HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

MASTER THESIS

Mr B.Sc.
Felix Hildebrandt

Designing Account and Token Architectures for Decentralized Social Economies on a Blockchain-based Chat Application

Mittweida, 2023

Faculty of Applied Computer Sciences and Biosciences

MASTER THESIS

Designing Account and Token Architectures for Decentralized Social Economies on a Blockchain-based Chat Application

Author:
Mr. B.Sc.

Felix Hildebrandt

Course of Studies:
Blockchain & Distributed Ledger Technologies (DLT)

Seminar Group:
BC20w1-M

First Examiner:
Prof. Dr.-Ing. Andreas Ittner

Second Examiner:
M.F.A Fabian Vogelsteller

Submission:
Mittweida, 27.10.2023

Evaluation:
Mittweida, 2023

Bibliographic Description:

Hildebrandt, Felix:
Designing Account and Token Architectures for Decentralized
Social Economies on a Blockchain-based Chat Application.
2023. - 9, 77, 18 p.

Mittweida, University of Applied Sciences,
Faculty of Applied Computer Sciences and Biosciences,
Master Thesis, 2023

Abstract:

Traditional user management on the Internet has historically required individuals to give up control over their identities. In contrast, decentralized solutions promise to empower users and foster decentralized interactions. Over the last few years, the development of decentralized accounts and tokens has significantly increased, aiming at broader user adoption and shared social economies.

This thesis delves into smart contract standards and social infrastructure for Ethereum-based blockchains to enable identity-based data exchange between abstracted blockchain accounts. In this regard, the standardization landscapes of account and social token developments were analyzed in-depth to form guidelines that allow users to retain complete control over their data and grant access selectively.

Based on the evaluations, a pioneering Solidity standard is presented, natively integrating consensual restrictive on-chain assets for abstracted blockchain accounts. Further, the architecture of a decentralized messaging service has been defined to outline how new token and account concepts can be intertwined with efficient and minimal data-sharing principles to ensure security and privacy, while merging traditional server environments with global ledgers.

This master’s thesis was supervised by LUKSO Blockchain GmbH

LUKSO

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ABI	*Application Binary Interface*. An interface document specifying how functions receive parameters and return values to allow programs to interact without having to exchange data beforehand.
API	*Application Programming Interface*. A set of rules that allows different software programs to interact with each other, facilitating the development and integration of various services.
DAO	*Decentralized Autonomous Organization*. An organization represented by rules encoded as smart contracts that are transparently controlled by organization members.
dApp	*Decentralized Application.* Program or service that operates on a globally decentralized network without being dependent on central data management.
DeFi	*Decentralized Finance.* A permissionless financial system built on blockchain technologies aiming to recreate and improve traditional financial systems without a central authority.
DID	*Decentralized Identifier*. A type of identifier created, owned, and controlled by the subject of the digital identity, anchored in decentralized networks.
ECDSA	*Elliptic Curve Digital Signature Algorithm*. A cryptographic algorithm to sign and verify messages widely used to secure transactions on blockchain platforms, including Ethereum.
ENS	*Ethereum Name Service*. A domain name service built on the Ethereum blockchain to translate human-readable names into blockchain addresses, simplifying the user experience.
EIP	*Ethereum Improvement Proposal*. An open-source document to propose protocol specifications, features, or API standards within the Ethereum community, to ensure consistency for development across EVM networks.
EOA	*Externally Owned Account*. The default user account on the Ethereum blockchain used to sign messages, execute network actions, and store assets.

ERC	*Ethereum Request for Comment*. An open-source standard of behavior, usually around smart contract interactions and conventions within the Ethereum community, to ensure consistency for development across EVM networks.
EVM	*Ethereum Virtual Machine*. A runtime environment for executing scripts on the Ethereum blockchain in order to create consensus on the network.
FT	*Fungible Token*. A type of cryptographic token that is interchangeable with other tokens of the same type, commonly used for cryptocurrencies.
GDPR	*General Data Protection Regulation*. A regulation in EU law on data protection and privacy for individuals within the European Union.
ID	*Identifier.* A unique sequence of numbers assigned to an object to efficiently distinguish and validate their uniqueness.
IDE	*Integrated Development Environment.* A software tool that provides comprehensive facilities to edit and debug code for software development.
JSON	*JavaScript Object Notation.* A lightweight data-interchange format that is commonly used to store easily readable, writable, and parsable information in a structured way.
KYC	*Know Your Customer.* A legal requirement that regulated entities must perform to identify and ascertain relevant information before doing business with them to prevent fraud and money laundering.
LSP	*LUKSO Standard Proposal.* An open-source standard of behavior, usually around smart contract interactions and conventions within the LUKSO community, to ensure consistency for development across EVM networks.
L2	*Layer 2.* A secondary network on top of an existing blockchain to deliver faster and cheaper transactions.
NFT	*Non-Fungible Token*. A type of cryptographic token on a blockchain that is unique and cannot be replaced with something else, often used to represent ownership of digital or physical assets.
NTT	*Non-Transferable Token*. A type of cryptographic token on a blockchain that can not be transferred upon initial allocation or creation.
QR	*Quick Response.* A type of matrix barcode that encodes associated information in a visual pattern that can be scanned with cameras, to let users access linked data.

RDF	*Resource Description Framework.* A standard and model for data exchange on the Internet, utilizing URIs to name relationships across multiple data types and facilitate merging.
SBT	*Soulbound Token*. A type of cryptographic token on a blockchain bound to an identity, imagined to work within social graphs and in community-oriented projects.
SMTP	*Simple Mail Transfer Protocol.* A protocol used to define message formats for transmitting email messages between computers across the Internet.
SSI	*Self-Sovereign Identity*. A digital data concept that allows individuals to own, control, and share their personal information without the oversight or intermediation of a centralized authority.
TBA	*Token Bound Account.* A blockchain based account that is attached to an NFT, so an asset can have its own wallet to own tokens and interact with other smart contracts.
URI	*Uniform Resource Identifier*. A string of characters that provides a simple and extensible way to locate a data asset.
VC	*Verifiable Credential*. A digital claim made by the issuer about a subject, which the holder can present anywhere without the verifier contacting the issuer.
XML	*Extensible Markup Language*. A descriptive notation that defines a set of rules for encoding documents in a human and machine-readable object format, primarily used to define and transmit data across the Internet.
ZK	*Zero-Knowledge.* A cryptographic, privacy-ensuring method by which one party can prove to another that a given statement is true, without conveying any information apart from the fact of it being true or false.

List of Tools

Writing and Figures

MS Office Word	Forms and Structures	F1, F2, F4
MS Office Excel	Curve Diagram Generation	F5
draw.io	Processes and Classification	F3, F6

Code Libraries

Remix IDE	Blockchain Token Deployment
React 18	TypeScript Chat Application Frontend
UP Browser Extension	Management of the Universal Profile
Ethers JS	Connection of App to the User
dm3 React	Frontend Chatting Component
dm3 Backend	Message Storage Container
dm3 Library	Direct Messaging Protocol
Bootstrap	Interface Design Library
ERC725 JS	Fetching Account Data

Code Implementation

Solidity	Programming Language (Digital Asset)
JavaScript	Programming Language (Application)
TypeScript	Programming Language (Application)

1 Introduction

Over the past decade, the evolution of decentralized systems and blockchain networks has been remarkable. Predominantly led by the financial industry, there is an aspiring shift towards leveraging this technology for social solutions and organizational applications. This thesis delves into the development of such new social data economies, focusing on two main aspects: the accounts and their data integration.

In the initial segment, the thesis will cover the current state of user management on the Internet, pinpointing its inherent flaws for social movements, interoperability, and legal compliance. This starting point presents the potential of decentralizing data exchange to address core architectural issues. Although the technology offers great opportunities, there are significant challenges regarding the convenience of user integration, especially compared to established Web 2.0 services. Therefore, barriers and the account standardization landscape are comprehensively analyzed to outline solutions. The crucial topics of this journey are the abstraction of the underpinning verification framework and the overall management of enriching accounts with programmable features.

In the second half, the thesis extends to data integration within decentralized societies, addressing identity-bound information and assets. The concepts and related infrastructure are thoroughly analyzed to evaluate opportunities and risks while underscoring the essential role of abstracted accounts in achieving shared social environments.

The combined knowledge of previous chapters is conducted into data handling guidelines for tokens and data processing, used to build out a restrictive asset standardization for abstracted accounts. Similarly, the architecture of a decentralized messaging service is defined to combine the industry's best practices for secure and privacy-preserving data exchange on a global ledger.

2 Traditional Web Service Management

The rise of decentralized data economies heralds a transformative change in how information is stored, shared, and monetized. Within this cosmos, accounts are an essential primitive, defining the secure anchor points for any blockchain interaction. The following chapter will analyze conventional account behavior to understand architectural challenges. This foundation provides rich context and benchmarks to evaluate decentralized models later on.

In a technical sense, the Internet is a global server network that uses protocols to transfer data between specific device addresses [1]. Users can transfer data to servers available for individual retrieval. When displaying plain page content, the data exchange through device addresses is sufficient. However, in the current era of online connectivity, defined as Web 2.0, most pages feature social interactions or act as services for big user bases. Individual providers, therefore, opt for handling user accounts for their customers. From a neutral standpoint, the goal is to offer people new tools to simplify or enjoy everyday life, making communities grow. In return, companies are given the appropriate marketing tools to develop infrastructures and gain profits. However, by making such platforms available, companies also access personal data from customers and create analytics [2] on user profiles to optimize value creation. Data nowadays is the leading business model and most important asset for web services.

Our online identities mainly consist of multiple user accounts that must be created for almost every platform. Upon registration or login, the user is granted access to the account's information on the operator's part. In this respect, the service provider is the sole administrator with complete control over the data management. Customers may also register by linking to existing accounts of larger account providers they collaborate with to increase convenience. The latter equally carries the risk of losing access to any account related to the provider if access is lost, attacks are perpetrated, the account is compromised, or the intermediate service is unavailable at some point.

2.1 Foreign Identity Handling

From the perspective of identity custody, multiple problems arise. Not only is one's data held externally, and users acquire the right to manage it, but linked providers can also monitor users' interactions related to their services at any time and create a business out of it. [3] The account data is managed in a centralized and concealed way. As service providers build specifically for their needs, they also have individual account architectures, meaning users can not easily take data to other services, fueling data duplication and management overhead.

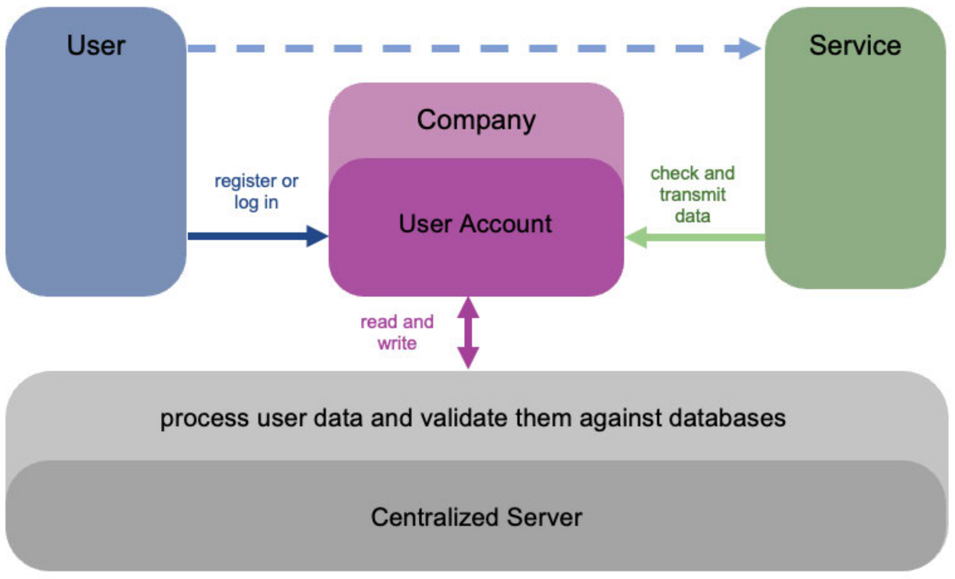


Figure 1: Simplified Web2 Login Model

The challenges of developing digital identities can be traced back to the outlined data transmission architecture in the figure above. The Internet protocols were designed for machines with unique device addresses but not for humans acting through them. No embedded systems verify individuals. There are only measurements to check access to devices or linked accounts. The lack of an identity protocol on the Internet is one of the leading causes of cybercrime and identity theft, causing enormous financial and personal damage. [4]

On top of the logins, there is also a considerable need for unique digital assets. In the real world, our signature represents our identity. However, the files we transmit online are copies of data. Typically, we scan verified documents to be able to use them digitally. With dozens of service logins, each associated with separate accounts, it is easy to lose track of ownership and whether the stored data is up-to-date. In this situation, users need to put lots of trust in platforms storing and securing their data. All this leads to the point where the dilemmas of security, privacy, and property cannot be addressed with the prevailing computer network architecture. [5]

2.2 Chatting and Social Media Patterns

Communication has always been the cornerstone of web technology. As protocols and platforms evolved, so have the ways of interaction between users. At the beginning of the Internet, e-mail was a fundamental way of online communication. Within Web 2.0, conversations expanded to forums and social networks. As embedded within the platforms, chatting allowed sharing experiences and emotions more fluidly in a public or private manner. It also enabled real-time reactions, knitting together global communities with shared interests and values. Chatting nowadays has blended with social media as even ordinary services support status messages, handles, video-share, or audio functionality. However, there is a set of points to criticize.

To begin with, chatting and social media platforms have unique features and user interfaces. While fragmentation aids in distributing weight and diversifying the landscape, it leads to interoperability issues as data is kept in silos, creating barriers in the architecture, protocols, and development processes. Users are unable to transfer or back up their chats to different platforms. If users lose access, their messages, memories, and digital history on that service are often lost forever. Even if users are friends on multiple services, they only remain with parts of their conversations. The non-existence of transfers of accounts or parts of their belongings hardens network effects, and migrating users to newer or safer platforms is a significant challenge. If most of an individual's social circle operates within a particular platform, the incentive to switch diminishes, even if there are valid concerns. This phenomenon ensures that specific structures, even if flawed, remain dominant as they are embedded in the infrastructure of user interactions.

As online communication becomes deeply ingrained in our daily lives, the security and privacy of these conversations gain the utmost importance, especially for data protection. Many chat platforms do not encrypt messages by default, leaving them vulnerable to unauthorized access. Furthermore, web services can regularly be found in data breaches and leaks [6], all raising valid concerns over the sanctity of personal space in the digital world. Even outages lead to worrying situations. Within the last few years, multiple instances have arisen where major platforms were unavailable, leaving millions unable to communicate and underscoring users' heavy reliance on platform-centered identity roots.

The behavioral restriction of social media platforms tied with identity management has become particularly evident with whistleblower reports from 2018 highlighting Facebook's Cambridge Analytica scandal [7]. Users who choose an account with a provider are also bound to their algorithms. Even if users only want to stay in touch with friends or maintain their public page, they can not direct how feeds are altered by the platforms' algorithms, potentially impacting relationships and public image. The operator's closed source algorithms quickly lead to perception manipulation. Digital systems and their underlying algorithms can prioritize and amplify radical or divisive content and connect those directly to someone's profile. [8] They frequently gravitate towards monopoly statuses to maximize advertising revenues fueled by disconnected networks. The reach-focused goals have tremendous implications for any campaign. Within Facebook's data breach, users were tracked for years, with their data, interactions, search queries, and visited places analyzed to serve targeted ads and influencing posts [9]. This bundled access over identities is threatening democratic processes. Services should be separated from identity management, allowing users to choose communication channels without losing their identity and reputation.

2.3 Legal Challenges for Platforms

The General Data Protection Regulation (GDPR) concluded that "anything that helps identify an individual, whether it relates to an individual's professional, private or public life" is considered a private data set and counts as an individual's property [10] , even if it resides on the company's server [11]. Accordingly, the digital identity rules refer to almost all information of such digital identity or account and its associated communication. However, even if users are given the right to view, manage, and delete the data collected about them transparently, companies can still process the data beforehand. How quickly companies can analyze data to their desired advantage is simply a matter of computing power. While the Data Protection Regulation restricts how this data can be obtained, companies can still obscure parts of data sets to exploit legal loopholes. In addition, innovations in data processing allow more and more information to be extracted from procured data, which thwarts the effectiveness of regulations [12].

As seen from the rapid development of terms and conditions over the last decade, companies are pursuing increasingly specific access rights to remain precisely at the limits of what is legally possible. Since the account exists directly on the platform, the user often has little choice. Agreeing to the new rules or closing his account, including giving up the built social life on this platform, cannot be withdrawn. Updates quickly become psychological overhead for their customers, as they do not own created values. Therefore, data sovereignty should be embedded into user rights by design.

