralized nature enhances security, ownership, and scalability while also introducing critical vulnerabilities. The increasing reliance on Web3 for finance,[8][9] digital ownership, and gaming underscores the urgency of addressing security risks, availability constraints, and scalability limitations. To this end, our research focuses on identifying attack vectors, analyzing malicious activity, and developing countermeasures to strengthen resilience and enhance scalability and availability, ensuring the efficiency and sustainability of decentralized technologies.

1.2 Contributions

While Web3 is often perceived as secure, built upon trustlessness, cryptographic guarantees, and transparent protocols, this dissertation challenges that premise by showing that even its core infrastructures can be exploited to support malicious activities. It presents original empirical evidence of real-world IPFS misuse, and introduces, for the first time, a fully functional Ransomware-as-a-Service architecture built with Ethereum and IPFS.

This central thesis is supported by a series of technical contributions that aim to address both the security risks and performance limitations of Web3 infrastructure. In this work, we analyze and address security and scalability challenges of Web3 by identifying new attack vectors and propose solutions to enhance availability and scalability. Our contributions can be summarized as follows:

- We provide a comprehensive overview of the Web3 ecosystem, focusing on its key components, including Ethereum, IPFS, and its core mechanisms such as Bitswap and IPNS, as well as Non-Fungible Tokens (NFTs) and their role in decentralized applications.

- We map key blockchain security threats reported in the literature, including structural, network, and application-level vulnerabilities, and categorize the

[8] https://www.whitehouse.gov/fact-sheets/2025/03/fact-sheet-president-donald-j-trump-establishes-the-strategic-bitcoin-reserve-and-u-s-digital-asset-stockpile/

[9] https://fortune.com/2025/02/14/elon-musk-us-treasury-blockchain-technology/

misuse of blockchain in malicious contexts such as malware, criminal smart contracts, and fraud.

- We realize the Ransomware as a Service (RaaS) attack vector, demonstrating how blockchain and decentralized storage enhance its resilience and anonymity. By leveraging Ethereum smart contracts for payments and IPFS for hosting malicious infrastructure, we highlight the challenges in disrupting such a system due to its decentralized nature.

- We analyze the IPFS network to detect suspicious activity, anomalous files, and potential abuse. By crawling and monitoring nodes, we assess network behavior and the ethicality of stored content, proposing countermeasures to mitigate associated risks.

- We investigate how malicious actors can exploit existing IPFS technologies to anonymously upload content. By analyzing the mechanisms for adding and accessing files through public services, we identify attack vectors that enable anonymity in IPFS, assess their feasibility through experimental evaluation, and discuss potential countermeasures to mitigate such exploits.

- We propose a method to improve the effectiveness of Bitswap, reducing reliance on the slower DHT. Through extensive experiments, we identify performance bottlenecks and introduce enhancements that increase the likelihood of locating content via Bitswap. This reduces response time from a median of 8 seconds to $\leq$1 second, particularly for less popular content.

- We design and implement a fully decentralized and self-sustaining game system that orchestrates heterogeneous decentralized services, embodying the Web3 paradigm. Our system introduces a decentralized "mint-in-sealed-box" mechanism using threshold cryptography and blockchains, ensures long-term availability of NFT assets via IPFS and Filecoin, and introduces the concept of evolvable NFTs, enabling asset transformation over time through name resolution services. It also facilitates new business models by enabling royalty payments at every resale of in-game assets.

1.3 Dissertation outline

The remainder of the dissertation is organized as follows. Chapter 2 provides an overview of the Web3 ecosystem, with a focus on Ethereum and IPFS. Chapter 3 reviews common blockchain vulnerabilities across multiple layers and analyzes the Ransomware-as-a-Service (RaaS) attack vector in decentralized environments. A proof-of-concept RaaS model is presented, leveraging Ethereum smart contracts and IPFS to demonstrate how decentralization enhances the resilience, anonymity, and trustlessness of such attacks, making disruption significantly more challenging. Chapter 4 investigates the security of IPFS through two complementary approaches. First, it analyzes network activity to detect suspicious behavior, anomalous files, and signs of abuse, using large-scale crawling and node monitoring to assess content distribution and ethical implications. Second, it explores how malicious actors could exploit IPFS to anonymously and persistently disseminate content by identifying and experimentally evaluating attack vectors that leverage public gateways and core protocols. Potential countermeasures are discussed in both contexts. Chapter 5 introduces a decentralized gaming system that features evolvable NFTs with royalty support, a mint-in-sealed-box mechanism, and NFT availability via IPFS and Filecoin. It also proposes optimizations to Bitswap that improve scalability and reduce retrieval latency by addressing performance bottlenecks. Chapter 6 discusses the application of blockchain and IPFS in broader security domains. It examines a system that combines the WoT framework with Ethereum to enable digital twins for access control in IoT environments, and describes a privacy-enhancing scheme for IPFS based on triple hashing, which protects content lookup operations from inference by intermediate nodes. Finally, Chapter 7 draws the final conclusions of this dissertation.

Chapter 2

Overview of the Web3 Ecosystem

Web 1.0 is often referred to as the read-only Web. For many years, the World Wide Web primarily served an informational and educational role through static content, where a small number of creators produced material for a broad audience. As users became more familiar and engaged with the Web, their desire to generate content increased. This shift led to the emergence of Web 2.0, a more interactive and participatory version of the Web. Despite its widespread adoption, Web 2.0 has faced significant challenges, including single points of failure, security vulnerabilities, and centralized control by large entities. Additionally, privacy concerns, such as the exploitation of personal data for marketing, have been long-standing issues due to centralized data storage. Web 3.0, which should not be confused with Web3 and is often referred to as the Semantic Web, emphasizes greater efficiency and intelligence by enabling the reuse and interconnection of data across websites through semantic technologies. Its core objective is to make Web content not only accessible to humans but also interpretable by machines, allowing for more accurate search results, personalized services, and automated reasoning [8].

Web3, positioned as the next phase in the evolution of the Internet, aspires to reshape the digital landscape by promoting decentralization, enhancing user agency, and fostering innovation across various sectors; including finance, governance, data privacy, and digital identity. At its core, Web3 envisions a more equitable, secure, and interoperable online ecosystem that prioritizes user owner-

ship and trustless interactions. Notably, Web3 is evolving and maturing in parallel with Web 3.0, with which it shares certain foundational technologies, such as linked data structures and machine-readable metadata. Figure 2.1 illustrates the historical trajectory of the Web's development, highlighting the distinct phases from Web 1.0 to Web3.

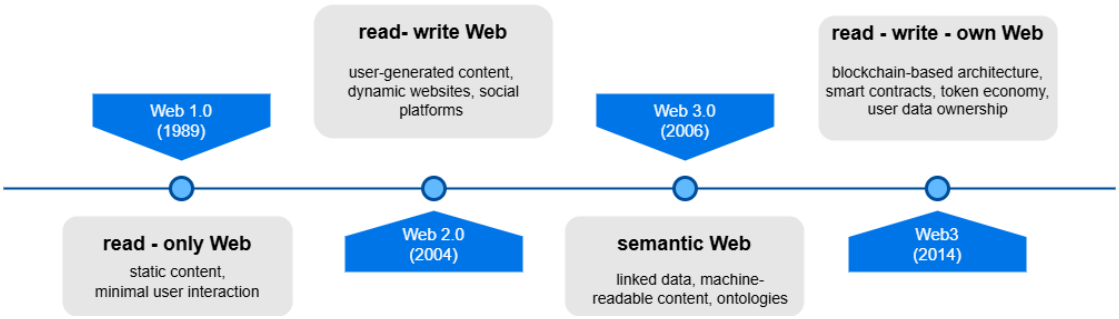


Figure 2.1: Chronological evolution of the Web through its distinct eras.

Web3 is often conceptualized as a layered architecture, where each level is responsible for specific functionalities that enable a decentralized, secure, and user-centric digital ecosystem. At the foundation of the Web3 architecture lies a robust infrastructure layer composed of P2P internet overlay protocols, such as libp2p,[1] which facilitate direct communication between nodes without the need for centralized intermediaries. Complementing this, platform-neutral computation description languages—most notably WebAssembly (WASM)—enable the definition and execution of logic across heterogeneous systems in a secure and portable manner. Building upon this base, the next layer introduces zero-trust interaction protocols, including consensus-driven networks like Ethereum [9] and Bitcoin, alongside data distribution technologies such as IPFS. In addition, transient messaging tools like Waku[2] support secure, ephemeral communication without relying on persistent storage. Further up the stack, we encounter mechanisms designed for performance and scalability. These include state channels, encrypted off-chain storage, e.g., Filecoin [10], Plasma protocols for handling intensive computation, decentralized oracles, and distributed secret management systems—all of which contribute to

[1] `https://libp2p.io/`
[2] `https://waku.org/`

enabling high-throughput, privacy-preserving applications. For developers, the architecture provides an interface layer offering protocol access through extensible tools and SDKs such as Web3.js[3] and Solidity,[4] simplifying the creation and deployment of decentralized applications (DApps). At the top of the stack lies the user interaction layer, where protocol-extensible interfaces like MetaMask[5] empower end-users to interact seamlessly with decentralized services across browsers, mobile devices, and desktop environments. The layered structure of Web3 [11] is depicted in Figure 2.2.

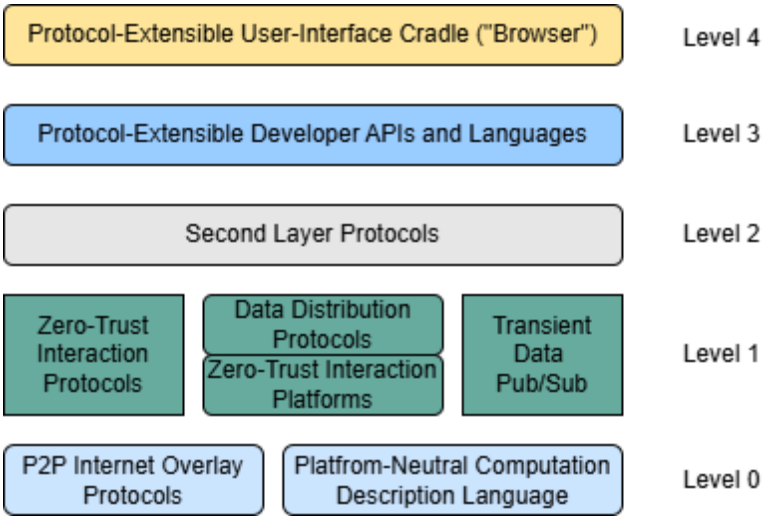


Figure 2.2: Protocol Architecture Overview

As previously discussed, Web3 introduces a more user-centric paradigm of the Internet compared to its predecessors. However, the immaturity of both the underlying technologies and the user base introduces significant risks, many of which stem from the very foundational features that define Web3. The absence of centralized oversight and regulatory frameworks has led to a rise in scams, rug pulls, and various forms of exploitation, with the burden often falling entirely on end-users. Additionally, the P2P nature of information dissemination accelerates the spread of inaccurate or harmful content, making it significantly more difficult to control or remove. Privacy risks are also a growing concern. As data is often

[3] `https://docs.web3js.org/`
[4] `https://soliditylang.org/`
[5] `https://metamask.io/`

replicated and stored across multiple user-operated nodes, the potential for leakage of sensitive information increases, especially in systems where participants may not fully understand the implications of storing or sharing encrypted or personal data [12].

The following sections delve into two central pillars of the Web3 architecture's Level 1: the decentralized ledger infrastructure provided by Ethereum and the distributed file storage capabilities enabled by IPFS.

2.1 Distributed Ledger Technologies

A ledger is a structured and chronological record of transactions whose origins date back to ancient Mesopotamia around 3200 BC. Throughout history, ledgers have served as a foundational tool to record economic activity and maintain trust among trading parties. From early inscriptions on clay tablets and stone to today's digital systems embedded within the financial sector, ledgers have played a central role in the evolution of capitalist economies. In modern times, they are deeply institutionalized within banking and financial infrastructures, forming the backbone of transaction processing, auditing, and financial accountability.[6] However, traditional ledgers, particularly those maintained by centralized institutions, are not without shortcomings. A major limitation lies in their lack of transparency, as access is typically restricted to trusted intermediaries or internal systems. This opacity can lead to information asymmetries, reduced accountability, and, in some cases, to manipulation or fraud. Transactions recorded in a centrally owned ledger may have been altered or tampered with, and records may not be complete or verifiable, especially in the absence of independent audit mechanisms. Moreover, such ledgers often operate within a homogeneous infrastructure environment, where all components—software, hardware, and networking—are standardized. While this may offer operational efficiency, it introduces a critical risk: an attack or failure affecting one part of the system can cascade through the entire network, potentially compromising all stored data [13]. Additionally, centralized systems

[6] https://medium.com/unraveling-the-ouroboros/a-brief-history-of-ledgers-b6ab84a7ff41

introduce single points of failure, making them vulnerable to data loss, cyberattacks, or institutional collapse. The reliance on intermediaries for validation and record-keeping also increases cost, latency, and bureaucratic overhead, particularly in multiparty or cross-border transactions. These limitations have motivated the search for alternative systems that ensure integrity, transparency, and resilience without relying on centralised control.

In 2008, a groundbreaking proposal emerged to address many of the limitations associated with centralized ledger systems. The publication of the white paper "Bitcoin: A Peer-to-Peer Electronic Cash System" [3] by the pseudonymous author Satoshi Nakamoto introduced a novel technological paradigm: the *blockchain*. This innovation presented a decentralized, append-only ledger that operates without the need for a central authority. By combining cryptographic techniques, peer-to-peer networking, and a consensus mechanism known as *Proof of Work*, the proposed system enables participants to validate and record transactions in a transparent, tamper-resistant, and censorship-resistant manner. Beyond solving the double-spending problem in digital currencies,[7] Nakamoto's design laid the foundation for a new class of trustless systems—capable of ensuring data integrity, resilience, and transparency without relying on any single entity or homogeneous infrastructure. Since the publication of Nakamoto's white paper, blockchain technology has undergone significant evolution and adoption across a wide range of domains beyond digital currencies [14]. Numerous blockchain networks have emerged, each with unique designs, consensus models, and use cases—extending from decentralized finance (DeFi) and supply chain management, to identity verification, governance, and data storage. As blockchain technology has matured, researchers and practitioners have proposed several classification models to capture the diversity of architectures. According to the literature [15], blockchains are most commonly categorized as public, private, hybrid, and consortium. Public blockchains, such as Bitcoin and Ethereum, are fully open, permissionless, and decentralized. Anyone can participate, verify transactions, and access the ledger's history, making them highly transparent but often limited in scalability and en-

[7] `https://en.wikipedia.org/wiki/Double-spending`

ergy efficiency. Private blockchains, like Hyperledger Fabric [16] and Corda [17], are restricted systems operated by a single organization or a closed group. They offer greater control, speed, and scalability, but at the cost of decentralization and transparency. Hybrid blockchains combine elements of both public and private models. They allow specific data and processes to remain private while others are made publicly accessible, offering flexibility for applications that require selective transparency, for example, in supply chain tracking or identity systems. Consortium blockchains are governed by a group of organizations rather than a single entity. These semi-decentralized networks are often used in sectors where collaborative data sharing is essential, such as banking or energy markets.

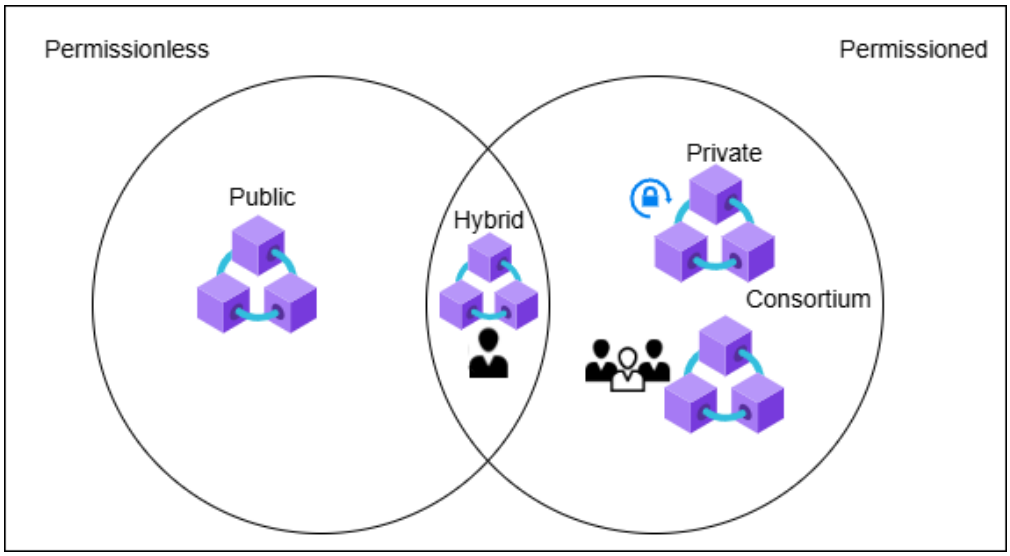


Figure 2.3: Classification of Blockchain Types

At the core of any blockchain system lies the block, the fundamental unit of data that collectively forms the chain. Each block consists of two main components: a block header and a body containing a set of validated transactions. The header stores crucial metadata that links each block to its predecessor, ensuring the chronological and cryptographic integrity of the entire chain. This typically includes the hash of the previous block, a hash representation of the transactions in the block, a timestamp, a nonce, and other fields depending on the consensus algorithm used. The body of the block contains a list of transactions that have

been validated and confirmed by the network, representing state changes such as token transfers, contract executions, or data uploads. By chaining blocks together through cryptographic hashes, the blockchain achieves immutability, where modifying any part of a previous block would require recalculating all subsequent blocks, a computationally infeasible task in most networks.

Another core component of blockchain systems is the consensus mechanism, i.e., the process by which distributed nodes in the network agree on the validity and ordering of transactions. In the absence of a central authority, consensus protocols are essential for maintaining a consistent and tamper-resistant ledger across potentially untrusted participants. These mechanisms address the fundamental problem of Byzantine Fault Tolerance [18], ensuring that the network can operate correctly even when some nodes behave maliciously or unpredictably. The choice of consensus algorithm often varies depending on the type of blockchain. Public, permissionless blockchains, such as Bitcoin and Ethereum, typically employ Proof-of-Work (PoW) or Proof-of-Stake (PoS), which are designed to be open and decentralized. These models rely on economic incentives, such as block rewards and staking, to secure the network. In contrast, permissioned blockchains, such as Hyperledger Fabric, adopt more efficient and deterministic algorithms like Practical Byzantine Fault Tolerance (PBFT), where a limited set of known participants coordinate to reach consensus. These models offer higher throughput and lower latency but trade off some decentralization for performance and control.

The concept of smart contracts was first introduced by Nick Szabo in 1994, long before the advent of blockchain technology. Szabo envisioned a system in which contractual clauses could be automatically enforced by computer programs, thereby eliminating the need for trusted intermediaries in various domains. The core idea behind smart contracts is that the execution of an agreement is not managed by a person, but rather by code, which ensures that the terms are carried out precisely and transparently once predefined conditions are met [19]. Today, smart contracts have become a fundamental component of blockchain platforms, particularly in networks such as Ethereum. Technically, a smart contract consists of two core elements: its code, which defines the logic and rules of execution, and

its state, which reflects its current data or configuration. When a smart contract is triggered, typically by a transaction, its code is executed deterministically across all validating nodes in the network. If all participating nodes reach the same result and reach consensus on the outcome, the resulting change in the contract's state is recorded on the blockchain. This ensures that smart contracts operate in a trustless, transparent, and tamper-resistant manner, enabling the automation of agreements without reliance on centralized authorities. Their tamper-resistance stems from the fact that, once deployed to the blockchain they cannot be altered. The programming language used to develop smart contracts varies depending on the underlying blockchain platform. Additionally, in many blockchain networks, the user or external agent triggering the smart contract is required to pay a transaction fee—commonly referred to as gas—to compensate for the computational resources consumed by the network during contract execution. These fees serve both as a spam prevention mechanism and as an economic incentive for validators or miners.

Despite its transformative potential, blockchain technology faces several inherent limitations. Low transaction throughput and high latency hinder scalability, particularly in public networks. PoW-based systems are often criticized for energy inefficiency, raising concerns about sustainability and the potential misuse of computational resources. Moreover, the threat of quantum computing further challenges the long-term security of existing cryptographic schemes [20]. Additionally, the growing storage requirements of blockchain ledgers, combined with the permanent visibility of transactional data, raise privacy and scalability concerns. Finally, smart contracts, while offering automation and trust minimization, are non-updatable once deployed, which can lead to persistent vulnerabilities if errors exist in their code [13, 14]. While blockchain technology offers a compelling vision for decentralized, transparent, and trustless systems, its widespread adoption will depend on the ability to address its current limitations—ensuring that innovation is matched with scalability, security, and responsible design.

2.1.1 Ethereum

Launched in 2015 by Vitalik Buterin and a team of co-founders, Ethereum [9] is a decentralized, general-purpose blockchain platform designed to enable the creation and execution of smart contracts and decentralized applications (DApps). Unlike Bitcoin, which was conceived solely as a peer-to-peer digital currency, Ethereum was developed as a programmable infrastructure, allowing developers to encode arbitrary logic into the blockchain using its built-in Turing-complete language. Ethereum leverages blockchain technology not only to maintain a secure and immutable record of transactions but also as a decentralized execution environment for smart contracts. This ensures that the contract logic is transparently executed and tamper-resistant, with every state change recorded and agreed upon across the network. At the core of Ethereum's architecture lies the Ethereum Virtual Machine (EVM), a Turing-complete runtime environment that allows contract code to be executed deterministically across all nodes in the network. Developers typically write smart contracts in *Solidity*, a high-level programming language designed specifically for Ethereum. This design enables the creation of complex logic on-chain, supporting a wide variety of applications, from DeFi to governance systems and identity management tools. Although it initially operated under a PoW consensus model, Ethereum transitioned to PoS through a series of upgrades collectively known as Ethereum 2.0, a process that culminated in "The Merge" on September of 2022, significantly improving the protocol's energy efficiency and laying the groundwork for future scalability [21].

The native cryptocurrency of the Ethereum network, Ether (ETH), serves not only as a medium of exchange and store of value, but also as the fuel that powers the execution of smart contracts and transactions through a mechanism known as gas. Gas represents the unit of computational effort required to perform operations such as executing smart contracts, processing transactions, or interacting with decentralized applications. Every operation in EVM is assigned a specific gas cost, and users must specify a gas limit and gas price when submitting a transaction. The total fee paid—denominated in ETH—is calculated as the product of these two values. This mechanism serves several key purposes: it prevents abuse of network

resources, ensures that computation is priced proportionally to its complexity, and incentivizes validators to prioritize and include transactions in blocks. Gas fees can fluctuate significantly depending on network congestion and demand, which has led to scalability challenges and motivated the development of Layer 2 solutions.

Non-Fungible Tokens (NFTs)

Ethereum's development and standardization process is driven by community proposals known as Ethereum Improvement Proposals (EIPs). These documents define suggested changes or enhancements to the protocol, covering areas such as consensus rules, virtual machine specifications, and network upgrades. A subset of EIPs, known as Ethereum Request for Comments (ERCs), establishes application-level standards, particularly for smart contracts. The most well-known examples include ERC-20, which defines a standard interface for fungible tokens, and ERC-721 [22], which introduced the concept of non-fungible tokens (NFTs). NFTs are digital assets that represent ownership of a wide range of unique items, tangible or intangible, often associated with digital goods such as images, music, virtual land, or identity credentials. On the Ethereum platform, NFTs are most commonly implemented via the ERC-721 standard, which defines a set of functions that a smart contract must implement to create, transfer, and manage NFTs. Beyond the core interface, ERC-721 includes an optional extension known as the metadata extension, which enables NFTs to be associated with external metadata—such as a name, description, or link to off-chain content—further enriching the token's utility and uniqueness. NFTs gained widespread popularity due to their versatility and found numerous applications across various industries, including gaming, digital collectibles, digital art, and fashion [23]. Their rapid adoption led to an explosive growth in market value, reaching a total market capitalization of over $400 billion in March 2022.[8]

This surge in popularity also attracted the attention of malicious actors, who sought to exploit the inherent vulnerabilities of NFT ecosystems for personal gain. One of the major concerns relates to intellectual property rights, as it is

[8] `https://coinmarketcap.com/nft/`

often unclear whether a seller truly owns the rights to the NFT they are offering. Verifying authenticity and rightful ownership before purchase remains a critical challenge. In addition, imitation and fraud are widespread: malicious users have impersonated well-known NFT artists to sell counterfeit works, while others engage in fake airdrops, phishing campaigns, and scam giveaways. Another significant vulnerability lies in the smart contracts that underpin NFTs. Poorly audited or insecure contract code can lead to exploits, permanent loss of assets, or abuse of platform mechanisms. Finally, NFTs are being used for money laundering[9], e.g., exploiting the lack of proper KYC practices. As the ecosystem continues to evolve, addressing these risks is essential for the long-term sustainability and trustworthiness of NFTs [24].

Ethereum Name Service (ENS)

Another widely adopted standard is EIP-137 [25], which defines the Ethereum Name Service (ENS), a distributed naming system built on the Ethereum blockchain. Similar in purpose to the Domain Name System (DNS) of the traditional Web, ENS maps human-readable names to Ethereum addresses, other cryptocurrency addresses, content hashes and other resources. The ENS architecture comprises three core components: the ENS registry, a smart contract that maintains mappings between names and resolvers; the resolvers, which are responsible for performing lookups and returning the associated data; and the registrars, which handle the allocation and management of domain names to users. By simplifying blockchain interactions and enhancing usability, ENS plays a crucial role in improving the accessibility of decentralized applications and services, yet it has already been abused by threat actors for phishing and other malicious activities [26].

2.2 Decentralized Storage Networks

In recent years, the digitization of all aspects of everyday life has led to a rapid increase in the volume of data being generated. As a result, the demand

[9] `https://home.treasury.gov/news/press-releases/jy2382`

for data storage capacity has grown significantly. The emergence of cloud services introduced new solutions offered by major technology companies such as Google, Amazon, and Microsoft. Through cloud services, users can store their data on platforms they trust, often at a lower cost compared to maintaining their own infrastructure, while also enjoying ubiquitous access to their files from any Internet-connected device.

In the era of Web3, where the user is at the centre of the digital experience, traditional cloud services stand in contrast to the principles of user autonomy and data ownership [27]. Issues such as documented cases of service outages,[10], and well-known incidents of data leakage, where sensitive user information was exposed due to misconfigurations or breaches, underscore the risks associated with centralized infrastructures. At the same time, centralized data silos represent attractive targets for attackers aiming to breach systems and monetize stolen data, further compromising security and privacy. Finally, pricing models that resemble cartel-like behavior[11] further reinforce the growing need for decentralized storage solutions that align with the core values of Web3. In the past, P2P data-sharing networks have enjoyed considerable popularity, with systems such as Napster, Gnutella, and later BitTorrent leading the way. BitTorrent, in particular, was the first to introduce incentive mechanisms to encourage participation, improve file availability, and create a more robust and resilient network architecture. Nevertheless, ensuring long-term availability of content remained one of the key limitations of such P2P systems, especially in the absence of persistent incentives or reliable node uptime [28]. Since the advent of Bitcoin in 2008 and the subsequent rise of other blockchain technologies, P2P networks have experienced a renewed resurgence. They have become integral in supporting a wide range of Web3 applications, including DApps, NFTs, decentralized gaming, and many other emerging use cases that have arisen in the Web3 paradigm. Among the most prominent decentralized storage platforms are IPFS, Filecoin, Storj, Arweave, and SIA. Decentralized storage networks aim to implement three key characteristics: proof of storage,

[10] https://www.crn.com/news/cloud/2024/the-10-biggest-cloud-outages-of-2024

[11] https://iclg.com/news/21598-germany-s-federal-cartel-office-intensifies-microsoft-scrutiny

consensus mechanisms, and incentive structures that ensure reliability and participation.