As rules progress within legislative periods, companies must constantly adapt to new regulations. [13] Users will be given more rights to delete data or find out where and when companies collect it. However, implementing systems to verify these rules is a huge task and could lead to the debilitating restructuring of digital ecosystems. As user rights and corporate transparency increase, the need for follow-up and oversight will continue to grow. As the variety of attacks and enforced requirements are tremendously raising the maintenance cost of central data centers, it is almost impossible to safeguard users comprehensively at scale. The lack of coverage puts data at risk, especially in smaller businesses. The current security stems from a top-to-bottom approach, and users have to bear with the company’s standards. Ideally, security would be in the hands of the users and embedded by design.

2.4 Measurement of Interaction Barriers

In terms of traditional web services, it is foreseeable that enterprises, government entities, platforms, and devices will need more robust identity management to counteract highlighted traffic imbalances. Rethinking how data is stored and managed is of enormous importance for society. The following matrix systematically categorizes the challenges associated with traditional web services, evaluating issues related to identity handling, social interactions, and legal hurdles across various operational aspects.

	Identity Handling	**Social Interactions**	**Legal Challenges**
Lock-In	Account setup and duplicated data for every service in use	No portability for reputation, posts, and chats	Created value is owned property of the platform
Backend	User data extraction and private value optimization done by corporations	Behavioral restriction based on algorithms and feeds	Race conditions from companies while processing data
Maintenance	Account owned by centralized companies and only managed through user logins	Reliability issues, closed source data formats, and backup issues	Massive effort to convert or update systems with user rights
Security	Risks through intermediates or device and password logins	Extended tracking and risk of external data breaches or leaks	Top-to-bottom safety model from company to customer
Values	No personal ownership for digital goods and profiles	Commercialized relationship models	Inverted identity relationship model based on real-world
Protection	Mandatory to trust company for data custody, active preservation	Privacy concerns based on platforms individual features and terms	Companies applying to the minimum of protection rules, interfering business

Table 1: Traditional Obstacles of Web Services

3 Data Economy on Global Ledgers

Based on traditional web services analyzed in Chapter 2, it should become a principle that users retain complete control and consent over their data instead of letting companies hold their accounts and information. The following chapters will describe how cryptographic networks can counteract these central structures to build a fundamental understanding of their abstraction and barriers.

Web3, the third evolution of Internet technologies, addresses the need for data ownership with one of its technologies. Blockchain networks define a decentralized way of data exchange to create fair and equal relationships between users and services. Due to its architecture, it is possible to attach tangible value to assets instead of sending copies of data while the networks are served user-centered.

Public blockchains in this context can be described as a publicly shared network of computers operated by individuals worldwide, connecting to form a network and aggregating an ever-growing chain of data blocks. Within the creation of these new network types, security and data ownership are core properties. They work without the need for central actors. Complex cryptography makes this distributed ledger secure and the built chain virtually immutable. Added data blocks cannot be subsequently replaced, making them the digital equivalent of stone engravings. The manifestation of committed blocks acts as a security measurement, allowing users to own unique digital assets and ascribe value to them. In contrast to previous iterations of Internet technology that focused on big data and browser-based improvements without changing the server infrastructures, blockchains are a fundamental redesign of the Internet’s backbone.

Blockchains offer a significant advantage: they introduce the desired and requested base for an identity layer. All interactions and datasets added within blockchain constructs must be digitally signed and permitted by users. The signature and its manifestation within blocks make it possible to exchange and own data without creating duplicates. All those actions relate to an entity, not just a device connected to a service provider. Through the novel data economy, multiple parties can request and verify the same information about an individual without storing it again.

The situation defines an entirely new starting point for data protection rights. While blockchain information is public and should not be used to store private identity manners directly, the identity layer can be utilized in subnetworks and platforms that dock onto decentralized ledgers accordingly.

3.1 Decentralized Architectures

With built-in cryptography on blockchain networks, digital signatures can be used to verify accounts instead of usernames and passwords commonly used within Web 2.0. The cryptographic mechanism uses a key pair of private and public parts. All actions are linked to a public key and can be compared to a reference of a person or one of his instances. The private key represents a handwritten signature or password. Only the user decides when, where, and what they sign with it, automatically tying it to the public part.

While using cryptographic keys is central to ensuring security and privacy in digital communications, their role extends beyond safeguarding individual users and their credentials. In the broader ecosystem of interactions, relationships established through these keys are not owned by a single entity, reflecting a collective existence similar to human interactions in the real world. This fairness marks a significant milestone for the Internet. In traditional web architectures, companies still have sovereignty over connections, and users are merely given more rights to access specific data points. Blockchain technology is helping to enable an unprecedented level of independence.

The enormous effort of complying with the law and verifying the integrity of personal data in the centric model for companies and states can now be solved more efficiently and user-oriented. Adding advanced cryptographic methods such as zero-knowledge (ZK) proofs [14] on top, even private datasets could be verifiably attached to these accounts without revealing them directly to third-party services.

Blockchains also stand out with exceptional security and resilience. Users' sole access to their data reduces companies' system management and IT security costs. Instead of a central server, geographically distributed computers run the same network software in parallel and verify information independently. In correlation, there is a strong trend toward publishing the source code, as management depends entirely on the accepted consensus of the protocol and functions autonomously. All operators have the right to review their exact specifications and where their security lies underneath.

Like dealing with valuable assets in the real world, decentralized networks bring more responsibility for the user, their belongings, and the community operating the network. In conclusion, three main things need to be standardized and expanded in functionality to enable a seamless transition to blockchain technology: asset embedment, accounts, and scalability. The following table elaborates on their challenges and proposed solutions.

Area of Concern	**Description & Challenges**	**Solutions & Development**	**Tech Department**
Asset Embedment	How data can be verifiably incorporated into user-centric accounts	• SSI Claims (DID, VC) • Token (FT, NFT, SBT) • Issuing Protocols • Recovery Mechanisms	Data Economy
Blockchain Accounts	How blockchain accounts can become feature-rich centers for identities	• Smart Accounts • Account Abstraction • Permission Integration • Data Maintenance Flows	
Network Scalability	How distributed ledgers can handle costs and speed for extensive user bases	• Sidechains • Rollups and Bundling • Network Layering • Data Compression	Infrastructure

Table 2: Development Subjects for the Decentralized Economy

The thesis primarily focuses on the data economy landscape, addressing scalability to a lesser extent. The following chapters elaborate on the distinct issues of blockchain accounts and how data can be embedded into them.

3.2 Data Relationships

When establishing user behavior to verify and manage data globally, actors must use a public and decentralized registry, as every participant needs unrestricted read access for verifying credentials or tracking their history. The principle is necessary to be able to check data connections independently. Operators then decide to only put references or actual data onto the publicly viewable blockchain storage. The handling of such an account scheme is called self-sovereign identity (SSI), as the operator is firstly responsible for possessing his account and its data. Even though the main objective of SSI is to detach authentication processes from web platforms, identity facilitators, and certification agencies, the principle can be used as a fundament for a wide range of regular or self-issued user data.

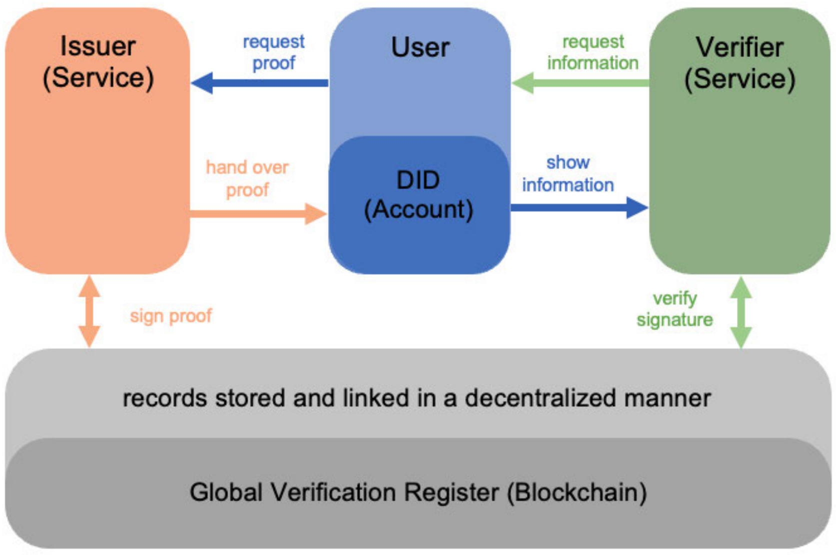


Figure 2: Decentralized Data Verification Model

There are three critical roles for verification and management as shown above: the issuer, the verifier, and the actual user. They all have a public address or key as a decentralized identifier (DID). As in the real world, users own an initially empty shell of their identities [15] and request a certificate from an issuer to attach to their accounts. After the request is fulfilled, the issuer can sign the certificate with its private key on the blockchain, referencing the user's account. Afterward, the certificate becomes a verifiable credential (VC): a signed proof in the global record [16]. The holder can continuously use services that need to check these credentials independently.

DIDs and VC can be applied flexibly, including private companies and service providers not necessarily connected to a blockchain. Nevertheless, compared to regular web infrastructure, the benefit is that users only need one account associated with a ledger and can use all the different services and their data issuing or requests. If the blockchain network is involved, credentials can even have a value connected to them. Through the user-centered methodology, network effects can be challenged without losing the roots of their confidential or public account data.

The concept can be extended to core elements in almost all areas of life where verification of arbitrary credentials, transcripts, or IDs is required. For e-commerce, verifying users before payment could be done using SSI integration. Banking services could use VCs and DIDs for claims to simplify bureaucracy and natively issue digital documents. Health documents could also be approved digitally instantly, or a history of illnesses regarding medical interventions could be displayed. Blockchains are unsuitable for data storage, especially for personal data, as the data remains immutable in the formed chain.

On the other hand, storage is enormously cost-intensive since the network has neither central computing nor specified storage units. Therefore, external storage solutions and the previously noted ZK proofs will become fundamental building blocks.

As the space evolves, a significant increase in projects is expected to explore different ways of implementing identity claims, pushing the boundaries of what is possible regarding privacy, security, and interoperability. Ultimately, the methodology only needs encryption keys, identities, and storage on both service ends. The equal flow will be used within the prototype of this thesis to manage the chat application's handles to establish connections.

It is essential to mention that SSI is not meant to restrict big data but rather to own and manage data flows and identity claims. Services will still measure behaviors or collect data. The primary benefit is the opportunity for built-in consent for owned private information. The user must sign the collection of verifiable data referenced on a ledger that only captures the current content. Reconfirmation for queries may be necessary as long-term storage of account information does not reside with the service provider anymore, significantly reducing duplicates.

3.3 Regular Accounts on the EVM

In distributed data sharing, the private key verifies the information. Within the blockchain, a basic framework is needed so cryptographic keys can become accounts and digital identities owning datasets later. On Ethereum's EVM, the world's most significant programmable blockchain protocol, a minimal user account is called an Externally Owned Account (EOA). Compared to the previously described public and private keys, this EOA also derives an address from the public part of the key, where the user can be reached and receive tokens. The address then serves as an identifier across the network. Creating an EOA happens offline, free of charge, and can be repeated as often as needed to create new accounts. Critically, however, there is only one account for each private key, meaning user participation relies on a single endpoint. EOAs do not consume any storage space in the network and are only referenced with data during their initial on-chain action. These actions, also called transactions as they modify the blockchain state, must be signed by the EOA before they can be executed and added within a block. [17]

As blockchain technology gained traction in the financial sector, the term wallet swiftly emerged as a standard descriptor for applications handling these EOAs. The initial configuration for wallets requires generating a series of words known as the seed phrase, which subsequently acts as a backup for the private key.

3.4 Integration of Contract Accounts

To create goods and economies, modern blockchains like the EVM feature a programmable application layer to deploy code. The code instances, named smart contracts or contract accounts, allow developers to deploy and execute custom logic on the blockchain. Similar to EOAs, they have their addresses upon instantiation. However, they come with a cost every time an EOA initiates the executive call due to their storage and computational needs. A contract account relies entirely on EOA calls as he can not execute transactions independently. As the blocks within the network, smart contracts are final in their functionality after integration. Only individual values of parameters can change subsequently. [18] Accordingly, smart contracts are widely used for relatively static assets. Since future accounts will be designed using the smart contract layer, they should retain flexibility and adaptability. This demand asks for the exploration of innovative solutions.

3.5 Economic Opportunities

As described in the relationship section, it is evident that integrating new solutions depends on leveraging the signatures of the underlying private keys. The challenge pertains to structuring the data infrastructure and flow dynamics to address scalability and functionality effectively.

For social constructs, Web3 and its drive to open source creates possibilities to mitigate network effects. Outlaid distributed ledger frameworks empower users, allowing them to select desired services and algorithms. More importantly, it will enable them to retain and transport their data, social interactions, and reputation across platforms, offering support for these networks. Because of the public visibility of source code within the field, developers can build unprecedented bridges.

Open standardization plays a significant role in more transparency, collaboration on code, efficiency in development, and much better implementation of new features and improvements, as many people assure code quality. Transparency fosters collaboration among developers and organizations working on blockchain networks, leading to the rapid development of innovative solutions.

At its core, blockchain technology can strengthen democratic principles, addressing shortcomings often seen in closed, platform-specific accounts of traditional web services. These foundational elements should inherently prioritize the protection of individuals and incorporate features widely known in today's digital world.

Despite its promise, the technology remains nascent, primarily geared towards financial applications. Creating new standards that will enable wider acceptance and general use is imperative.

4 Barriers to Blockchain Adoption

Managing identity on a large scale requires recovery, the assignment of rights, and references to public data points. Before diving into the prevalent account standardizations, navigating the challenges of blockchain accounts is essential.

4.1 Permissions and Backups

Regular accounts do not come with granular permission. Losing the wallet's seed phrase of an EOA means that stored assets become irretrievable, excluding further actions from the user account. This dilemma hinders user-friendly onboarding, known from regular web services, as individuals need extensive research on backup solutions.

On top of that, every private key is granted immediate administrative privileges. Users often split their assets across multiple EOAs to minimize risks, creating a significant problem when reconnecting values and maintaining social interactions. Relying solely on one unchanging passphrase to secure an entire account or identity is extremely risky and seen as an anti-pattern for adoption across the industry.

Additional multi-signature applications like Safe [19] offer upgraded security when managing valuable assets, diminishing the significance of keys. Thereby, multiple EOAs are needed to sign to execute an action on the network. However, they are primarily designed for groups rather than individuals. Accepting everyday interactions around a single identity with varying personas feels out of place for user accounts.

4.2 Paid Services and User Protection

As a community of node operators run blockchain networks, there are costs associated with their use each time new data is written to the block. Immediately after setting up a wallet, users must connect to a crypto exchange to acquire coins and execute operations on the network. The verification on crypto exchanges can take days, among other things, until users are in a position to act. Here, indirect or token payments could foster onboarding. Service providers could thus be able to pay for their customers' transactions and consider other compensation options.

Even after the users receive their currency to power their transactions, there are no safeguards for incorrect data entries. Mistakes, such as transfers to incorrect recipient addresses or wrongly typed amounts, are irreversible. On top of that, a significant concern is the lack of consent and spam protection. As everyone in the network can send transactions to any possible address, the openness is exploited for fraud. Due to transactions often being presented in a very technical way, inexperienced users might not always understand the actions they sign. This opportunity resulted in a whole wave of social-engineering wallet exploits, where marketplaces like Opensea [20] had to implement ways to hide assets within the front end to reduce their direct exposure. The topic of protection is crucial when digital goods or certificates are account-bound, i.e., they cannot be transferred after their initial receipt. Counterparts should always be able to approve actions affecting their account to establish a healthy data economy that complies with the presented legal frameworks.

4.3 The Gap of User Information

Based on the EOA structure, writing user-based information onto the account is unfeasible. The limitation stands in heavy contrast to modern web platforms facilitating shared user interactions. In this context, the account cannot directly hold claims or data as outlined in SSI. Without further concepts on top or beyond the protocol layer, an identity would be solely about the assets an account holds.

The limitation reflects well with the asset-focused development that ended up causing the NFT hype in 2021. During this time, a wide range of avatar crypto collections launched, and people used the images of acquired NFTs on various platforms to create and interact as fictional identities. While the approach heavily fostered community building [21], the underlying technology of assets could not meet people's desired integrational needs. The engagement decreased, and most people settled back to regular web services.

The most widespread project aiming to add more context to addresses is the Ethereum Name Service (ENS) [22] . The idea of ENS is similar to today's web domains. Instead of typing the raw address of a website's server, browsers handle human-readable domains that resolve to the correct address underneath. As long as the domain is owned, the operator can refer it to any webpage. ENS provides similar functionality for blockchain addresses, linking names to various crypto addresses as long as the domain is owned via a subscription model. On top of that, the name can house text records like references to social media platforms, email addresses, profile pictures, or custom text entries.

However, once access is lost, a domain can not be restored. People can only acquire their names if their old subscriptions run out. The approach of renewal is what complicates identity management. If the subscription expires, all historical actions on the network dissociate from the original name, erasing the collected reputation and references. Individuals could acquire expired domains and inherit a user's historical identity associated with the name. Reputation and properties are most important, especially in social collectives or creative environments. Therefore, it is critical to link data to the active account directly.

4.4 Embedment of Digital Goods

Since the asset economy is not integrated into the accounts of the protocol, it is impossible to make data queries from them. Its associated contract has to be invoked individually to verify the ownership of a particular token. Even if a project's platform is open source and deployed on the blockchain, many services predominantly rely on centralized backends to grasp account information. To comprehensively view an account's assets, services must go through complex and continuous retrieval operations using node machines or centralized block explorers and data services like Etherscan [23].

Smaller dApps can avoid external data points by statically integrating their associated or allowlisted addresses and calling them individually. However, platforms that need comprehensive asset information without overburdening their technical infrastructure or making demands on their users resort to calls to external providers. The reliance on external APIs intensifies the dependency on centralized entities and brings in vulnerabilities, including potential system outages or errors.

Not having an integrated data point makes reading broad user information tremendously tricky. Because an account can own assets without having them in their transaction history, the above steps also involve storing and maintaining the results, as services could not retrieve the data ad hoc. A service has to perform the following steps to fetch owned assets:

1. Get the account addresses of an identity.
2. Check the results of the extensive data analysis against the accounts.
3. Pass all owned assets back to the application interface.
4. Call every resulting contract address for their data.
5. Embed contract data into the service's front end.

The following diagram shows the complex flow of broad asset data fetches.

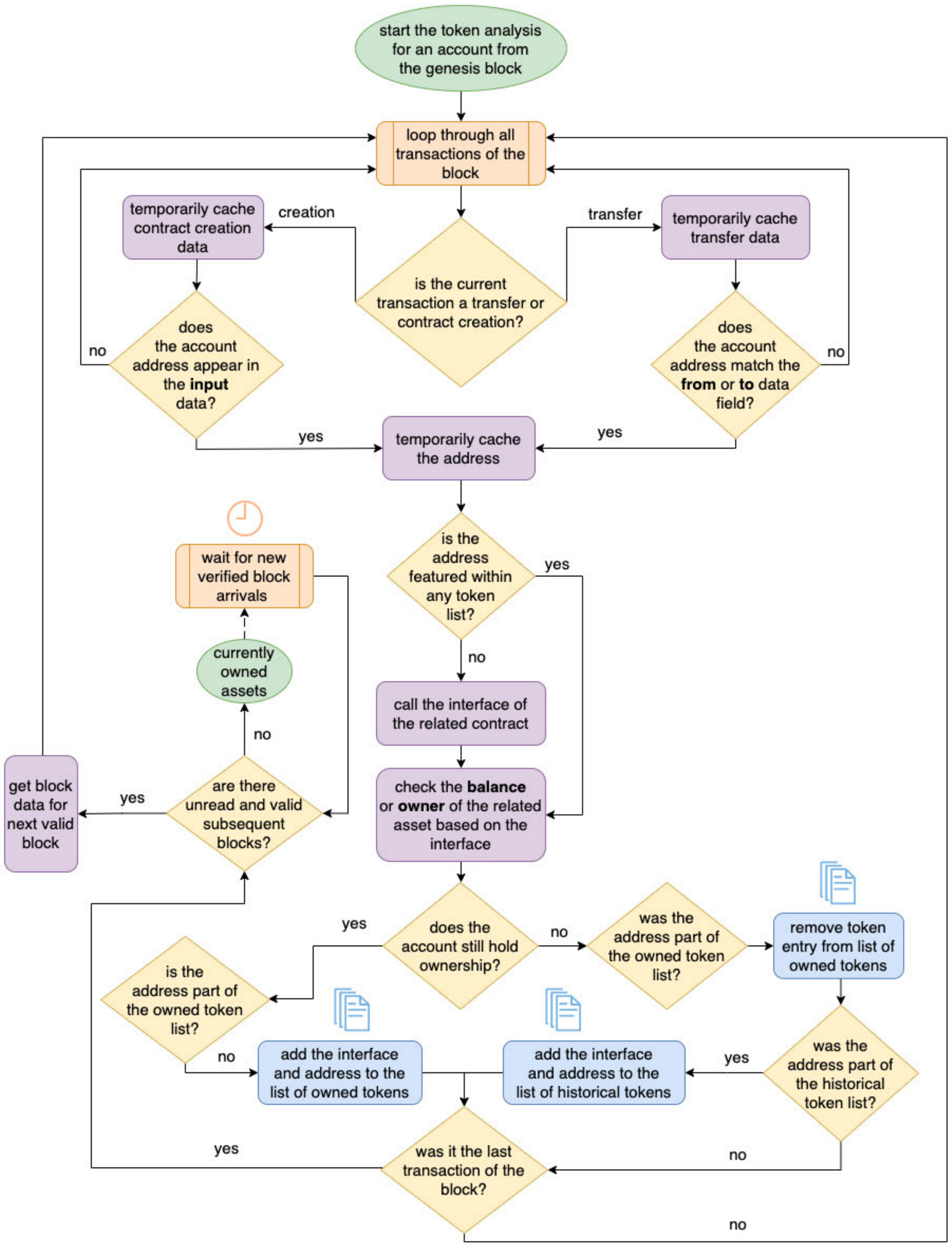

start the token analysis for an account from the genesis block
loop through all transactions of the block
temporarily cache contract creation data
creation
is the current transaction a transfer or contract creation?
transfer
temporarily cache transfer data
does the account address appear in the **input** data?
no
does the account address match the **from** or **to** data field?
no
yes
temporarily cache the address
yes
wait for new verified block arrivals
is the address featured within any token list?
yes
currently owned assets
no
call the interface of the related contract
no
get block data for next valid block
yes
are there unread and valid subsequent blocks?
check the **balance** or **owner** of the related asset based on the interface
yes
does the account still hold ownership?
no
was the address part of the owned token list?
remove token entry from list of owned tokens
is the address part of the owned token list?
no
add the interface and address to the list of owned tokens
add the interface and address to the list of historical tokens
yes
was the address part of the historical token list?
yes
was it the last transaction of the block?
no
no

Figure 3: Verifying Ownership of Unknown Assets

Further external checks must happen to gain shared ownership insights. Similar extra effort is needed for most NFTs to fetch their actual issuer, as they are often outside the transaction data during initialization when minted from the zero address. Therefore, accounts should seamlessly integrate into the on-chain asset economy to prevent operations from leaving the application layer, improving security using a chain of smart contract calls. An integration could also enable streamlined consent for asset transfers, as addressed in Chapter 4.1.

Another lack of embedment can be seen in the assets, relating to how the valuable data is attached to the smart contract. Typically, the token's data is not stored within the smart contract. Instead, it refers to an off-chain file often saved in a JSON format.

Full off-chain referencing comes with significant limitations. Notably, smart contracts cannot read these references, making them unsuitable for any other smart contract built to work with those datasets. Additionally, data references use URLs rather than direct cryptographic hash references, which ensures the data remains unaltered. Future asset standards should incorporate hash references in the contract storage, enabling consistent data verification, irrespective of where it is stored. The compound would allow updating the asset's storage solution while verifying the original data, making decentralized assets more sustainable.

4.5 Navigation of User Interaction

As it turns out, the accounts serve as a pivotal anchor for digital communication, much like it has always been the case in traditional Web 2.0. While transitioning into the decentralized domain, creating modern identities using regular EOAs becomes a challenge. The following table classifies the core data economy obstacles.

Category	Challenges	Affected Blockchain Layer
Permissions	No granular roles	Application
	Immediate administrative privileges	Protocol/Application
Payment	No indirect payment	Protocol/Application
Backups	Static backup phrase	Protocol/Application
Protection	No safeguards for on-chain actions	Application
	Lack of spam protection	Application
	No consent on the transaction	Application
Description	No way to attach account data	Application
	Sole asset-focused economy	Application
Integration	Heavily complex data analytics	Protocol/Application
	Assets are separated from the account	Protocol/Application
	Asset verification relies on storage	Application

Table 3: Analyzed Data Economy Obstacles

Addressing these challenges could be done entirely at the application layer. However, the associated costs and possible scalability must be considered. If something is exclusively on the application layer, it fosters custom implementations without direction while increasing complexity over contracts and off-chain server infrastructure. The topics of payment, integration, and permission should be embedded in the protocol long term to improve the network's base functionality. On top of that, specific accounts could be designed with the needs of the user base in mind.

Every transaction going through a smart contract is more expensive than sending it with a plain EOA. The more functionality has to be triggered, the pricier the execution. To keep future cost increases within an acceptable range, open-source development of smart contracts has to settle on generic standardizations. In the future, platforms could also redirect costs using indirect payment, as we know from the web today.

5 Abstraction of EVM Accounts

As blockchain accounts have faced various barriers for nearly a decade, multiple approaches have appeared to overcome them. Vitalik Buterin, one of the Ethereum founders, states that EOAs are a main constraint for mass adoption, and the abstraction of accounts remains one of the industry's most important topics [24]. This chapter covers proposed solutions and constraints for integrating later features and data interfaces.

Account abstraction is a continually evolving area in the Ethereum ecosystem aimed at making accounts more flexible, secure, and feature-rich. While not a single standard, it is a process that has been in active development since 2016, involving multiple Ethereum Improvement Proposals (EIPs) and Ethereum Request for Comments (ERCs), covering both network or contract-related standardizations. Each submission has features that contribute to a more robust account system.

In summary, account abstraction tries to standardize how the body of future blockchain accounts should be set up and what transaction flow they follow. The fundamental challenge of it is the verification of transactions by network operators. Traditional EOAs have the advantage that they allow for straightforward verification because a transaction is guaranteed valid if parameters like balances and computation prices are met and signed by the address key. In addition, EOAs offer a unique nonce value that can be tracked to determine the transaction order. However, this verification mechanism gets even more complex if the account can no longer be assigned to a single operator. Within the field of account abstraction, the goal is to make this verification flexible yet quickly executable via EVM code while maintaining safety standards across the existing setup.

5.1 Historical Standardizations

Introduced in early 2016, **EIP-86** was one of the first glimpses at account abstraction on Ethereum. The protocol and EOAs regularly use a static verification method, ECDSA, for every action of its whole network. The proposal made it possible to abstract away signature verification and nonce checking from EOAs by forwarding them to custom contracts. The most significant take was that these contracts could become accounts directly handling such operations, making transaction processing obsolete and paving the way for greater functionality to bundle actions, allowing for indirect payment, or upgrading the signature algorithm for security without being tied to the network. [25] After a lack of attention and demand, it stayed inactive while the concept was revisited in subsequent standardizations.

ERC-725, proposed in late 2017, provided an interface for smart contract-based accounts, allowing users greater control via exchangeable EOA keys while keeping the regular account unchanged. The initial proposal included a preview of a manager to regulate which key could do what and how claims could be attached. Later on, these functionalities split into separate standards: ERC-734 [26] and ERC-735 [27]. The vision was that smart accounts could include features like social recovery and greater security for assets, while abstraction does not have to be solidified within the protocol. The concept allows developers to work on feature-rich account ecosystems to sense possibilities and transaction verification in parallel. One of the proposal's highlights was the option to attach storage, providing a potential basis for identity and profile management on-chain. While the EOA controllers can rotate, these identities could have set backups on hardware wallets or instances for limited access. [28] This flexibility allows for improved onboarding, where a service provider initially creates a profile, but the user can take control afterward. The standard resulted in the ERC-725 Alliance [29] that officially developed and prototyped it. Due to the lack of features and structure at the time being, the majority of identity moved on to VC solutions. However, development beyond ERC-725 never stopped to build on social economies.

Midway through 2018, **ERC-1271** provided a standard method for verifying smart contract signatures. While signing could regularly be done on EOAs due to their private key, contract accounts could not interact with dApps utilizing these signatures. Usually, smart contracts had to rely on cumbersome and potentially less secure methods to validate signatures, often requiring off-chain verification or complicated on-chain logic. Limited signatures often resulted in multiple callbacks to the EOA, breaking the validation chain on the backend. The complexity especially presented a barrier to seamless interactions for multi-signature wallets when granting rights for asset transfers. The proposed improvement with the standard signature validation allowed smart contracts to authenticate signatures directly through a single function, simplifying the architecture of dApps and keeping the security measures within contract-to-contract interactions. The verification method has been particularly beneficial for multi-signature wallets and identity development. [30] The standard was the first within the account abstraction field to become final and greatly expanded contract communication for various projects.

EIP-2938 later sought to redefine user and contract interactions by introducing a new category of abstracted transactions in mid-2020. The proposal would allow contracts to initiate and cover the cost of transactions, much like EOAs, paving the way for more versatile and user-friendly account functionality. The standardization was the first abstraction concept focusing on transactions instead of the account itself. While it was a big step forward for accounts with relatively small protocol interventions, the standard remained inactive due to the unclear direction of accounts. [31] However, the concept continued to remain as a prototype idea that was incorporated into later standards. One of the key benefits of going with a new transaction type is that the protocol ensures they meet validity criteria before even sending it off to the network's transaction pool. In an ideal solution, abstraction must tackle both transactions and account specifications. Otherwise, transactions could spam the pool while waiting for a smart contract to validate them.

As a final milestone for the history of abstraction, **EIP-3074** proposed another protocol operation allowing users to delegate control of their EOA to smart contracts. The outlaid structure allowed regular accounts to behave like contracts without changing the EOA framework. The benefit of introducing a new mechanism, rather than creating a new transaction type, as in EIP-2938, is that it does not require any changes to existing wallets and dApps. However, both solutions have their negative points. The idea of EIP-3074 was similar to ERC-725, but instead of keeping EOA controllers, they could envoy rights to contracts themselves. These could then initiate transactions on behalf of the user. Here, transaction fees can be paid from an account different from the initiator. On top of that, it would come with greater flexibility, as sponsored transactions with custom tokens could be used as refunds. [32] Because the original EOA framework remains unchanged, the initial key retains immediate administrative privileges even if unused, potentially posing a safety risk. This issue was tackled by combining the proposal with the extension of EIP-5003 [33], which makes it possible to revoke any previous signing key of an EIP-3074-based account directly in the protocol. Both proposals have already become part of the core standards track of Ethereum and are likely to be used within new account changes.

5.2 Account Imbalances

Based on the history of standards, it is apparent that the issues surrounding signing and account features represent two distinct challenges for account abstraction. On the one hand, EOAs are the exclusive account entities able to sign. On the other hand, smart contracts bring all the functionality. So far, EOA controllers for smart contract accounts, as described by ERC-725 or used within multi-signature applications, still represent the state-of-the-art functionality but are often seen as second-class citizens within operational networks that mainly onboard users through EOAs. When updating the network's account system, the network's entry point should be consistent across all users, while backward compatibility with the previous EOA system is guaranteed. Such considerations are necessary for the risk of dividing user groups. In this regard, abstraction is also a matter of equality.

Due to EOAs serving dual roles of account management combined with signing, it is not easy to introduce new features without altering the protocol. The signing functionality should be separated from the account frame To tackle both problems modularly. Such a split allows for independent development of account features while keeping ownership rights from the cryptographic keys already widespread.

5.3 ERC-4337 Abstraction

Based on the smart contract limits outlined within account imbalances, the Ethereum network is on the verge of a pivotal transformation in account management, brought about by the standardization of ERC-4337. The proposal aims to revolutionize how accounts are treated in the network to spread the adoption of smart contract wallets. Hence, it is positioned as the definitive framework for realizing account abstraction and is already part of Ethereum's official standards track [34].

The concept outlines how the previously described separation of signer functionality and account can be integrated into the application and protocol layer. As the split acts as an ideal foundation, the proposal also includes features from previously stagnated standards such as token payment, bundling, or custom signature schemes.

On the technical side, ERC-4337 introduces a new architecture where every account becomes a smart contract deployed via a factory contract during its initial transaction. Unlike traditional Ethereum accounts, which derive their addresses from a single private key offline, these new abstract accounts have addresses generated at the application level using wallet-specific data. [34] This update paves the way for greater functionality and security.

The proposal replicates the behavior of existing EOAs to ensure backward compatibility. Instead of immediately generating an address on account creation, a signer key pair is handed out without any on-chain deployment. Afterward, the public key can be checked against a smart contract interface to obtain the on-chain account address. This call can be done immediately after the initial setup to receive funds at the address. The account, however, can be fully deployed later on whenever their first transaction is sent using this original key.

One of the most noteworthy features is the account flexibility once deployed. Since the signature scheme is no longer hard-coded into the account, transactions can carry multiple or custom types of signatures. The enhancement allows an account to become a native multi-signature wallet, where multiple keys can be added or removed while keeping the same address. The static anchor point mainly solves the hurdles of identity management and reputation. However, freedom of choice regarding signature schemes also enables multi-calls in applications, where various user interactions from different smart contracts can be executed within a single transaction. It is a powerful feature for saving network fees and facilitates the development of more feature-rich but efficient dApps. These new signatures will be validated on-chain, adding an extra layer of security. Consequently, signatures must also be validated on-chain as described within ERC-1271 rather than checking the ECDSA key offline.

To handle the enhanced functionalities, ERC-4337 introduces a new type of transaction. The so-called user operations have a separate, higher-level transaction pool within the network. The separate pool ensures that the new transaction type does not interfere with traditional transactions and the block building. The separation is particularly needed as the new signature schemes will bundle multiple user operations into a single transaction [35], which is sent to the main pool later. [36] The bundling reduces overall network traffic and costs, thereby enhancing scalability.