2.2.1 InterPlanetary File System (IPFS)

The InterPlanetary File System (IPFS) [29] is a P2P protocol for file storage and sharing. It was developed by Protocol Labs as an open-source project in 2015, and has since undergone continuous development and improvement. IPFS is considered a cornerstone of the Web3 ecosystem, as it is widely supported and adopted across multiple domains, including NFTs, decentralized gaming, and social media platforms. Since 2018, public gateways for IPFS have been introduced, serving as bridges between the traditional Web and the emerging Web3 ecosystem. These gateways allow users to access IPFS content through conventional browsers without running a full IPFS node, significantly enhancing accessibility. As a result of these factors, IPFS has experienced significant adoption, with more than three million Web client accesses and more than 300,000 unique nodes actively serving content on the P2P network each week [4]. It is worth mentioning that, in alignment with the principles of Web3, IPFS was used in 2017 to disseminate and preserve information related to the Catalan independence referendum, after attempts by the Spanish government to censor online content.[12]

IPFS employs a content-addressable model in which each piece of content is identified by a unique Content Identifier (CID) generated through cryptographic hashing. Unlike traditional Web protocols such as HTTP, which use URLs that point to specific servers, IPFS CIDs refer directly to the content itself, making the system independent of storage location and promoting decentralized data distribution. A CID consists of four components: the base encoding format (e.g., Base32), the version of the CID (either v0 or v1), the multicodec, which indicates how the content is encoded (such as protobuf, JSON, or CBOR), and the multihash, which encodes the hash function, length, and content hash.

A core component of IPFS is the Distributed Hash Table (DHT), a tailored

[12] `https://edri.org/our-work/no-justification-for-internet-censorship-during-catalan-referendum/`

version of the Kademlia DHT [30], specifically adapted to the needs and properties of the IPFS network. The DHT manages three types of mappings, including Provider Records, which indicate which peer stores a given piece of content; Peer Records, which contain information about the network addresses of peers; and IPNS Records, which enable the mapping of persistent identifiers to dynamically changing content. The DHT used in IPFS is based on the XOR metric, which is employed to organize the network's routing and lookup functions. Specifically, each peer and each piece of content is assigned a unique identifier which is 256 bits long. Moreover, every IPFS peer maintains at most 256 KBuckets, which are data structures used to store information about known remote peers. The x^{th} KBucket of a node contains peers whose IDs share a common prefix of length $x-1$ with the node's own ID. When given a target ID, IPFS uses Kademlia's XOR-based distance metric to locate the closest peers in a logarithmic number of steps, enabling fast and decentralized peer discovery.

Once a new node connects to the IPFS network, it is designated as a DHT Server if it has a public IP address. Otherwise, e.g., if it is behind a NAT—it operates as a DHT Client. This distinction is managed by a mechanism called Autonat, which determines a node's reachability. DHT Servers are responsible for storing and serving content, while DHT Clients only issue queries, a separation that enhances the network's overall efficiency. When a user publishes a file to IPFS, the file is first split into chunks, typically 256 KB each. Each chunk receives a unique CID and is organized into a Merkle Directed Acyclic Graph (DAG) before being added to the network where the root CID represents the entire file. This cryptographic linking ensures that content is tamper-proof and verifiable, since even the slightest modification will result in a completely different CID. As part of this process, two types of records are stored in the DHT, each distributed across 20 different nodes. The first, known as the Provider Record, identifies which node is hosting the content. It includes two important parameters: a republish interval (12 hours by default), which reallocates the record if original nodes become unavailable; and an expiration interval (24 hours by default), which verifies that the content provider is still online. The second, the Peer Record, maps the peer's ID to its

physical network address. Notably, when content is added to IPFS, the file itself is not replicated across the network. Instead, only routing records pointing to the uploader are distributed. The uploader's node automatically pins the file locally, ensuring its availability only while the uploader remains online. Replication occurs opportunistically, i.e., if another user retrieves the file, it is cached locally. If the original uploader disconnects, the file's continued availability depends entirely on whether other peers have cached it. Figure 2.4 illustrates the aforementioned process.

In IPFS, each node maintains a set of active peer connections called the *swarm*, typically consisting of 600 to 900 peers, with the lower and upper bounds known as the low and high water marks, respectively. When a user requests a file, the Bitswap protocol is triggered. It broadcasts a "want-have <root CID>" message to peers within the swarm. Each peer checks locally whether it holds the corresponding CID, and if so, responds with a "have" message. Upon receiving a "have" message, IPFS initiates a dedicated session for that CID, including all peers that indicated content availability. From that point forward, only these peers participate in the data exchange session [31]. If no peer responds within one second, the query is escalated to the DHT. The DHT operates in two phases: first, it searches for the Provider Record. Upon retrieving it, it fetches the corresponding Peer Record. Once the lookup is complete, Bitswap resumes, establishing a direct exchange with the peer that holds the desired data [4]. A notable recent

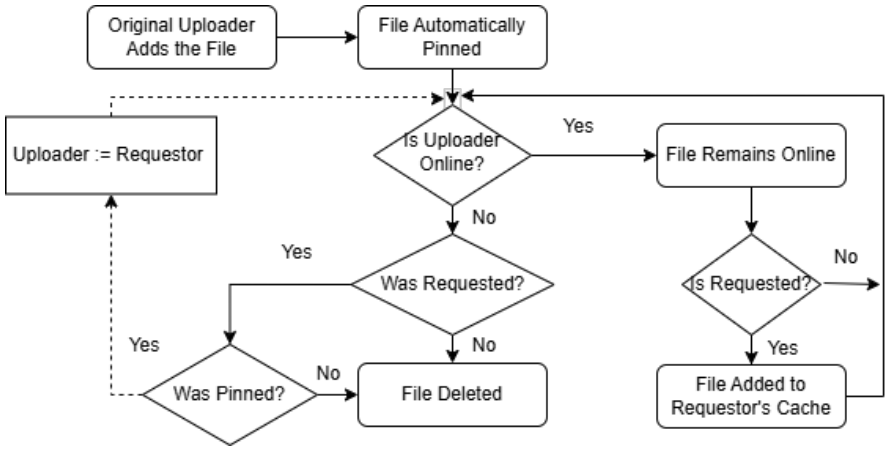


Figure 2.4: The File Lifecycle In IPFS

development in the IPFS ecosystem is the introduction of InterPlanetary Network Indexers (IPNI), a more centralized alternative to the DHT, specifically designed for efficient indexing of provider records. Targeted primarily at large-scale content providers, IPNI complements the DHT by offloading the responsibility of provider discovery, while trying to maintain the decentralized structure of the broader network [32].

InterPlanetary Name System (IPNS)

IPNS[13] is the component of IPFS responsible for creating persistent addresses that can point to mutable data. Each IPNS address is essentially the hash of a public key, and to use it, the user must first generate a corresponding asymmetric key pair. Updates to the underlying content can be made by simply publishing new data using the same key, thus maintaining the same address while modifying the linked content. IPNS records are both stored in the DHT and propagated through it, enabling decentralized name resolution. These records are versioned, so when a node queries the DHT for a particular IPNS address, it receives the most recent version of the associated data. It is also important to note that IPNS records have a default lifetime of 24 hours, requiring periodic republishing to ensure that the address remains resolvable and up to date.

[13] `https://docs.ipfs.tech/concepts/ipns/`

Chapter 3

Security Threats and Malicious Use Cases in the Blockchain Ecosystem

Blockchain technology has rapidly evolved from a novel mechanism underpinning cryptocurrencies into a foundational layer for decentralized systems and digital trust. However, as adoption has accelerated, so too has the discovery of security weaknesses, both inherent in the underlying protocols and emergent through complex applications such as smart contracts. This chapter aims to explore the security landscape of blockchain ecosystems by presenting a two-fold examination. First, it provides a comprehensive overview of vulnerabilities that have been identified across various layers of blockchain architecture, including consensus mechanisms, networking protocols, and application-level components. This part highlights the technical and operational challenges faced by both public blockchain deployments. Second, the chapter focuses on a concrete and evolving threat scenario: the abuse of blockchain and decentralized storage platforms such as the InterPlanetary File System (IPFS) to support Ransomware-as-a-Service (RaaS) operations.

3.1 Blockchain Vulnerabilities

Blockchain systems are inherently multi-layered, and each layer introduces specific security challenges [33]. These layers can broadly be categorized as follows:

- Protocol Layer: Includes cryptographic primitives, and block validation logic.
- Peer-to-Peer (P2P) Network Layer: Governs node connectivity and message propagation.
- Consensus Layer: Defines how agreement is reached across distributed nodes (e.g., PoW, PoS).
- Application Layer: Encompasses smart contracts, wallets, and decentralized applications.

While each of these layers exposes unique attack surfaces, this work focuses primarily on vulnerabilities in the P2P Network Layer and the Application Layer, as these are commonly exploited in real-world blockchain abuse scenarios.

3.1.1 P2P System Attacks

Majority Attack

Majority Attack, also known as 51% attack, is mostly a threat for PoW-based systems. For this attack to be achieved, the attacker has to come with hash rate greater than the network's majority. This gives the attacker the ability to cancel transactions, double spend or even fork the main blockchain. There is also a special version of this attack, known as rental attack, during which the attacker rents computing resources in order to achieve the 51% of consensus.

Network Attacks

Network-layer attacks target the P2P communication protocols of blockchain systems, aiming to disrupt node connectivity, isolate participants, or manipulate the propagation of information across the network [33].

Name	Symbol	Hash Rate	Attack cost per Hour
Bitcoin	BTC	915,565 PH/s	$1,527,672
Bitcoin Cash	BCH	2,831 PH/s	$4,724
EthereumClassic	ETC	284 TH/s	$7,159
LiteCoin	LTC	2 PH/s	$89,757

Table 3.1: https://www.crypto51.app/

1. DNS Attacks

   When a node connects to the network for the first time, it doesn't have any knowledge of the active peers' IPs. To discover and communicate with at least one of them, DNS seeds are used, or an active IP has to be entered manually. For a successful attack, the attacker should tamper the seeds and inject an invalid list in the open source Blockchain software or poison the DNS cache.

2. BGP Hijacks and Spatial Partitioning

   Most Blockchain applications have two types of nodes, full nodes which keep an updated and complete version of the blockchain and lightweight nodes which contact full nodes to acquire a snapshot of the Blockchain. When a full node is compromised then its related lightweight nodes will also be. The centralization of Bitcoin nodes, as highlighted by Apostolaki et al. [34], increases the network's vulnerability to BGP routing attacks, which can have severe implications for both network reliability and security. Mining pools use stratum protocol for bitcoin mining. Stratum servers, which collect mining outcome, have a public IP. As a result, they are also vulnerable to routing attacks, which would delay the block propagation up to 20 minutes. This scenario raises the risk of doublespending or intentional forks.

3. Eclipse Attacks

   The P2P Protocol used by blockchain is vulnerable to the so-called Eclipse Attacks. In this attack, a group of malicious nodes isolates a neighbor node in order to tamper its incoming and outgoing traffic and change their view

of blockchain. While the victim is connected with at least one honest node, it keeps the normal version of blockchain. When the connection is lost, malicious nodes have the opportunity to surround it. In Bitcoin every node has 8 active connections, in contrast to Ethereum, where it would have 13. Because of this, Ethereum is seemingly more secure. Unlike Bitcoin, Ethereum employs a Kademlia-based peer discovery protocol, which is ID-oriented rather than IP-oriented. This design introduces certain vulnerabilities, as shown by Marcus et al. [35], who successfully executed an Eclipse attack on the Ethereum network using only a single machine and multiple virtual node identities, effectively exploiting the protocol's reliance on node IDs.

DoS Attacks

Denial-of-Service (DoS) attacks in blockchain networks aim to overwhelm nodes or the entire network with excessive data or computation requests, potentially degrading performance, delaying consensus, or causing temporary unavailability of services.

1. Stress Testing

   A version of DoS attack is Stress testing, which is caused by the limited capacity of transactions per block at a given time. For example, Bitcoin can process up to 7 transactions per second, which is very low compared to VISA Credit which can process up to 2000 transactions per second. An attacker may exploit this shortcoming using Sybil identities or multiple wallets to flood the network with dust transactions (≈ 0.001 BTC/tx) so that the legitimate users will not be served.

2. Mempool Flooding

   Another version of DoS attack is caused by mempool resulting in augmentation of the mining fee. Mempools, or transaction pools, are cache memories of unconfirmed transactions and their size is watched by the miners. When there are many unconfirmed transactions, the competition for mining raises and thus users pay greater mining fee. Saad et al. [36] introduced an attack

in which sybil nodes flood the mempool with unconfirmed transactions. Such an attack causes panic to the users who have to pay higher mining fee whilst the attacker's transactions are not mined.

Time Related Attacks

1. Consensus Delay

   In this attack the attacker injects faulty blocks to delay the peers from reaching consensus. This can be achieved with stale blocks or double spend transactions. The nodes which are unaware of the faulty blocks will spend some time for verification.

2. Timejacking Attack

   In blockchain applications, full nodes keep an internal counter which denotes the network time. Network time is given to a node during the bootstrapping procedure. If the median of the network time is given and overcomes 70 minutes, then the network time is set to the system time. This situation creates an attack opportunity: as long as the attacker sends to the victim different timestamps from sybil nodes, the median of which overcomes 70 minutes, and siphons the network blocks with time difference 50 minutes. Hence, the blocks arriving at the victim have a time difference of 120 minutes to the victim's clock and get automatically rejected. As a result, the victim becomes isolated from the network.

3.1.2 Application Oriented Attacks

The application layer encompasses the components that interact directly with end users, such as wallets, smart contracts, and DApps. While enabling key blockchain functionalities, this layer also exposes a broad attack surface, making it a common target for threats like cryptojacking, private key theft, and contract-level exploits [33].

Cryptojacking

Cryptojacking is a kind of attack in which the attacker performs PoW for blockchain-based cryptocurrencies without consent, while the victim is unaware of this action.

1. Cloud-based Cryptojacking
   Malicious miners have found a way to expand their hash power by hijacking processors of remote devices for mining. It involves hijacking a target device to perform PoW calculations for the attacker. Initially, these attacks were launched against cloud service providers, where malicious users performed covert mining operations on virtual machines and exhausted cloud resources.

2. Web-based Cryptojacking
   Web-based cryptojacking is used by attackers who inject malicious JavaScript code into websites that secretly mine tokens without the consent of their visitors. In browser-based cryptojacking, the web browser on the client device executes JavaScript code that establishes a WebSocket connection with a remote server. The server collects the computed PoW hashes on behalf of the attacker. Throughout this process, the device owner remains unaware of this background activity and seamlessly continues to browse the website.

3. Malware Cryptojacking
   Malware cryptojacking[1] is another way that cryptojackers have found to enslave users' processors. It spreads like a classic malware through corrupted software either on a computer platform or a mobile platform. When the victim executes the infected application, a miner is loaded onto the memory and it starts mining while staying hidden in the victim's device. Two-dozen Android applications whose code turns user's phones into cryptocurrency mining workers recently appeared in Google Play. Some of those have targeted users in the US by using the guise of educational tools. Combined, they have been downloaded more than 120,000 times.[2]

[1] `https://www.malwarebytes.com/cryptojacking/`

[2] `https://nakedsecurity.sophos.com/2018/02/01/cryptomining-is-it-the-new-ransomware-report/`

Wallet Theft

A wallet is a software program that stores private and public keys and interacts with blockchain platforms to enable users to send and receive digital currency and monitor their balances. However, wallet theft remains a chronic threat: in February 2025, approximately $1.5 billion in Ether were stolen from a Bybit exchange wallet during a routine cold-to-warm transfer,[3] in an attack attributed to the Lazarus-linked TraderTraitor group. More broadly, over $1.8 billion in cryptocurrency losses have been reported in the first half of 2025 alone, due to private key compromise.[4]

1. Key Exposure and Theft

   Having possession of the private key and keeping it secret is critical because in most cryptocurrencies using private key, someone can sign transactions, spend money or tokens and deploy smart contracts on the blockchain. If attackers acquire the private key belonging to a user, then they can spend all of the user's balance.

2. Software Client Vulnerabilities

   Public blockchains like Bitcoin and Ethereum have open source software, which users use to connect to the network. This software is vulnerable to many attacks and, although new software versions are being released, many users do not update to the newer versions. This situation gives the attacker the opportunity to take control of the wallet software and the victim's balance.

3. Weak Private Keys

   In Ethereum the procedure for generating credentials is as follows:

   (a) The Private Key is generated from a random number between 1 and 2^{256} using the OS random number generator.

[3] https://www.reuters.com/technology/cybersecurity/cryptos-biggest-hacks-heists-after-15-billion-theft-bybit-2025-02-24/

[4] https://cointelegraph.com/news/otal-hacks-down-q2-after-record-losses-2025-h1

(b) The public key is calculated from the private key using elliptic curve multiplication.

(c) Address is derived from the Keccak-256 hash of the public key keeping the last 20 bytes.

ISE,[5] a security research group, made an experiment in which they scanned the Ethereum Blockchain to find keys that are weak or lack randomness due to key truncation [37]. Key truncation is the process according to which a random 256-bit key is generated but only a subset is used due to coding, compiler or framework errors. Since scanning all the potential addresses is impossible, ISE split the 256-bit space in 8 intervals and performed a brute force scan for each one of them setting the other intervals to zero.

H	G	F	E	D	C	B	A
256	224	192	160	128	96	64	32	0

Table 3.2: The intervals of a 256-bit key

- Group A: 000...00000001 to 000...0FFFFFFFF
- Group B: 000...100000000 to 000...FFFFFFFF00000000
- ...
- Group H: 00000001...0000 to FFFFFFFF00...00000

The group found more than 750 weak keys, 450 of which are in group A, responsible for 49060 transactions. They also noted that there is a specific address, to which they gave the alias *blockchain bandit*, which interacts with many of the weak addresses. ISE sent an amount $1 worth of Ethereum to one of those weak addresses and it was instantly transferred to the *blockchain bandit*. This address has once held almost $54 million worth of ether and currently owns $6 million due to market correction. Finally, ISE found a shortcoming in the Parity Wallet which generates

[5]https://www.securityevaluators.com/

the public key using a user's passphrase as randomness. They found out that Parity accepts an empty passphrase as a valid passphrase and generates a specific address. There have been 8772 transaction on this address, and every incoming transaction is transferred to one of many keyholders monitoring this address.

Smart Contract Attacks

Smart contracts, while offering programmable logic and automation within blockchain environments, have introduced a new class of security vulnerabilities. Due to their immutable and public nature, any flaw in their design or implementation can be permanently exploitable, often leading to significant financial losses and systemic risks. Some of the most well-known smart contract vulnerabilities, which have been repeatedly documented and exploited in real-world attacks, include the following [38]:

1. Reentrancy

   Fallback function is the only unnamed, no argument function a contract can have. It is executed on a call to the contract if none of the other functions match the given function identifier or when the contract receives plain ether. In a reentrancy attack, the attacker calls recursively the fallback function before its termination, so that the balance is not updated before sending ether. This act usually leads to loss of ether.

2. Stack Size Limit

   Every time a contract invokes another contract, the call stack associated with the transaction grows by one frame. The call stack cannot grow larger than 1024 frames and, when the limit is reached, possible further invocation throws an exception. Until October 18th 2016, it was possible to exploit this vulnerability. Since then this flaw has been resolved by a hard fork. The best known attack exploiting Stack Size Limit is the Governmental Ponzi Scheme.

3. Keeping Secrets

   Fields in contracts can be public, i.e. can be read by everyone, or private,

i.e. other users or smart contracts cannot access them. However, declaring a field as private does not guarantee it's secrecy. The main reason for that is Blockchain's transparency, as a result of which, the state of a smart contract is visible to anyone using a Blockchain explorer.

4. Immutable Bugs
   Once a contract is deployed on the Blockchain, it cannot be edited. If a contract contains a bug there is no way to fix it. Programmers, during the coding phase, have to foresee and provide the user with the option to terminate a vulnerable smart contract.

5. Generating Randomness
   The execution of EVM bytecode is deterministic, which means that every transaction mined by every miner should have the same result in the state of a contract. In order to generate randomness, programmers usually take advantage of the hash or the timestamp of a block, either the present block or a future one. A malicious miner could craft his/her block to predetermine the result of random generator.

6. Time Constraints
   Programmers in many cases use time constraints to decide whether some actions are permitted or not. To achieve these constraints they take advantage of block timestamps. Since the miner who crafts the block is responsible for choosing its timestamp within a degree of arbitrariness, the contract may be exposed to attacks.

7. Integer Overflow/Underflow
   When an unsigned integer reaches its byte size, the next element added will return the first variable element. Programmers using high level languages are not familiar with this situation which may expose the smart contract to attacks.

The most known Smart Contract attack is the D.A.O. attack performed on June 18th 2016. The attacker exploited a combination of vulnerabilities to take

under his possession $60 million. This attack was the reason Ethereum had a hard fork and its price was reduced from $20 to $13.

3.2 Malicious Use Of Blockchains

While blockchain technologies are primarily designed to enhance transparency, integrity, and decentralization, their architectural features can also be exploited for malicious purposes. The immutability, censorship resistance, and pseudonymity they offer have attracted not only developers and innovators, but also cybercriminals. This section explores the malicious applications of blockchain systems, from malware distribution and botnet coordination to criminal smart contracts and financial fraud.

3.2.1 Blockchain and Malware

In 2015, Harsh Patel[6] wrote about a new kind of malware which utilizes blockchain, the **Blockchainware**. In his article he mentioned that there are two different ways in which malware and blockchain can cooperate:

- Blockchain as a storage for malware components.
- Blockchain as a command and control center.

Proof of Existence for malware components.

Moubarak et al. [39] studied the blockchain potential in a K-ary malware. A K-ary malware instead of holding the virus instructions in a single file, consists of k parts which union results to the malware. Each one of these parts is executable and seems to be an innocent file. There are two main categories of k-ary malware:

- First category: The k parts are working sequentially.
- Second category: The k parts are working in parallel.

[6]https://www.linkedin.com/pulse/blockchain-ware-next-stage-malware-evolution-harsh-patel/

In their paper they leveraged blockchain to create a 4-ary malware of the first category, they used blockchain technology with Proof Of Existence. They split the viral payload into 4 chunks and they deployed each one of them on to the Blockchain. The blockchain, keeps a record of each file that anyone can verify simply by using a blockchain explorer. The OP_RETURN opcode [3] is a script opcode which can be found in a transaction. It allows a user to add 80 bytes of arbitrary data into the blockchain. Another important aspect of OP_RETURN is that the transaction's output is unspendable so it does not belong to UTXO and thus is not transferred to a node's cache. The latter makes the OP_RETURN transactions less expensive for the network and a way to burn BTCs. Proof of Existence is a way for someone to prove that a file existed at a specific time. Someone can exploit the OP_RETURN opcode and insert into the blockchain the file's hash value [40].

Malware coordination using Blockchain

Some malware, such as ransomware, must keep in touch with their owner, either to receive orders from him or to be able to send him some loot from the victim. Initially, the owner's IP was hardcoded to malware, so it was easy to make communication perceived and blocked by the host. Then, the malware writers hid the communication via the IRC protocol, but with the passage of time it became obsolete, so it was again easy to make the communication perceived by the host. Recently, malware writers hide communication under the HTTP protocol, mixing it with the rest of the host's communication with the web. To avoid blocking the IP by the host, they apply a technique called domain fluxing. They use a Domain Generation Algorithm (DGA), which for a specific time interval, produces a long list of potential domains and one of them leads to the Command & Control server (C2). The problem in this case is the many look -ups that lead to NXDomain, a pattern in network traffic that an anti-virus can easily discover. Pletinckx et al. [41] report a completely new group of DGA that locate the C2 server information in the bitcoin blockchain without generating any NXDomain responses.

This technique is actively used by Cerber ransomware,[7] as follows:

1. The first time the host runs the malware, it connects to a block explorer and asks for the transactions of a wallet, the wallet address is hardcoded to the malware.

2. This wallet, let it be W, appears to trade with some other wallets, let them be W_i. In these transactions W sends some money to W_i which is instantly returned to W. However, W_i are temporary because the first six characters of their address followed by .top TLD is the address to connect with C2 server.

3. The latter address does not host the C2 server but is a gateway for the Tor network.

Finally, the authors report that this malware could be a typical example of a ransomware -as -a-service model. In a system in which affiliates propagate malware to potential victims, which are led to a malware writer's microsite. Upon the victim's payment, the writer gives the percentage that corresponds to the affiliate.

Managing Botnets using Bitcoin

Botnet is a network of computers infected with malicious software and controlled as a group, without the owner's knowledge, by an entity known as the botmaster. Every bot needs to communicate with the botmaster or the C2 center to receive commands. This communication channel can make the botnet perceived by the host. Like ransomware, botnets have followed various C2 techniques such as IRC chatrooms, HTTP rendezvous points or P2P networks. Ali et al. [42] claim that bitcoin is an ideal means of spreading commands for a botnet. The advantages of bitcoin include the small cost of maintaining the C2 infrastructure, relative anonymity, and more importantly, that no entity can shut down the bitcoin network or address. The authors leveraged bitcoin to implement their own botnet named ZombieCoin. ZombieCoin's main points of operation are:

[7]https://www.avast.com/c-cerber

1. The botmaster creates a Bitcoin key pair *(pk, sk)* and then hardcodes the pk in the bot file. The bots are also equipped with a set of commands to translate and execute the commands sent by the C2 center.

2. The botmaster using classic infection techniques or a zero-day exploit spreads the botnet. When a host becomes infected the bot generates a unique bot identifier.

3. Bots then connect to the bitcoin network as Simplified Payment Verification (SPV) nodes, receive and propagate incoming transactions.

4. The botmaster periodically sends commands embedded in transactions. The bots to discover commands scan the ScriptSig field of transactions with the *pk* input of the master. Commands are part of the 80 bytes available for OP_RETURN opcode. Finally, bots decode and execute commands.

To validate ZombieCoin, the authors built a 14-node botnet using BitcoinJ. Their application is 7 MB in size and stored locally about 626 KB of blockchain data. They issued over 250 instructions with an average response time of ≈ 6 seconds and total cost of $ 7.50. The latter is a trivial cost compared to the profits he may have renting the botnet. The price for 50,000 bots, which will constantly attack a target for a duration of 3,600 seconds with a 5-10 minute cool down timer set for 2 weeks, is between $3000 - $4000.[8] Finally, for upstream communication they used the rendezvous points technique, in which the botmaster sends an IP address available for a short time so that bots upload data through this address.

3.2.2 Criminal Smart Contracts

Juels et al. [43] conducted a study on criminal smart contracts. They contend that Bitcoin may be a criminal playground for two main reasons. Because of its anonymity and because it does not need a trusted 3rd party. Thus, no entity can interfere and interrupt a transaction, as for example, a bank could do. However, apart from simple transactions, it's scripting language does not allow

[8] https://anonhq.com/mirai-ddos-botnet-is-back-for-renting/

more complex activities. The authors argue that by using Non-Interactive Zero-Knowledge (NIZK) proofs and smart contracts over Ethereum, more sophisticated transactions can be made with even stronger anonymity.

Darkleaks

Darkleaks is a decentralized black market where one can sell or buy information. There is no identity, no central operator, and no interaction between leaker and buyers. Hollywood movies, government secrets and zero-day exploits are some of the goods to be sold on Darkleaks. How Darkleaks works:

1. The Contractor C prepares the offering:
   (a) Partitions the secret M into n segments $M = m_1||m_2 \ldots ||m_n$
   (b) Encrypts each m_i under a symmetric key κ_i
   (c) Publishes ciphertexts $C = c_1||c_2 \ldots ||c_n$
   (d) Reveals keys for random subset $\Omega = \{\kappa_i\}, i \in \mathbb{N}^*$
2. Potential purchasers P decrypt the sample.
3. If purchasers like the sample, they give offer.
4. On accepting offer, C reveals a full set of keys $K = \{\kappa_i\}, i \in \mathbb{N}$ and thus M.

Darkleaks approach for fair exchange:

1. Contractor creates a Bitcoin private key $sk_i = SHA256(m_i)$
2. Public key PK_i and address $Addr_i$ are generated.
3. C derives a symmetric key $\kappa_i = SHA256(SHA256(PK_i))$
4. P deposits offer for m_i to $Addr_i$
5. To spend offer from $Addr_i$, C must reveal PK_i,thus κ_i, making decryption of m_i available.