Within the separate network pool, a new entity of a bundler plays a crucial role. The bundler collects user operations, packages them into standard Ethereum transactions, and covers the transaction fees. The bundled transactions are sent to a single entry point smart contract for validation. The entry point extracts each user operation from the transaction, validates them separately by calling the abstracted user accounts, and checks their signature, nonce, and ownership. Upon success, the entry point executes the user operations and initiates refunding the bundler fees from the user accounts based on their used computation.

The bundler can be upgraded with a separate smart contract paymaster module for more versatile payment options. The module allows the bundler to specify a particular token that is allowed to be used for the refunding process. Before aggregating user operations into a single transaction, the bundler's system verifies that the user accounts have sufficient balances and funds the paymaster contract with the network's native coin. Instead of using its wallet to cover the transaction costs, the bundler points to the funded paymaster contract to handle the network fee for the transaction. In return, the paymaster contract demands its refunds as ERC-20 [37] compatible tokens previously allowlisted in the transaction metadata. Within the verification phase, the entry point checks the balance on the paymaster to ensure it has enough funds to complete the transaction. It then waits for the paymaster to finish its internal token validation process again. If everything is correct, the entry point regularly executes the user operations. It then triggers the paymaster to deduct the appropriate token amount from the user accounts to cover the computation costs incurred. This payment mechanism is designed so the paymaster cannot act in bad faith. The entry point contract controls the initiation of the payment or refund, ensuring a secure and trustworthy transaction process. Detailed views of the bundler and paymaster can be found in Appendix Part A.

ERC-4337 is designed to operate both at the application and protocol levels. Initially, off-chain bundlers or paymasters will be standalone servers facilitating transactions to nodes. In this case, every server maintains its separate pool of user operations. While the application layer approach functions correctly, it can come with multiple downsides. The largest providers would have the most extensive user operations to choose from, bundling more efficiently with lower refund fees and effectively marginalizing others. On top of that, setting up off-chain servers can be complex to maintain and carries the risk of centralization if a few prominent bundlers dominate the market. Therefore, these roles are imagined to be integrated into the protocol in later stages. The embedment would mean nodes becoming bundlers or paymasters while running the regular network. The protocol integration would help keep the network fair and allow for a bundling economy inside the network, improving speed and operation costs.

Alongside optimization, another reason for the protocol embedment is the unification of accounts on Ethereum, aiming that every account becomes abstracted. Without the change, they will only be used for a network subsection, remaining as second-class members in the ecosystem designed for EOAs. As ERC-4337 can emulate similar functionality more modularly, the standard eliminates the need for a separate account and address generation. With the protocol integration and simultaneous removal of EOAs, every single user on the network could be onboarded in a more user-friendly way, only requiring the plain cryptographic keys on the back. The freedom of choice makes it suitable to test and further specify the concept within live environments before statically embedding it into the EVM protocol when adoption is present.

6 Insights into Proxy Ecosystems

Like the development of account validation mechanisms, management solutions operating on top of them have also developed steeply. Where account upgrades focus on token payment and rotatable keys, smart contract ecosystems help to add functionality using interconnected standardizations. This chapter shows leading implementations of proxy contracts with a focus on the standards developed by LUKSO to find suitable solutions for building the thesis' asset restriction and give an outlook on the broad account evolution within web services.

As mentioned in the previous chapter, the concept of ERC-725 impacted its developer alliance, which has been researching rotatable EOAs for years. Instead of being heavily involved within ERC-4337, the focus was mainly on outlining what can go beyond and solve additional missing features. Over the years, Fabian Vogelsteller, creator of ERC-725, further fine-tuned modular components to enhance the base entity model. The primary standard was divided into two parts: One for executing program code or creating new smart contracts and the other one for defining an expandable storage list. This list exists as a parameter and can be filled with data elements like links or VCs, making the handling extremely flexible without changing the contract itself.

Many projects within the ERC-725 Alliance focused on core identity solutions for finance regarding uncollateralized loans, investments, KYC, or better custody. For instance, ERC-3643 outlaid a solution for permissioned issuance. The financial focus heavily aligns with the outcome described in Chapter 3.5. Nevertheless, Fabian Vogelsteller and the LUKSO project built a fully standardized ecosystem for social and creative applications, utilizing the vast potential for a drastically new economy beyond DeFi. The project and related proposals then became the main driving force for proxy smart accounts. They actively fostered the development of a developer library to interact with and decode information about such accounts.

6.1 Standardized LSP Ecosystem

As the history of abstraction has shown, aligning concepts and thoughts of a significant developer community is often complicated, resulting in roadblocks. The problem worsens when several standards are developed in parallel and built on each other. Within the LUKSO project, the standard development has been pursued outside the Ethereum communities ERCs. While having a unique term for smart contract proposals, LSPs, all standardizations are still fully compatible with the EVM and can be used across chains. The detachment was necessary to build several standardizations in parallel.

The standard of ERC-725, translated into LPS0, represents the foundation of a smart account [38]. Beyond others, it now also utilizes ERC-1271 signature validation for abstraction. By combining the account with the LSP1 Universal Receiver standard, this account can receive notifications about incoming and outgoing actions. To these on-chain events, developers can attach custom flows and behaviors. An example would be rejecting or approving certain digital goods or currency transactions, solving the issue of consent. If the recipient does not accept certain payments, they could be sent back to the original address. [39] Redirects or blocklists to reduce spam are also conceivable, as actions can be delegated.

The receiver standard can then be bundled with LSP5 Received Assets [40] and LSP12 Issued Assets [41], a storage framework to directly see owned or issued digital goods of the account. When the asset is entirely spent or sent, the received assets list is directly updated within the call. This feature is another milestone for the infrastructure of data economies in decentralized networks. As described in Chapter 4.4, the collective viewing of currencies was previously only possible for centralized services that scan the blockchain and display transactions in a readable format. Now, smart contracts can act completely decentralized by directly querying the addresses of accounts with a simple call of the ERC-725 library instead of running a node or connecting to complex and centralized off-chain systems.

Another critical point is the LSP6 Key Manager, which evolved from ERC-734, giving the EOAs different roles and rights. Until now, simple accounts could only cover full access and offered little security for managing content. Even ERC-4337 only provides a base concept for admin key rotation, leaving additional permission for developers. The key manager standardization is a suitable answer. By default, the proposal gradually offers nine permissions across signing, transfers, and ownership. When acting through the account, the key manager in front of it is called first, checking if the controller has the correct permissions. If access is granted, the transaction can be executed. [42] As everything is modular, key manager interfaces are exchangeable and can even be attached to other smart contracts to enable tokens with governance.

One of the standards also specifies the memory register for ERC-725 accounts, as the storage is practical only if one knows how to access and interpret it. The generic LSP2 JSON Schema ensures that metadata within the account is both readable and writable in an automated way [43]. It sets the guidelines for attaching information, such as asset ownership or file references. Another criticism analyzed within the blockchain adoption problems was that current goods on a blockchain can be verified, but often not their attached data. To this end, the attached data has a hash key indicating whether the original information is still unaltered for any storage key.

The storage concept can also add LSP3 Profile Metadata to the smart account. This enhancement turns a basic anonymous account into a Universal Profile that attaches publicly viewable information directly to the account, similar to how people are used to within the current social media landscape. [44] Since this metadata link is stored directly in the account's storage, the profile can be used for any action on the blockchain without creating a new profile for each application. Similarly, external services can write custom information to the storage as well.

The approach of the Universal Profiles can be cited as "public first, private second, "as the account can already have general information as in regular social media apps. Later, services will utilize the convenient structure underneath to dock private claims onto them or even stay fully anonymized. As David Silverman said in his talk: "If you build a project for private purposes first, it is locked in, and going public would not be an option. So being public and flexible is better for placing directions and power into developers' hands, not making decisions for them." [45]

A similar LSP4 Digital Asset Metadata standard brings equal features for exchangeable or unique assets on the blockchain, allowing for updatable metadata. Here, multiple sources of information, like media files, names, or descriptions, can be attached. [46] Even profiles of numerous artists can be linked directly to solve the current reputation problem of EOAs. Added profiles could then quickly enter the data economy of the digital good by utilizing royalties or reputation. If there is a link with the key manager, rights assignments for subsequent content modification can also be implemented for assets. The modularity of the ecosystem comes in handy, as the new asset standards, namely LSP7 Digital Asset [47] and LSP8 Identifiable Digital Asset [48], can all combine previous concepts to enhance regular FTs and NFTs. Among other things, these include notifications, the batch transfer of goods, links to the creators, and updatable datasets.

Figure 4 demonstrates how the entirety of the standardized features comes together to an account architecture and lifecycle for digital values.

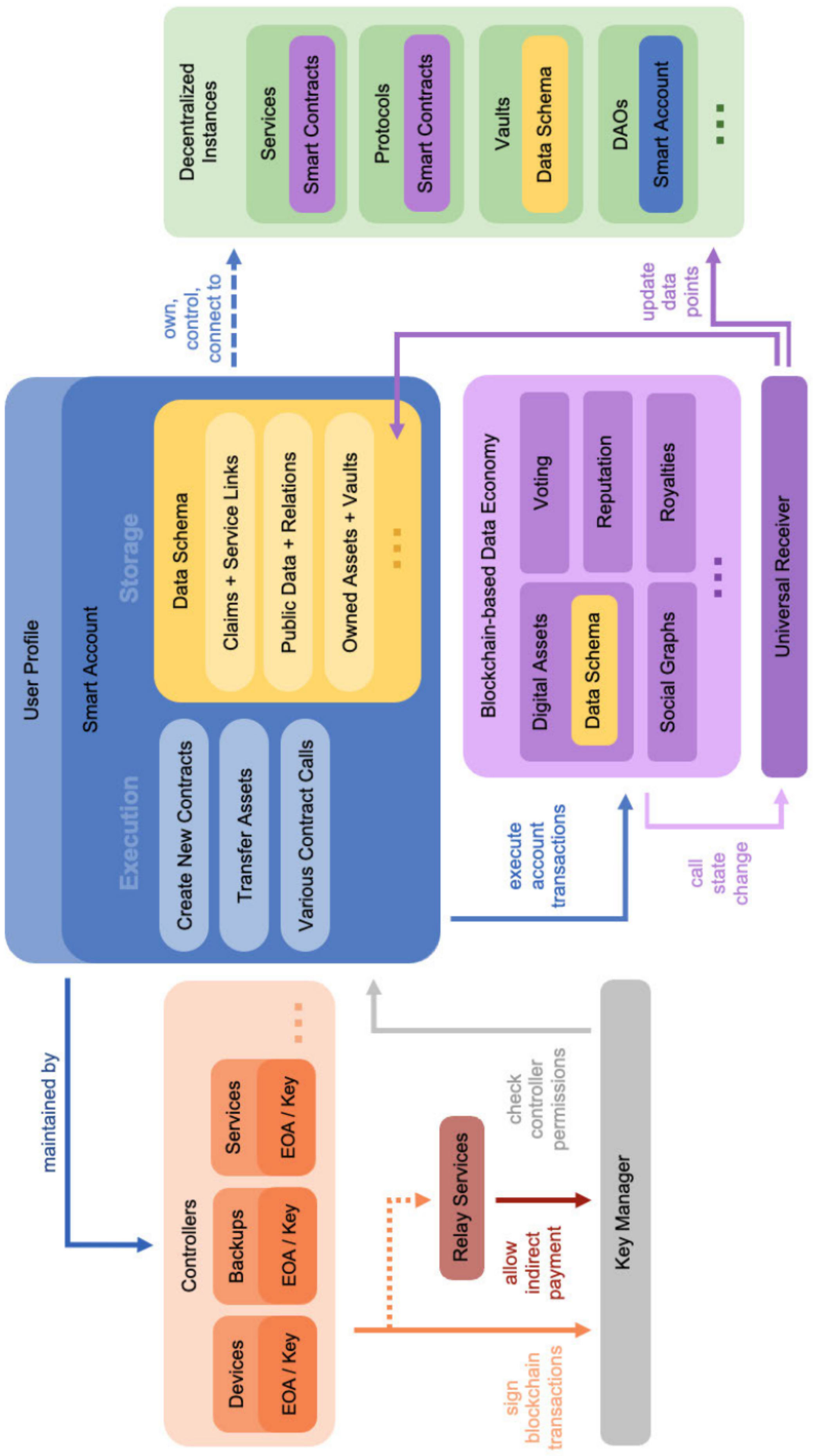


Figure 4: Abstract LSP Ecosystem Flow

In the future, a lot of data traffic will likely happen through such new smart accounts, adding a layer of complexity to asset navigation. As illustrated in Chapter 4.4, even current active blockchain users need help managing many digital goods. The issue was tackled as separate standards, LSP9 Vault [49] and LSP10 Received Vaults [50]. Here, accounts can create multiple different smart contract instances that behave like subfolders of a profile to organize data and assets. Not only can they be used to sort a person's possessions or provide additional security, but they also allow trusted applications to open or write into specific profile subfolders. While doing so, users can enjoy applications without signing each action individually.

The concept of shared management but self-sovereign ownership can be bundled with a separate standardization for relay services. In these, transactions are not sent directly from the account to the blockchain but are passed to an external service that executes them on behalf of the user. Bundlers and paymasters make a similar approach without their protocol integration. Here, LUKSO has brought its own RPC API to let services remotely pay for the transactions while remaining protected by cryptographic signatures. Users can then utilize indirect payments until fully integrated into the EVM. There will likely be a new market to allow users to participate in the network entirely without their crypto assets. These concepts are similar to mobile phone providers' monthly data volume. In return, users can use advertising, free-to-play concepts, or subscription models.

6.2 Lightweight Abstraction Approaches

While LUKSO is focused on developing interoperable standards for the core accounts of the future, numerous other initiatives focused on more minor, protocol or wallet-specific features regarding proxy execution. For instance, Argent X [51] is actively developing a 4337-compatible web wallet that includes proxy fraud detection with double signatures, definable spending limits for held tokens, and cookie-based keys for web pages where a wallet is created and can already sponsor their first transactions. After a certain amount of time or asset value, the user can be asked to overtake the created wallet. The adaptive design could be a significant breakthrough for in-browser sessions to manage users on the go, which may take their belongings into self-custody after some value generation. Games could profit from session keys to manage in-game items within a wallet. On top of that, it has a guardian functionality to create backup keys with a time gate to restore the account.

Another on-chain approach for profiles was laid out by the Lens Protocol in 2022, using an NFT as a social media profile. The protocol focuses on managing modular and decentralized social graphs where users can take their followers, reputation, and interactions across any app building. [52] While a single EOA still holds everything, the NFT can reference a profile manager smart contract to delegate social actions to a different wallet. The composite approach would mean a profile could safely be stored using browser wallets with less security that can only execute certain operations. However, every functionality is specifically for its social media protocol. As an indirect payment, a second proxy wallet, called a dispatcher, can be sponsored with the network's native coin to pay for every transaction. The intermediate wallet then acts as the signer for every transaction going through the Lens Hub contract. With the delegated signing privileges to the dispatcher, every transaction can be forwarded without additional signing on the user's front end. [53] A full interaction flow of the proxy wallet can be seen in Appendix Part B. Still, operators must acquire platform-independent values funds first. Compared to Universal Profiles, there would be a separate profile and token payments for each protocol instead of a global account solution. The concepts could even be combined, having a universal identity but distinct profiles for every central area of digital life.

6.3 Cross-Layer Network Evolution

Beyond the described proxy solutions, there has been a trend toward using subordinate or distinct networks. Argent X is building on StarkNet [54], which is focused on scalable and privacy-preserving transactions. Lens Protocol runs on Polygon [55] and has its own Momoka network [56] to map all social interactions. As outlined within the navigation of user interaction, smart contract operations are more expensive than regular network transactions. Here, speed and price are the main reasons for using L2 solutions [57] that dock onto Ethereum instead of using it directly. Costs are the reason why Ethereum is focusing on embedding bundling for user operations, as regular costs related to abstraction would not be eligible by default.

Besides the costs, another issue has already been raised in Chapter 4.3, but it becomes evident for LUKSO and why they are building on their network and extension. If abstracted contracts interact with regular EOA wallets, the counterpart will hinder convenience as it can not adapt to functionality. Yet, the problem of the second user class does not arise if each user is given an abstracted account directly upon entry. By instantiating a new network, the entry can be done directly via the abstracted LSP0 account to use all features. For existing EOA networks, the counterpart would be the protocol integration of account abstraction.

StarkNet, for instance, is a more scalable bundling network that uses a direct protocol-based abstraction model. While still able to dock onto Ethereum, they already got rid of EOAs, meaning that contracts are derived from the user's keys and created implicitly. After users compute a contract address from their keys, they can already receive funds that can be accessed after the related smart contract is deployed by using the received coins. Here, every developer can directly deploy various custom account features into the account's core while complying with the protocol standardization. [58] Users can later sign transactions using their private key, whose signature is checked against the generated smart contract address and its protocol-based nonce. [59] The network presents a semi-integrated abstraction less extensive than Ethereum's ideas.

Compared to LUKSO's Universal Profiles, developers working with StarkNet have the flexibility to embed features directly into the core and determine their structure and appearance. However, this also means that the user experience and security can heavily vary from wallet to wallet, and interoperability will not exist across the board. In the future, they will try to enrich their abstraction design, similar to the features of ERC-4337.

With such separate standalone networks for different economies in the blockchain field, general scalability for mainstream adoption can be securely raised. Additionally, early users can be subsidized through lower network participation. Subsidization and entry point convenience could not work out on an already settled network.

However, L2 solutions are not without drawbacks. Governed by multi-signatures, these subnetworks employ a relatively centralized way of constructing blocks, compromising asset security. They can also act as data silos that trap users' assets due to high transfer costs or bottlenecking during high-demand periods. Furthermore, only strict assets can be easily transferred, while social graphs and interactions remain largely unportable. [60] Where L2s provide an excellent solution for purpose-driven environments and first adoption, they do not seem fitting for generic accounts that act as the center of a user's decentralized identity. Here, abstraction has to be awaited on more robust, primary networks.

6.4 Curve of Smart Account Adoption

According to Vitalik Buterin's remarks at the 2022 Ethereum conference in Bogota, the integration of ERC-4337 could be a complex process, potentially taking more than half a decade to implement fully. [61] It has yet to be decided how exactly EOAs will be removed. Two solutions would be modifying the transaction, a concept leaning on EIP-2938, or the account ruleset outlined in EIP-3074 and EIP-5003.

The entry point contract of ERC-4337 was finally audited and deployed on the Ethereum mainnet in March 2023, validating bundled user operations. However, the adoption of ERC-4337 is expected on subordinate L2 or unoccupied networks first, as described in the previous Chapter.

Broader account solutions have already been rising, using EOAs on their backend. Where plain account abstraction is the right direction for the blockchain space, it can not compete with contract systems built today, like LSPs. Even further, current solutions will seamlessly switch to pure signing keys, allowing optimization later on.

A similar combined approach can be applied to today's relay solutions for indirectly paying transactions. Such servers could switch to become bundlers or paymasters directly in a much more convenient way. Here, both network and application development act in a supportive manner by already rudimentary showing what would be possible to implement and simplify account management later on strategically.

Previous trends can be directly transferred to the already known table from Chapter 4.5 by the evaluation to see the delimitations more clearly. Table 5, which can be found below, shows how backend infrastructure like payment, backups, and scalability are separated from the individual data exchange problems directly facing the users. The bottom line, however, is that these run parallel to overcome data-economy problems analyzed in Chapter 4.

Category	Challenge	Standardized Solution
Payment	No indirect payment	Relay Service APIs, Account Abstraction (Mainly ERC-4337)
Backups	Static backup phrase	
Scalability	Static Signature Schemes	
Permissions	No granular roles	Smart Contract Ecosystems (Mainly LSPs), Asset & Security Features (EIPs / ERCs)
	Immediate administrative privileges	
Protection	No safeguards for on-chain actions	
	Lack of spam protection	
	No consent on the transaction	
Description	No way to attach account data	
	Sole asset-focused economy	
Integration	Heavily complex data analytics	
	Assets are separated from the account	
	Asset verification relies on storage	

Table 4: Leading Decentralized Data Economical Solutions

The integration graph shown in Figure 5 below elaborates on the path taken by different account systems. Currently, EVM users start with a pure protocol and key-based account with low costs and functionality. Over time, these accounts can be enriched with additional features via smart contracts, elevating their mainstream functionality while moving to the application layer. These smart contract-based accounts can be further optimized and integrated with transaction bundlers, reducing transaction costs to enhance scalability. The goal is to effortlessly manage all activities involving accounts directly within the protocol while maintaining previously acquired functionality. The unification would result in a fair and decentralized network with unified user onboarding.

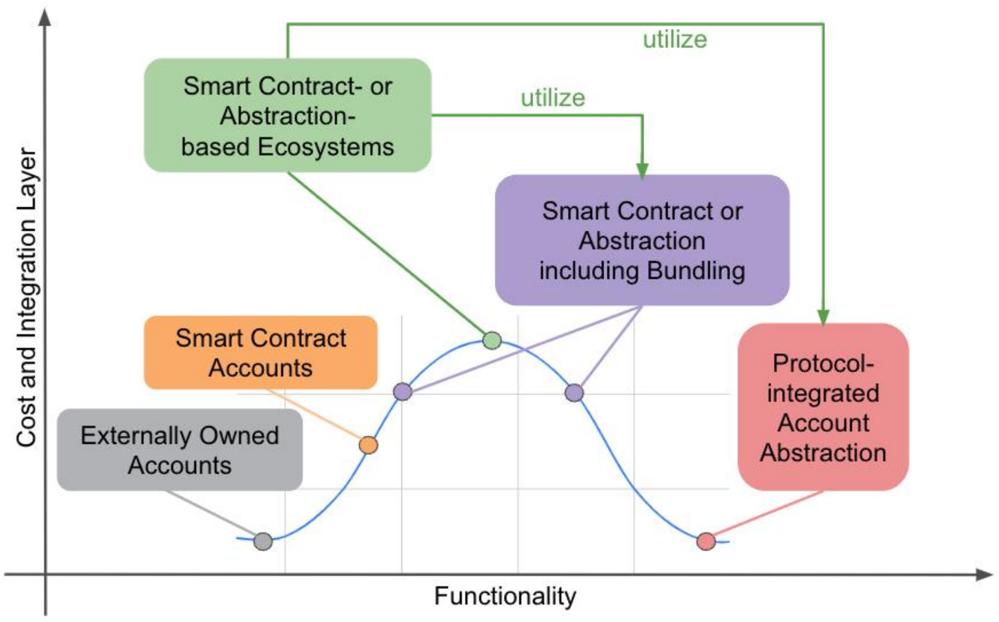

utilize
utilize
Smart Contract- or Abstraction-based Ecosystems
Smart Contract or Abstraction including Bundling
Smart Contract Accounts
Externally Owned Accounts
Protocol-integrated Account Abstraction
Cost and Integration Layer
Functionality

Figure 5: Evolution of Abstraction Levels and Integration Layers

As an overall picture, account abstraction standardizes the body of future blockchain accounts and their validation principles. At the same time, extended user schemes will be created on top, utilizing the gained possibilities through extensions or dApps. Several prototype wallets have already attempted to integrate rotatable keys, bundling, and token payments into extensions, focusing mainly on the existing DeFi world. More interesting, however, will be upcoming social environments utilizing functionality for profiles and reputation, which will be discussed in the following chapter.

7 Decentralized Societies

As the main driver of this thesis, social blockchain-based interactions mainly surfaced through initiatives like CryptoPunks [62] or Bored Ape Yacht Club [63], which indirectly cultivated the idea of decentralized societies. Within communities, identity and reputation are fostered based on on-chain assets. This chapter will introduce blockchain socials, summarizing the industry-leading platforms, their architectures, and dynamics before deep-diving into the underlying tokens.

In a broad view, decentralized societies are represented through memberships, commitments, and credentials, all anchored within a blockchain account. This data economy forms a concept of dynamic souls that offer a rich, multifaceted representation of an individual’s affiliations, achievements, and values. [64]

7.1 Social Integration Dynamics

In 2017, blockchain initiatives heavily began to delve into the concept of a blockchain-based identity for engaging in communities, yet these efforts were limited to token ownership and lacked integration with a decentralized social framework. Where Decentralized Autonomous Organizations (DAOs) facilitated governance, proposals, and token transactions on-chain, community engagement largely remained on Web2 platforms like Discord, X, Reddit, and Telegram. The shift lead to a disjointed experience with scattered profiles and inconsistent reputation management.

The asset-based blockchain identity, often based on Web 2.0 associations, leads to problems in confirming the authenticity and value of digital assets, as it is not natively anchored in the chain. For example, on-chain assets, like those based on the ERC-721 standardization [65], do not adequately link their creators, with many NFTs originating from a zero address. Therefore, the topic of decentralized societies is seeking a more resilient connection between the relationships of accounts and their interactions. The goal is to create a blockchain environment where identities and authenticities are inherently connected, allowing the blockchain to directly confirm individual identities and the originality of digital assets, using abstracted and multi-functional accounts.

7.2 Decentralizing Social Media

As analyzed in Chapter 2.2 and beyond, decentralized societies can be seen as the solutions to patch present Web 2.0 issues of data security, privacy, and fragmented user experiences with a more open, user-centered, and secure interaction landscape. Due to self-sovereign account management, users can choose platforms and biases individually, fostering environments not governed by stringent algorithms.

This freedom of choice encourages the development of portable and open social graphs, thereby promoting a culture of diversity and inclusivity. Interoperability ensures that users have the autonomy to transfer or back up their chats seamlessly across diverse platforms, effectively preserving their digital histories and memories. Moreover, it diminishes the network effects that have kept users tethered to flawed platforms, empowering them to migrate to newer, safer platforms without losing their identities and the essence of their digital interactions.

A cardinal benefit Web3 offers is the strengthened security protocols, prioritizing end-to-end encryption, significantly reducing the chances of unauthorized access and data breaches that have become commonplace concerns today. Establishing stringent data protection norms restores faith in digital conversations, ensuring that personal spaces in the digital realm are sacrosanct and protected from unwarranted intrusions.

Where abstraction separates the signing functionality from the account, decentralized societies foster the separation between service platforms and their accounts. Users gain complete control over their profiles, choosing communication channels. This separation restores the integrity of democratic processes, mitigating risks associated with data misuse and biased content amplification.

7.3 Current Social Architectures

Where centralization often sidelines user needs and sacrifices individual experience for corporate gain, the first pioneering open media infrastructures are already available. Two flagships building on decentralized ledger technology are the Lens and Farcaster protocols.

Lens manages modular and decentralized social graphs based on an NFT collection to map user profiles. Everyone can acquire a profile asset to customize an identity. While the profile's metadata and interactions are strictly defined, external apps can access the data but interpret it differently to create unique feeds or views. The protocol and its interactions are open-source smart contract standardizations. A central hub is responsible for minting profiles and keeping track of all user information, whereas custom modules are responsible for mechanisms regarding followerships or postings. Such dynamics can include time- and reputation-based restrictions or paywalls. The personal interactions are written into the profile's NFT storage and can be minted as a token. Due to the static NFT address, identity references can be maintained even if the related account gets updated. All user interactions are managed through a more scalable but separately governed subordinate network. [52]

Farcaster, on the other hand, is a more minimal open protocol that apps can dock onto. Users acquire unrestricted and unique Farcaster IDs from a smart contract registry on-chain. However, the ID can also be referenced to a specific governed name registry for better accessibility. All data, like user information, posts, messages, or comments, is stored on nodes running a decentralized and open subnet of nodes. The system is designed so that nodes within it build and hold delta graphs created from signed messages, retaining this data for up to a year to ensure network stability and data integrity. [66] Like Lens, social media apps can aggregate deltas, build feeds, and create archives to store older messages.

Both solutions integrated proxy signing for their EOA user base, meaning they can direct signing rights for the protocol to a different wallet or device. For Lens, the profile NFT can be stored safely within a hardware wallet, while interactions can be initiated throughout other extensions. Both protocols then walk through the chain of smart contracts for verification.

Regarding recovery and permissions, Farcaster comes with a handy recovery address pointing to the Farcaster ID. This EOA can trigger the ownership transfer of the handle after a week without intervention. On the other hand, Lens comes with permissions to restrict protocol-based operations of wallets with signing rights, which limits the risk of false interactions [53] but does not increase the security of the wallets overall.

Both applications are built on the EVM but are limited to the current account model. Here, identity data only exists on a separate network or within an asset that might get sold from an account. Here, a user becomes a frame of what he holds or which registry he signed up for. In the future, such media data could be directly embedded in abstracted ecosystems, further simplifying the protocols and giving the account more social presence.

8 Bound Token Economy

While account features and public identities have evolved, more user coordination and interactions are needed. All sorts of entity-related properties should be able to be shifted into the smart contract landscape, then acting as the foundation of decentralized social relationships. This chapter will thoroughly cover bound assets' goals, dangers, and standardizations to develop guidelines and classifications for a generic restriction standard.

Concerning binding information to a persona, non-transferable Soulbound [67] Tokens (SBTs) arose to build more robust decentralized social structures. Once issued, they belong to a specific account and cannot typically be transferred or sold to a new account, making them non-financial rewards with personal value. They could imitate special, inalienable certificates, achievements, proofs of presence, social currencies and bonds, or interactions and reputations of a fictitious personality. [64] In other words, precisely what constitutes identity in the first place. Beyond the transfer limitation, SBTs are imagined to come with novel attributes, including mechanisms for social or communal recovery or exclusive issuing. One example is the study group's completion or validation of authors that worked on a specific paper- each confirming others to prove certain skills [68].

SBTs are imagined to work well with non-financial rewards, achievements, and reputation. While social applications hand out a vision, SBTs could bring a new similar wave of adoption for Decentralized Autonomous Organizations (DAOs) on the chain. As outlined before, communities are currently only managing votes or assets within smart contracts but fall back to several other Web 2.0 platforms for interacting with each other. On top of that, NFTs with monetary value encourage people to buy into communities instead of showing genuine interest and involvement. Here, SBTs could become the new tool to manifest interactions within an organization directly on a chain.

8.1 Requirements for Bound Assets

As they are on-chain assets, it must be ensured that there is a valid token and social structure. If bound goods had to be reissued every time a key was lost or updated, this would result in immense additional expense. The complexity of recovery or carryover for locked assets is also why current proof of attendance issuers, POAP [69] being the most prominent representative, opt for transferable certificates, although the sale is optional. SBTs create a system where attestations are accurate and verified, fostering a culture where trust is not just expected but guaranteed.

A binding has far-reaching implications in fields such as academia, where attestations grounded on SBTs could revolutionize the credential verification processes, making them more secure and reliable. SBTs can offer a framework where voting, open-source contributions, and attestations are grounded on verifiable and non-transferable identities.

For governance, they can ensure that tokens are linked to indisputable identities, promoting transparency and accountability. Governance tokens are often available for purchase, a practice that predominantly enables wealthy individuals to accumulate voting power or buy into communities. Their value and related asset speculation causes holders to be less willing to vote. The market connection fostered a prevailing trend of plutocratic governance systems, where the affluent disproportionately influence decision-making processes. Furthermore, these structures seldom incorporate safeguards for the protection of minority interests, thereby contradicting the principles of equality professed by many protocols. Using SBTs, participation or achievements could be needed to vote, get airdropped goods, or join certain groups. By holding individuals accountable through a system that is incorruptible and transparent, SBTs facilitate a governance model that is more fair, democratic, and participation-friendly. However, such structures also provide the opportunity to reflect negative aspects - for example, unfair behavior or the accumulation of debts.

In the open-source community, the identity of contributors is integral to fostering a collaborative and reliable ecosystem. SBTs facilitate the recognition and accreditation of contributors by tethering contributions to immutable identities. The affiliation ensures that contributors are recognized and rewarded for their input and promotes a culture of trust and mutual respect, which are the bedrock of open-source environments.

There should also be a consensus between the issuing and receiving parties. Otherwise, unintentionally credible identities would be at risk of spam without a way to remove it from their accounts. Selling entire accounts is also dangerous: objects would linger on the exact identity address even though the owner changes. SBTs only unfold their power in trustworthy social networks and should be linked to specific requirements. If SBTs were transferred, people would likely recognize it and outlaw the seller's reputation if he "sold its soul." In the long term, ways must be found to restrict bots from forming social circles and fake relationships.

For decentralized societies, however, there is a chicken-and-egg problem when using EOAs: Regular wallets need SBTs to become valid identities. However, these cannot be issued if one quickly loses access to them. On the other hand, EOAs need authentic social group networks for identity recovery, which can be implemented exclusively with SBTs. An EOA is just the sum of its particles: Since they cannot carry any information, the social space is only about what the address has acquired. Such accounts become hollow placeholders without the anchoring personality with depth and content.

The intertwining of SBTs and identities steers toward Name Bound Tokens or Account-bound Tokens with similar goals but derivates from the soul concept. However, a substance of personality needs to be available and integrated into the network, creating a holistic identity ecosystem.

Account abstraction, especially the Universal Profile ecosystem, solves the problem. It provides a static address as an identity, delivers natively upgradable security, and links the account to public user data that can generate reputation: It enables data to be stored and shared more securely from the outset. Thanks to the configurable rights interface, variable recovery methods are now conceivable. Here, the emphasis is no longer on what a user accumulates but on who they are and how they present themselves to the outside world. Universal Profiles solve the identity problem and make up the ideal soul framework.

Concepts for community-based backups are possible results. In this way, "lost souls" could be helped back on their feet without requiring action from the user. Vaults or rights could be cleverly combined to embed backups in social structures. SBTs can strengthen the dynamics of a DAO and blur the boundaries between Web 2.0 and Web3.

8.2 Advising Social Dangers

While promising, bound tokens introduce significant concerns and potential social dangers. Blockchains are inherently immutable and permanently available. With their data persistence, blockchains fundamentally violate the present GDPR outlined by the analysis in Chapter 2.3: the right of deletion on the custodial medium. This problem is deeply rooted in blockchain technology and has brought the issue of privacy to the forefront once again. In a realm where each crypto user's financial history is already immutably stored on platforms, social goods add another layer of complexity by tightly linking tokens to individual identities.