The authors contend that Darkleaks suffer from some shortcomings. For example, C can refrain from spending payments until M loses its value, or C can choose to avoid disclosing some of M's segments. To overcome these shortcomings, the authors propose a smart contract on top of Ethereum written in Serpent. Their implementation integrates a withdraw function in case C aborts, a function to return the donations unless all the secret keys are simultaneously revealed. Finally, they leverage the NIZK proofs technology to verify that κ_i can decrypt c_i. Smart contract plays the role of the verifier.

Calling Card Crimes

Calling card is a signature token or a characteristic of a crime used by a serial criminal. The authors have presented an authenticated data feed system called Town Crier (TC), to act as bridge between smart contracts and existing websites. They argue that calling cards, alongside authenticated data feeds, can support a general framework for a wide variety of Criminal Smart Contracts. They propose a protocol for construction of a criminal smart contract based on a calling card.

1. Perpetrator (P) provides a commitment to a calling card to a smart contract.
2. After the commission of the crime, P proves that calling card corresponds to his initial commitment.
3. Smart contract refers to some trustworthy and authenticated data feed to verify that the crime was committed, and the calling card matches the crime.
4. If both conditions are met, the smart contract pays a reward to P.

Exploiting this protocol, many calling-card crimes become available, like website defacement, DoS attack, assassination, kidnapping and terrorist attacks. Again, using smart contracts there is no need for direct communication between the perpetrator and the criminal.

3.2.3 Financial Frauds

With the interest of the public towards electronic currencies increasing, many skilled frauds have found the opportunity to deceive users in order to steal their money. In this section, we will make a summary about the distinct cases of financial frauds.

Fraudulent ICOs

ICO stands for Initial Coin Offering, it is a procedure by which an entity, usually a person or a company, sells tokens for an amount of cryptocurrency. One could buy such a token, hoping that it will get used a lot and its high circulation will raise its value. In the short history of cryptocurrencies, many fraudulent ICOs have happened. Investors bought tokens in order to increase their profits, but they ended up owning worthless tokens. Fraudulent ICOs have costed over $35 million to cryptocurrency users.[9]

Fake Wallets

Fake wallets, usually met as Android apps, steal user's private key or seed and consequently transfer his/her funds. In many cases they have been found in Play Store as clones of known wallets.

Ponzi Schemes

Ponzi scheme is a fraudulent investment operation where the operator generates returns for older investors through revenue paid by new investors, rather than from legitimate business activities or profits of financial trading. Chen et al. [44] propose an approach to detect Ponzi schemes on blockchain by using data mining and machine learning methods. Using combined account data and smart contract's bytecode the authors built a classification model in order to distinguish Ponzi schemes from other smart contracts.

[9]https://www.businessinsider.in/The-SEC-is-charging-two-cash-and-car-loving-crypto-founders-with-fraud-after-their-32-million-initial-coin-offering/articleshow/63588096.cms

3.3 Ransomware-as-a-Service Usings Smart Contracts and IPFS

The emergence of Ransomware-as-a-Service (RaaS) represents a significant development within the broader evolution of Malware-as-a-Service (MaaS) [45]. As part of this commodified cybercrime ecosystem, RaaS exemplifies the shift from bespoke malware campaigns to modular, subscription-based offerings. This transformation reflects wider trends in the digital economy, where complex tools and attack infrastructures are repackaged as services accessible to a broader range of actors, including those with minimal technical skills. RaaS is, in essence, a "ransomware affiliation program": affiliates spread ransomware to potential victims who, upon infection, pay an amount to the ransomware author in exchange for a (file) decryption key; the affiliate who performed the infection receives a percentage of the ransom. RaaS platforms typically offer not only ransomware payloads, but also backend infrastructure, payment handling, user documentation, and even technical support. We argue that while the model works, we expect it to shift to more enterprise status. The shift is expected since it is known that Law Enforcement Agencies (LEAs) and other organizations try to monitor or even backdoor forums and, as a result, reveal the identity of malware authors. We believe that the key future venue would be further exploitation of decentralized models. Hence, a model where the various func- tionalities are split, operate individually, and are orchestrated through a blockchain is a viable alternative that we will face more in the near future. The research question is therefore to predict this trend and identify weaknesses and gaps to be used to counter such threats. In this work [46], we demonstrate a RaaS attack vector that leverages blockchain and decentralized storage technologies to enhance operational resilience and anonymity. By utilizing Ethereum smart contracts for ransom payments and IPFS to host malicious infrastructure components, we illustrate how the decentralized architecture of such systems significantly complicates mitigation and takedown efforts. In the considered system, we fully utilize blockchain and IPFS technologies. We take advantage of the Ethereum blockchain by using smart contracts as a registration and ran-

som payment service for robust, low cost, fair and anonymous transactions. We also leverage IPFS to reliably host Web pages used for interacting with Ethereum smart contracts, as well as the malicious executable files. This system achieves the following: (1) The amount of time a ransomware author needs to be online is minimal. (2) The ransomware author does not have to have a stable network address. (3) The identities of the ransomware authors and the affiliates are hidden. (4) Affiliates do not have to pay money upfront; instead, malware authors receive a commission from the ransom. (5) Once up and running, it is too hard to take offline the RaaS system, as well as the registration and payment systems.

3.3.1 Related Work

The works discussed in Section 3.2 demonstrate that blockchains and decentralized infrastructures can be misused as persistence, coordination, or communication channels for malware. However, these efforts remain limited compared to a complete Ransomware-as-a-Service (RaaS) ecosystem.

Early work by Moubarak et al. [39] focused on integrity validation by storing only hashes of malware components on the Bitcoin blockchain, without providing distribution or monetization. Subsequent studies such as ZombieCoin [42] and Cerber [41] leveraged Bitcoin for botnet C2 and ransomware coordination, respectively, showing resilience against takedowns but limiting decentralization to communication; Cerber further introduced an affiliate model, though revenue sharing was managed centrally. Zhong et al. [47] extended this line with duplex and stealthy communication across the Bitcoin main and test networks. In parallel, Patsakis and Casino [48] explored IPFS for malware coordination, showing how its content addressing and replication properties could be abused to host and disseminate malicious content.

Table 3.3 summarizes these approaches across key dimensions, including the underlying platform, utilization of infrastructure, support for decentralized storage and file distribution, monetization, and their potential to enable MaaS. As illustrated, prior works either lacked decentralized storage, relied on centralized rendezvous points, or did not integrate payment mechanisms. Among them, Cerber

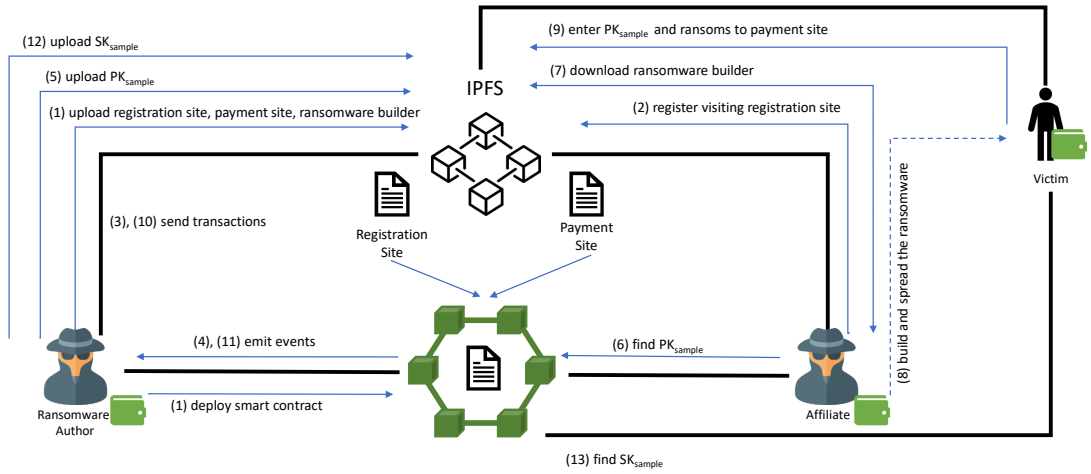


Figure 3.1: An overview of the considered blockchain-based architecture.

stands out as the only concrete RaaS, yet revenue sharing with affiliates remained manual and centralized. By contrast, our work is the first to combine decentralized storage and distribution via IPFS with automated monetization through Ethereum smart contracts, supporting asynchronous communication and trustless revenue sharing, thereby delivering the first fully decentralized RaaS ecosystem. Crucially, the use of smart contracts eliminates the need for backend servers, making the operator location-agnostic and further strengthening resilience against takedowns.

3.3.2 System Design

We now present the design of the considered architecture (illustrated in Figure 3.1). The system is composed of the following entities: the ransomware *author* who creates the original ransomware, some *affiliates* who buy the ransomware from an author and try to infect *victims*. From a high-level perspective, these entities interact with each other as follows. Authors store their ransomware in IPFS and make it available using an Ethereum smart contract. Affiliates obtain it and try to infect other users. Infected users (i.e., the victims) use another Ethereum smart contact to pay the ransom and receive the decryption key. The author and the corresponding affiliate share the ransom. This functionality is implemented using the following steps.

Table 3.3: Comparison of related works with our proposed RaaS system

Paper	**Year**	**Platform**	**Utilization of Infrastructure**	**Decentr. File Storage**	**Storage Cost**	**Monetization**	**File Distribution**	**Need Online C2**	**Enabling MaaS**	**Revenue sharing**
Developing a K-ary malware using Blockchain [39]	2018	Bitcoin	Decentralized (PoE validation)	✗	N/A	✗	✗	N/A	✗	N/A
Zombiecoin 2.0: managing next-generation botnets using bitcoin [42]	2018	Bitcoin	Hybrid (Tx downstream + rendezvous servers upstream)	✗	Rendezvous service	✗	✗	✓	✓(leasing)	✗
Malware coordination using the blockchain: An analysis of the cerber ransomware [41]	2018	Bitcoin + Tor	Hybrid (Wallets + Tor service)	✗	Rendezvous service	✓	✓	✓	✓	✓(manual)
Hydras and IPFS: A Decentralised Playground for Malware [48]	2019	IPFS	Decentralized (IPFS Resource Identifier Generation Algorithm)	✓	✗	✗	✓	✓	✗	N/A
Dustbot: A duplex and stealthy p2p-based botnet in the bitcoin network [47]	2019	Bitcoin mainnet + testnet	Decentralized (Duplex C2)	✗	N/A	✗	✗	✓	✗	N/A
Our work	2020	Ethereum + IPFS	Decentralized (Smart contracts + IPFS)	✓	✗	✓	✓	✗(async events)	✓	✓(auto)

Setup

During the setup phase, the author creates an Ethereum account and stores it in a wallet. We will refer to the Ethereum address of the author as $A_{address}$. Then, she creates a *registration* Web page, a *payment* Web page, and a ransomware *builder*, and uploads them on IPFS. Moreover, she has to create the corresponding smart contracts and deploy them on the Ethereum blockchain. In order for the author to upload the created files in IPFS, she needs an IPFS node running on a computer under her control, but after the files are received by other IPFS nodes, her local node is no longer required. The registration Web page is used by affiliates to join the affiliates program, whereas the payment page is used by victims to pay the ransom. Both pages interact with the corresponding smart contracts and are realized on IPFS for robustness. Finally, the author sets up a C2 application which interacts with all other components *only through the Ethereum smart contracts*. The C2 application is connected to the Ethereum network and it is configured to "listen" for specific events.

Affiliate registration

An affiliate joins the affiliation program through the registration page. During this process, the affiliate registers his Ethereum address, which is stored in the corresponding smart contract, and downloads the builder. For each user he wishes to infect, the affiliate uses the builder to create a ransomware *sample*. Whenever a new sample is created, it uses the registration smart contract to request a *public key* PK_{sample}. This request creates an event which is broadcasted in the Ethereum network; hence it is received by the C2 application; the C2 application generates a public-private key pair using as seed the transaction hash, and returns the public key to the smart contract. The smart contract verifies that PK_{sample} was sent by $A_{address}$. The smart contract stores all PK_{sample} associated with an affiliate address in a data structure.

Victim infection

An affiliate may distribute the ransomware using various mechanisms, e.g. exploiting a system vulnerability, spear phishing, etc.; however, the means to do it are beyond the scope of this work. When the ransomware is executed by a victim, it creates a symmetric encryption key Key_{temp}, which is used to encrypt some critical for the victim files. Then, the ransomware encrypts Key_{temp} using PK_{sample} and stores the ciphertext locally. Finally, the ransomware notice is presented to the user containing the PK_{sample}, the URL of the payment page, and the amount of money, in ether, that the victim must pay to get his files decrypted.

Ransom payment

In order for the victim to get his files decrypted, he has to visit the payment page and enter PK_{sample}, as well as to deposit the predetermined amount. This amount is transferred from his Ethereum wallet to the smart contract's balance, which in return transfers a proportion of the ransom to the affiliate and the rest to the ransomware author. This action emits an event, which is received by the C2 application. The application retrieves the corresponding secret key SK_{sample} and sends it to the smart contract. The smart contract verifies that SK_{sample} was sent by $A_{address}$. The ransomware uses SK_{sample} to decrypt Key_{temp}, and then it uses Key_{temp} to decrypt the files of the victim.

3.3.3 Implementation

We developed a proof of concept implementation of the blockchain-based architecture and we deployed it on the Rinkeby test network. Ethers in the Rinkeby network do not have any real value. Furthermore, the deployed system does not contain a true infection mechanism and is thus not directly usable, i.e., harmless.

The smart contracts of the system were developed using Solidity. These contracts implement seven functions, corresponding to the actions of the system. The first function is used for the affiliate registration; it stores the necessary information in the blockchain, and it emits an event when the process is completed.

Two other functions are used for setting and getting a PK_{sample}. A fourth function implements ransom payment, and it is responsible for triggering the corresponding event. A fifth function is responsible for splitting the ransom between the affiliate and the author. Finally, there are two functions for setting and getting SK_{sample}.

As we have already mentioned, the ransomware author is expected to create the two sites and upload them to IPFS. The sites are implemented as React applications with a simple user interface. The sites are interacting with the smart contracts, using the web3 JavaScript library.

Finally, the implementation of this system includes the C2 application that is executed on behalf of the ransomware author. This is a script developed in Node.js.

3.3.4 Evaluation

Performance and Cost Evaluation

All actions performed in the system involve the invocation of the smart contract functions discussed in the previous section. Two of them only read the state of the blockchain; thus, they have no cost, no significant delay, or serious overhead. The deployment of the smart contract in the blockchain network (in our experiments was the Rinkeby test network), as well as the cost (measured in gas) for invoking the contract's functions, are shown in Table 3.4.

Actor	**Operation**	**Cost measured in gas**
Ransomware Author	Deployment	505822
	PK_{sample} Upload	29881
	SK_{sample} Upload	22144
	Ransom Split	37515
Affiliate	Affiliate Registration	22796
Victim	Ransom Payment	28326

Table 3.4: Cost of the construction building blocks

Three of those five operations are initiated by the ransomware author, one

by the affiliate, and one by the victim. Therefore the malware author is billed with a total cost of $TC = 505822 + 29881 \cdot \rho + 22144 \cdot \delta + 37515 \cdot \mu$ gas units, where ρ is the number of registrations, δ is the number of SK_{sample} keys generates, and μ is the number of payments received. At the time of writing, the equivalent of 1 Ether in fiat currency is \$175,59[10] and the gas price is set to 1Gwei. Assuming that $\rho = \delta = \mu = 100$, the total cost for the ransomware author for 100 registrations and payments will be $\approx \$1,67$ which is minimal by any standard.

In addition to the gas cost, Ethereum also adds an execution time overhead related to the time a transaction needs to be mined. This time depends on the gas price, which is the amount of Ether that a user pays per unit of gas. The higher the gas price, the faster the transaction will be mined. On average, an operation in Ethereum is executed in ≈ 13 seconds.

Security and Privacy Properties

The system has some intriguing security and privacy properties. The Ethereum blockchain offers a high degree of anonymity, as it is hard to track an Ethereum address back to its real-world owner. Therefore, authors and affiliates cannot be easily detected. Similarly, once the ransomware and the Web pages are stored in IPFS, the author does not have to participate in the IPFS P2P network.

An author does not have to be constantly online; neither has he to use the same device or network location. An author's C2 application is triggered by Ethereum events; however, these events are broadcast on the whole Ethereum network. The smart contract controls who can write PK/SK_{sample}. This access control is implemented by examining the Ethereum address of the entity that made the corresponding transaction; hence it does not reveal any real-world information.

3.3.5 Discussion

Two key properties of our system are that a) it can easily bootstrap, and b) it is hard to take it offline.

[10] `https://coinmarketcap.com/currencies/ethereum/`

The system can be easily adopted, since it does not require any upfront payment by the affiliate, apart from the cost for interacting with the blockchain. Instead, affiliates share the ransom with the ransomware author. Ransom sharing is done automatically by a smart contract.

IPFS and Ethereum are two robust systems where information is permanently stored. Therefore, once the ransomware and the Web pages are in the IPFS, and the smart contracts are deployed in Ethereum, they cannot be removed. IPFS provides a "blacklisting" functionality, which is optional. Similarly, smart contracts can only be removed using chain "forking", which is an extremely unlikely process. On the other hand, smart contracts cannot be modified; hence they must be carefully designed so as to be bug-free. The major drawback, for criminals, of the proposed scheme is the fact that all the transactions are visible from any participating node of the blockchain. Therefore, a LEA can identify, for instance, when a new affiliate has been recruited, a payment has been made, etc. Nevertheless, the same applies to many wallets which have already been identified with criminal activity. This is the reason for the rise of crypto laundry services, which can efficiently anonymize such revenues [49].

It is important to mention that field values in smart contracts, even though they are not declared as public, are still accessible. A smart contract's state is visible to anyone using a blockchain explorer [38]. It is also important to note that even if the smart contract stores only information about the last registration, someone can find information about the previous registrations on the blockchain. But note that every piece of information sent through the smart contract can be encrypted with the recipient's public key so that only he can decrypt it using his private key, locally.

Chapter 4

Security Threats in IPFS

The InterPlanetary File System (IPFS) has emerged as a key building block for the Web3, offering content-addressable, peer-to-peer file storage and retrieval. Designed to overcome the limitations of traditional client-server architectures—such as centralized control, single points of failure, and high infrastructure costs, IPFS aims to provide increased availability, censorship resistance, and resilience. However, these benefits come with a host of new security and privacy challenges that remain insufficiently addressed.

This chapter aims to explore these multifaceted threats by (i) analyzing the architecture and design choices that expose IPFS to abuse, (ii) presenting real–world examples of malicious activity and privacy leakage, (iii) discussing current and proposed countermeasures for securing the IPFS ecosystem.

4.1 Malicious IPFS nodes under the magnifying glass

As IPFS becomes increasingly integrated into Web3 infrastructures, its open-access and decentralized architecture is being leveraged not only by legitimate users but also by malicious actors. Unlike traditional web infrastructures, IPFS lacks centralized oversight, allowing any node to participate freely in content storage and distribution. This raises pressing questions regarding the integrity and

trustworthiness of the nodes that constitute the network. To this end, we aim to unravel the structural elements of the IPFS net- work, and the nodes, focusing on suspicious activity. Initially, we crawl the IPFS network to enumerate it and make the first contact with the nodes. Following that, we collect intelligence from different sources regarding the aforementioned nodes. Moreover, we collect the exchanged data by nodes and analyse them to have a deeper understanding of the consistency of the network. Finally, we try to determine the extent of possible abuse of IPFS for copyright infringement [50].

4.1.1 Background

`JARM` [51] is an open-source, active fingerprinting tool by `salesforce`. JARM uses TLS to identify a target host and extract information from it. Specifically, JARM crafts ten different `Client Hello` packets of the TLS handshake process and collects all data from the process, e.g. cipher negotiation and supported cipher suites. The corresponding responses received from the host vary depending on the underlying operating system, software, libraries, version, order of calling the libraries etc. Consequently, JARM aggregates the responses and produces a hybrid fuzzy hash which are 62 characters long. This hash can be broken down into two parts. The first 30 bytes try to describe how the server reacted to each of the ten client hello messages. The rest 32 characters are a SHA256 hash which summarises the extensions sent by the server without the certificate data. JARM is used by the community as a software-wise host clustering tool, therefore it is also eligible to detect malware Command & Control (C2) servers as their fingerprints are often very unique and therefore distinguishable.

4.1.2 Related Work

P2P networks have been of interest to the scientific community for many years, and while their popularity fluctuates, they have never been outdone. In recent years, the advent of cryptocurrencies and blockchain technology has brought them back into the limelight. Thus, while P2P node profiling has been extensively

studied in the past, research in the context of Web3 is minimal. Web3 is in a very early phase and its decentralised components are still under heavy development. Hence, the current research regarding its nodes is still in its infancy. Henningsen et al. in [52] make one of the first attempts to explore the IPFS network. Adopting a hybrid design, passively and actively, they aim to enumerate the IPFS network and profile its nodes. The authors note that the overlay network outperforms the overlay induced by buckets. Furthermore, they observe that an overwhelming percentage of nodes, i.e. 94%, did not react to the authors' attempt to connect to them. The reason this happens is twofold. The first is because many nodes are behind NAT and thus advertise their local IP address. The second is that a large portion of users uses IPFS in an opportunistic way, therefore their footprint remains in buckets for longer than they remain online and connected.

Recently, researchers discovered a botnet hiding in the IPFS ecosystem [6]. The latter, named InterPlanetary Strom (IPStorm) and estimated size of 9000 devices, utilises IPFS at multiple levels. Initially, the researchers found that it uses the libp2p DHT to discover nodes. Bots identify each other with the attribute `Agent Version`: "storm". In addition, the botnet utilises the Pub/Sub protocol as a communication channel over specific topics. Finally, the botnet uses IPFS to share files so that it can be updated to a newer version.

Trautwein et al. [4] further to providing a basic guide of IPFS' design, they collected data from three different sources to shed light on various metrics related to IPFS performance. Initially, they crawled the IPFS network to gather information about peers. Among the conclusions drawn is that IPFS nodes are geographically distributed in 152 countries, yet more than 50% are located in just two countries, US and China. Furthermore, more than 50% of the IPs are covered by five automated systems, yet only 2.3% of the nodes are in some cloud infrastructure. The last insight extracted from this dataset is that the IPFS network suffers from high rates of churn, with 87.6% of peers having an uptime of less than 8 hours. Finally, the authors wanted to study the time performance in downloading data. To this end, they experimented with different AWS regions and recorded the download duration from the data they produce each time. In 50% of the cases, the download

took less than 3s, and in 90% of the cases, less than 4.5s.

4.1.3 Profiling IPFS nodes

Data collection methodology

To enumerate the IPFS network we used the IPFS Crawler [52]. The IPFS crawler is a tool written in Go and is based on libp2p (v0.11.0).

Acting as a Kademlia node the crawler uses precomputed keys to extract all the entries from most buckets for every node it encounters. In essence, it invokes FindNode actions repeatedly using the appropriate precomputed keys. Finally, the crawler produces two files: (i) a JSON file storing the tuple <`PeerID, multiaddress, agent, reachability`> for every distinct node met, and (ii) a CSV file containing all the pairs of connected nodes.

We conducted a series of consecutive crawls. Initially, the crawls were performed iteratively, every ten days during the period from March to April 2022. Each crawl series spanned over a day (24h) totalling about 360 crawls in a row per day. From the data in the JSON file, for each PeerID we extracted the IP addresses. Each IPFS node maintains an address book retaining information for the nodes it encounters. If any of the encountered nodes advertises a new address, then it is appended in the address book for reachability purposes. As a result, a single PeerID may correspond to more than one IP address. We studied each different address considering it as a unique node. Moreover, nodes behind a firewall or NAT use `p2p-circuit`, a libp2p relay transport protocol, to avoid connectivity barriers. In essence, these nodes advertise addresses through relay nodes. As a consequence, they do not reveal their real IP address but the IP address of the relay. The aforementioned peers as well as those which advertise only local IP addresses are excluded from our analysis. Clearly, the absence of such IP addresses prevents us from studying or fingerprinting the corresponding hosts.

Node Profiling

In this section, we present general information regarding the IPFS network and its nodes. We should mention that in the following findings, every different IP address is considered a different node. Although, we found that unique Peer IDs advertise multiple IP addresses since our study focuses on the "fabric" of the IPFS network. Thus, we want to enumerate and analyse every different IP address.

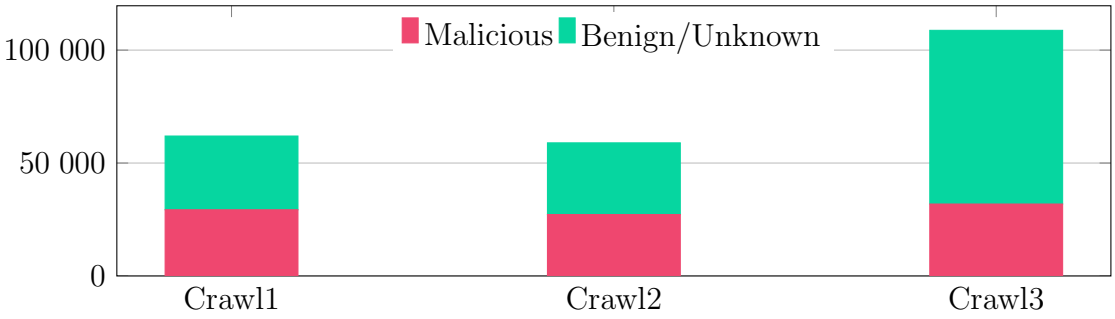
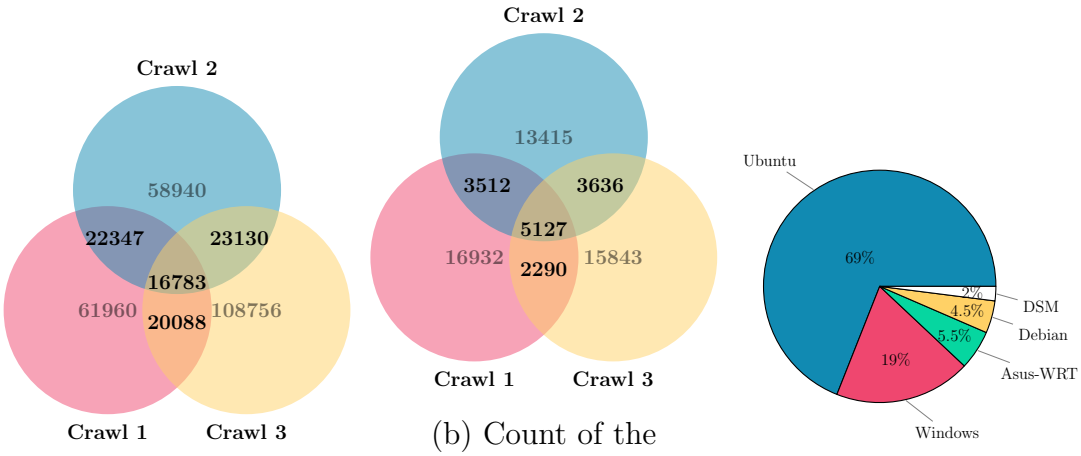


Figure 4.1: Malicious nodes per crawl.

Figure 4.1 illustrates the nodes per crawl and the count of malicious nodes for which we collected intelligence. In Figure 4.2a, the exact results of IP addresses per crawl can be found. Moreover, from the same figure, we can observe that 16783 were found online in all three crawls. We can assume that the aforementioned nodes were found online at least once a day in the span of the whole month. Given the periodic changes of IPs, we can assume that most of these IPs belong to some infrastructure that has been devoted to constantly working with IPFS.

A node's agent version can be an indication of malicious activity. Nodes' agent version is public and advertised, thus, it can act as an identifier for malicious nodes to discover and track each other. The latter is a technique already implemented by "storm" agents. Figure 4.3 illustrates the ten most used agent versions we found in each crawl. We should highlight that the counts depicted correspond to the agents from the nodes we managed to connect to. In each crawl we found 50%, 61%, 49% respectively, unreachable peers, i.e., we found their address stored in the DHT but they were offline. Moreover, IPFS is open-source software; therefore, it is at the user's discretion whether to display the agent version. The latter results



(a) Count of IP addresses per crawl.

(b) Count of the malicious IP addresses per crawl.

(c) Top five OS used by IPFS nodes.