Unlike traditional Internet environments, where posts can be stored as screenshots and retained by third parties, the blockchain system allows users to prove the hash and details of the message indefinitely, creating immutable accountability that can be dangerous. Even if data is kept off the blockchain, transactions or connectors can be used for movement analysis. Due to the finality of assets, subjective social statements could become objective truths, potentially misleading individuals into accepting claims at face value and bypassing the evaluation process [70]. Later, the links could result in an unwanted negative reputation [71]. In the future, individuals may overshare information that could be viewed negatively, such as a low credit score, religious connections, political parties, etc.

Data restrictions could further result in exclusion with social or political consequences, potentially becoming a systematic problem for minorities. Here, the surrounding dynamics of personal data markets could become dangerous. While intended to offer enhanced choice and control over personal data, some users may feel compelled to give up their privacy, a compromise not demanded for financially secure groups of people. Such inequality could pose individual risks and challenge the integrity and equity of technologies dependent on personal data. [72] Thus, the analytics issues that arise in centralized content moderation could become even more complex in decentralized contexts, as monitoring and auditing information flows in a global space becomes much more difficult, if not impossible [73]. Especially in an open network, handling bound assets should be reconsidered carefully.

Regarding protection and obliviousness, efforts should be made to incorporate the right to dissociate and block non-consensual minting [74], as stated by Tim Daub (TD), creator of multiple bound token standards. It would be possible for the connection to the user account to be capped and for an association to be made only after mutual consent. However, this step presents several difficulties, especially in scenarios involving more than two people, which can lead to consensus blocking and power imbalances.

Another common argument against social tokens is their susceptibility to censorship, mainly because the issuer retains the power to control reissuing. Procedures should safeguard against unilateral control and manipulation. Shared control could be solved through protocols or previous permission managers in proxy accounts, extending equally to tokens.

A broader view of potential abuse should be included for standards to provide a roadmap for constructing a self-managed identity that balances control, access, and protection, among others. Such a nuanced approach is essential to balance anonymity and privacy that respects individual preferences while ensuring community safety.

There have been some outstanding standardization attempts in the past. Among others, the Web of Trust Initiative (WoT) established ten principles for self-sovereign identities [75]. To form a global guideline, a comprehensive matrix of key regulations was derived with identity rules such as the Laws of Identity [76] by Kim Cameron (KC), Identity and Access Architect at Microsoft, and the Principles of a Digital Being [77] by the Privacy Standardization Architect Natsuhiko Sakimura (NS). The following table shows the user-centered principles applied to token development.

Aspect	Principles	Derived Guidance
Sovereignty	Independent existence and control (WoT), complete control and consent (CL), accountable expressions (NS), and dissociation (TD)	Users must have control over their identity-bound asset, creating, managing, and expressing their digital being autonomously as long as the association is wanted.
Management	Access, transparency, and minimization regarding data (WoT), minimal disclosure for constrained use (CL), and fair data handling (NS)	Users must be able to control and limit access to their tokens, focusing on transparency while minimizing unnecessary data collection.
Agreements	Consent on all actions (WoT), control and consent for justifiable parties (KC), and consent during issuing (TD)	Users must express consent with a clear description of parties justified in using the identity-bound information.
Adaptation	Portability and interoperability (WoT), diversity of operators with consistent experience across contexts (KC), and adoption friendliness (NS)	The identity-bound asset must promote portability and interoperability across various platforms, ensuring a consistent experience.
Safety	Data protection (WoT) and universal benefit (NS)	Users' rights and data must be protected, fostering a system beneficial for all entities, including individuals, companies, and governments.
Lifecycle	Data persistence (WoT), directed identity data (KC), upholding the right not to be forgotten (NS), and dissociation as the end of data lifecycle (TD)	Long-lived asset data must be assured while respecting the right to be dissociated and facilitating the directed mechanisms that are secure and private.
Experience	Human integration (KC) and human-friendly design (NS)	Systems must be developed with a human-centric approach, catering to individual differences and focusing on inclusive, integrative solutions.

Table 5: Guiding Matrix for Identity-based Token Development

These principles advocate the notion of a user having independent existence and control over their identity, ensuring access to personal data, and advocating for transparency in systems and algorithms. They emphasize the importance of long-lived, transferable, interoperable identities and stress the need for user consent, data minimization, and protection of user rights. By focusing on decentralization in social tokens, a robust mechanism can be implemented to protect against undue influence and ensure a fair playing field for all actors.

8.3 Differentiating Tokens and Claims

Despite the inherent risks in the dynamically evolving digital landscape, tokens promote interactions and trust within decentralized environments. However, the security concerns lead to the question of why interactions should become tokens instead of SSI claims in the first place.

Both SBTs and VCs rely on the decentralization model from Chapter 3.2 but differ in their motivations and privacy considerations. SSIs inherently protect users' private and sensitive information and aim to decouple entity authentication from centralized registries, identity providers, and certificate authorities, thus enabling a decentralized approach to identity verification. In contrast, SBTs seek a blended approach to private data to encode social trust networks on the blockchain, creating provenance and reputation within decentralized entities such as DAOs. Therefore, the overarching goal of SBTs is to foster a personality-based token ecosystem that mirrors the trust relationships and affiliations in real-world networks, indispensably needing transparency. As with the current internet, the data economy is rarely understood as private property [64], and it is up to the user to decide how much to disclose. Some data has to be on-chain intentionally as a common dataset. Otherwise, centralized companies will create services for this, constraining gained freedoms. Universal Profiles are a prime example of keeping everything open without restrictions or governance but making it possible to operate globally under services.

However, it is essential to make a distinction regarding the storage of identity-related data. While VCs are the data whose signed reference is written to a decentralized ledger, SBTs only represent metadata to describe the asset. All concrete information that may be attached as a separate link is still stored off-chain, as with SSI, and allows for additional protection. Still, SBTs could unintentionally empower harmful intermediaries by exposing too much control and data on chains. Nevertheless, having an always-available token with metadata may be useful for preventing the provider from simply removing the associated record.

Regarding custody, SBTs can be seen as a crypto-native approach, as the specific token form requires the blockchain to unify value and ownership in a decentralized context. VCs do not necessarily need to be anchored in the blockchain layer. Storing credentials on the blockchain brings several benefits by enabling cost-effective and reliable data retrieval while coordinating every participant in a neutral environment. It uses blockchains to avoid the tradeoff between inconvenience and centralization [71], even if the initial cost exceeds those of signed claims.

SBTs envision a pluralistic network of values where personal information is programmable. It provides a flexible approach to granting access to underlying data, social credentials, affiliations, or government-issued documents. This method opens avenues for a bottom-up, decentralized coordination mechanism that can redefine the coordination of social groups and communities and overcome the limitations imposed by government-issued identity documents. [73]

In summary, while SSI and SBTs share a common goal of decentralizing identity and trust mechanisms, they follow different paths. SSI tends to take a more privacy-oriented approach that promotes individual control over personal data. In contrast, SBTs are heading toward a decentralized reputation system anchored in the blockchain. As the digital landscape evolves, harmonizing these approaches and mitigating their inherent challenges remains vital in fostering a trusted and user-centric digital identity ecosystem. However, as the previous chapters have shown, the SBT landscape needs a robust account system with good asset integration to securely and automatically query participant connections and facilitate approvals.

8.4 Standardization Landscape

In 2022, the topic of restrictive tokens experienced a considerable upswing, mainly as the idea was spread by Vitalik Buterin, referencing soul-binding from the gaming industry [67], where items are bound to characters after completing in-game challenges. Based on the initial idea, many projects have been proposed to solve barriers for several use cases. The biggest NFT marketplace OpenSea introduced such a locking as a possible feature for trading [78]. Another group addressed the private data issue by combining ZK proofs with an SBT [79]. The following chapter presents and evaluates a historically ordered list of SBT and restriction-related ERCs to weigh opportunity and use cases. All their vital elements of the Solidity code are listed in Appendix Part C. Until the 10th October 2023, 26 individual restriction-based standardizations were analyzed.

The two main topics of discussion are how to solve the transfer permissions and give consent. A common practice experienced early on was using regular asset standards with an empty or reverting transfer function. While complying with the general setup, digital assets would throw an error once they are called, breaking interfaces that cannot detect the lock before calling the function. Therefore, interfaces and life cycles must be established so marketplaces, recovery services, and social apps can represent tags or interactive buttons accordingly.

8.4.1 Fungible Proposals

In 2018, the first locking mechanism for fungible assets was introduced by **ERC-1132**, creating an extension for ERC-20 tokens to self-lock token amounts for a specific period. Even if it did not come with its interface, users can check locked and transferable token balances during a future timestamp or increase time or amounts on the fly. [80] The continued management becomes particularly useful for any governance or participation-specific fields where identities earn reputation.

Years later, **ERC-3643** introduced a multi-layered standardization to facilitate regulation for fungible securities. While SBTs were not introduced, the goal was built to ensure that assets remain compliant with various restrictions across jurisdictions while interacting with the DeFi world. However, this also suits the topic of socials, ensuring that certain types of tokens are not traded without adherence to guidelines. At its core, the standardization functions as a token permission system, ensuring that only approved and registered identities can perform specific actions. At the same time, allowlisted agents can impose token restrictions, updates, and backups or even freeze and retain tokens. In the context of SBTs, the authorization party could specify which addresses can mint, restore, or disassociate a token. The standard can also manifest licensing authorities in a DAO during restructures, imposing re-issuing or removals. [81] Because the identities exist independently of the wallet addresses, community leaders could implement token freezing and instance backups for members. The same applies to the tokens, meaning all token regulations can be updated and bound separately to allow updated restrictions or government rules.

Another attempt regarding bound assets was made with **ERC-6808** and ERC-6809, representing the equal backward compatible standardization for FTs and NFTs. The concept splits the responsibility between the holder and owner, authorizing how a second wallet can spend owned assets. This separation as an enhanced security approach works using time-bound restrictions and transfer approvals. [82] Here, both standardizations feature multi-layered concepts that allow for authorizing, adding, or removing multiple designated key owners and allow for several transfers for a specific period.

8.4.2 Non-Fungible Proposals

Where FTs always focus on time-locking or regulation, the first NFT-exclusive concept of restriction was a static, non-tradable token. The original idea stemmed from ERC-1238 in 2018, later converted into a multi-token standardization. However, **ERC-4671** continued the original name to represent personal possessions handed out by institutions. At its core, the proposal includes a minting and revoking mechanism by which the receiver has to approve the minting before it can be issued. Once minted, the asset is bound to a specific user account but can permanently be revoked by the owner. If the owner revokes the statement, all information and ownership remain unchanged, but the token will return an invalid status that can be publicly queried. Besides its transaction behavior, the standard also features an interface so marketplaces can differentiate and adjust their front ends based on the binding.

It also comes with a whole set of modular extensions regarding token-specific metadata, a global storage contract to keep track of multiple assets and their states, or address renewals, where users can pull over their tokens to a new address. [83]

After Vitalik Buterin outlined the original idea of SBTs from the gaming industry in 2022, **ERC-4973** implemented an account-bound token without any transfer functionality. The token or item can be consensually given out or taken to an account acting as a soul. If the user does not want to show the asset, it can be unequipped by the owner upon receipt. However, users can always re-equip it. [84] Here, the handout functionality, in particular, acts like an airdrop to the signed EOA address, equal to in-game behavior.

ERC-5058 also presents an extension for NFTs, where owners can grant approval to issuers and lock it up to a future block time. While locked, the transfer is prohibited until the issuers unlock the asset or its locking period is over. Without user actions, assets stay tradable as regular. The standardized interface also outlines the idea of bound NFTs, replica tokens of the original assets, and their metadata created during the locking process. The twin could then be handed out to rent the original asset until the expiration time is reached and the twin destroyed. [85] With this functionality, NFTs could be locked securely in hardware wallets while still being used on dApps, supporting token standardization. The security approach is similar to **ERC-6809 [86]**, where the holder and owner are separated, with the only difference being that it allows for multiple instead of singular owners at a time.

Due to various specifications and criticism of SBTs discussed in Chapter 8.2, the community tried to define multiple minimal proposals, removing the context discussion of how to bind and focusing on locking. Developers can then choose how to implement it strictly, only knowing that the transfer event has to revert whenever the asset is locked. **ERC-5192** became one of the first SBT-related standards that was finalized. With this minimal approach, the standardization is an extension and specifies two simple events for locking and unlocking an asset. [87] Marketplaces can then check the interface and lock status within a single view.

The second final minimized standardization **ERC-5484** focused explicitly on the consensus part of an SBT for regular NFTs. Before a soulbound token is issued, the issuer and the receiver must agree on who can burn the token. This authorization is permanent and cannot be altered post-issuance. The operator must then present the token metadata to the receiver and obtain the receiver’s signature to bind the asset. This method would force the data and removal settings to be available off-chain or by having an inheriting contract where the recipient can retrieve and sign the metadata and burn authorization before creating the individual SBT. Any changes to the metadata post-issuance are prohibited. To support a wide variety of use cases, both parties can choose to enable the disassociation from the owner, issuer, both, or none. [88]

More niche, **ERC-5753** outlined a minimal locking interface similar to ERC-5192. Within the proposal, the lock and unlock functions were split and defined as a single address to free the asset once locked. [89]

ERC-6147 proposed another step to enable new NFT use cases by providing expiration dates for specific guards of an asset. The specification allows the owner to retain holding rights while temporarily giving transfer functionality to a time-limited guard until full control returns. [90] What was initially meant to decrease the risks of losing the EOA key could give institutions or protocols the right to remove or stake personal accomplishments in an SBT context.

In addition to the minimal extensions already discussed, **ERC-6454** brings another approach to transferability determination. Here, the standard introduces an interface with a single function to check transferability from a specific issuer to a recipient. By including both addresses, the standard provides a more agile way of allowing transfers for backup reasons. It does not come with individual events to not restrict implementation or combination with other mechanisms. Where other standards utilize separate burn or revoke functions, ERC-6454 embeds them directly into the regular transfer function, as an initial assignment or final removal can be utilized by leaving the issuer or recipient as the zero address. [91]

ERC-6982 proposed an interface for minimal and efficient locking by minimizing on-chain events to optimize operation costs further. The standard features a default event stating the status for all future minted tokens whenever a token contract is initialized. In correlation, there is also a separate function to set the default locking value for all existing or future tokens later. Individual token IDs can also be modified. [92]

Last but not least, another minimal mechanism was added by **ERC-7066,** assigning a single, infinite unlocker to an individual token ID. The standard describes two main functions to allow retroactive locking for regular assets or restrictions during transfer. [93] Here, SBT use cases are seen as a glimpse, mainly reaching for wallets to lock their valuable assets to a hardware device or multi-signature contract. Other than ERC-5753, however, a single function is used to transfer and lock simultaneously.

8.4.3 Hybrid and Multi-Fungible Proposals

As mentioned in Chapter 8.4.2, the non-transferable tokens were initially proposed with **ERC-1238** and turned into a multi-type standard. Within the proposal, so-called badges or experience points were lifted off as fixed statements about someone's EOA. Those were later seen as primitive for SBTs, so the standard was further fine-tuned [94]. Although the tokens cannot be transferred, the idea was that they could be staked and potentially lost post-staking or even set to expire, with manifold use cases across reputation, achievements, or DAO integrity. The initial draft focused on NFTs only [95], while the later architecture was heavily inspired by ERC-1155 [96] and its capacity to manage multiple token types under a single contract. In addition, the standard makes it possible for each token ID to attach a custom data link for the badge or experience currency they gained. Besides letting the owner remove the badge at all times, the core part of this standardization is the authorization during the mint time [97], with a receiver interface [98] that returns an EOA-specific approval value for an EIP-712 signature [99]. Overall, the standardization became a broad mixture of features, aiming to enable extensions like storage links, expirable properties, or holder separation for any fungibility type. [100]

In 2020, **ERC-3525** introduced a semi-fungible, restricted token. While it also comes with regular token IDs like NFTs, the standardization adds slots and values. While the ID ensures that every token can be distinguished separately, a slot represents the property or characteristic differentiating tokens within the same ID category. On top, the value field distinguishes the number of tokens within a specific slot. A classic use-case example would be having voting tickets with different weights for a specific election. DAO members could acquire tickets of multiple weights, fungible within the same weight ranking. The included locking mechanism introduces approvals, so owners can allow operations for specific slots and entire IDs or attach custom metadata to them. [101] Related to SBTs, such approvals help restrict minting and manage social member trees.

As ERC-5058 did for NFTs, **ERC-5516** presented an interface to bind assets from multiple types to several owners. On top of that, the standardization uses a pending state for the binding directly on-chain. Before the minting, the single token is transferred to one or multiple recipients and enters a pending state where the received asset can be signed or rejected. Parties can agree individually, resulting in an on-chain event and the fixed binding to the signed address. If not, the token is shown as rejected but unaltered metadata. As it integrates the previously mentioned ERC-1155, the standard also allows the blend of fungible and non-fungible characteristics by supporting the transfer of multiple tokens in a single transaction. However, the proposal does not provide content-disassociation. [102]

ERC-5633 initiated another extension for tokens of different fungibility. By having a non-obligatory soulbound property, bound and non-bound assets could coexist within the same contract, as each token ID can be addressed individually. Once bound, all transfers except creation and removal will be denied. [103] One great use case would be games that generate rare or unique items of the same asset type that can never be sold while leaving the majority unaffected.

Quickly after, **ERC-5727** took the approach of slots from ERC-5325 and mixed it with previous SBT concepts from ERC-5192 and ERC-5484 to create a multi-tool close to real-world use cases. It is a semi-fungible, non-transferable token standard with removal permission, including separated issuer and verifier parties. Each token ID can be individually verified through an external party and even gain or lose credit, stating the importance or rarity of the accomplishment. Moreover, it also comes with a set of add-ons to approve statuses for shared governance, token expiration, and backups for SBTs. [104]

Another minimal for multi-token add-on was proposed on **ERC-6268**, outlining a minimal untransferable indicator. [105] As within ERC-5633, a locked parameter can be assigned for unique IDs. However, this time, the standardization also comes with individual events to lock or unlock one or multiple tokens, making queries for dApps more efficient.

8.4.4 Atypical Proposals

During the last two years of SBT development, even unconventional binding approaches have come to light. Because of the previous hurdles of having a strict and static private key for every EOA address and the chicken-and-egg problem outlined in Chapter 8.1, another idea was that soulbound items should relate to a name or property with rotatable keys, just as smart accounts have solved. The original idea of **ERC-5107** was to bind an NFT to an ENS name that acts as a universal anchor point on-chain but never became a concrete specification. [106] Here, splitting the entity from the account to be maintained externally was seen as the right amount of modularity until upcoming account changes face adoption. By doing so, the proposal aimed to inherit all security features from the name registry. [107]

After years of the badge idea originally proposed with ERC-1238, **ERC-5114** made another attempt at irrevocable soulbound badges and picked up the question of how an on-chain identity may look. As profile pictures were the main driver of the digital asset narrative, the proposal standardizes how non-transferable badges can be bound to NFTs. Here, unique pictures of on-chain characters could be able to bind particular traits, clothing, or accessories to their souls, while the attached content cannot be censored or altered later on. [108] This idea can become an even broader concept of creating marketplaces for hybrid SBTs that only get bound once redeemed to another asset. More importantly, the parent NFT can stay transferable while its properties are individually bound.

As DeFi remains the most common use-case of blockchain, SBTs were also seen as a significant opportunity to port over real-world use cases of banks. **ERC-5252** outlines the architecture to connect an account-bound token with the DeFi realm to enable reputation or even hand out uncollateralized loans. The standard describes a design pattern for account-bound finance. An investor's deposit is associated with a bound NFT and directed to a personalized finance contract, maintaining an investor entity across multiple wallets. This NFT can be minted and burnt depending on specific transfer and operator approvals. [109]

Speaking of connecting more real-world use cases, **ERC-6239** tries to embed the Resource Description Framework (RDF) from the World Wide Web Consortium into the metadata of tokens. Most tokens use regular fallback metadata for NFTs like ERC-712. The World Wide Web Consortium, also responsible for SSI development, developed a scheme to capture, store, and manage social metadata in a structured and interconnected manner. Included JSON and XML structures enable better creation of relationships between attributes and facilitate the integration and sharing of social data across various applications. Most significantly, the proposal aims to close the gap between the regular and decentralized areas of the Internet. The related contract implementation, therefore, houses different events and calls to create, update, remove, or access token data similar to regular database or API fetching in Web 2.0. [110]

ERC-5114 already mentions a radical shift for bound items attached to an NFT, **but ERC-6551** goes further by mixing SBTs and abstracted accounts. While an asset usually gets bound to accounts, the standard describes the opposite concept of how accounts can be bound to a single NFT, meaning a digital asset can have a wallet. The idea behind this mechanism is to allow an asset to own data and interact with other smart contracts. The Token-Bound Account (TBA) then has its signature using ERC-1271 to verify the transactions. From an architectural standpoint, each TBA is a minimal proxy account, ensuring a deterministic address within a registry. This account then delegates execution to the external business logic whenever a transaction comes in. [111] With the concept, the owner of the NFT can individually grow the asset's social interaction graph while the creators designed its functional capabilities beforehand.

ERC-6956 proposes a more abstract standard for digital twin NFTs as the interest in connecting blockchain and real-world assets grows. The proposal aims to bind physical and digital assets with NFTs while each asset is connected through a unique anchor. An oracle must then verify and attest this anchor, ensuring that control over the asset equals control over the NFT. Afterward, the anchor can be transferred or destroyed concerning its real world's lifecycle, creating an asset-to-asset binding. [112]

While most atypical proposals do not directly fit the SBT construct, they outline similar locking mechanisms. In a broader picture, they also help the development of restriction- and identity-based on-chain economies.

8.5 Evaluation of Token Standards

As seen through the standard analysis of SBT-related models, many proposals include features regarding the current lack of account abstraction. However, as the broad adoption of smart accounts has yet to be present, security-related topics often get integrated into tokens, hindering proposals to focus on the core feature. On top of that, while locking mechanisms increase security, they can also lead to more overhead by not losing the secondary seed phrase.

Making SBTs an extension to regular tokens benefits the already-built adoption. However, almost all gathered proposals focus on NFTs. As restriction can be seen as the foundation for many different use cases once personas can act through secure and static smart accounts, the field of SBTs can be expected to grow much more comprehensive than non-fungible assets. Here, regular standards often lack the farsightedness to open them up for increased possibilities regarding issuer, owner, and combinability. The following figure shows the complete survey of analyzed asset standards.

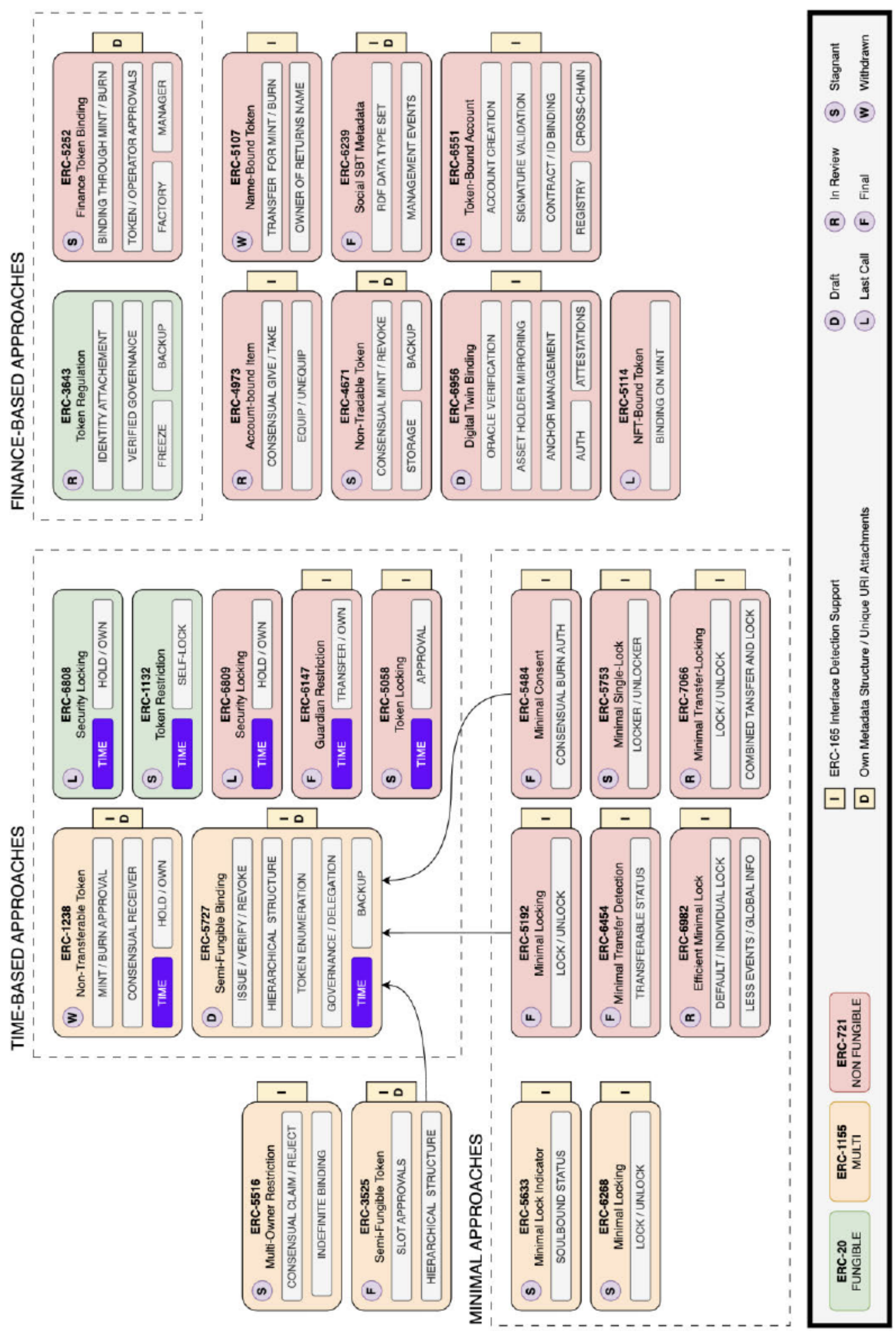


Figure 6: Classification of Restriction Standards

Another primitive often seen during the standards analysis is the variety of add-ons or extensions that seem minimal in the first place. However, on closer inspection, these only offer superficial capacities. These should be independent standardizations referencing their base concept to foster organized and individual development. As seen in the diagram, many standards are in a stagnant or year-long review state. The difficulty of progress could correlate to the opacity of features, as 3 out of 8 minimal standardizations could be finalized within a few months. Compared to all other approaches, only 16.7% of ERCs have the status “finalized.”

The tendency to tokenize assets with features has sparked an essential dialogue about utility and necessity. While tokens, particularly soulbound tokens, are perceived as integrating rights and permissions directly into user accounts, their widespread use raises the fundamental question about their need. Many rights, obligations, voting, and login permissions for DAOs and other blockchain-based entities are already encapsulated in a single contract that anyone can check individually. Tokens should not be used to map direct rights, only indirect permissions or achievements that might be used for benefits. Compared to real life, users might show friends their goals or pictures of their work but not the account login or key to their office, which they manage on the backend. If backend circumstances change while they have permissions, they need to be changed, ending in meaningless parallel maintenance of two instances. With the right frameworks, only some pieces of data have to become an expensive token. Attendance or acknowledgments could be a simple claim or non-divisible currency with additional metadata. On top of that, developers should continually optimize for systematic and cost-effective approaches regarding decentralized data management.

Addressing the adaptability of permissions and rules, a structural connection between an account manager and the DAO could facilitate similar dynamic adjustments based on users’ achievements and time allocations. If the account address remains static, the application of ranks and permissions could be mapped directly to accounts, enhancing the adaptability and responsiveness to member achievements. Similarly, consent functionality could become a direct part of communities if their abstracted accounts provide suitable interfaces. Users could then look up if there is a notification regarding their global account instead of relying on individual token implementations.

9 Implementing Restriction

As discussed in Chapters 6.4 and 8.5, smart accounts represent the ideal framework for a more open transfer restriction interface and future data economy. This chapter will combine the previous identity-related token guidelines and standardizations to develop a generic Solidity standardization, serving as a foundation across presented use cases.

As a pioneer, the modular concept of Universal Profiles already solves the need for token-based identity, security, backups, and asset storage. Therefore, the standard is implemented as LSP to extend the social profile ecosystem further. A generic restriction standard for token transferability could then be used in a whole spectrum of use cases:

- Soulbound Tokens
- Identity or human-limited tokens or vaults
- Community-restricted claims or their recovery
- Property-restricted bindings for services like domains
- Assets with added accessories or related appliances
- Reputation for skill-based and nuanced governance
- Accurate proof of attendance
- Non-financial rewards or recognition
- New membership opportunities
- Simple or hybrid lock-in or staking processes

When outlining the core setup of LSPs and restriction, the notification and asset integration already exist utilizing the Universal Receiver. Both asset types, the Digital Asset and Digital Identifiable Asset, come with individual and expandable storage and hooks. As a final piece, the ERC725 Account will be used as a base for attachments and comes with the Key Manager to allow rotatable EOA operators.

9.1 Denominators across Use-Cases

To establish a suitable standard, listing guidelines to comply with is essential. These principles are based on the conducted guiding matrix for identity-based tokens from Chapter 8.2. On top of this, existing and finalized proposals from Chapter 8.4 give a direction for standard best practices. Both can be combined with the convenience of smart contract accounts. The universal restriction rules will look like the following:

1. Binding the asset to an account address must be possible.
2. Only metadata should be anchored directly.
3. The link to personal data must be completely removable by the owner.
4. The recipient must confirm transfer rules before the handout is executed.
5. The recipient must confirm removal rules before the handout is executed.
6. The recipient must confirm the token contents before executing the handout.
7. Bound assets can no longer be transferred without further intermediate steps.
8. Bound assets and rules can no longer be modified without reauthorization.
9. Restriction types must allow for final, hybrid, or temporary binding.
10. A variety of allowed operators must be supported to perform transfers.
11. Various people must be able to remove the asset from the account.
12. The owner must always be able to remove the asset.
13. The standard must exist as an extension to regular tokens.
14. The standard must not commit to a fungibility type.
15. The bound asset must have a detectable interface.

On behalf of the guidelines, the following features are carried forward from finalized ERC standardizations, even if the proposals cannot be utilized thoroughly:

- **ERC-5192**: Minimal Lock and Unlock Events
- **ERC-5484**: Consensual Burn Authentication

Based on the feedback and discussions during their review stages of the standards, the following concepts of standardization can be utilized more broadly:

- **ERC-5753**: Having an individual set of unlockers
- **ERC-5192, ERC-6454, ERC-5633**: Having a lock indicator
- **ERC-6982**: Having a default asset lock type

9.2 Formation of the Proposal

Based on the preparation, a suitable Solidity standardization was designed and published as **LSP22 Transfer Restriction** within the LUKSO Improvement Proposals. The core layout and signature setup can be found in Appendix Part D. The detailed standardization paper and interface calculation can be retrieved from the following code repository:

`https://github.com/fhildeb/ma-standardization`

9.2.1 Restriction Architecture

LSP22 outlines a generic restriction standard based on previous rules and impressions. It inherits and requires the following standardizations to achieve those goals:

- **EIP-165**: A retrievable interface identification allows frontends to identify the standard and related functionality without direct interaction.
- **EIP-712**: A specific typed data hashing and signature method is implemented to allow consent during the handout or locking process.
- **LSP4**: The interface can be attached to any fungibility type or asset using a specific metadata standardization needed for consensual signing.
- **LSP5**: A unified storage scheme for owned assets is utilized to embed and retrieve locked assets from smart contracts directly.
- **LSP6**: The key manager must validate signing permissions of operator keys to prove ownership for abstracted smart accounts.

Within the setup, LSP22 has four restriction types that can be applied to individual token IDs and owners within a public registry of the smart contract:

- **None**: No restriction was set, meaning it will fall back to the default type of the contract that must be set during the initialization of the smart contract.
- **TempLock**: A dynamic locking mechanism where an asset can toggle between locked and unlocked states, governed by custom logic. The switching allows for asset recovery or community-based transfers while maintaining security. Here, the lock status can be altered in response to particular events or conditions, ensuring adaptability and controlled access.
- **SoftLock**: A condition in which an asset, post-acceptance by the owner, is locked to its assigned address with provisions for deletion or removal under specific circumstances. Only designated addresses can remove the asset, balancing security and flexibility.
- **HardLock**: A status where an asset is irrevocably locked to its assigned address once accepted by the owner. Under this condition, the asset cannot be transferred or removed, ensuring permanent retention and immutability.

On top of that, each smart contract has to come with a default restriction type that can be used to control a range of token IDs or owners at once.

Similar to the restriction types, the standardization introduces a set of permissions related to the transfer behaviors of bound assets:

- **None**: The default value indicates that the address has no privileges regarding the locking method of the asset, meaning that the address was not included within the asset lifecycle.
- **CanRemove**: Indicates that the address has the right to disassociate from the asset in case of expiration or particular condition. The allowed address can burn or transfer the asset to the zero address.
- **isOperator**: Indicates that the asset can be transferred to another address regarding backups or shared community management when unlocked. Being able to transfer also means that operators can remove the asset anytime.

With this setup of roles, issuers, operators, and removal rights can be strictly separated. Each smart contract has a public hierarchical entry list that can handle multiple management addresses and their permissions for each asset.

Regarding the locking itself, there is also a public entry list for the transferability indicator of an asset. Here, the current restriction of assets can not only be verified but also comes with additional information like when the status was changed and by which address.