Figure 4.2: Crawl statistics.

are aligned with the ones in [4]. In the third crawl we observe that there is an increase in nodes using the agent called `Hydra Booster`.[1] Hydra Booster is a node having many different Peer IDs over a common routing table. It is designed to accelerate IPFS' processes carried out through DHT-like content resolution, routing and discoverability. The existence, as well as the operation of these nodes, brought about an increase in the number of nodes of the third crawl. One of the features of open software, which has been hotly debated lately, is that upgrading to a newer version is at the user's discretion. Observing the crawling results of Figure 4.3, one can observe that there are many different software versions running and communicating simultaneously. For example, `go-ipfs 7.0` was released in July of 2020 while `go-ipfs 11.0` in August of 2021. Moreover, although the measurements were made in mid-2022, and version `go-ipfs 12.0` had already been released, we can conclude from the bar charts that the versions which are more widely used are the older ones. In addition, we must mention that agent storm, which has been found in all three crawls with a non-negligible number, is characteristic of the nodes belonging to the IPStorm botnet we have already mentioned.

In what follows, we study the maliciousness of nodes, so we used Virus Total

[1] `https://github.com/libp2p/hydra-booster/`

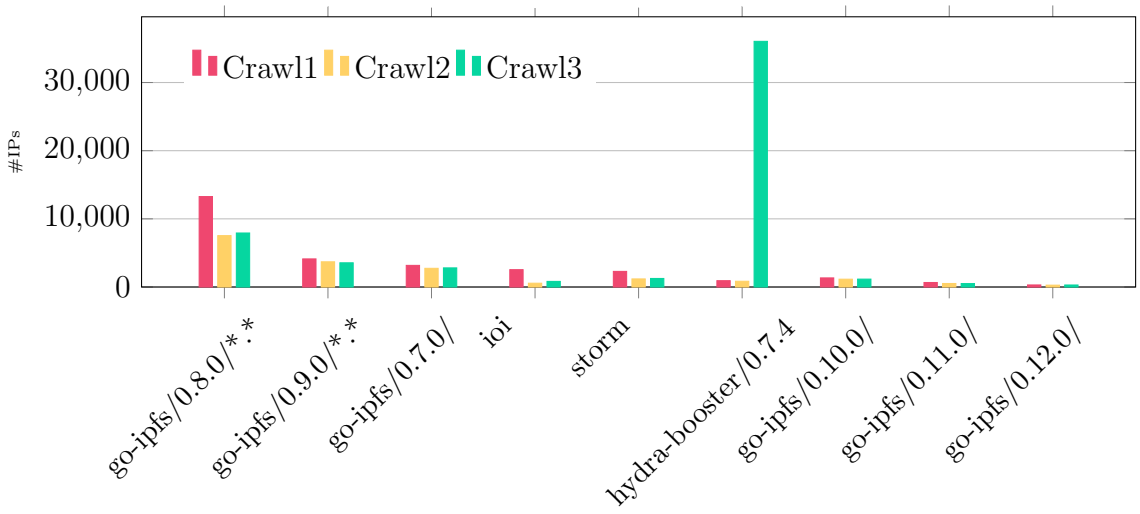


Figure 4.3: The ten most commonly used agent versions in each crawl. The `*.*` denotes varying subversions combined.

to assess the corresponding IPs. Nevertheless, Virus Total also provides valuable insights regarding the geographic distribution of the various nodes, regardless of whether they are malicious or not. The vast majority of the nodes are located in two countries, namely the United States and China. We notice that our results are aligned with [4].

To conduct a more in-depth analysis, we passed the crawling results to intelligence services. Namely, we used Shodan,[2] a network monitoring tool, to fingerprint each node. Shodan returned intelligence for approximately 40960 unique nodes. Figure 4.4 illustrates the ten most commonly used ports by the total of nodes we examined. Port 22, the most widely used port by IPs related to IPFS, is typically used for Secure Shell (SSH) connections, which allow users to log in to a host and execute commands remotely. Port 80 is used as the default port for HTTP (Hypertext Transfer Protocol) traffic, port 8080 is an alternative to port 80 and moreover the default port of the IPFS gateway, and port 443 for HTTPS. Port 3389 is typically used by hosts running Microsoft Remote Desktop Protocol (RDP) to allow remote access to the host's desktop. Finally, port 4001 is used by default for IPFS traffic, but users can also set up a custom port. Regarding

[2] `https://www.shodan.io/`

the operating system running on IPFS nodes, Shodan's results, depicted in Figure 4.2c, indicate that the lion's share uses Ubuntu Linux. The next runner-up is Microsoft Windows 10, followed by Debian Linux. The latter is also exhibited by the most used services, Figure 4.5, where most hosts appear to be using SSH as opposed to RDP. Moreover, most of them seem to have a web server (nginx and then Apache).

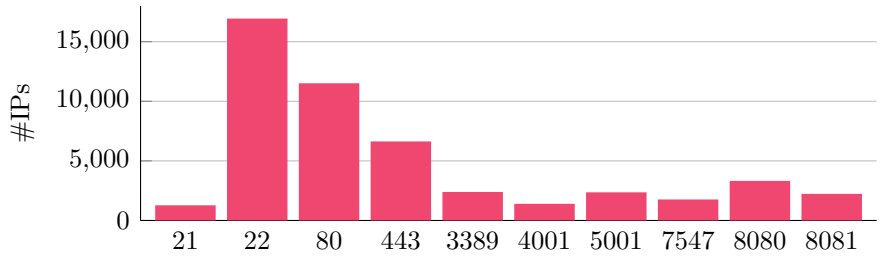


Figure 4.4: The ten most common ports.

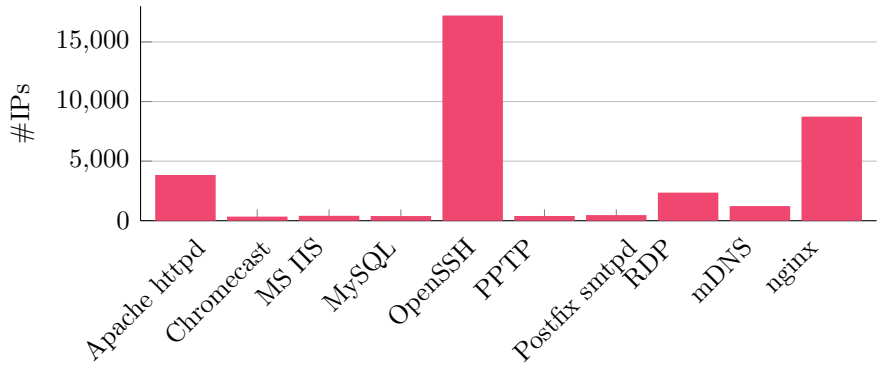


Figure 4.5: The ten most commonly used services.

JARM [51] is an open-source fingerprinting tool that generates a string based on the response of the host to ten TLS packets. JARM is used by the community as a software-wise host clustering tool, therefore it is also eligible to detect malware Command & Control (C2). We use JARM strings, extracted from Shodan and Virus Total, to detect any similarities among the different nodes. Finally, we combined them since for the same IP different services can provide varying information. For 1002 IP addresses, we found information in both services, so we considered both records. The JARMs indicate that there are several clusters

of IPs in which servers have the same TLS configuration, which implies that the same entity is behind them. The most common ones are illustrated in Table 4.1.

JARM	# IPs
2ad2ad0002ad2ad00042d42d0000008aec5bb03750a1d7eddfa29fb2d1deea	2070
2ad2ad16d2ad2ad22c2ad2ad2ad2adfd9c9d14e4f4f67f94f0359f8b28f532	1378
15d3fd16d29d29d00042d43d000000fe02290512647416dcf0a400ccbc0b6b	577
15d3fd16d29d29d00042d43d0000009ec686233a4398bea334ba5e62e34a01	562
15d3fd16d21d21d00042d43d000000fe02290512647416dcf0a400ccbc0b6b	489

Table 4.1: Most common JARMs.

Malicious Activity

In this section, we investigate the moral character of IPFS nodes, i.e., we examine whether and to what extent there are malicious nodes. To this end, we collect and leverage existing intelligence to create and present their profile. Our goal is to assess the network structure, keeping IPFS users and the related community alert to the existence of malicious activity in the IPFS network. Due to the current IPFS rules, every node maintains several active connections varying from 600 to 900 peers. Thus, we argue that it is very important for each node to know what kind of alignment, i.e. neutral or malicious, the node it interacts with has.

Initially, we leveraged the intelligence provided by two popular services, namely Virus Total (`https://virustotal.com/`) and SpamHaus (`https://www.spamhaus.org/`), to get a baseline for the reputation and past activity of nodes. SpamHaus uses several methods to find information about an Internet resource. It uses sensors in large networks, i.e. a data-sharing community, from which it collects data about network traffic. In addition, SpamHaus deploys honeypots to attract malicious users. Along the same lines as SpamHaus and VT, in addition to monitoring more than 70 anti-malware and IP blocking services, it relies on data generated and shared by an already large community. Both the aforementioned

services provide APIs to interact with their knowledge base and generate a JSON formatted output for each request. We combine the extracted output information with the SpamHaus output and we consider malicious those nodes with at least one record in one of the aforementioned services.

Moreover, in Figure 4.2b, we notice that from the 27861 different IP addresses we encountered during the first crawl, 5126 of them, $\approx 18\%$ remained online throughout the whole month. The latter indicates that there is a number of nodes that constantly utilise the IPFS network for malicious purposes. Compared to the 16783 found online in all three crawls, as depicted in Figure 4.2a, a significant part of them, i.e., 30.5%, are known to be malicious. Based on SpamHaus' results, we conclude that the majority of malicious nodes were discovered using the `DNS Sinkhole` technique. According to this technique, security researchers create, at various levels, a DNS record of a known malicious URL pointing to an address they own, usually a sinkhole server. The gain from applying this technique is twofold: On the one hand, they prevent communication between bot and C2, and on the other hand, researchers can find which computers are infected, i.e. ask to connect to known malicious URLs.

In Table 4.2a, the five most commonly requested and sinkholed URLs in the number of unique IP addresses are illustrated. Note that several URLs such as `differentia.ru`, `atomictrivia.ru`, `amnsreiuojy.ru` and `restlesz.su` are known to be leveraged as C2 by malware. `disorderstatus.ru` is a relatively newly created domain reported to be mostly used for spamming. To draw deeper conclusions about the URLs, we isolated the `Top Level Domain` (TLD) of the different requested URLs. To our surprise, while most requested URLs have a ".ru" TLD, this is not reflected among the unique TLDs. On the contrary, we notice that the most commonly encountered is ".xyz", a relatively new TLD offering many domains that would traditionally be registered by legitimate users. The fact that they are new and cheap and that traditional domain names are available has led `xyz` domains to be widely exploited[3]. Given that 11227 `xyz` domains are hosted

[3] `https://www.spamhaus.com/resource-center/getting-the-low-down-from-xyz-registry-on-combating-domain-abuse/https://www.bleepingcomputer.com/news/security/these-are-the-top-level-domains-threat-actors-like-the-most/`

by these addresses makes us conclude that some adversaries use nodes of IPFS for hosting malicious domains in addition to C2 infrastructure.Tinba a portmanteau of the words Tiny Banker, is a trojan that leverages packet sniffing to determine whether the user visits a bank's webpage. In that case, the trojan tries to steal the keystrokes and sends them to a C2. Nymaim and Ranbyus are well-known trojans, which steal information from the user and consequently send them to a C2. Some of their variants have been found to use domain fluxing to communicate with their orchestrator, and some have been found in DoS attacks. Mirai is used to infect Internet of Things (IoT) devices and turn them into bots that can be used to launch large-scale network attacks. The Mirai botnet was initially discovered in 2016 and was part of various high-profile cyberattacks, including distributed denial-of-service (DDoS) attacks that brought down popular websites and online services. The most frequently displayed campaigns are gathered in Table 4.2b.

URL	**Count**
`differentia.ru`	38681
`disorderstatus.ru`	15504
`atomictrivia.ru`	7049
`amnsreiuojy.ru`	5662
`restlesz.su`	2180

(a) The five most sinkholed URLs and the number of unique requests.

Campaigns	**Count**
tinba	30019
conflicker	22650
nymaim	22228
andromeda	6403
ranbyus	4845
mirai	3750

(b) Malware campaigns with the largest participation from the encountered nodes.

Table 4.2: Extroversion of malicious nodes: Which groups do they belong to and what webpages they seek to visit.

Finally, we studied the JARMs of malicious nodes to better frame our research. As we have already mentioned, we combined knowledge from all intelligence services to produce the results. Notably, among them, we found a cluster of 68 nodes corresponding to the JARM fingerprint `15d3fd16d29d29d00042d43d00`

`00009ec686233a4398bea334ba5e62e34a01` which is attributed to the notorious `emotet` botnet.

As already mentioned, the crawler we used, in addition to information about the nodes encountered, produces an edge list with each pair of connected nodes. Based on this, we constructed a mapping from one PeerID to the several PeerIDs we found connected during the second day. In essence, we built for each peer its buckets expanded to the span of a day. Consequently, we converted the aforementioned mapping to the corresponding IP addresses. This way, we can investigate whether there is a clique between the malicious nodes. The findings indicate that there is no such clique, as the median percentage of malicious nodes in the buckets of a malicious node is 7%, and the average is 9.5%. Along the same lines, the median percentage of nodes in the buckets of a benign node is also 7%, with the average being 9.2%.

4.1.4 File Investigation

Despite the processes and functionality IPFS offers through libp2p and its other components, its main purpose is undeniably storage-related. The largest NFT marketplaces use IPFS for the data storage and integrity it provides, while its widespread utilisation has already brought about the need for cooperation with other Web3 layers, such as ENS, which natively offers names corresponding to CIDs. No wonder the increasing popularity has also caught the eye of cyber criminals. A recent research[4] highlights that the volume of malware samples hosted in IPFS has increased during 2022. Moreover, researchers report the `Agent Tesla` malware, which using phishing techniques, leads to an IPFS public gateway, disguising the download of malicious content. To better frame our research into the storage of the IPFS ecosystem, we also researched the file side. Our research is twofold, in the first case, we eavesdropped on the files requested by IPFS users, while in the second, more actively, we searched for files we randomly downloaded from well-known torrent sites.

[4] `https://blog.talosintelligence.com/ipfs-abuse/`

Bitswap Eavesdropping

According to the operating rules of IPFS, when a user searches for a file, a one-hop inquiry is first performed through Bitswarm, requesting it from nodes with an active connection to the initiator. If none of them responds, the query is then served by the DHT. To collect data, we tweaked our node so that it maintains active connections with around 4000 nodes; that is, according to our measurements, approximately 20% of the network's active nodes at that time. So when one of those nodes was looking for a file, thanks to Bitswap's functionality, that information would also go through us. This way, we could eavesdrop on about 20% of the network's requests and, in turn, request back to retrieve them. In total, we monitored the requests for 24 hours while we set each request to last no more than 15 seconds. This way, we avoided downloading very large files while, on the other hand, we cancelled the search in case it was routed through the DHT. In total, we collected 49155 files with a size of about 13.7 GB. To have a more complete picture of the type of files requested, we used the Python `mimetypes` module[5] to find the MIME type of each file. We shall mention that it managed to classify 13691 of the files. The latter can be attributed to Bitswap's design. When a user requests a file from Bitswap, the search is performed by the root CID of the file. The aforementioned file contains links to the chunks of which it is composed. Thus, when the requester receives the root CID and learns the CIDs of the chunks that make up the file, it requests through Bitswap consecutively all the chunks, which are essentially blocks of data. The file results illustrate that 3716 are image files with MIME types "image/png", "image/gif", "image/jpeg", and 9148 are JSON files, which is the most common format for NFT metadata. The latter clearly demonstrates and confirms our initial statement that IPFS is a cornerstone of NFT data storage and Web3 in general. Among others, we fetched 177 Javascript files and 27 videos of type "video/mp4". We then fed the image files to the Python `Not Suitable For Work (NSFW) Detector` module to determine whether IPFS is being used for inappropriate content. From the 1636 image files it examined successfully, it found

[5] `https://docs.python.org/3/library/mimetypes.html`

33 unsuitable.[6] The above indicates that some users leverage IPFS' anonymity to host inappropriate content that is difficult for LEAs to track and take down.

Torrent Files

Very often, inappropriate files are found in the form of torrent files disseminated through torrent search engines. We downloaded a sample from various widespread torrent sites, ten popular torrents in total. We computed their CIDs locally to determine whether they are shared on the IPFS. This way, not only did we not add any illegal files to the IPFS network, but we also limited the possibility of tampering with the results of our upcoming searches. The ten different torrent files yielded 72 different root CIDs. Each torrent file can contain a video file, a cover image for the video file, a text file with information about the file, etc. In turn, we made 72 requests to the DHT for providers of these CIDs. We found providers for seven of them, and in fact, for most of them, more than one. The latter implies that IPFS users may also share the same content in torrents and that intellectual infringement content is also distributed through IPFS.

4.1.5 Countermeasures

The amount of malicious nodes connected to IPFS is alarmingly high. Given the P2P nature of IPFS and its continuous exploitation, we believe that pruning nodes from the network might provide an initial measure of sanitising the network; otherwise, the benign peers facilitate the malicious ones. To this end, we opt for a periodical blacklist approach that is resolved through InterPlanetary Name System (IPNS). In essence, we propose using the proposed data crawling methodology to monitor the nodes on a daily basis, the IPs are collected and using intelligence services, we determine whether the IP should be blocked or not. Each IP is four bytes long, so the expected size is rather small and easy to manage. For instance, using our experiments as a baseline, using the worst estimate of 32000 malicious nodes, the blocklist would be around 125KB if the IPs were directly stored (4 bytes per IP). Given its size and possible optimisations (e.g. use binary search

[6] `https://pypi.org/project/nsfw-detector/`

over the sorted list), searching whether the connected peers are malicious can be very efficient. Moreover, since the amount of nodes is tolerable, the collection of data from intelligence services can be rather fast. Of course, one could hide the IPs using approaches based on Bloom filters [53]. In this case, one would need less than half of this storage (almost 56KB) to store these IPs with 0.01% possible false positive. However, the issue is that this error would be persistent, meaning that the nodes that would be false positives would be considered malicious by everyone without being able to rectify this error. Nevertheless, with the growth of IPFS and the increase of malicious nodes, probabilistic structures such as Bloom filters might be more optimal.

IPFS is becoming institutional, after all, many organisations are participating in it and supporting it. Recent research efforts indicate that it could frame the existing banking system [54], while at the same time, it constitutes a cornerstone of Decentralised Finance (DeFi). Our research does not intend to act as a brake on its use; on the contrary, it intends to inform, alert and promote its secure use. For instance, the network administrator of an organisation participating in the IPFS network can block the traffic towards and from a suspicious IP address by adding a rule to the firewall. Note that it can also remove alert fatigue from SOCs who might observe malicious IPs connected to the monitored infrastructure due to IPFS traffic. Finally, while IPFS provides the ability to disconnect from a node, it does not provide natively the option for the user to maintain a blacklist.

4.1.6 Conclusions

Open and decentralised systems are, by their very nature, prone to several attacks. However, given the crucial role of IPFS for Web3, it is essential to protect the ecosystem. Our measurements indicate that an alarming number of IPs reported as malicious through intelligence services are using IPFS. Rather than making it centralised, we opt for soft measures that allow nodes to isolate malicious ones selectively. We argue that this isolation can significantly benefit the network as the content of most of these nodes may be malicious, leading legitimate ones to facilitate nefarious acts and malicious campaigns. Therefore, their isolation, in

the long run, may increase the robustness of the network and trust in it.

IPFS seems to have sacrificed part of the privacy to succeed in terms of performance, speed, and robustness [7]. This shortcoming can be exploited for malicious purposes, but it can also be leveraged by security analysts to monitor malicious nodes. Thus, apart from the fact that we can obtain critical information regarding a malicious node, such as its IP address, we can also monitor it from a content point of view, i.e., its requests as well as what it provides. Therefore, a future direction of this work is an extension of the implementation of the proposed filter so that it associates malicious nodes with the corresponding content.

4.2 Investigating Anonymity Abuse in IPFS

As IPFS continues to grow in popularity, a variety of supporting services—such as pinning services and public gateways—have emerged to enhance its functionality and accessibility. Pinning services play a crucial role in maintaining file availability across the network. These services allow users to ensure that specific files remain accessible by hosting them on dedicated nodes, even if the original uploader goes offline. In parallel, public gateways act as bridges between the IPFS network and the traditional Web, enabling users to retrieve IPFS-hosted content through standard HTTP protocols without running a local node.

Recent works have shown that malware increasingly leverages benign Internet services to distribute payloads and evade detection. This includes both centralized platforms such as GitHub and Dropbox [55], and large-scale abuse of cloud services like Discord, Mediafire, and Google Drive [56]. Our work extends this threat model to decentralized infrastructures like IPFS, where anonymity, content immutability, and the absence of centralized moderation create an even more permissive environment for abuse. In the following analysis we investigate how malicious actors can exploit existing technologies within the IPFS ecosystem to anonymously upload and distribute content. We begin by mapping the current landscape of tools and protocols used to add and access content on IPFS, including pinning services and public gateways. We then design and evaluate practical at-

tacks that leverage these mechanisms to achieve anonymity and persistence within the network. Finally, we explore potential countermeasures to mitigate such exploits.

4.2.1 Adding a File to IPFS

There are several ways to add a file to IPFS. In this section, we explore different methods and their respective modi operandi. Additionally, we examine the information about the original uploader that can be retrieved for each method and the duration that the files remain online.

IPFS Node For the average user, the primary option for connecting to the IPFS network is the IPFS Desktop application, which supports the most operating systems and includes the functionality of an IPFS node within a user-friendly graphical interface. There is also a command-line version available called Kubo. A detailed description of this process can be found in Section 2.2.1. It is also worth mentioning that the Brave Browser natively supports the use of IPFS in conjunction with a local node [57], yet earlier versions provided the ability to add files via Public Gateways.

Pinning Services IPFS, according to its design principles, does not provide a mechanism to ensure that files added to the network remain online if the original uploader deletes them or disconnects from the network. Files are primarily cached by requesters to ensure their availability to other nodes. The more popular a file is, the higher its chances of staying online for an extended period. Additionally, every IPFS node runs a garbage collector to free up storage space. As a result, cached files are periodically removed, leading some files to disappear from the network over time [28]. To prevent the garbage collector from removing a file, the user must pin it. Pinning can be categorized into two types: local pinning, where the user configures their node to retain the file, though it will fade once the node disconnects from the network; and remote pinning, where an external provider takes the responsibility to ensure that the file remains pinned [58].

A plethora of pinning services is available, with Pinata, Filebase, Fleek, and 4EVERLAND being among the most popular. These platforms offer user-friendly graphical interfaces for adding files to the IPFS network, simplifying the process for the average user. Moreover, they provide free storage space for uploading and pinning files, making them accessible to a wide range of users. Once added, the files can be retrieved through public gateways, which act as HTTP access points to the IPFS network.

Although Web3.Storage and NFT.Storage[7] are not strictly classified as pinning services, their functionality closely resembles traditional pinning solutions, so we include them in this section for completeness. These open-source services, developed by Protocol Labs, are designed to store general and NFT-related data, respectively, in the Web3 era. Both services operate decentralized, leveraging IPFS for content addressing and Filecoin for long-term data preservation rather than offering a pinning service. Web3.Storage is notably free for the community, while NFT.Storage operates under a paid model. NFT.Storage was excluded from further experiments, as it specializes exclusively in NFT metadata storage, which falls outside the scope of our analysis focusing on general-purpose file uploads.

Public Gateways Public gateways act as HTTP entry points to the IPFS network, bridging the Web2 and Web3 ecosystems. They process HTTP requests containing CIDs and relay them to an IPFS node, enabling broader access to the network through conventional Web protocols. Although users cannot directly upload files through a gateway, indirect methods enable this functionality, justifying their classification in this section. Furthermore, the HTTP servers underpinning these gateways leverage caching mechanisms, most commonly the Least Recently Used (LRU) strategy which optimizes performance and user experience by evicting the least recently accessed content when the cache reaches its capacity [4]. Based on the above, it is evident that even if the original uploader disconnects from the IPFS network, the file may remain accessible, cached by gateways, with its persistence primarily influenced by its popularity. During the preparation of this

[7] `https://nft.storage/`

study, we identified 10 online gateways.[8] Using the fingerprinting tool WhatWeb,[9] we found that nine gateways utilize either Nginx software or Cloudflare proxies, which employ the LRU caching strategy to manage content efficiently.

The fact that public gateways serve as a bridge between the traditional Web and the P2P ecosystem of IPFS makes them very crucial for launching and countering several attacks. For instance, an adversary may host a phishing page on IPFS; however, the content must be rendered from the victim's browser. Thus, the bridge fetches the content from IPFS and brings it to the Web. It must be noted that while there is no official deletion mechanism for IPFS [59], some public gateways follow blocking mechanisms to prevent specific content from reaching the Web [60]. Nevertheless, not all gateways follow the same blocking mechanism and, of course, this does not remove the content from IPFS.

4.2.2 Exploiting IPFS for Anonymity: Attack Scenarios

The anonymity offered by IPFS can be exploited by malicious actors. In this section, we analyze how attackers leverage methods discussed in Section 4.2.1 to achieve anonymity, presenting and evaluating two distinct attack scenarios.

The Pinning Service Attack

Pinning services ensure that a file remains online. Therefore, it is logical to consider that an attacker could exploit these services to upload a file and guarantee its availability. However, since our focus is on evaluating the level of anonymity, we first examine the information each pinning service requires from users to allow file uploads, i.e., the Know Your Customer (KYC) procedure. We selected Pinata, Filebase, Fleek, Web3.Storage, and 4EVERLAND based on a systematic Internet search. Specifically, we performed Google queries such as "top IPFS pinning services" and "most popular IPFS pinning services," identifying the services most frequently mentioned in developer documentation, technical articles, and community discussions. Academic literature specifically evaluating IPFS pinning services

[8] `https://ipfs.github.io/public-gateway-checker/`
[9] `https://github.com/urbanadventurer/whatweb`

remains limited, further justifying the need to consult current developer ecosystems and real-world service availability. Besides the selected providers, our search also highlighted Infura and Temporal. However, Infura currently restricts access to pre-qualified customers,[10] and Temporal appears to have discontinued operations. Thus, our study focuses exclusively on active and publicly available services, realistically representing the infrastructure accessible to potential anonymous attackers.

The Pinata, Fleek, and Filebase services require an email address for user registration. To achieve higher levels of anonymity, we attempted to use a temporary email service. A temporary email is a disposable email address that allows users to receive emails for a short period, often used to maintain anonymity or avoid spam during registration processes. During December 2024 and January 2025, we tested the registration process on Pinata, Fleek, and Filebase using email addresses generated by the service TempMail (`https://temp-mail.org`). Both Pinata and Fleek accepted the first temporary email we generated, allowing us to create accounts successfully. After four attempts with different temporary email addresses, Filebase accepted the registration, suggesting that its filtering against disposable emails may be incomplete.

In all three cases, the platforms required us to verify the email address using a one-time password (OTP). 4EVERLAND, on the other hand, does not use email-based registration but instead requires a cryptocurrency wallet. Using Metamask, we successfully created an account on the platform, noting that even for creating the Metamask wallet, no email was needed. Finally, while Web3.storage accepted the temporary registration email, uploading files required linking a payment account, even though the platform also offers a free plan. This suggests that, although temporary emails are allowed, the payment account requirement serves as an additional verification step for users, limiting its suitability for fully anonymous abuse scenarios. Table 4.3 presents a summary of these findings.

To simulate malicious behavior, we developed a Python script compiled into a Windows executable ($\approx$ 7 MB) using PyInstaller.[11] It mimicked keylogging, dummy process injection, basic file manipulation, and failed network connections.

[10] https://docs.metamask.io/services/reference/ipfs/

[11] `https://pyinstaller.org/`

Table 4.3: Registration requirements & free storage for pinning services.

Pinning Service	URL	KYC	Temp Mail Accepted	Free Storage	Registered Country	DMCA Compliant
Pinata	`https://pinata.cloud`	E-mail	✓	1 GB	USA	✓
Filebase	`https://filebase.com`	E-mail	✓	5 GB	USA	✓
Fleek	`https://fleek.co`	E-mail	✓	5 GB	USA	✓
Web3.Storage	`https://web3.storage`	Credit Card	✓	5 GB	USA	-
4EVERLAND	`https://4everland.org`	Crypto Wallet	N/A	5 GB	AUS	✓

The file was safe by design, yet flagged by multiple antivirus engines on VirusTotal[12] due to behavioral heuristics. No harmful payload or external communication was included. To ensure unique Content Identifiers (CIDs), we created a distinct version of each script for each pinning service under evaluation. One of the key questions explored in this section is how pinning services handle files clearly marked as malicious, aiming to better replicate the perspective and actions of a potential attacker. In addition to the simulated malware, we also tested uploading known deprecated malware, specifically the WannaCry ransomware, to the pinning services. The result was identical: the file was successfully uploaded, and its CID was generated. Furthermore, we confirmed its accessibility through the public gateways. Notably, all files, including WannaCry, were immediately accessible, highlighting the absence of mechanisms in public gateways to evaluate the maliciousness of uploaded content. This raises significant concerns about the potential misuse of the IPFS network.

As previously discussed, in IPFS, the physical address of the node hosting a file can be identified. However, when files are hosted by pinning services, attackers are not concerned about their own address being exposed. The only potential exposure point is during the interaction with the pinning service's website for registration and file upload. To mitigate this risk, an attacker could use a public network or leverage the Tor [61] network to enhance their anonymity prior to registering and uploading files to the pinning services. Since many services implement protections that restrict access via Tor, we conducted a series of tests to verify the feasibility of using Tor to access these services. Our tests confirmed that

[12] `https://www.virustotal.com/`

files could be successfully uploaded, and the recorded IP address differed from our actual address, ensuring the attacker's anonymity.

It is important to note that visitors to these files, once uploaded by the attacker, may include either unsuspecting users who were targeted by phishing [60] or malware campaigns, or, in CyberCrime-as-a-Service scenarios [46], collaborators of the attacker, such as affiliates. Even in the latter case, leveraging the Tor network can effectively mitigate the risk of exposing their identities or the nature of their activities.

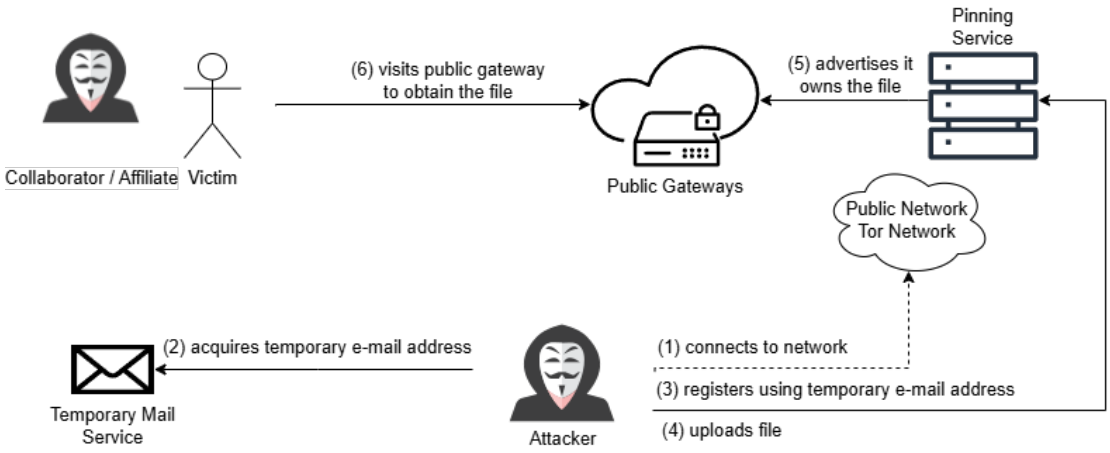


Figure 4.6: Design of the "Pinning Service Attack".

Figure 4.6 presents the steps that a malicious actor must follow to execute the "Pinning Service Attack". It allows the attacker to leverage the Tor network for anonymity and anonymously upload files to IPFS. By utilizing pinning services, the attacker ensures that uploaded files remain persistently online.

The Public Gateway Attack

As mentioned, Public Gateways of IPFS do not provide a direct method for uploading a file to the network. However, their caching might indirectly serve as a pinning service, providing file availability. In this section, we initially examine whether and for how long a file remains cached.

To better understand this phenomenon, we conducted a systematic experiment focusing on caching behavior across multiple gateways. The methodology

we followed is as follows. From the 10 gateways identified in Section 4.2.1, we selected five based on their strong association with well-known Web companies (e.g., Pinata, Infura) and official status within the IPFS ecosystem. Specifically, we chose (a) `ipfs.io` (the official gateway maintained by Protocol Labs), (b) `gateway.pinata.cloud`, (c) `infura-ipfs.io`, (d) `flk-ipfs.xyz` and (e) `4everland.io`. For each selected gateway, we created four different files resulting in 20 different files. First, we wanted each gateway to have different files to avoid cross-caching scenarios. Second, for each of these, we created four different files corresponding to the 4 time scenarios we are studying: 1 hour, 6 hours, 12 hours, and 24 hours. We use these intervals to request the respective files from the gateways to understand how popular a file needs to be to remain cached.

Subsequently, we used an IPFS node to add the files, ensuring our node ran as a DHT server. Then, to confirm that all the gateways cached all files, we sent up to four requests per file to verify their caching status. The four requests were performed in a negligible amount of time, less than five minutes, and the files became available. After successfully ensuring that all files were cached across the gateways, we disconnected the node from the network, leaving the gateways as the sole source of file hosting. The latter allows us to isolate the role of gateway caching in maintaining file availability independent of the original node. By doing so, we could analyze how the caching mechanisms of public gateways sustain file accessibility over time.

We automated the process of sending requests to the gateways based on the aforementioned periods and recorded the responses for more than three days. The results indicate that caching duration varies significantly between gateways, with some maintaining availability longer than others, which could be attributed to differences in caching strategies or the relative popularity of each gateway. Figure 4.7 illustrates the ratio $\frac{\nu}{5}$ per hour, where ν represents the number of gateways caching our files at a given time across the different time scenarios. As depicted, two out of the five gateways removed our files from their cache shortly after 16 hours, while the remaining three continued to retain them online. For ethical reasons, we refrain from disclosing which ones retained or removed the files.

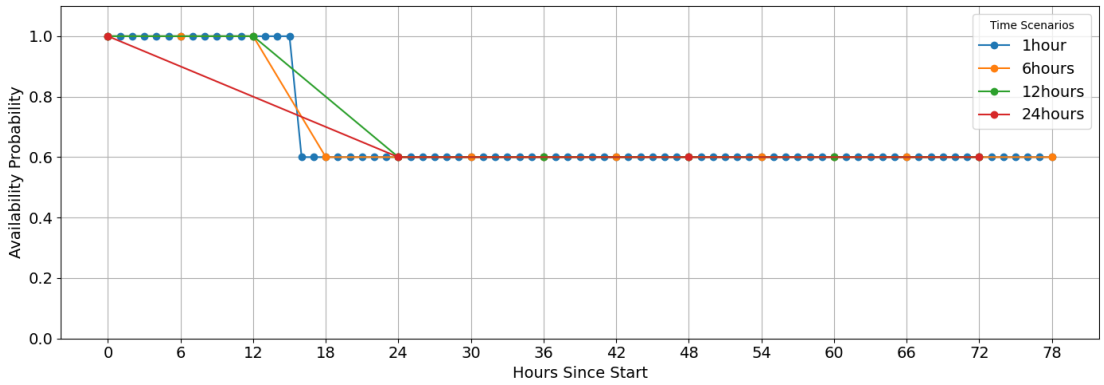


Figure 4.7: Time-Dependent File Availability Analysis.

In conclusion, we have demonstrated that a malicious actor could potentially exploit Public Gateways to maintain files on the IPFS network anonymously. The process involves first uploading the files to the IPFS network and generating artificial traffic by repeatedly requesting these files. This ensures that the Public Gateways cache the files. Once the files are cached, the actor can sustain their availability by periodically sending requests for the files, preventing them from being removed from the cache due to inactivity. This approach allows the actor to leverage the distributed infrastructure of Public Gateways to maintain file availability while preserving anonymity, eliminating the need for a dedicated pinning service. At this point, it should be noted that during the attack, the attacker only risks revealing their physical address while uploading the files via the local node. As previously mentioned, this process requires minimal time, significantly reducing the exposure window for the attacker. Additionally, the attacker could perform this step through a public network to further obscure their physical location. The subsequent periodic requests to the public gateways can also be accomplished through a public network or Tor. Additionally, the attacker could utilize a botnet under their control to generate artificial traffic towards the files without revealing their identity. By distributing requests across multiple geographically dispersed nodes, the botnet obscures the origin of the traffic, making it significantly harder to trace back to the attacker. Note that in the past, the IPFS network has been a victim of such botnet activity [6]. A step-by-step implementation of the attack is illustrated

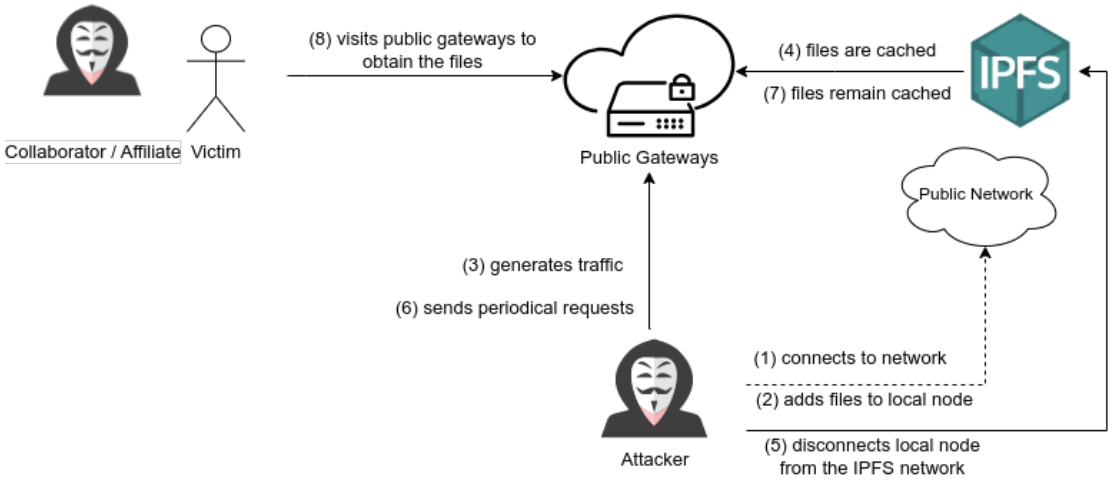


Figure 4.8: Design of the "Public Gateway Attack".

in Figure 4.8.

Double Extortion Attack

Typically, ransomware attacks encrypt the victim's files and demand a ransom to be paid to hand over the decryption key. Nevertheless, modern organizations have invested in backup systems that limit the damages of a potential ransomware attack, significantly decreasing the amount of ransom they would be willing to pay. As a countermeasure, ransomware gangs siphon sensitive data to their premises, threatening their victims by leaking the data and creating what is often called a "double extortion".

The siphoning of the data can be performed in multiple ways, however, methods like DNS tunneling, while effective, can be very slow. Therefore, ransomware gangs tend to abuse cloud service providers to upload their "loot". For example, the notorious Conti group used RClone to upload data to multiple cloud storage providers. [13] With IPFS and the poor KYC practices of pinning services, ransomware gangs can have another more robust option. They may harvest sensitive information from the infected hosts and upload them to IPFS through pinning

[13] `https://news.sophos.com/en-us/2021/02/16/conti-ransomware-attack-day-by-day/`

services. Beyond exploiting KYC to gain the necessary storage, ransomware gangs may also exploit whitelisted domains and the lack of content takedown mechanisms. Note that cloud service providers respond to takedown notices, e.g., the victim notifies the cloud service provider that leaked sensitive data are hosted and must be taken down. However, pinning services cannot remove content from the IPFS once it has been uploaded. Although pinning services comply with DMCA policies and can remove a pinned file from their hosted storage, this does not translate into the deletion of the file from the IPFS network. The decentralized nature of IPFS makes this nearly impossible, while the existence of public gateways, many of which do not adhere to the badbits list (as mentioned in 4.2.2), further complicates takedown efforts. Figure 4.9 illustrates this abuse scenario.

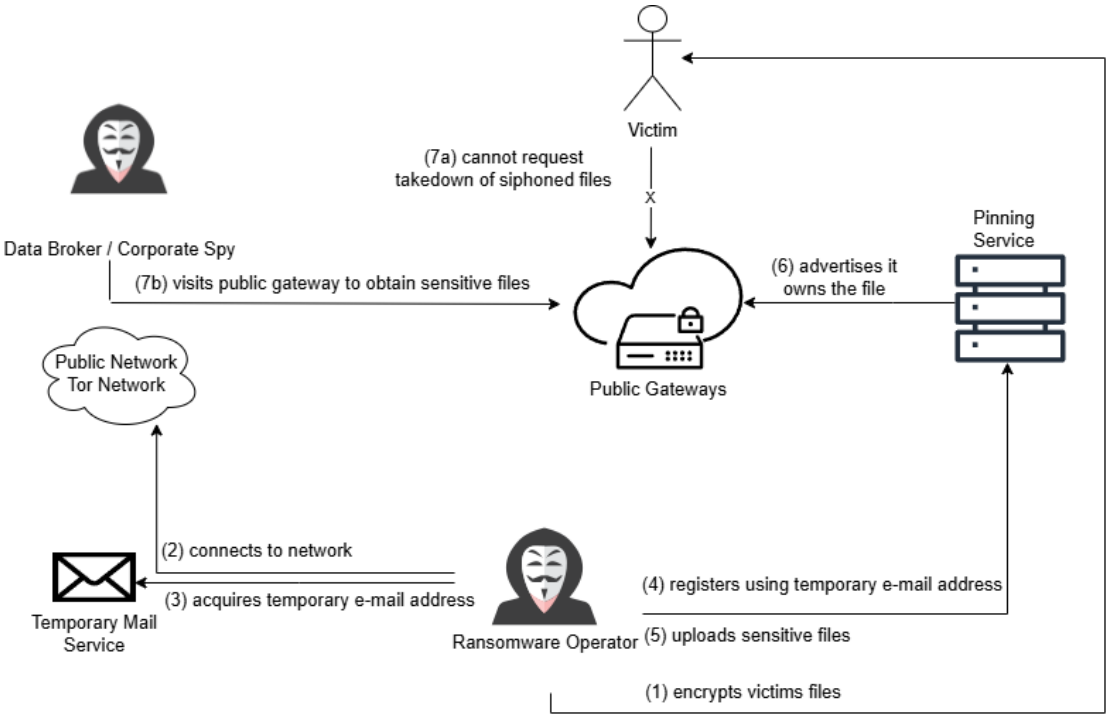


Figure 4.9: Design of the "Double Extortion Attack".

Real-Life Evidence of Malicious Exploitation

While previous work such as [60] investigated the presence of malicious or illegal content across the IPFS network, our approach specifically targets pinning

services, i.e., entities that intentionally maintain long-term availability of hosted content. By focusing on CIDs advertised by major pinning providers, our analysis offers a more precise view into deliberate, persistent misuse of the IPFS ecosystem, and links it directly to infrastructures that facilitate anonymity and permanence.

We utilized `ipni-cli`[14] to monitor CIDs advertised by Pinata, Filebase, and Fleek pinning services on the `cid.contact` indexer for 24 hours. For all providers, we repeatedly executed the following command:

```
ipni ads get --ai=<provider addr> --head
```

This command retrieves information about the latest advertisement from the specified provider, including the number of CIDs it contains. Once we obtained this information, we proceeded to extract the actual CIDs using:

```
ipni random <provider addr>
```

With the parameter `n`, this command returns `m` CIDs from a random selection of the most recent `n` advertisements. By setting `n=1`, we ensured that the selection always targeted the most recent advertisement. Since the previous command had already provided us with the exact number of CIDs, we could request all of them at once. This approach enabled us to systematically retrieve all hashes from every advertisement recorded since the beginning of the experiment. By continuously executing these queries and storing the results, we effectively built a historical record of all advertisements and their associated CIDs from each provider. During the 24-hour interval, we collected **(i)** $1,124,780$ CIDs from Pinata, **(ii)** $718,578$ from Filebase, and **(iii)** $339,684$ from Fleek. For each of these, we standardized the format of the CIDs to match the entries in the `Bad Bits Denylist`,[15] ensuring compatibility for an accurate comparison. The Bad Bits Denylist is a list maintained by Protocol Labs, updated upon email recommendations to filter undesirable files, such as malware, phishing content, or copyright-infringing materials. Note that the list is enforced on the public gateways operated by Protocol Labs but is advisory for all other nodes within the IPFS network. By matching the monitored CIDs against the entries in the denylist, we discovered that within 24 hours,

[14] `https://github.com/ipni/ipni-cli`
[15] `https://badbits.dwebops.pub/`

the pinning services advertised five CIDs included in the Bad Bits Denylist. It is worth mentioning that one of these CIDs was advertised by all three services, while two were common to two services. We consider the presence of these blocked CIDs –and even more so their simultaneous advertisement on the same day by multiple pinning services– a strong indication of malicious actors' organized exploitation of the anonymity provided by pinning services. Finally, we managed to retrieve three of them, discovering that one was a JavaScript file involved in a Bank of America phishing scam, the second was a login phishing webpage targeting a Korean webmail service, and the third was an image, likely used for malicious purposes.

4.2.3 Related Work

A growing body of research has shown that malware increasingly abuses centralized Web and cloud platforms for infrastructure, persistence, and evasion. Yao et al. [55] propose Marsea, a concolic execution engine that detects malware interaction with benign Web applications such as GitHub and Dropbox, revealing how these services are repurposed for malicious use. At a broader scale, Allegretta et al. [56] analyze threat intelligence from 36 vendors and identify over 22,000 abused benign domains, including services like Discord and Google Drive, used to distribute malware. These works demonstrate that even trusted, centrally managed services are vulnerable to abuse. In this work, we show that decentralized infrastructures like IPFS introduce new and arguably more permissive abuse surfaces, due to their inherent anonymity, lack of content moderation, and resistance to takedown.

In recent years, Web3 has emerged as a new paradigm for the Internet, prioritizing user anonymity and privacy. These features are especially significant as concerns about user privacy and tracking escalate. However, numerous studies indicate that these features are often compromised. Kshetri [62] highlights several vulnerabilities within Web3 and the metaverse, particularly the extensive data collection and exposure of personal and sensitive data due to numerous security breaches on Web3. Furthermore, the author points out that anonymity may be compromised through the traceability of blockchain transactions on Web3

platforms, potentially linking personal identities and actions to public transaction records.

On the other hand, other studies focus on how anonymity and privacy are compromised on Web3. Wang et al. [63] explore how Web3 social platforms, such as friend.tech,[16] impact user privacy and anonymity. In particular, they identified that the integration between Web3 and legacy Web2 platforms could significantly undermine Web3 anonymity and lead to privacy leakage. This occurs because user actions on Web2 platforms can be associated with accounts on Web3 platforms since these actions are immutably written on blockchains. Then, the recorded actions can be linked and traced back to the users. To address these problems, the authors argue that a balanced approach between transparency and privacy in Web3 is needed. Additionally, Torres et al. [64] focus on how wallets and Decentralized Applications (DApps) manage user data. The authors conclude that current privacy measures are insufficient, highlighting that Web3 applications, particularly wallets, often expose sensitive user data, such as wallet addresses. This exposure directly contradicts the foundational privacy promises of Web3 by compromising user anonymity and privacy.

A central element of Web3 and a core focus of our study are distributed file systems, with IPFS being the most prominent. Previous research has demonstrated that IPFS can be exploited by malicious actors across various domains. For instance, studies have shown its use in Malware as a Service systems [46], while others have reported the presence of phishing files or copyright violations within the IPFS network [60]. Moreover, IPFS also has some privacy violations. In particular, Balduf et al. [7] showcase a privacy attack on the IPFS network by leveraging the Bitswap protocol and introducing a set of attack vectors. The authors state that every IPFS node is susceptible to each of the introduced attacks, and moreover, they succeed in exploiting it by deploying a number of nodes with extended connectivity to passively monitor the Bitswap channel and demonstrate their attack methodology by discovering the PeerId of the public IPFS HTTP gateways.

[16] `https://www.friend.tech/`

In addition to attackers, security analysts can leverage Bitswap's privacy shortcomings. Son et al. [65] propose *IF-DSS*, a digital forensics investigation framework for Decentralized Storage Services (DSSs). They analyze the most critical DSSs from the point of view of digital forensics and apply the proposed framework to IPFS. To collect appropriate and sufficient data, they separate them into those that exist on the local side as well as remotely. Finally, they suggest tackling the dissemination of illegal material in three steps: (i) Content filtering, i.e., blacklisting of the inappropriate content, (ii) stop content sharing, i.e., turn the node from server to client, and finally, (iii) shutting down the node.

On the other hand, some works try to enhance IPFS privacy. Katsantas et al. [66] focus on hiding the identity of content on IPFS by using only hash functions. The authors aim to prevent intermediaries from detecting the retrieved contents without relying on trusted third parties. Furthermore, Daniel et al. [67] point out that as IPFS follows the ICN paradigm, a client requests content directly rather than visiting an address. Thus, Bitswap queries all the client's neighbors for content, resulting in the client's interest leaking. Aiming to reduce interest leakage, the authors propose three privacy-enhanced standards for content discovery. By using these protocols, on the one hand, the level of privacy of the client is improved, but that of the provider is reduced. More specifically, they propose `Bloom-Swap`, a solution using bloom filters in which the provider sends its inventory to the client, and he, in turn, checks locally whether the requested content is a Bloom Filter member to ask the block directly. `PSI-Swap`, which uses Private Set Intersection (PSI), reduces and improves privacy levels on the provider's side as well. Finally, the `BEPSI-Swap`, which combines the two previous ones, improves the efficiency of PSI-Swap, at the cost of making PSI probabilistic. The authors then implement a proof of concept of the proposed protocols and study them from the security and efficiency perspectives.

4.2.4 Countermeasures & Conclusions

The decentralized nature of the technologies we study, combined with the fact that the majority of the software is open-source, makes enforcing rules for

implementing countermeasures challenging. From the perspective of pinning services, KYC practices must become stricter. Measures such as filtering temporary emails, implementing blockchain-based identity systems, e.g., cryptocurrency wallets with benign transaction history, applying stricter criteria for users operating through Tor networks, enabling content scanning mechanisms, and adhering to a centralized deny list like Bad Bits should be enforced. Public gateways act as bridges for Web2 users to access the Web3 ecosystem. For the average user, requiring a blockchain-based identity would deter them from utilizing these gateways. However, all gateways could be required to comply with the Bad Bits, a policy currently enforced only on gateways managed by Protocol Labs. Moreover, even if a CID is listed on the Bad Bits Denylist, a malicious actor can circumvent it by simply choosing an alternative chunking size when adding the file to IPFS (**RQ5**). This approach generates a different CID that is not associated with the blacklisted one [60], making content filtering on gateways significantly more challenging.

In this study, we examined the vulnerabilities of IPFS pinning services and public gateways, highlighting how malicious actors can exploit their anonymity features or lack of proper KYC policies to share undesirable content. By implementing and testing two distinct attack methodologies, we demonstrated not only their feasibility (**RQ3**) but also observed instances of malicious activity occurring within the IPFS ecosystem (**RQ4**). Our findings reveal critical issues, including the lack of robust KYC practices in pinning services (**RQ1**), insufficient content filtering mechanisms (**RQ2**), and the challenges posed by the decentralized and open-source nature of the IPFS ecosystem. These gaps enable attackers to take advantage of the anonymity features of the system while avoiding accountability. Since current KYC practices in pinning services can be easily bypassed, the use of stricter measures, of even the consideration of blockchain-based identity verification methods, such as zero-knowledge proofs (ZKPs), e.g., zkLogin [68], would allow users to verify their legitimacy without exposing their full identity.

It should be stressed that the decentralized nature of IPFS raises significant legal and regulatory challenges, particularly in the enforcement of content moderation and compliance with existing digital laws. While platforms operating

in centralized environments are bound by regulations such as the Digital Services Act (DSA),[17] decentralized systems like IPFS lack clear accountability structures. This creates a regulatory gap that malicious actors can exploit to distribute illicit content while avoiding legal repercussions. One of the main concerns is jurisdictional ambiguity. Since IPFS content is hosted on a distributed network of peers, it is often unclear which jurisdiction has the authority to enforce takedown requests or prosecute offenders. This is especially true on platforms like IPFS, where there is no deletion mechanism and data ownership is not always known. Pinning services, many of which operate in different countries with varying legal requirements, further complicates the enforcement process.

Nevertheless, this sparks the debate surrounding IPFS security and other such platforms regarding the trade-off between privacy and censorship resistance. While decentralization offers increased resilience against state-sponsored censorship, it also enables unmoderated content proliferation, including, but not limited to, extremist propaganda, child sexual abuse material, and malware distribution. The ability of malicious actors to exploit anonymity for illegal activities creates a dilemma where content moderation mechanisms must be introduced without undermining the fundamental principles of decentralized storage.

Strengthening the security of IPFS and the surrounding ecosystem is essential not only to prevent its misuse but also to promote its adoption as a reliable and privacy-preserving tool for decentralized file sharing, which is fundamental to the Web3 paradigm. To this end, future research could focus on the development of automated tools to detect malicious CIDs in a decentralized and scalable way. Another approach would be decentralized content moderation, where community-driven flagging mechanisms allow for voluntary filtering rather than direct deletion. Likewise, user-driven reputation systems for pinning services and nodes could help differentiate legitimate operators from malicious ones. By assigning trust scores to nodes based on their activity and compliance with community standards, users could make informed choices about which nodes to trust for content retrieval and caching.

[17] `https://eur-lex.europa.eu/eli/reg/2022/2065/oj/eng`

Chapter 5

Security Challenges and Solutions in the Web3 Application Layer

The Web3 application layer constitutes the uppermost tier of the Web3 stack, enabling user-facing DApps that leverage blockchain technologies, decentralized storage systems, and cryptographic protocols. This layer abstracts the complexity of the underlying infrastructure, e.g., smart contracts, DLTs, naming systems, and focuses on delivering secure, user-driven services across domains such as Decentralized Finance (DeFi), social media, and gaming. The application layer is responsible for orchestrating secure asset interactions, enforcing ownership logic, ensuring long-term availability, and preserving digital content through P2P infrastructures. In this chapter, we investigate security concerns specific to the Web3 application layer, with a focus on decentralized gaming and peer-to-peer content retrieval. First, we introduce a fully decentralized NFT-based gaming architecture that addresses trust, ownership, and sustainability through smart contracts, IPFS, and name resolution mechanisms. We then turn our attention to the foundational infrastructure that supports such applications: IPFS and its Bitswap protocol. We present a protocol enhancement that improves both retrieval latency and content availability, critical properties for scalable and reliable DApps. Together, these two contributions demonstrate how application-layer design and P2P storage optimization can jointly address core security requirements such as asset integrity, content availability, and decentralized control.

5.1 The case of Blockchain Gaming

Distributed Ledger Technologies (DLTs) have found application in many aspects of our lives, as they promise secure, trustworthy, and decentralized transactions with the use of cryptographic techniques. Thanks to their fast growing popularity, DLTs have lately caught the eyes of game development industry. Offering solid proof of uniqueness and ownership for assets, they are fertile ground for the development of various types of games.

Gaming models have changed over the years, targeting to keep pace with the evolution of technology, trying to be attractive to users, and at the same time more profitable for the gaming industry. The traditional *Pay-to-Play* model, where users pay upfront for the game and/or the console, has lately been replaced by the *Free-to-Play* model. In the Free-to-Play model, users acquire the game at no cost, but they are incentivized to spend money for in-game assets. The latter has brought to the fore the gaming career path and the eSports industry. The advent of DLTs combined with the trend of gamers to earn an additional income from gaming, has given birth to the *Play-to-Earn* (P2E) model. In the P2E gaming model, not only do users play for free, but they can potentially earn cryptocurrencies. All in-game merchandise they earn playing rely on Non-Fungible Tokens (NFTs) owned by them, and not by the company, which can be sold or exchanged for cryptocurrency. In 2017, blockchain trading games made their appearance using NFTs and since then, they have been growing in popularity, as well as in market capitalization. From the pioneering Cryptokitties,[1] to Axie Infinity,[2] the most recent pokemon-like game following the P2E model with over $1,100,000,000 total volume.[3] As player communities continue to grow, an increasing number of game categories are adopting the NFT model, making it clear that NFT-based games are not a fleeting trend. In the third quarter of 2023, the blockchain gaming sector experienced 12% growth, with average daily unique active wallets reaching 786,766 [69]. By August 2024, the industry had reached a new record of 4.2

[1] https://www.cryptokitties.co/
[2] https://axieinfinity.com/
[3] https://nomics.com/assets/axs2-axie-infinity/

million daily unique active wallets.[4] This exponential growth underscores the rising importance and permanence of blockchain gaming within the broader digital entertainment ecosystem. As these applications scale, the Web3 application layer is increasingly challenged to deliver robust, secure, and highly available infrastructure for managing digital assets. Ensuring ownership persistence, content integrity, and resistance to central points of failure is not only critical for user trust, but also for the long-term sustainability of decentralized gaming platforms. Therefore, secure storage, naming, and interaction mechanisms become central design concerns at the application layer.

Blockchain games are classified as DApps, offering new models for ownership and interaction. Although they utilize blockchain-based asset representation, the centralization of media files, such as game artwork and metadata, has been a critical limitation. In most implementations, these assets are hosted on the servers of the game companies, introducing risks related to long-term availability, content integrity, and user ownership [70]. To address these issues, a prior system [71] was proposed, introducing a fully decentralized trading game architecture. The system leveraged the InterPlanetary File System (IPFS) to ensure that in-game assets remain available and verifiable, regardless of the status of the game operator. This initial design sought to answer foundational questions such as: "Who truly owns the artwork of a game?", "What happens if the gaming company discontinues the service?", and "Does the asset retain any value independently?". Building on that foundation, the extended work [72] identified key limitations in the use of IPFS alone, particularly regarding storage incentives and update mechanisms. As a response, it integrated Filecoin to enhance long-term availability via economic incentives, and replaced IPNS with the Ethereum Name Service (ENS), offering improved robustness and user-friendly management of evolving NFT metadata. In parallel with the rapid expansion of NFT-based gaming and the Play-to-Earn (P2E) model, the system architecture was enhanced to support decentralized interaction between gaming companies, artists, and players in a tamper-resistant, auditable, and trustless manner. Core technologies underlying this design include

[4]https://dappradar.com/blog/blockchain-gaming-reaches-new-record-4-2-million-daily-active-users

smart contracts on Ethereum, IPFS/Filecoin for decentralized storage, and threshold cryptography for secure asset control. To meet the evolving requirements of decentralized gaming under the expanding P2E paradigm, the proposed system introduces a fully decentralized and self-sustainable architecture. By orchestrating heterogeneous Web3 components such as blockchains, decentralized storage (IPFS and Filecoin), and naming services (ENS), the system addresses limitations related to asset availability, ownership guarantees, and trustless interaction. Notably, it implements a decentralized version of the "mint-in-sealed-box" concept using threshold cryptography, preserving asset secrecy and uniqueness until explicitly revealed. The system also supports evolvable NFTs, allowing in-game assets to dynamically change over time. This is achieved through the use of mutable name resolution mechanisms, for which ENS was selected over IPNS due to its enhanced robustness and performance. Finally, the architecture supports royalty-aware asset resale, enabling new business models where artists and other stakeholders can receive automatic compensation for secondary market activity.

5.1.1 Background

IPFS Ecosystem

Web3.storage and NFT.storage are open-source services created by Protocol Labs targeting to store general data and NFT related data, respectively, in the Web3 era. Both work in a decentralized manner, leveraging IPFS and Filecoin and are framed by Javascript libraries. It is notable to mention that both services are provided to the community at no cost to the user.

Shamir's Secret Sharing

Shamir's Secret Sharing (k,n) [73] is a threshold cryptography scheme, used to secure a secret. Initially, the secret is divided into n fragments, called shares. For the secret to be revealed, at least k of the shares are required. Thus, if the n shares were distributed to n actors, at least k of them must coalite to recover the secret.

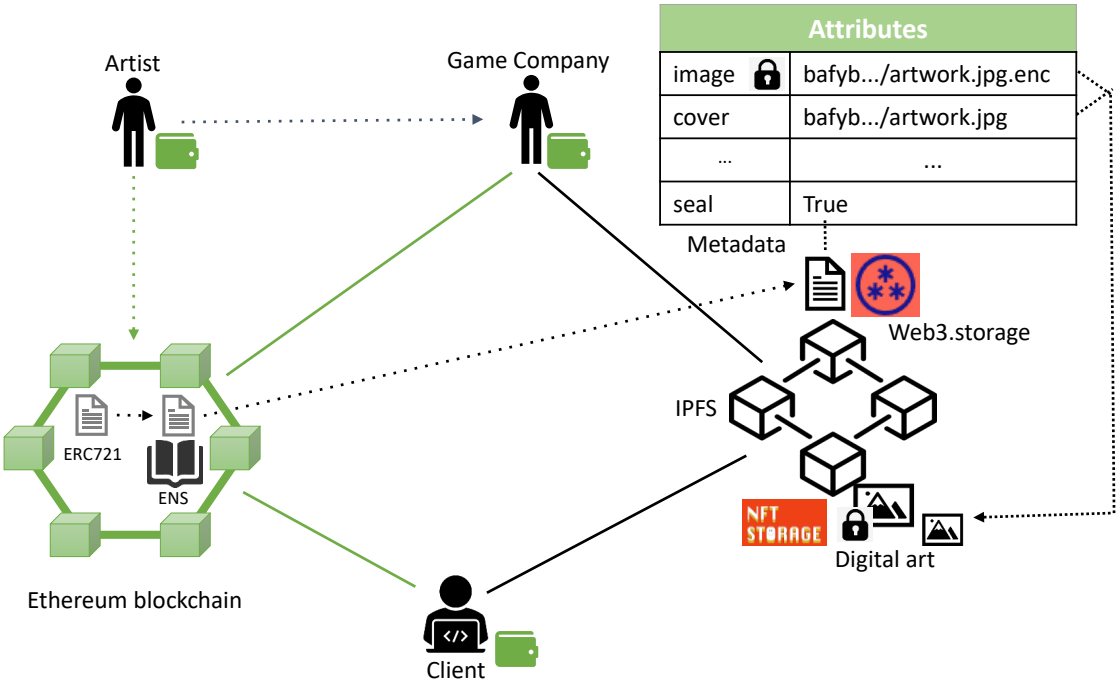


Figure 5.1: An overview of the system's architecture.

5.1.2 System Design

In this section, we present the architecture of our proposed system, which is illustrated in Figure 5.1. Our architecture is composed of the following entities:

- The Ethereum blockchain, the IPFS, and Filecoin infrastructure
- The required smart contracts, one that creates and manages the NFTs and the smart contracts that implements the ENS (registry, resolver, and registrars)
- The gaming company (game administrator/creator) that owns the trading game
- The artists that create the digital arts
- The clients that want to purchase and trade the NFTs and the corresponding digital arts

The actors of the system, namely the *game administrator*, the *artists*, and the *clients*, should own a blockchain wallet in order to be able to interact with the blockchain network. This wallet is also used as an address on the blockchain network and as a secure storage for the acquired NFTs. Moreover, for the encryption part of the digital assets, a symmetric key is generated, split, and shared among the three actors. Finally, the game administrator should own an API key to be able to upload the metadata file and the corresponding digital art on the Web3.storage and the NFT.storage respectively. From a high level perspective, the entities of our system interact with each other as follows. The artist creates the digital art and sends it to the gaming company. Then, the gaming company creates the NFTs, initializes the corresponding ENS entries, and uploads the encrypted digital art and the appropriate metadata file on the IPFS. Finally, clients can acquire NFTs by paying the defined amount of money (in ethers) on the smart contract. The flow of the system is described in more detail in the following phases.

Setup

Initially, the gaming company implements the smart contract that creates and manages the NFTs, based on the ERC-721 token standard, and deploys it on the Ethereum blockchain. We settled on ERC-721, despite the variety of token standards, e.g., ERC-1155, due to its popularity among the blockchain games. Moreover, it is considered as a perfect fit for our system, since we are focusing purely on NFTs, and not on both fungible and non-fungible tokens. The address of the smart contract is considered well-known. Then, the gaming company mints the tokens, receives the media files (e.g., character avatars), which are created by the artists, and it initializes the corresponding ENS entries.

We now describe the flow that happens for each NFT and the corresponding media file. The gaming company generates a symmetric encryption key to encrypt the media file and uploads it on the NFT.storage (the gaming company is considered a trusted entity). Subsequently, the gaming company uploads the metadata file on the Web3.storage. The metadata file contains information about the NFT. Namely, it contains the name and the description of the NFT, the CID of

the encrypted media file that corresponds to this NFT, and the CID of the sample cover file. The metadata file contains also a map showing the current owner of the NFT, and a value that shows whether the media file has been downloaded and decrypted. These information are also stored in the smart contract that handles the NFTs. Upon uploading the media file on the NFT.storage and the metadata file on the Web3.storage, the gaming company modifies the ENS entry to point to the IPFS hash (CID) of the metadata file, in which there is an entry with the hash of the media files on the NFT.storage on IPFS. Finally, it modifies the token URI field of the NFT in the blockchain to point to the corresponding ENS entry.

NFT purchase

From this point on, a client can acquire NFTs, either from the gaming company or from another client. For the first case, the client can read the blockchain, by invoking the appropriate function of the token smart contract to find out the available NFTs. Then, if she wants to purchase an NFT, she has to pay the defined amount of money (in ethers) on the smart contract. Subsequently, the token is "transferred" to her account (blockchain wallet) and she is able to see the metadata file of the token stored on IPFS. When a client acquires an NFT, sold for the first time, the gaming company, using Shamir's Secret Sharing (2,3) threshold scheme, splits the decryption key into three parts, and each role of the system, artist, gaming company, and client, gets one.

On the other hand, if a client wants to acquire an NFT that is already owned by another client the following flow occurs. She should come to an agreement with the client that owns the NFT for its price. If they come to an agreement, then she has to pay the agreed amount of money, which is transferred to the smart contract's address. The NFT is "transferred" on the smart contract's address too. The client that sold the NFT, has to send his share of the decryption key to the client that bought the NFT, through the smart contract. Then, an event is generated that eventually is "caught" by the gaming company, which verifies that the key is the actual key and not a fake one. Finally, if everything is as expected, the gaming company sends a transaction on the blockchain, to "transfer" the money on the

previous owner' address and the NFT on the new owner's address. Otherwise, the NFT is transferred back to the previous owner and the money is transferred back to the client wanted to acquire the NFT. Due to our design, there is no need for generating a new encryption key or for keeping this specific share secret, since it does not matter if the previous owner of the NFT copies the key or anyone else read the blockchain and acquire the key, as the media file cannot be decrypted with only one share of the key.

NFT retaining

In this phase, we assume that a client has already acquired an NFT. If the client wants to download and decrypt the media file, she has to ask the other two parts (artist and gaming company) for their decryption keys. Then, the gaming company checks from the metadata file or the smart contract that the client is indeed the owner of the NFT that she wants to decrypt, and if that is the case, the gaming company and/or artist send their part of the key to the client, offline and off-chain. After that, the list on the metadata file, as well as the appropriate fields of the smart contract, is updated and shows that the client downloaded and decrypted the media file. The price of the token is adjusted accordingly, based on the status of the media file, e.g., if an asset has not been decrypted by any client yet, its price remains high ("mint in sealed box" feature, see below).

5.1.3 Implementation

To better illustrate the advantages of our system and to fully quantitative evaluate it, we developed a proof of concept implementation.[5] The developed system is composed of two main parts; the *core service module* and the *smart contracts* that exist on the blockchain network. The *core service module* is a piece of software developed in Node.js. It implements several functions that enable the gaming company to interact with the IPFS-related services, such as the NFT.storage and Web3.storage, as well as with the ENS smart contracts. Moreover, we provide

[5] `https://github.com/mmlab-aueb/Tokenmon`

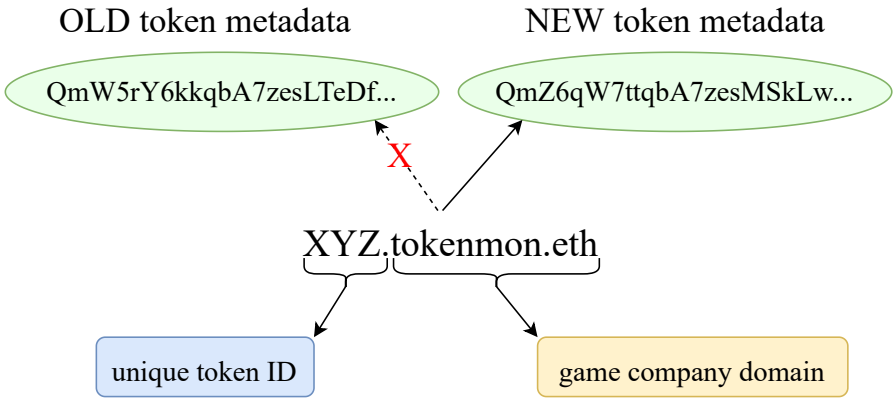


Figure 5.2: Diagram illustrating the use of an ENS address in our system.

an alternative implementation using IPFS' native name resolution service, IPNS, instead of ENS.

Core Service Module

As we have already mentioned above, the *core service module* runs on behalf of the gaming company and implements functions for managing the IPFS and ENS related services. The first function, called `uploadToken`, is used for uploading a token's full sized encrypted artwork, a low quality sample cover, and the token's attributes on IPFS, and in particular on the NFT.storage. Other auxiliary information about the tokens, such as its name, its description, the encrypted artwork file's CID on IPFS, the sample cover file's CID, etc., are included on the metadata file, which is uploaded on the Web3.storage, using the same function. In order for the game administrator to use the two services, two different API keys are required - one for NFT.storage and one for Web3.storage. Each key can be acquired through the each service's website by creating an account. The metadata file also contains information about the owner of the NFT, and whether the digital artwork has been decrypted. This file is encoded using JSON.

The second function of the *core service module* is called `createToken` and is used to create a unique static ENS address for a token. This function is used whenever either a completely new NFT is generated or a token's metadata changes due to a system feature, i.e., evolution and fusion. The function claims a new sub-

domain, which is used together with the gaming company's domain to identify the NFT. This is illustrated in Figure 5.2. Another basic function of our system is the `updateToken` function. This function is used to update the CID of the previously mentioned static ENS address associated with the NFT. Every time that an NFT's metadata is changed, this function is used to update the address to point to the latest version of the metadata file. Finally, the last function is the `decryptToken`, which downloads an NFT's encrypted artwork from the NFT.storage and decrypts it, using two of the stakeholder's shares of the key.

Two other important functions are implemented separately in a cryptographic module; an `encrypt` and `decrypt` function. Both of them utilize the Advanced Encryption Standard (256-bit AES) via Shamir's threshold secret sharing scheme. The encrypt function is used to encrypt the artist's artwork before going public on the IPFS, via the uploadToken function. A random key is generated and three separate shares are created and distributed to the three entities participating; the gaming company, the artist, and the owner of the NFT. At its current state, the information regarding the keys is stored in a database hosted by the gaming company. Finally, the function results in a fully encrypted file, which can be uploaded on NFT.storage. On the other hand, the decrypt function is used to decrypt the token's artwork using two out of the three secret shares.

1b) Core Service Module using IPNS

As we have already mentioned, every IPNS address is in essence the hash of the public key of a key pair. Thus, for every NFT, a public-private key pair must have been generated in advance. The differences between the two implementations are minimal and are located in the `createToken` and `updateToken` functions. `createToken` generates a new pair of public-private keys that is directly associated with the token's ID. After the creation, the IPFS node automatically saves the pair and an IPNS address is generated via publishing the token's metadata CID by hashing its public key. Finally, a new entry is inserted in the game company's local database including the token's ID, its current metadata CID, and its static IPNS address.

On the other hand, considering that a key pair already exists and that the goal is to update a token, `updateToken` publishes the new metadata CID via the already existing token's public key. At this stage, the static IPNS address points to the newest version of the token's metadata. Then, the function proceeds to update the token's metadata CID field in the local database. In order for the implementation to work properly during its early production stages, between random time intervals, a support function iterates the database and republishes the CID for every token ID stored in the database. This is done as a precaution to the IPNS records exceeding their life time.

Blockchain

The other part of our presented system is related to the Ethereum blockchain and the smart contracts. The core smart contract of our system is the NFT smart contract, which is developed using the Solidity programming language and it is deployed on the Rinkeby Ethereum test network (ethers in this test network do not have real value). This smart contract implements the functions that the ERC-721 token standard dictates. In addition to these functions, our smart contract implements three more functions that correspond to the system's actions; `createToken`, `fuseTokens` and `breakSeal`. The `createToken` function is used to mint an NFT, given a specific URI that corresponds to the ENS address pointing to the metadata file. The `fuseTokens` function is used to merge two NFTs and mint a new one (evolvability feature). Lastly, the `breakSeal` function is used to simulate the seal of a non-digital collectible being broken, thus supporting the "mint in sealed box" feature. As we have already mentioned, the owner of an NFT is able to choose whether she wants to download and decrypt the digital artwork, namely revealing the NFT's full scale artwork. If such a decision is made, then the break seal function is called, and an event is emitted. The event is eventually "caught" by the core service module software, which updates the metadata file accordingly. Furthermore, the smart contract implements some (view) functions (introduce zero cost) in order to allow the actors of the system to read and learn auxiliary information about the NFTs. The software that interacts with the blockchain network and

the smart contract uses the web3.js JavaScript library. The gaming company can develop its own smart contracts based on the core contract that we implemented, to support more complex game mechanics.

5.1.4 Evaluation

One of the contributions of our work is that it provides evolvability. We argue that to achieve this desired feature, a name resolution service is necessary. In particular, we consider two naming systems; IPNS and ENS. In this section, we present the findings from the experiments we conducted, of our twofold implementation, the first using ENS and the second using IPNS.

ENS

The usage of blockchain technology is obvious and has a significant impact on the time performance and responsiveness of the system. Furthermore, the invocation of a smart contract function introduces some monetary cost measured in gas units. Various tests have been conducted on the Ethereum Rinkeby test network to estimate the average response times and the cost of the fundamental functions of the core smart contract. The results are are shown in Table 5.1.

Smart Contract Function	**Average Response (sec)**	**Cost (gas)**
createToken	27.27	85532
fuseTokens	22.10	99005
breakSeal	17.29	48394
transferFrom	18.35	61523

Table 5.1: Average response times and costs in the Rinkeby network.

It must be noted that the presented average response times are significantly exaggerated by sparse long waiting times. By excluding those infrequent spikes, the pure-average waiting times are near the 15 second mark. Other important time measurements have been conducted regarding the ENS-based implementation.

One of the most substantial differences between the ENS-based and IPNS-based implementation is accessing the blockchain more frequently to claim a subdomain, update, and retrieve its content hash field. The three functions have also been tested on the Rinkeby test network. The results for these three functions are shown in Table 5.2.

ENS Function	**Average Response (sec)**
setSubnodeRecord	27.23
setContenthash	14.67
getContenthash	1.69

Table 5.2: Average rsponse times of ENS functions.

IPNS

To have a clear picture, in order to compare the two different name resolving systems, we conducted the same measurements in the IPNS network. What we observe in Table 5.3 is the time duration from the moment NFT's address is announced over IPNS up to the moment an average user can visit it, through a public gateway. As we have already mentioned, IPNS leverages the DHT. So, the aforementioned times vary, depending on the number of active connections the announcing node keeps. To have a better understanding of the IPNS' robustness and consistency, we conducted the experiments in different versions regarding the number of active connections. Moreover, to have unbiased results every HTTP request is made among a set of different public gateways, and more specifically: (i) `https://gateway.ipfs.io`, (ii) `https://cloudflare-ipfs.com`, (iii) `https://gateway.pinata.cloud`, (iv) `https://ipfs.io` and finally,
(v) `https://ipfs.fleek.com`.

In Table 5.3, we can see that even using the default settings of IPFS 56% of the total queries timed out, meaning it took more than 5 or 10 minutes, depending on the gateway, to respond. Moreover, we observe that even in the extreme case

of over 3000 active connections, which was about the 20% of nodes found online[6], the 27% of requests were not served. From those served, the average response can be also found at Table 5.3. Both the percentage of non-served requests, as well as the average response time cannot be considered negligible and increase the chance that our system will malfunction. Additionally, we should point out that every IPNS record has by default a 24h validity, therefore the game company should reannounce it periodically.

Active Connections	**Served**	**Timed Out**	**Average Response(sec)**
[250 - 600]	28%	72%	23.9
[600 - 900]	44%	56%	20
[3000 - 4000]	73%	27%	15.9

Table 5.3: Response metrics using IPNS.

Finally, it is worth mentioning that measuring the exact waiting times of the various combinations of the system functions is at some level meaningless and misleading. For example, the creation, update, and other token mechanisms offered, rely heavily on the user's connection speed. The average mechanism waiting time was observed to be around 1 minute with the create mechanism waiting time being the longest one of them all due to its nature. Measuring such a quantity can be helpful to determine the user waiting times of the application that is going to be developed by the gaming company. One can easily observe that blockchain waiting times are very noticeable and need to be dealt with in a clever way by the decentralized application in order to ensure the user is immersed. The response times presented in Tables 5.1 and 5.2 are expected to be longer in the Main Network.

5.1.5 Discussion

Properties

Our system has many compelling security properties. Initially, it provides availability of metadata. Usually, in games as the one presented there, the gam-

[6] `https://trudi.weizenbaum-institut.de/ipfs_crawler.html`

ing company that produced the game hosts the in-game assets locally, on their (centralized) servers, in order to make them available to customers. However, if an outage happens, the servers would become unreachable and the data would become unavailable, causing the whole system to collapse. This is the case also if a gaming company uses IPFS. The gaming company has to provide the media files and the metadata of the in-game assets to the IPFS through their servers. On the other hand, our presented system achieves increased availability due to Filecoin's replication rules, in the sense that data will remain online in any case, even after the game company loses interest or goes bankrupt. Moreover, the proposed system is immune to single points of failure and resilient to Denial of Service (DoS) attacks, as all of its components are decentralized. Furthermore, it is tamper-resistant since all the data are stored on the blockchain and the metadata on IPFS, which is content addressable, i.e., for every chunk of data uploaded a unique CID is generated. Last but not least, thanks to Ethereum's blockchain, it is highly auditable and provides a degree of anonymity known in the literature as pseudonymity. Every NFT includes a metadata field pointing to the related asset, provided by the ERC-721 standard. For integrity reasons, the aforementioned should not be modified or else NFT loses a part of its value. Dat et al. [24] state that there is a number of marketplaces, among the most popular, which allow tampering with NFT's metadata. Our system considers evolvable in-game assets paired with the corresponding evolvable metadata. To achieve the aforementioned property, we leverage a decentralized name resolution service, i.e., ENS. Thus, when the digital art is updated, there is no need for change in the smart contract, keeping the cost from gas consumption low. The introduction of ENS to the system has offered two crucial advantages: a significantly more user-friendly appearance to the NFT's metadata and lower transaction fees. In order to update NFT's metadata, its ENS address content field has to be changed. The transaction fee for this action is on average 56854,25 Gwei ($0.10) and it is lower than directly updating NFT's URI via the smart contract, which costs on average 90546,66 Gwei ($0,16).[7] While the difference might seem insignificant, it greatly increases the game company's long term

[7] as measured on June 4, 2022.

revenue, considering the amount of transactions that might occur once the game grows in popularity. On the other hand, it is evident that ENS increases the complexity of the system. Despite the initial implications, mentioned in section 5.1.5, the advantages outweigh the disadvantages.

As we have explained in Section 5.1.2, the owner of the NFT is able to choose whether she wants to reveal the NFT's full scale artwork. This mechanism enables variable token prices, according to their seal state (closed or opened) and could have a significant impact on the revenue generated from the platform both for the business and the artist. Furthermore, the use of decentralized storage combined with the $(2, 3)$ threshold cryptosystem guarantees that even if the company stops supporting the game the owner will be able to fetch and decrypt the artwork.

We have already discussed that the vast majority of blockchain games that utilize NFTs suffer from many shortcomings concerning the actual ownership of the metadata files. For example, a CryptoKitties NFT is a digital, collectible "kitten" built on the Ethereum blockchain. It can be bought and sold using ether and bred to create new cats with different traits. The concept is very simple, but as the creators mention in their whitepaper, NFTs fail to succeed due to *"Provider Dependency; The existence of a digital collectible is dependent upon the existence of the issuing authority. If a digital collectible is created and the creator ceases to exist, the digital collectibles also cease to exist"* [74]. Our solution focuses heavily on the decentralization and preservation of the NFT metadata, breaking the dependency of the issuing authority. If the game company that minted the NFTs ceases to exist, the token continues to live on, as its artwork and metadata are safely stored on IPFS via the NFT.storage and Web3.storage services. The rights of ownership are also preserved by the smart contract on the Ethereum blockchain. Other games, such as Sorare[8] and Axie Infinity focus on using their own instance of the blockchain and do not specify the exact technology used to store NFT artworks and metadata. More importantly, the creators do not clarify what would happen in case of a server failure.

Another issue with legacy games is the general lack of NFTs use outside the

[8] `https://sorare.com/`

game environment, also pointed out in [74], as another reason of failure; *"Lack of Function; Physical collectibles are popular because of their intended purpose. Art is a great example: people collect it, it can be worth a lot of money, and it serves a purpose by hanging on the wall as a thing of beauty"*. Such a claim can now be considered outdated, due to the growing adoption of NFTs. The popular social media platform "Twitter" has initiated its own integration of NFTs, as user profile pictures in late January 2022.[9] This is just one of the many examples concerning the future usage of NFTs in various other applications outside of their original uses. However, it is obvious that such a rapid integration by third parties can compromise the integrity of NFTs. For example, if a user displays their NFT as a profile picture, the original full-size artwork is available to the public. A malicious user, e.g., a forger, could steal the original artwork and mint their own token. Suddenly, two identical looking versions of the NFT exist and the forger could contest ownership rights. This problem is successfully addressed in our system, as the full-size original artwork gets uploaded on IPFS encrypted. A smaller, low quality version of the artwork gets uploaded on IPFS unencrypted, as a sample of the original. The above description fits the purpose of a profile picture; a low quality picture being used to represent the account. Of course, forgers could also download the sample used as a profile picture, but it lacks the quality of the original. If they decide to mint a copy, the result would be poor and evident of fraud.

Challenges

All these different Web3 components are still in their infancy. So, it is unsurprising that their orchestration would lead to various challenges. Initially, the content hash processing algorithm of ENS automatically converted the CID v1 provided by NFT.storage to CID v0. The latter turned out to cause an error because NFT.storage produces an encoded ("dag-cbor") object, which cannot be translated to v1. To address this issue, we host the metadata file on Web3.storage and the artwork on NFT.storage. In the metadata file, there is a field pointing to

[9] `https://twitter.com/twitterblue/status/1484226494708662273?s=21`

the address of the artwork.

We have previously stated that a name resolution service is an essential component of the proposed system, which provides updatability and evolvability to in-game assets. Both ENS and IPNS complete our system and the use of each brings some advantages as well as some disadvantages. So, while in [71], we proposed the use of IPNS, ENS is proven to be more robust than IPNS, in the sense that every request to it is served. On the other hand, as we have seen, even with a large number of active connections, there is a high probability that a request will not be processed in IPNS, causing inconvenience to the users of the system. However, IPNS seems to be slightly faster than ENS under some circumstances, i.e., in case of extremely high number of active connections. Furthermore, it is very important to mention that the usage of IPNS comes at no monetary cost, unlike ENS, in which every registration or update action incurs fees. Finally, we observed that IPNS increases the complexity of using and supporting the system, since, due to the expiration of IPNS registrations, they should be announced regularly on the network. We claim that, overall, the use of ENS is a more rational choice than the choice of IPNS, as there are more points, where it has an advantage, keeping our system more sustainable.

5.1.6 Related Work

The research regarding blockchain and NFT gaming is still in its infancy. Pittaras et al. [75] discuss the feasibility of blockchain-based games. They perform an extensive evaluation of blockchain gaming to showcase the advantages and disadvantages. Min et al. [76] also conduct an in-depth review of blockchain gaming area, categorizing games based on the way they benefit from blockchains; rule transparency, asset ownership, asset reusability and user-generated content. Although they have included various NFT games in the category that leverages blockchains for asset ownership, they do not take into consideration actual ownership of the digital art. Moreover, the reviewed architecture is centralized, storing game assets on a company's file server.

Wang et al. [76] present an overview of the NFT domain. They study the

NFT ecosystem from multiple points of view, they conduct a security evaluation and demonstrate opportunities and challenges. The authors present two different protocols for NFT creation. In the first protocol, the artist creates tokens and sells them directly to potential buyers, while in the second, an NFT template is created and NFTs are produced utilizing the aforementioned template. Our system can serve more complex business models, based on decentralization for all interactions. Finally, the authors conclude that NFTs have a great potential in the gaming industry.

On the same wavelength, Rehman et al. [77] conduct a very detailed review of the NFT research area. They propose a categorization of NFTs based on their applications to digital art, fashion, collectibles, games (boosting game potential), domain names, virtual worlds and finally sports. Although we believe that the boundaries of the various categories are ambiguous, our system can be seen as a member of more than one of these categories, i.e., collectibles, digital art, and games. Moreover, the authors present multiple challenges NFT technology is called upon to overcome, e.g., security issues, legal issues, etc.

Fowler et al. [78] study the potential of NFTs for game development. They perform an in-depth investigation from different perspectives. They state that using NFTs in gaming can boost players' motivation through consumer-created content in addition to professional artists. Both can benefit by the use of NFTs, due to royalties at every resale in open or in-game markets. Finally, the authors highlight their severe security concerns. Cases in which the company stops hosting the file or the artwork is altered after being sold are some of them. Our scheme overcomes these obstacles, as it considers decentralized storage which is, thanks to IPFS, tamper-resistant and thanks to Filecoin, robust and permanent.

Finally, Muthe et al. [79] highlight the multiple shortcomings arising from game centralization. They argue that centralized storage and computation may lead to privacy leakage and cause high latency. To overcome the aforementioned barriers, they propose an architecture based on proxies for computing and IPFS for storing data. The proposed architecture incentivizes proxies with rewards on Ethereum. Although the design is generic enough, it cannot serve different business

models, e.g. outsourcing artwork of the game to artists, who are paid royalties. Moreover, the authors leverage IPFS for data storage without taking into consideration that with the current rules of IPFS, files are hosted (essentially only) by the original uploader; data get disseminated and cached by other nodes only if they become popular. Our proposed system instead supports artist royalties and deals with IPFS limitations utilizing Filecoin and in particular Web3.storage and NFT.storage.

5.1.7 Conclusions and Future Work

We developed and presented here an architecture that aims to create a fully decentralized and self-sustainable system using various Web3 components. More specifically, we leveraged NFTs backed by the Ethereum blockchain in the role of collectibles, as well as the IPFS ecosystem as decentralized file storage. We compared two different name resolution services, namely ENS and IPNS and selected ENS because of its robustness, even though it introduces monetary cost. The proposed system manages to overcome past obstacles related to NFT artwork, not only by giving the NFT owner the full-control of the file, but also the opportunity to keep its value high by choosing not to "open the box."

We realized a proof of concept implementation of our solution to evaluate it under realistic conditions and to compare the two versions of the name resolution service. Furthermore, an API is provided that enables users to easily set up such a system. Two radically different architectures are made available in order for users–typically, gaming companies–to select which one is more appropriate for their specific needs.

Lately, intense discussion is under way regarding the value of an NFT's digital art.[10] Although our system supports adding value to the artwork with the sealed-box emulation, another very interesting direction is the utilization of steganography. We argue that the use of steganographic techniques will be well suited to provide solid proof of authenticity of the digital art.[11] Thus, it is in our

[10] https://www.bbc.com/news/technology-59262326

[11] https://hackernoon.com/hiding-secrets-steganography-in-digital-arts-and-nfts-531z35f0

immediate plans to experiment with and support these techniques in our solution. Finally, an intriguing extension of our architecture would be a decentralized and company independent way for NFT-related actors (artists, buyers, et al.) to communicate with each other, not only for NFT-related topics, but also through a secure channel to exchange keys.

5.2 Enhancing Availability and Performance in IPFS Bitswap

The Web3 application layer depends not only on secure and decentralized infrastructures, but also on systems that can ensure the persistent availability of data over time while maintain a smooth user experience. IPFS, as the core storage protocol in this stack, supports content-addressed data retrieval over a P2P network. However, its performance and availability guarantees degrade in the presence of low-content popularity, limited replication, or peer churn. Swift service provision is imperative for IPFS-dependent applications to meet their demands. To enhance IPFS efficiency we propose a method to refine the Bitswap system [80], reducing dependence on the DHT, which can be adversely influenced by network dynamics. To this end, we initially conducted a sequence of experiments intending to assess the DHT's response time to a query aiming to locate providers for content that we had previously added to the network, thus making it unique. Our findings from 1800 experiments reveal that the median response time is 8 seconds, which is considered non-negligible. We then introduced our own enhancements to boost the chances that a content search will employ Bitswap rather than have to resort to the DHT. This strategy aims to decrease the total response time to ≤ 1 second. Our findings demonstrate that our method significantly increases the likelihood of finding content via Bitswap, especially for content that is not widely popular.

5.2.1 Measuring Times

In our experimental setup, we employed two distinct nodes: a client and a server. The server, situated on a workstation with a public IP address and not confined behind a firewall, played a crucial role. This server was responsible for generating random files, adding them to the IPFS network, and then announcing the respective root CID on a database (to avoid direct communication between the nodes).

The client node was utilized to retrieve the CID from the database and subsequently inquire the IPFS network for the corresponding content through the DHT. We meticulously measured the time it took for the DHT to respond to each inquiry. Specifically, the client, after the server added the file to the network, executed the command `ipfs dht findprovs <hash>` which returns up to 20 Peer IDs corresponding to providers of the requested file. The aforementioned experiment was systematically repeated for 24 hours, with a total of approximately $1,800$ randomly generated files. By executing this series of experiments, we aimed to evaluate the efficiency and responsiveness of the IPFS network in handling requests for randomly generated files. The measured response times from the DHT provide insights into the performance of IPFS in the context of file retrieval for a substantial dataset.

To enhance the objectivity of our results, we excluded 105 experiments out of the initial 1,800 that experienced timeouts, i.e., more than 2 minutes. Among the experiments that were successfully served, the average response time was 12 seconds, with a median of 8.4 seconds. These values, regardless of the specific metric used, are considered non-negligible. More detailed information regarding the distribution of response times are depicted in Figure 5.3.

This curation of results, by excluding timed-out experiments, ensures that the analysis is based on a subset of data where interactions with the IPFS network were successfully completed. The average and median response times provide insights into the typical performance observed during the retrieval of files from the IPFS network in our experimental setup.

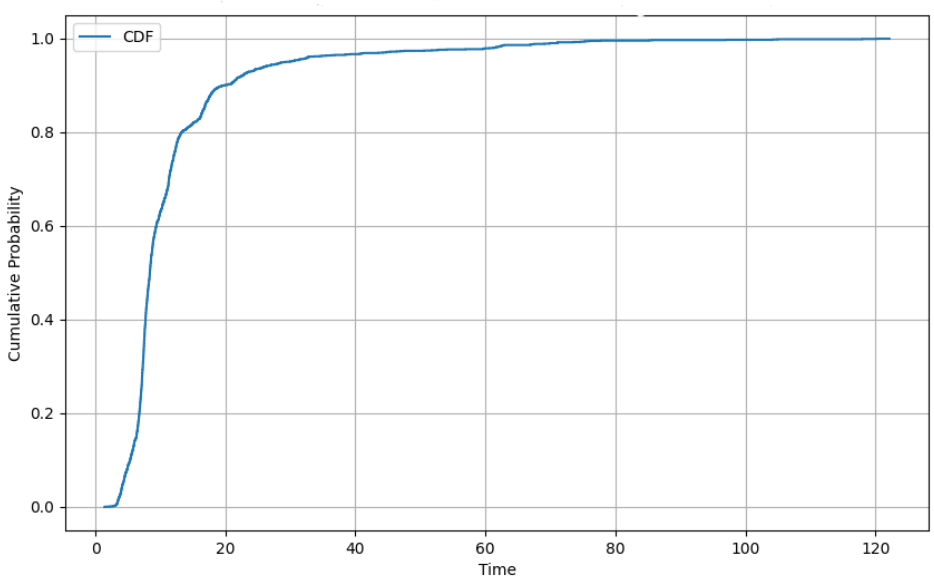


Figure 5.3: Cumulative Distribution Function plot of response times in seconds.

5.2.2 The `know` message

In Section 5.2.1, we observed that DHT response times in IPFS are not negligible and can potentially impact the functionality of an IPFS-based service. To address this issue, we propose an enhancement to Bitswap aimed at reducing the likelihood of a query going unanswered by Bitswap, which would cause the requester to resort to the, much slower, DHT. The proposed enhancement involves introducing a new message type called `know` to the list of Bitswap messages.

The process unfolds as follows: when a client broadcasts a request for a root CID, the nodes within the swarm individually check if they locally store the file. If so, they respond with a `have` message, as usual. As an additional step, we suggest that nodes check if they store a Provider Record for this CID, that is, if they know someone that stores the content (as opposed to having the content themselves). If a Provider Record is found, the node responds with a message in the format `know PeerID`, where PeerID is the identifier of the node that, at some point, advertised that it provides the file. Subsequently, the client incorporates that Peer ID into the ongoing session. The aforementioned procedure is depicted in Figure 5.4.

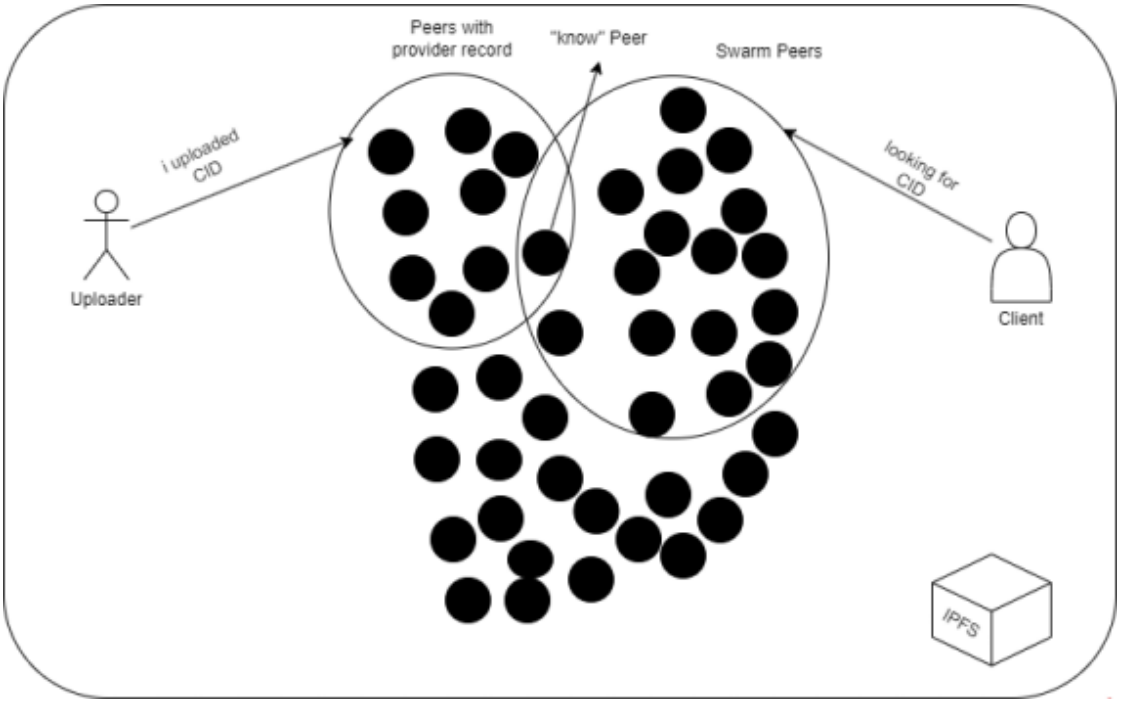


Figure 5.4: Bird's-eye view of the proposed method.

Merely possessing the provider's PeerID is insufficient for establishing contact; what is required is the network address of the peer. To avoid immediately resorting to peer discovery, such as a DHT walk to locate the Peer Record, we propose utilizing the nodes' address books. The process is as follows: When a node gets a `want-have` message, it first checks if the file is stored locally and, if so, replies with a `have` message. If the file is not found, the node checks for a Provider Record associated with that Content Identifier (CID) and responds with a `know` message. Additionally, the node consults its address book to see if it stores a Peer Record, which, if found, is sent along with the `know message`. Consequently, when the requester receives the `know` message, they will also be informed of the Peer ID and, ideally, obtain the Peer Record, enabling them to link the Peer ID to a physical address. If this information is not available, the process transitions to the Peer discovery stage, where the requester first checks for a stored peer record for the specific Peer ID before turning to the DHT for more information. The flow of the aforementioned procedure is depicted in Figure 5.5.

This proposed modification aims to enhance the efficiency of Bitswap by

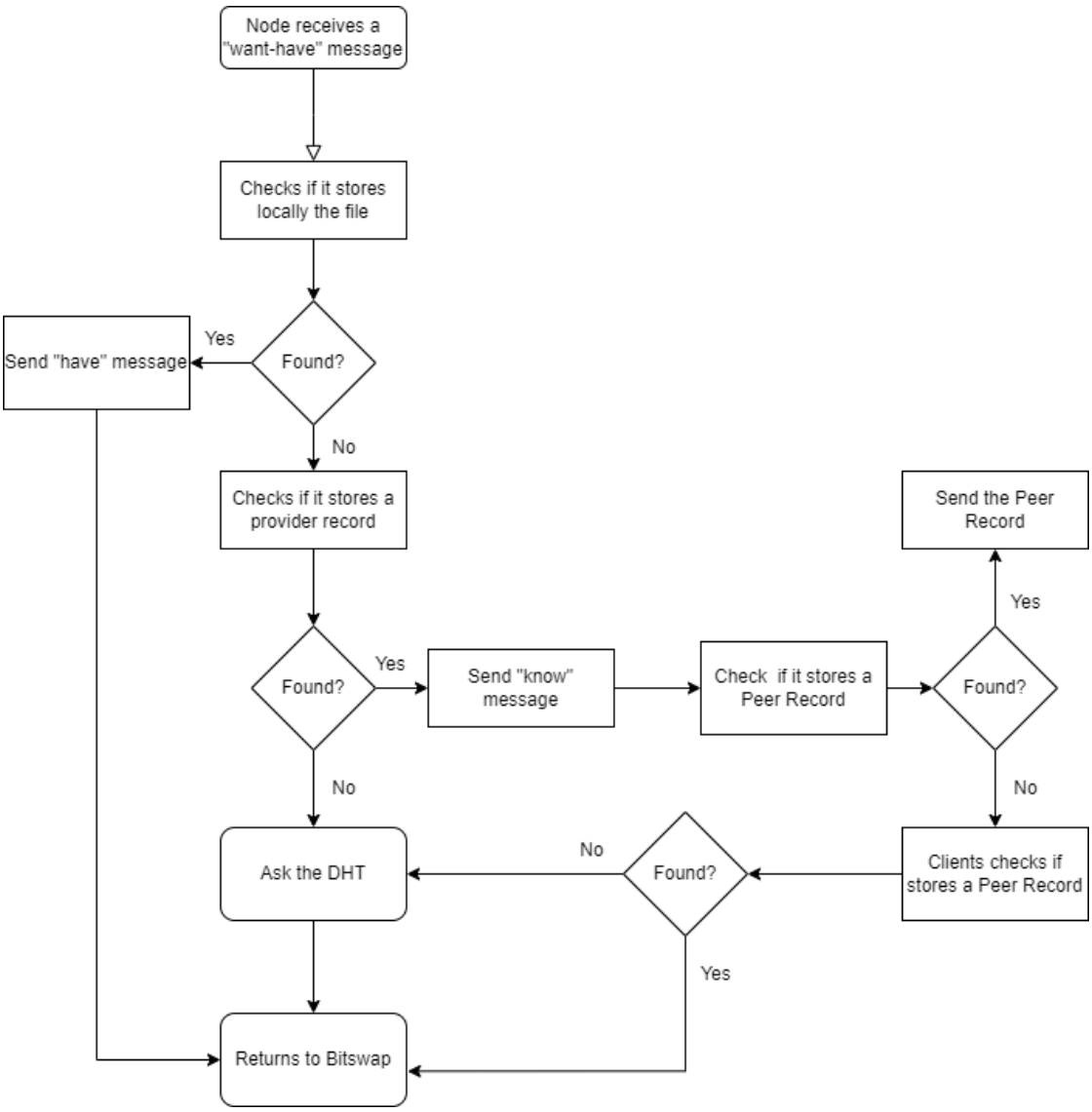


Figure 5.5: The operation flow of the proposed enhanced Bitswap.

providing the client with information about nodes that have previously advertised their capability to provide the requested file. By introducing the `know` message, the client can potentially reduce its reliance on the DHT and improve the success rate of queries served by Bitswap, contributing to a more reliable and responsive IPFS-based service.

5.2.3 Evaluation

To estimate the impact of our proposed scheme, we first need to have an estimate of the size of the IPFS network. For this reason we used the Nebula DHT Crawler [81]. The measurements show that during the measurement period the network size was about $\nu = 15300$ nodes. Let p be the probability that a node storing the provider record of the original uploader, is also part of the client's swarm. We approach the problem as follows. We consider each of the broadcasted messages as an attempt to select at least one Peer from the 20 that have the provider record. The probability of finding no one on the first try is $\frac{15280}{15300}$, the second is $\frac{15279}{15299}$ etc. So in general the probability of not selecting even one is $q = \prod_{i=1}^{\nu_s} \frac{15280 - i}{15300 - i}$, where $\nu_s = \#$ peers in the swarm. The probability we are looking for is $p = 1 - q$, which is the probability to find at least one.

We also conducted a series of Monte Carlo experiments to delve into the performance nuances of two versions of Bitswap, the baseline and the proposed. The primary objective was to ascertain the probability of a query receiving a response from Bitswap across these two distinct versions.

Recognizing the multifaceted nature of network dynamics, particularly in relation to file availability and peer participation, we meticulously designed our experiments to incorporate varying levels of file popularity. Additionally, we explored the impact of swarm size, which as we mentioned before ranges from 600 to 900 participants. We categorized swarm sizes into three distinct scenarios: a conservative setting with $\nu_s = 600$ peers, representing a constrained network environment; an intermediate scenario with $\nu_s = 750$ peers, reflecting an average network configuration; and an optimistic scenario with $\nu_s = 900$ peers. Each sce-

nario underwent rigorous evaluation through 100,000 iterations, ensuring robust statistical analysis and reliable conclusions. The results are illustrated in Figure 5.6.

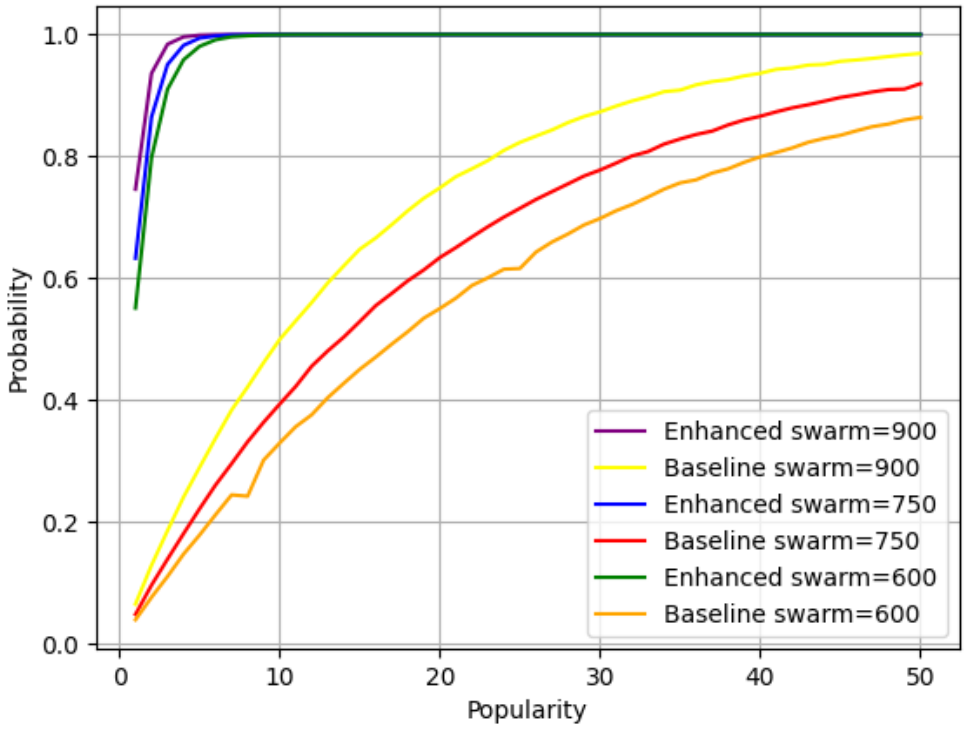


Figure 5.6: The probability of response per popularity of file.

Our analysis reveals a marked improvement in the likelihood of query responses with the introduction of the proposed solution, particularly evident in scenarios featuring files with low popularity levels. To illustrate, when examining files with a popularity rating of 1 and a swarm size of 600, the probability of receiving a response under the current system stands at a mere 4%. However, with the implementation of the proposed solution, this probability surges to a notable 55%. Similarly, for files boasting a popularity rating of 10 under the same swarm size conditions, the probability escalates from 32% with the existing system to an impressive 99.2% with the proposed upgrade.

It is worth noting that as file popularity increases, the gap in probability

between the current and proposed solutions gradually narrows. Nevertheless, the substantial improvement afforded by the proposed version remains evident across the entire spectrum of file demand. This enhancement signifies a significant advancement in the IPFS, particularly in its ability to facilitate efficient file exchange, especially in scenarios characterized by low file traffic.

Another significant limitation confronting Bitswap under its current rules of operation is scalability. In order to maintain its present level of effectiveness, the swarm's size must remain in proportion to the network's size. For instance, if we imagine a scenario where the network comprises $15,000$ nodes and each node's swarm has 600 active connections, then if the network expands to $60,000$ nodes, each node's active connections should increase to $2,400$ in order to sustain the same success rate. We conducted a series of experiments which indicate that leveraging the `know` message enhances Bitswap's efficiency without needing to expand the swarm size. In our experiments, we maintained file popularity at an average level [82], with 5 replicas, and examined how response probabilities changed with larger network sizes. These tests were carried out for three different swarm sizes, comparing the standard operation to the operation utilizing the know message, each over $100,000$ iterations. As shown in Figure 5.7, even when the network size reaches $65,000$ nodes—roughly four times larger than our initial measurements—the standard method with $\nu_s = 600$ only achieves a success rate below 10%, whereas the modified approach achieves a 60% success probability. In the extreme scenario where the network expands to $100,000$ nodes, the suggested approach maintains a 45% success rate, significantly outperforming the baseline's modest 3% probability, making Bitswap futureproof. The findings of the experiments are elaborately presented in Figure 5.7.

5.2.4 Conclusions and Future Work

Given the crucial importance of IPFS in the Web3 ecosystem, we contend that operational efficiency is paramount. Despite ongoing upgrades, there remains room for improvement, particularly concerning response times. In our paper, we introduce a method aimed at enhancing the performance of Bitswap. Our goal

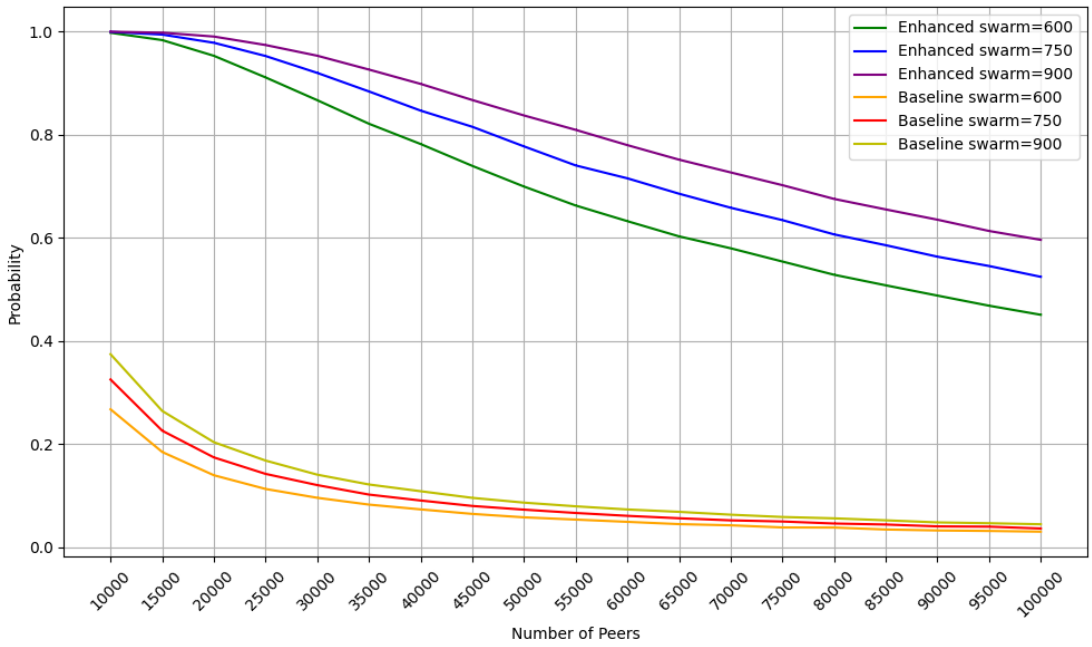


Figure 5.7: The probability of response with growing network size.

is to ensure that queries are resolved by Bitswap itself, eliminating the need to rely on the DHT, which, as observed, can be time-consuming. In the near future, our objective is to conduct measurements using Testground,[12] a simulation tool tailored for the IPFS network. Through this initiative, we aim to validate the efficacy of the proposed method and assess benefits that stem by its use.

[12]https://docs.testground.ai/master/

Chapter 6

Security Advancements in Decentralized Architectures

Building upon the technical foundations, architectures, and implementation strategies discussed in the preceding chapters, this chapter provides a thematic synthesis and reflective discussion of the dissertation's core contributions. It explores how blockchain technologies can serve as enablers of secure and auditable IoT systems, and how privacy-preserving techniques can be integrated into decentralized storage infrastructures like IPFS. By revisiting select case studies and complementary research works, the chapter highlights the broader implications of decentralized architectures for real-world security and privacy challenges. The two sections that follow analyze, respectively, the role of blockchain-enabled digital twins in IoT control, and the enhancement of privacy in peer-to-peer storage protocols. Together, they demonstrate how the key research themes of this thesis, trust decentralization, system interoperability, and privacy awareness, can be realized in practical, scalable ways.

6.1 Blockchain-Enabled Security Architectures for the Internet of Things

The Internet of Things (IoT) is envisioned as a networked environment where interconnected devices seamlessly integrate the physical and digital domains, offering a wide spectrum of services aimed at enhancing human life. Nevertheless, the IoT landscape is marked by several enduring challenges. A fundamental issue is fragmentation, as the ecosystem comprises a multitude of heterogeneous devices from different manufacturers, each employing distinct communication protocols and standards.

To address this, the Web of Things (WoT) initiative,[1] developed by the W3C working group, proposes a standardized, Web-based framework to achieve interoperability across diverse IoT platforms. WoT builds upon widely adopted Web technologies—such as RESTful APIs and HTTP(s)—facilitating device discovery, access, and integration within a common application layer. This approach effectively mitigates the fragmentation and interoperability issues that plague many IoT deployments.

Beyond interoperability, security in IoT systems presents a complex challenge. Traditional cryptographic mechanisms are often ill-suited for the resource-constrained nature of IoT devices, many of which lack the computational power required for executing heavyweight security protocols. Consequently, there is a pressing need for lightweight and efficient security solutions that align with the inherent limitations of IoT hardware. Additionally, the direct interaction between IoT systems and the physical world elevates concerns related to security, safety, and privacy. Devices are frequently deployed in publicly accessible environments and are exposed to potential manipulation or misuse. One promising approach to mitigating these risks is the introduction of digital twins—virtual representations of physical IoT entities [83]. Although traditionally used for testing, monitoring, and simulation purposes, digital twins in this context are proposed as intermediary layers of abstraction and protection. Rather than interacting with the physical

[1] `https://www.w3.org/WoT/`

device itself, users engage solely with its digital twin. Any validated changes to the digital twin's state are then securely propagated to the actual device, which performs the corresponding actions. This design introduces a layer of indirection that enhances both control and security in IoT deployments.

Building upon this and aiming to overcome the limitations previously discussed in work [84], we propose a system in which the WoT framework and the Ethereum blockchain collaborate to implement digital twins for IoT environments. In this system, consumers do not directly interact with IoT gateways or devices. Instead, they communicate exclusively with the digital twin of the virtual entity, which is implemented as a smart contract and deployed on the Ethereum blockchain. When a consumer wishes to trigger an actuation operation, they deposit a specific number of tokens into the smart contract, which holds them in escrow, and submit a transaction containing the desired action and its parameters. The consumer can query the blockchain freely to discover the available actions and their expected parameters. The smart contract verifies the validity of the transaction and the sufficiency of the escrow tokens; if both checks pass, it emits an event. IoT gateways are configured to monitor the blockchain for such events. Once detected, the gateway maps the event to a concrete actuation task and forwards the appropriate command to the IoT device(s), based on vendor-specific communication protocols. In the same spirit, the work presented in [85] explores the use of a permissioned blockchain, namely Hyperledger Fabric, instead of Ethereum. This design choice is motivated by the need to mitigate performance-related limitations such as high transaction latency, scalability constraints, and elevated operational costs. Moreover, permissionless blockchains may be unsuitable for certain use cases—such as smart home environments—due to their inherently public and transparent nature, which may conflict with privacy or control requirements. Finally, the work presented in [86] summarizes and extends our previous efforts by implementing the proposed solution on two distinct blockchain platforms: Ethereum and Hyperledger Fabric, each offering unique advantages tailored to different application contexts. Specifically, we design, implement, and evaluate an IoT system that employs smart contract-based digital twins to enable secure sensing and actuation

in distributed environments.

6.2 Enhancing IPFS Privacy

IPFS is a P2P system, and as is often the case with such architectures, privacy is frequently sacrificed in favor of performance. Numerous studies have shown that IPFS exhibits significant privacy shortcomings, as it tends to expose user and content-related information openly across the network. Researchers in [7] demonstrate a privacy attack on the IPFS network by exploiting the Bitswap protocol. They introduce a set of attack vectors which give the attacker the ability to track nodes that are requesting a specific content identifier (CID), monitor the full set of data a particular node is attempting to retrieve, and reveal the historical content interests of a node. In addition to malicious attackers, security analysts can also take advantage of the privacy limitations inherent in the Bitswap protocol. In [65] researchers propose IF-DSS, a digital forensics investigation framework for Decentralized Storage Services (DSSs). Their work includes a comparative analysis of major DSS platforms from a digital forensics perspective and applies the proposed methodology specifically to IPFS, demonstrating how its design choices expose useful artifacts for forensic investigation. One proposed privacy-enhancing solution tailored to IPFS involves the use of Bloom filters instead of plain CIDs, in order to obfuscate request and response messages exchanged via the Bitswap protocol [67]. This approach aims to conceal the exact data interests of nodes, thereby mitigating exposure to tracking attacks. In our work [66], we enhance IPFS privacy by introducing a triple hashing scheme for content storage and lookup operations. This approach obscures the identity of the content being requested, particularly from intermediate peers, who are unable to infer what a node is searching for. During the actual content exchange phase, the mechanism transitions to single or double hashing, ensuring efficiency without compromising privacy.

To make an object available on IPFS, an uploader begins by generating the object's Content Identifier (CID), derived by hashing the content along with some metadata. Then, using the Kademlia Distributed Hash Table (DHT), it identifies

the k nodes whose PeerIDs are closest to the CID and stores at each of them a Provider Record pointing to itself, along with a corresponding Peer Record. In our approach, uploader additionally computes a *triple hash* of the object, resulting in what we denote as CID^3. It repeats the DHT insertion process, this time using CID^3 instead of the original CID. For clarity, we refer to the original content hash as CID^1, the double hash as CID^2, and the triple hash as CID^3. When a client attempts to retrieve the object, it starts from the known CID^1, which may have been obtained through previous access, a web source, or another reference. The client computes CID^3 and sends a `FINDVALUE` RPC to the three closest nodes to CID^3 in its routing table. If the first node contacted does not store the content, it responds with a referral to a closer node. The latter, being among the k closest to CID^3, responds with a Provider Record indicating the node that holds the requested object. At this stage, the client switches to the Bitswap protocol to initiate the actual data transfer. It sends a `WANT_HAVE` message to provider node using CID^3. The node replies with a `HAVE` message, this time including CID^2. The client verifies the validity of this response by computing CID^2 from CID^1, confirming that the provider node possesses the correct content. Finally, the client sends a `WANT_BLOCK` message containing the true CID^1. Node verifies that the client had prior knowledge of the original CID and returns the object in a `BLOCK` message.

Our solution supports optional content encryption, which can facilitate privacy-preserving caching. Moreover, the privacy and security analysis demonstrates that, beyond improving anonymity, the proposed scheme is effective against a variety of DDoS attacks targeting IPFS. Importantly, the design introduces no routing delays, avoids reliance on trusted third parties, and incurs negligible computational overhead, making it practical and scalable.

Chapter 7

Conclusions and Future Work

In this chapter, we provide an overview of the core findings and conclusions drawn from the studies discussed in the previous chapters. We also suggest possible paths for future research, building on the theoretical and empirical groundwork laid by this dissertation.

7.1 Conclusions

Web3 has emerged as the next evolutionary phase of the Internet, emphasizing user empowerment across multiple dimensions, including financial ownership and control over personal data. While the vision of Web3 may appear utopian, in practice it faces significant challenges. Many of these stem from user behavior in environments lacking centralized regulatory or governance structures. Moreover, the Web3 ecosystem remains vulnerable to malicious actors—who are by no means a new phenomenon—but who now exploit both the technological immaturity of the infrastructure and the inexperience of its users for their own gain. With these considerations in mind, we investigated potential attack vectors that combine various components of the Web3 stack, such as DLTs and IPFS. Our study also examined the ethical alignment of IPFS nodes. Finally, we propose application-layer solutions, specifically within the domain of blockchain gaming, one of the most prominent sectors at this layer, to enhance system availability and improve overall performance.

Initially, we observed that the Ransomware-as-a-Service model is being realized within the decentralized ecosystem through the use of smart contracts deployed on Ethereum and content hosted on IPFS. We found that malicious actors benefit significantly from this setup, as it enables them to automate transactions at minimal cost, maintain a high degree of privacy, and operate in an environment where takedown by LEAs is considerably more difficult. Beyond this specific attack vector, we also explored how malicious actors can leverage components of the IPFS ecosystem, such as public gateways and pinning services, to distribute and persist malicious content. Through empirical measurements, we found that due to the inherent characteristics of these entities, it is indeed possible for an attacker to anonymously share and sustain the availability of malicious files within the network. In addition, we investigated the presence of malicious activity within the IPFS ecosystem at the node and file levels, and found clear evidence of exploitation by malicious actors. A significant portion of nodes were identified as malicious by reputable threat intelligence services, and many of these nodes remained active for extended periods—often longer than a month—leveraging the network for their own benefit. Furthermore, we identified malicious usage at the file level by passively monitoring the content exchanged between nodes as well as the files uploaded to public pinning services.

Alongside this, we designed and proposed application-layer solutions within the Web3 ecosystem, specifically focusing on the blockchain gaming domain, with the objective of ensuring reliable file availability. Our approach addresses previous challenges related to NFT artwork by giving NFT owners not only full control over their associated files, but also the ability to maintain or even increase the value of their digital assets. We also propose a complementary solution targeting the IPFS layer, which enhances both availability and performance, ultimately leading to a smoother user experience at the application level.

In summary, this dissertation highlights the dual nature of decentralization in Web3: while offering enhanced user empowerment and innovation, it simultaneously introduces complex security and trust challenges. By identifying and analyzing emerging attack vectors and misuse within decentralized infrastructures

like IPFS and Ethereum, we provide a deeper understanding of the current threat landscape. At the same time, our proposed application-layer solutions demonstrate that resilience and performance can be achieved without compromising decentralization. Ultimately, this work aspires to contribute both to the critical assessment and the responsible advancement of Web3 technologies.

7.2 Future Work

While this dissertation provides a foundational analysis of security and availability challenges in the Web3 ecosystem, several avenues remain open for further exploration. First, future research could focus on the development of more robust and decentralized countermeasures against malicious activity in content-addressed networks such as IPFS. Importantly, these solutions should aim to preserve the peer-to-peer and decentralized nature of such systems, avoiding reliance on centralized mechanisms like global blacklists, which undermine core Web3 principles. Investigating reputation-based or community-driven verification mechanisms could offer effective alternatives. Extending this analysis to other peer-to-peer storage networks—such as Filecoin and Storj—would provide a broader understanding of how similar infrastructures handle availability, persistence, and resistance to malicious actors.

Moreover, the proposed application-layer solutions—particularly in the context of blockchain gaming and NFT file availability—could be further evaluated under real-world deployment conditions. Emphasis should be placed on assessing their scalability, user experience, and resilience against adversarial behaviors in live environments. Expanding these mechanisms to additional domains—such as decentralized social media, distributed scientific data sharing, or decentralized finance platforms, could further validate their utility and adaptability.

A final promising direction could focus on designing adaptive threat detection mechanisms that operate effectively in peer-to-peer contexts, without centralized control or identity verification. In particular, developing decentralized trust models, anomaly detection frameworks, and abuse-resistant incentive schemes will

be essential to safeguarding the long-term viability of Web3 infrastructures.

Appendix A

Acronyms

C2 Command and Control

CIA Confidentiality - Integrity - Availability

CID Content ID

DAG Directed Acyclic Graph

DAO Decentralized Autonomous Organization

DApp Decentralized Application

DDoS Distributed Denial of Service

DGA Domain Generation Algorithm

DHT Distributed Hash Table

DIDs Decentralized Identifiers

DLTs Distributed Ledger Technologies

DNS Domain Name System

DoS Denial of Service

DSA Digital Services Act

DSS Decentralized Storage Service

DeFi Decentralized Finance

EIPs Ethereum Improvement Proposals

ENS Ethereum Name Service

ERCs Ethereum Request for Comments

ETH Ethereum

EVM Ethereum Virtual Machine
HTTP Hypertext Transfer Protocol
IPFS InterPlanetary File System
IPNI InterPlanetary Name Index
IPNS InterPlanetary Naming System
IoT Internet of Things
KYC Know Your Customer
LEAs Law Enforcement Agencies
LRU Least Recently Used
MaaS Malware as a Service
NFT Non-Fungible Token
NIZK Non-Interactive Zero-Knowledge
OTP One-Time Password
P2E Play to Earn
P2P Peer to Peer
PBFT Practical Byzantine Fault Tolerance
PoS Proof of Stake
PoW Proof of Work
RDP Remote Desktop Protocol
RaaS Ransomware as a Service
SP Service Provider
SPV Simplified Payment Verification
SSH Secure Shell
TLS Transport Layer Security
WASM WebAssembly
WoT Web of Things

Bibliography

[1] R. Huang, J. Chen, Y. Wang, T. Bi, L. Nie, and Z. Zheng, "An overview of web3 technology: Infrastructure, applications, and popularity," *Blockchain: Research and Applications*, vol. 5, no. 1, p. 100173, 2024.

[2] I. Makhdoom, B. Alzahrani, M. R. Bakar, F. K. Hussain, and A. Gani, "Web 3.0 and sustainability: Challenges and research opportunities," *Sustainability*, vol. 15, no. 4, 2023, projection: 1 billion users by 2027. [Online]. Available: https://www.mdpi.com/2071-1050/15/4/3259

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[4] D. Trautwein, A. Raman, G. Tyson, I. Castro, W. Scott, M. Schubotz, B. Gipp, and Y. Psaras, "Design and evaluation of ipfs: a storage layer for the decentralized web," in *Proceedings of the ACM SIGCOMM 2022 Conference*, ser. SIGCOMM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 739–752. [Online]. Available: https://doi.org/10.1145/3544216.3544232

[5] C. Karam and V. Kamluk, "Blockchainware-decentralized malware on the blockchain," *Black Hat ASIA*, 2015.

[6] Bitdefender, "Looking Into the Eye of the Interplanetary Storm," https://www.bitdefender.com/files/News/CaseStudies/study/376/Bitdefender-Whitepaper-IPStorm.pdf, 2020.

[7] L. Balduf, S. Henningsen, M. Florian, S. Rust, and B. Scheuermann, "Monitoring data requests in decentralized data storage systems: A case study of IPFS," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2022, pp. 658–668.

[8] S. Wan, H. Lin, W. Gan, J. Chen, and P. S. Yu, "Web3: The next internet revolution," *IEEE Internet of Things Journal*, vol. 11, no. 21, pp. 34 811–34 825, 2024.

[9] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[10] J. Benet and N. Greco, "Filecoin: A decentralized storage network," *Protoc. Labs*, vol. 1, pp. 1–36, 2018.

[11] P. P. Ray, "Web3: A comprehensive review on background, technologies, applications, zero-trust architectures, challenges and future directions," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 213–248, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667345223000305

[12] D. Sheridan, J. Harris, F. Wear, J. C. Jr, E. Wong, and A. Yazdinejad, "Web3 challenges and opportunities for the market," 2022. [Online]. Available: https://arxiv.org/abs/2209.02446

[13] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," National Institute of Standards and Technology, Tech. Rep., 2018.

[14] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics Informatics*, vol. 36, pp. 55–81, 2019. [Online]. Available: https://doi.org/10.1016/j.tele.2018.11.006

[15] P. Paul, P. Aithal, R. Saavedra, and S. Ghosh, "Blockchain technology and its types—a short review," *International Journal of Applied Science and Engineering (IJASE)*, vol. 9, no. 2, pp. 189–200, 2021.

[16] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[17] M. Hearn and R. G. Brown, "Corda: A distributed ledger," *Corda Technical White Paper*, vol. 2016, p. 6, 2016.

[18] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the works of leslie lamport*, 2019, pp. 203–226.

[19] M. G. Vigliotti, "What do we mean by smart contracts? open challenges in smart contracts," *Frontiers in Blockchain*, vol. 3, p. 553671, 2021.

[20] J. J. Kearney and C. A. Perez-Delgado, "Vulnerability of blockchain technologies to quantum attacks," *Array*, vol. 10, p. 100065, 2021.

[21] J. Ahn, E. Yi, and M. Kim, "Ethereum 2.0 hard fork: Consensus change and market efficiency," *The Journal of The British Blockchain Association*, 2024.

[22] W. Entriken and D. Shirley and J. Evans and N. Sachs. (2018) EIP-721: ERC-721 Non-Fungible Token Standard. [Online]. Available: https://eips.ethereum.org/EIPS/eip-721

[23] W. Rehman, H. e Zainab, J. Imran, and N. Z. Bawany, "Nfts: Applications and challenges," in *2021 22nd International Arab Conference on Information Technology (ACIT)*. IEEE, 2021, pp. 1–7.

[24] D. Das, P. Bose, N. Ruaro, C. Kruegel, and G. Vigna, "Understanding security issues in the nft ecosystem," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 667–681.

[25] Nick Johnson. (2016) EIP-137: Ethereum Domain Name Service - Specification. [Online]. Available: https://eips.ethereum.org/EIPS/eip-137

[26] F. Casino, N. Lykousas, V. Katos, and C. Patsakis, "Unearthing malicious campaigns and actors from the blockchain dns ecosystem," *Computer Communications*, vol. 179, pp. 217–230, 2021.

[27] J. Ernstberger, J. Lauinger, F. Elsheimy, L. Zhou, S. Steinhorst, R. Canetti, A. Miller, A. Gervais, and D. Song, "Sok: data sovereignty," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2023, pp. 122–143.

[28] E. Daniel and F. Tschorsch, "Ipfs and friends: A qualitative comparison of next generation peer-to-peer data networks," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 31–52, 2022.

[29] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[30] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International workshop on peer-to-peer systems*. Springer, 2002, pp. 53–65.

[31] A. De la Rocha, D. Dias, and Y. Psaras, "Accelerating content routing with bitswap: A multi-path file transfer protocol in IPFS and filecoin," ProbeLab, Tech. Rep., 2021.

[32] Y. Wei, D. Trautwein, Y. Psaras, I. Castro, W. Scott, A. Raman, and G. Tyson, "The eternal tussle: exploring the role of centralization in {IPFS}," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 441–454.

[33] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. Nyang, and A. Mohaisen, "Exploring the Attack Surface of Blockchain: A Systematic Overview," *arXiv e-prints*, Apr. 2019.

[34] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 375–392.

[35] Y. Marcus, E. Heilman, and S. Goldberg, "Low-resource eclipse attacks on ethereum's peer-to-peer network." *IACR Cryptology ePrint Archive*, vol. 2018, p. 236, 2018.

[36] M. Saad, L. Njilla, C. Kamhoua, J. Kim, D. Nyang, and A. Mohaisen, "Mempool optimization for defending against ddos attacks in pow-based blockchain systems," in *2019 IEEE international conference on blockchain and cryptocurrency (ICBC)*. IEEE, 2019, pp. 285–292.

[37] I. S. Evaluators, "Ethercombing: Finding Secrets in Popular Places," https://www.securityevaluators.com/casestudies/ethercombing/, 2019, [Online; published 23-April-2019].

[38] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts," Cryptology ePrint Archive, Report 2016/1007, 2016, https://eprint.iacr.org/2016/1007.

[39] J. Moubarak, E. Filiol, and M. Chamoun, "Developing a K-ary malware using Blockchain," *arXiv e-prints*, Apr. 2018.

[40] M. Crosby, P. Pattanayak, S. Verma, V. Kalyanaraman *et al.*, "Blockchain technology: Beyond bitcoin," *Applied Innovation*, vol. 2, no. 6-10, p. 71, 2016.

[41] S. Pletinckx, C. Trap, and C. Doerr, "Malware coordination using the blockchain: An analysis of the cerber ransomware," 08 2018.

[42] S. T. Ali, P. McCorry, P. H.-J. Lee, and F. Hao, "Zombiecoin 2.0: managing next-generation botnets using bitcoin," *International Journal of Information Security*, vol. 17, no. 4, pp. 411–422, Aug 2018. [Online]. Available: https://doi.org/10.1007/s10207-017-0379-8

[43] A. Juels, A. Kosba, and E. Shi, "The ring of gyges: Investigating the future of criminal smart contracts," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 283–295. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978362

[44] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, and Y. Zhou, "Detecting ponzi schemes on ethereum: Towards healthier blockchain technology," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2018, pp. 1409–1418. [Online]. Available: https://doi.org/10.1145/3178876.3186046

[45] C. Patsakis, D. Arroyo, and F. Casino, "The malware as a service ecosystem," in *Malware: Handbook of Prevention and Detection*. Springer, 2024, pp. 371–394.

[46] C. Karapapas, I. Pittaras, N. Fotiou, and G. C. Polyzos, "Ransomware as a service using smart contracts and ipfs," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2020, pp. 1–5.

[47] Y. Zhong, A. Zhou, L. Zhang, F. Jing, and Z. Zuo, "Dustbot: A duplex and stealthy p2p-based botnet in the bitcoin network," *PloS one*, vol. 14, no. 12, p. e0226594, 2019.

[48] C. Patsakis and F. Casino, "Hydras and IPFS: A Decentralised Playground for Malware," *International Journal of Information Security*, vol. 18, no. 6, pp. 787–799, December 2019. [Online]. Available: https://doi.org/10.1007/s10207-019-00443-0

[49] T. de Balthasar and J. Hernandez-Castro, "An analysis of bitcoin laundry services," in *Secure IT Systems - 22nd Nordic Conference, NordSec 2017, Tartu, Estonia, November 8-10, 2017, Proceedings*, ser. Lecture Notes in Computer Science, H. Lipmaa, A. Mitrokotsa, and R. Matulevicius, Eds., vol. 10674. Springer, 2017, pp. 297–312. [Online]. Available: https://doi.org/10.1007/978-3-319-70290-2_18

[50] C. Karapapas, G. C. Polyzos, and C. Patsakis, "What's inside a node? malicious ipfs nodes under the magnifying glass," in *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer, 2023, pp. 149–162.

[51] J. Althouse, "Easily Identify Malicious Servers on the Internet with JARM," 2020. [Online]. Available: https://engineering.salesforce.com/easily-identify-malicious-servers-on-the-internet-with-jarm-e095edac525a

[52] S. Henningsen, M. Florian, S. Rust, and B. Scheuermann, "Mapping the Interplanetary Filesystem," in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 289–297.

[53] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[54] A. A. Mamun, S. R. Hasan, M. S. Bhuiyan, M. S. Kaiser, and M. A. Yousuf, "Secure and Transparent KYC for Banking System Using IPFS and Blockchain Technology," in *2020 IEEE Region 10 Symposium (TENSYMP)*, 2020, pp. 348–351.

[55] M. Yao, J. Fuller, R. P. Kasturi, S. Agarwal, A. K. Sikder, and B. Saltaformaggio, "Hiding in plain sight: An empirical study of web application abuse in malware," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 6115–6132.

[56] M. Allegretta, G. Siracusano, R. González, M. Gramaglia, and J. Caballero, "Web of shadows: Investigating malware abuse of internet services," *Computers & Security*, vol. 149, p. 104182, 2025.

[57] T. V. Doan, Y. Psaras, J. Ott, and V. Bajpai, "Toward decentralized cloud storage with ipfs: opportunities, challenges, and future considerations," *IEEE Internet Computing*, vol. 26, no. 6, pp. 7–15, 2022.

[58] B. Guidi, A. Michienzi, and L. Ricci, "Data persistence in decentralized social applications: The IPFS approach," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2021, pp. 1–4.

[59] E. Politou, E. Alepis, C. Patsakis, F. Casino, and M. Alazab, "Delegated content erasure in IPFS," *Future Generation Computer Systems*, vol. 112, pp. 956–964, 2020.

[60] S. Sokoto, L. Balduf, D. Trautwein, Y. Wei, G. Tyson, I. Castro, O. Ascigil, G. Pavlou, M. Korczynski, B. Scheuermann, and M. Król, "Guardians of the galaxy: Content moderation in the interplanetary file system," in *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*, D. Balzarotti and W. Xu, Eds. USENIX Association, 2024.

[61] R. Dingledine, N. Mathewson, P. F. Syverson *et al.*, "Tor: The second-generation onion router." in *USENIX security symposium*, vol. 4, 2004, pp. 303–320.

[62] N. Kshetri, "Privacy violations, security breaches and other threats of Web3 and the metaverse," International Telecommunications Society (ITS), 32nd European Regional ITS Conference, Madrid 2023: Realising the digital decade in the European Union – Easier said than done? 277993, 2023.

[63] B. Wang, T. Liu, W. Wang, Y. Weng, C. Li, G. Xu, M. Shen, S. Zhu, and W. Wang, "The Illusion of Anonymity: Uncovering the Impact of User Actions on Privacy in Web3 Social Ecosystems," 2024. [Online]. Available: https://arxiv.org/abs/2405.13380

[64] C. F. Torres, F. Willi, and S. Shinde, "Is your wallet snitching on you? an analysis on the privacy implications of web3," in *Proceedings of the 32nd USENIX Conference on Security Symposium*, ser. SEC '23. USA: USENIX Association, 2023.

[65] J. Son, G. Kim, H. Jung, J. Bang, and J. Park, "If-dss: A forensic investigation framework for decentralized storage services," *Forensic Science International: Digital Investigation*, vol. 46, p. 301611, 2023.

[66] T. Katsantas, Y. Thomas, C. Karapapas, and G. Xylomenos, "Enhancing ipfs privacy through triple hashing," in *2024 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2024, pp. 1–6.

[67] E. Daniel and F. Tschorsch, "Privacy-enhanced content discovery for bitswap," in *2023 IFIP Networking Conference (IFIP Networking)*, 2023, pp. 1–9.

[68] F. Baldimtsi, K. K. Chalkias, Y. Ji, J. Lindstrøm, D. Maram, B. Riva, A. Roy, M. Sedaghat, and J. Wang, "zklogin: Privacy-preserving blockchain authentication with existing credentials," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 3182–3196. [Online]. Available: https://doi.org/10.1145/3658644.3690356

[69] G. Jaferian, D. Ramezani, and M. G. Wagner, "Blockchain potentials for the game industry: A review," *Games and Culture*, p. 15554120231222578, 2024.

[70] D. Das, P. Bose, N. Ruaro, C. Kruegel, and G. Vigna, "Understanding security issues in the nft ecosystem," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 667–681.

[71] C. Karapapas, I. Pittaras, and G. C. Polyzos, "Fully Decentralized Trading Games with Evolvable Characters using NFTs and IPFS," in *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–2.

[72] C. Karapapas, G. Syros, I. Pittaras, and G. C. Polyzos, "Decentralized nft-based evolvable games," in *2022 4th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE, 2022, pp. 67–74.

[73] A. Shamir, "How to Share a Secret," *Commun. ACM*, Nov. 1979.

[74] D. Labs, "CryptoKitties Whitepaper," in *CryptoKitties: Collectible and Breedable Cats Empowered by Blockchain Technology*, 2018, pp. 4–9. [Online]. Available: https://drive.google.com/file/d/1soo-eAaJHzhw_XhFGMJp3VNcQoM43byS/view

[75] I. Pittaras, N. Fotiou, V. A. Siris, and G. C. Polyzos, "Beacons and Blockchains in the Mobile Gaming Ecosystem: A Feasibility Analysis," *Sensors*, vol. 21, no. 3, 2021.

[76] T. Min, H. Wang, Y. Guo, and W. Cai, "Blockchain games: A survey," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–8.

[77] W. Rehman, H. e. Zainab, J. Imran, and N. Z. Bawany, "NFTs: Applications and Challenges," in *2021 22nd International Arab Conference on Information Technology (ACIT)*, 2021, pp. 1–7.

[78] A. Fowler and J. Pirker, "Tokenfication-The potential of non-fungible tokens (NFT) for game development," in *Extended Abstracts of the 2021 Annual Symposium on Computer-Human Interaction in Play*, 2021, pp. 152–157.

[79] K. B. Muthe, K. Sharma, and K. E. N. Sri, "A Blockchain Based Decentralized Computing And NFT Infrastructure For Game Networks," in *2020 Second International Conference on Blockchain Computing and Applications (BCCA)*, 2020, pp. 73–77.

[80] C. Karapapas, G. Xylomenos, and G. C. Polyzos, "Enhancing ipfs bitswap," in *International Conference on Information and Communication Technology for Intelligent Systems*. Springer, 2024, pp. 189–199.

[81] D. Trautwein, "Nebula – A crawler for networks based on the libp2p DHT implementation," 7 2021. [Online]. Available: https://github.com/dennis-tra/nebula-crawler

[82] B. Confais, B. Parrein, J. Lacan, and F. Marques, "Characterization of the IPFS Public Network from DHT Requests," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems LV*. Springer, 2023, pp. 87–108.

[83] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A survey on digital twin: Definitions, characteristics, applications, and design implications," *IEEE access*, vol. 7, pp. 167 653–167 671, 2019.

[84] I. Pittaras, N. Fotiou, C. Karapapas, V. A. Siris, and G. C. Polyzos, "Secure, mass web of things actuation using smart contracts-based digital twins," in *2022 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2022, pp. 1–6.

[85] I. Pittaras and G. C. Polyzos, "Secure and efficient web of things digital twins using permissioned blockchains," in *2022 7th International Conference on Smart and Sustainable Technologies (SpliTech)*. IEEE, 2022, pp. 1–5.

[86] I. Pittaras, N. Fotiou, C. Karapapas, V. A. Siris, and G. C. Polyzos, "Secure smart contract-based digital twins for the internet of things," *Blockchain: Research and Applications*, vol. 5, no. 1, p. 100168, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S209672092300043X