The standardization comes with four individual functions that can be called to lock, unlock, remove, or redeem a restricted asset. In order to lock the asset, the receiver's signature is mandatory for the execution. Therefore, the owner has to sign the asset's metadata, restriction type, and an array of all permissioned addresses and roles in order to give consent. The locking can either be combined with the initial mint or transfer or called later. The related signing is done off-chain to balance costs and give services freedom regarding the interaction flow. However, the structured data schema must be strictly defined within the smart contract so ownership signatures can be verified in a standardized way. Here, the key manager comes into play to allow consent across abstracted smart contract accounts like Universal Profiles. The locking function retrieves the signature permission of the used key and authenticates that it belongs to the owner. If the signature is valid, the function continues if the sender is included in the operator list and sets the lock status for the asset. Upon receipt, the asset must be added to the receiving smart contract's list of owned assets.

If the restriction type is soft or hard-locked, the locking can only be done once before removing it. While an asset is locked, any transfer function will fail. For temporary restriction types, the asset can be unlocked by an operator's address to transfer to a new address. Upon re-locking an asset, the receiver has to sign again so that it is impossible to change properties without permission. Nevertheless, the locking process may introduce new operators or an updated restriction type.

Similar to the unlocking of an asset, it can be removed by anyone having related permission, granting the right of disassociation for temporary or soft-locked assets. The function effectively hard-locks the ownership to the zero address, blocking further interaction. Upon removal, the asset must also be removed from the owned asset list of the owner account.

It is optionally possible to allow owners to permanently lock an asset to their own or an owned and allowlisted address. When an asset is redeemed, the restriction type gets modified to a more strict category. The redeemability further allows for use cases like voluntary acquisition or even the binding of accessories, properties, or appliances onto other digital assets.

9.2.2 Signature Validation

Regarding the consent for abstracted accounts managing restricted assets, the LSP22 standard comes with multiple mandatory variables defined by the EIP-712 standard:

- **Nonces**: Every address has its nonce counter to ensure unique signatures whenever an operator key of a smart contract grants a lock permission. It increases with each locking operation to prevent replay attacks.
- **Domain Separator**: A specific hash value for the asset's smart contract that must be implemented in the constructor to prevent signature replay attacks across different contracts and blockchains. It includes details such as the contract name, version, chain ID, and the address.
- **Domain Type Hash**: The standardized identifier for the domain separator's contents used to retrieve the operator address based on hash and signature.
- **Lock Type Hash**: The LSP22-specific data structure used to generate the hash. It will be signed from an operator key of the owner's smart contract to lock an asset. The structure includes the lock function name, the owner, the asset's metadata, restriction type, the related array of permissioned addresses, the signer, and its nonce.

As described before, the permission of the smart contract's operator key that signed the typed data structure has to be authenticated using the key manager. Whenever the lock function is called, the hash of the data structure is created using both type hashes and the domain separator. Once generated, the signer's nonce gets raised by one. The generated hash is used when calculating the signer's address based on the provided ERC-712 signature parts. Once finished, the generated address is used to call the key manager of the owner's smart contract to retrieve the permission key. If the key's value is greater or equal to the sign permission, the locking mechanism can proceed.

9.3 Asset Integration

The messaging protocol's architecture described in Chapter 10.3 will show how the new decentralized data economy can be handled in a responsible and future-oriented manner. Therefore, the characteristics of an SBT could be incorporated as achievements by utilizing LSP22. This chapter outlines a potential integration flow.

Optional achievements could be realized as a fungible token representing experience points. When sending messages or logging in to the protocol daily, these may be earned and incremented locally in the browser's cache. Users could then claim and transfer the account-specific experience to their Universal Profile before the local cache might be cleared. As soon as the user agrees to the transfer restriction by signing all the token information outlined in Chapter 9.2.2, the experience will get credited on-chain, and the local counter may reset.

The counter would have to be moved to a temporary unit during the claim process to prevent duplication of experience points, discarded when the smart contract event is received. To ensure that the experience points are earned honestly and remain credible over the entire user base, the total amount of experience would only be increased with each on-chain claim. A private hashing and salt procedure could be combined with the user's messaging signature to prevent local counter modification.

Local data preservation and individual global settlement periods provide the right balance. Thus, every user could decide on the importance of the settlement, depending on how often the points are claimed.

10 Architectural Chat Framework

Based on Chapters 2.2, 6.1, and 8.1 evaluation, interaction is needed across smart account-based communities. However, decentralized social layers must avoid relying on central social media providers to preserve the reliability and independence of Web3 accounts. The following chapter will combine pioneering abstraction and restriction standardizations to build out the architecture of a chat application to enable global messaging through Universal Profiles. The service is imagined to become a prime example of future social economies, as users could earn an on-chain reputation using secure and private communication channels. Following this master thesis, the outlined concept will be implemented as a web service.

10.1 Messaging Ecosystems

To define the architecture, an appropriately decentralized chatting protocol has to be selected on the backend. Unlike the Web 2.0 era, where the public protocols were the foundation upon which companies built and generated value, the trend has shifted towards creating nurturing communities and platforms [113] that operate and share in the project's success. The shared principle is further made possible by incorporating blockchain and token economics principles.

An example of chatting would be the Push Protocol, which comes with chatting across EOAs, notifications for smart contract services on top of its node subnetwork, and tokens that will later be used for incentivization. While not a lightweight construct, it is widely used for Ethereum dApps and their notifications as it comes with JavaScript package integration and its API to connect to the network. However, the subnet of nodes is still operated by the founders. [114]

Another initiative is Status, which runs an open messaging network that will soon feature operator incentive structures. The project distributes its protocol's usage through mobile and desktop apps with a wallet and an in-app browser. Under the hood, it utilizes the peer-to-peer gossip protocol Waku to relay and spread encrypted messages across all operators automatically. [115]

One of the most present chatting protocols is XMTP. It is a communication network akin to SMTP from Web 2.0, used to facilitate emails over the Internet. Designed to be a public good, XMTP tries to serve as the Web3 backbone interoperable with all EVM chains. Like Status, the core of XMTP nodes runs on the Waku stack for synchronization. Clients are responsible for encrypting, submitting, and retrieving messages from the network. Users must generate or use existing wallets through XMTP-compatible apps upon use. These will then generate an identity within the subnetwork that can receive messages through an inbox. Due to its openness, the benefit here is that users gain continued access to all messages even if specific dApps are discontinued. As of October 2023, however, all the nodes are operated by their central software developer, XMTP Labs, which is still investigating a public incentive layer. [116] Due to the transparency and need, the XMTP network is already used across multiple aspiring social media apps built on the Lens Protocol and the community hub Common Ground [117]. A third yet centralized messaging app currently getting traction is called Nfty Chat [118]. Across these solutions, people can use regular wallets to sign in and chat while linking their on-chain possessions. Lenster is using NFTs. Others fall back to fully centralized profile storage that has to be manually linked to the communicating wallet address.

A more minimal approach was pursued with dm3, a Web3 protocol for direct messages through ENS. The solution has an open-source backend software that can be run permissionless by any dApp provider, web service, or private person's server. Users only specify a user profile entity on the blockchain through their wallets, including a public signing and encryption key and a list of delivery services to which the apps will forward the encrypted message containers. The storage encryption key is derived from the EOA via a local signature to allow end-to-end encryption. Each delivery service's profile is also posted on-chain to ensure that information remains confidential while verifying its accurate delivery to the server instance. As for users, the server profile includes a public signing key for applied postmarks on the encrypted message containers, a public encryption key for encoding the delivery information, and the address of the server instance itself. The core of the concept is how the on-chain anchor points are stored. Here, ENS was selected as the anchor to tap into an existing blockchain user base, allowing for the immediate onboarding of up to 745,000 unique owners [119] without further redo. As mentioned in Chapter 4.3, the domains feature custom text records that can be attached to the name. This functionality is utilized to manifest their public and updatable chatting credentials. Conveniently, ENS also includes the CCIP technology to reference off-chain data entries. With it, dm3 can function with users who own an on-chain domain and use their own ENS entry to create dm3 subdomains for newcomers. While equally secure, users could choose a less decentralized, cheaper onboarding by storing their chatting credentials on the company's server. [120]

The discussed protocols can be curated into an evaluation matrix to provide a comprehensive overview and facilitate a nuanced understanding. It will also help analyze the core properties and attributes, offering insights into their strengths, weaknesses, and application areas. All protocols come with native end-to-end encryption. Extended properties were analyzed and rated within Table 6 using the following indicators:

░ perfect to good ▒ ok to moderate ▓ poor to unpleasant

	Push Protocol	Status	XMTP	dm3
Architecture	permissionless network of nodes	permissionless network of nodes	permissionless network of nodes	permissionless mapping of standalone servers
Technology	ecosystem-specific open-source subnetwork of delivery servers	project-specific open-source subnetwork based on Waku & Geth	broad open-source subnetwork based on Waku	broad open-source server-containers connecting with ENS
Operation	centralized, soon to be decentralized	centralized, soon to be decentralized	centralized, soon to be decentralized	shared and independent server net
Incentivisation	soon to be token-based	soon to be token-based	under investigation	fully separated from protocol
Node Operation	soon to have token-based incentivisation	soon to have token-based incentivisation	private without direct profit	public without direct profit
Onboarding	wallets & EOAs, centralized profile creation	wallets & EOAs, centralized profile creation	wallets & EOAs, external app-based profile creation	wallets & EOAs, profile creation through ENS
Data Retrieval	network-based identity	network-based identity	network-based identity	messaging credential
Concept	open multi-layer notification service and chat service	individual application ecosystem	open messaging base layer for	open and efficient chatting with minimal data sharing

Table 6: Evaluation of Leading Web3 Messaging Protocols

As seen from the table, the blockchain industry is witnessing a movement towards permissionless yet unique ecosystems. Software systems often get more extensively embellished than necessary for their core purpose. As seen through the guiding figures of the landscape, protocols are typically integrated with separate shared networks of nodes, prioritizing decentralization and fostering redundancy across all players. Due to the complicated establishment of shared operation and incentivization, networks are commonly managed by the protocol's initiators. Nevertheless, as they expand, there is a need to incentivize node operating systems to offset the operational costs. Here, the question arises as to why everything has to be an explicitly shared network.

From an economic point of view, adhering to the principle of minimal data sharing should be beneficial when it comes to private messaging services. Generally speaking, data organization becomes much more accommodating with regular server structures. For many services, including chat, it could be way more convenient and efficient to call up a few standalone server containers inaccessible to their providers instead of interconnecting with hundreds to thousands of nodes. The sharing principle becomes a challenge regardless of the encryption. There should be a data footprint-reducing balance between regular server technology and the connection to the global ownership register. The global ledger can only be used as an anchor point to get and verify data. By having individual connected servers, scalability could be guaranteed while users maintain their roots and data storage provider independently.

Because of the excellent starting point of a lightweight, end-to-end encrypted, and open source-based protocol, the dm3 protocol [121] was chosen as the base layer. Application developers can permissionless create one or multiple server instances for encrypted backups. However, it must be modified to work directly with abstract accounts, specifically Universal Profiles.

10.2 Protocol Modification

As abstracted accounts can store their data in a list based on keys and values, the centralized ENS registry is no longer needed to store information about it. While ENS acts as primitive spam protection, as people need to acquire ENS names, attackers could always use CCIP to mimic off-chain subdomains as dm3 does for more accessible onboarding purposes. By not restricting the type of storage for the user profile, protocol calls can be even more efficient, shrinking down the needed steps to gather the data from a registry, as they can be directly read from the connected contract. Where dm3 is building on a decentralized ledger utilizing a central registry by default, abstracted account support can allow truly decentralized networks of individual accounts without underlying subscription or governance.

The dm3 integration is done through 2 different packages: **dm3-react** and **dm3-lib** [121]. React provides a chat interface for browser-based applications and handles the connection to the EOA. The library package then mirrors the actual protocol that is called in the background. For the custom execution, the following protocol parts must be modified:

1. **Credential Storage**: The initial onboarding check of dm3 is to see if there is already a dm3 chatting credential for the EOA. If not, it has to be generated first. This step must be covered by the customized app that checks or writes the chatting credential directly into the on-chain profile before it can be used. Yet, the credential will still be based on the EOA, now acting as operator key.
2. **React Interface**: As the library typically only uses the EOA to query the registrar of ENS and calls the registry to get the appropriate text record, there is no interface for a related smart contract address. Here, the handover parameters of the protocol's interface must be adjusted to feature the connected Universal Profile address to fetch information directly from the abstracted smart account.
3. **Protocol Library**: The data queries to the ENS service can be omitted entirely from the protocol. With the passed Universal Profile property, a call to the storage key can be executed to get the chatting credentials within a single ERC-725 library call.

10.3 Application Design

An application framework should be created in TypeScript to merge all the presented software pieces from Chapter 10.2. As the existing dm3 interface has to be embedded within the application, the widely used React 18 is a good fit. It is one of the most widespread tools for creating web-based user interfaces. Its critical features are custom, reusable elements that can be nested independently. The finished framework should include the general page menu to separate the reputation system from the actual chat rooms visually. Shared memory is mandatory to implement the local counters and cross-page queries regarding the network, extension, and protocol connections. Further, user interfaces should be customized with Bootstrap to provide responsive and uniform-looking elements. The initial app frame can be retrieved from the following repository:

`https://github.com/fhildeb/up-chat`

A suitable interface must be provided since users log in and sign messages with blockchain-based EOA keys. Here, ethers is one of last year's best-maintained and used blockchain libraries, also utilized within the dm3 interface. It can access the Ethereum object of the browser to exchange data packets over the blockchain's RPC. This object would then be used to check the correct network connection before being able to use certain functionality.

The Universal Profile browser extension houses the Universal Profile and operator keys the user holds. The account is set up individually. After initialization, it already has the regular functionality, such as the LSP1 Universal Receiver, to query properties or access its storage register using the LSP2 ERC725 JSON Schema.

Concerning chatting, the DM3 packages must be imported into the framework. Here, the customized packages from Chapter 10.2 have to be built and loaded manually. For testing purposes, the backend needs to be installed on a single server. An encrypted container should be set up, hosting the DM3 server instance. Services can then communicate with the server using its profile credential. If multiple servers are specified in the credential, several connections are maintained in parallel, enabling the distributed and semi-decentralized system of backups if one party is not reachable anymore.

Individual encrypted containers are created for each DM3 user profile on every connected server based on the initially used EOA. At the startup of the app, users have to sign the following message: "Connect the dm3 app with your wallet. Keys for secure communication are derived from the signature. No paid transaction will be executed. Nonce: 0" to derive a storage signature for the server connection from the EOA. The signature generation can be seen as a login to the app instance. Each profile publication to the server is then used to register. To fetch messages from the server, users authenticate themselves using the sign-in. Upon a successful handshake, encrypted data packets can be transmitted and decrypted at the user's end. If the storage encryption key has to be changed, the nonce can be raised, updating the registered profile on the server.

Social gamification could be added after the chatting backend can be accessed through the modified libraries. Here, the sign-in should count up the locally hashed counter. For this step, a simple, fungible version of global experience points was outlined using the LSP22 Transfer Restriction standard from Chapters 9.2 and 9.3. Later, the asset should be deployed on the LUKSO Testnet using the Remix IDE. In correlation, the static asset instance can be set up within the dApp using its ABI and address through ethers. When chatting across Universal Profiles, the user should then be able to head to the appropriate reputation page and manifest the chatting experience on the profile. During the initial transfer, the asset is automatically added to the standardized LSP5 Received Assets register and can be retrieved directly from the application using the ERC-725 library. In the future, this experience could give people access to advanced features, highlight their profiles, give access to unique chat rooms, or provide additional digital goods.

10.4 Future Development

The initial version of the blockchain-based chat architecture pursued its goals to showcase future data economy models. Nevertheless, some improvements are still missing to allow for long-term use of any service utilizing the framework. These modifications include lightweight frontend changes and more in-depth protocol add-ons.

- **Improved User Search**. What is most noticeable is that users either have to know the address of the person they want to chat with or search for them using an external profile explorer. It would be nice to add a search function that can retrieve and search a register of created blockchain profiles depending on the user input. Here, the regular version of dm3 is still a step ahead, as there is currently no name service within the LUKSO ecosystem.
- **Distributed Server Management**. Another improvement would be a window to manage the servers in the backend. In the initial framework, only one server is controlled, meaning messages are stored completely centrally, despite the decentrally stored credentials. Here, it should be possible for users to see their own or a selection of storage containers, from which they can select several in parallel and add them to their chatting credentials. This way, there are always multiple connections, making the storage fail-safe and persistent.
- **Multi-Device-Sessions**. If users chat through their Universal Profile with different devices assigned different keys, they can initially only read messages sent from the current key. On top of the framework, applications should be able to generate a link or QR code to connect multiple sessions and share the encryption key, as done by regular chat apps.
- **Storage Key Rotation**. As essential as the selection of multiple servers is the rotation of keys. Although the Universal Profile can exchange keys and assign different rights, the minimal approach of the dm3 protocol does not support updating the encryption keys of a container yet. The storage encryption key is always derived from the initially connected EOA, which can later be shared. If the Universal Profile removes the initial EOA key, the storage encryption key can not be freshly re-generated. Removal could result in users no longer being able to access their messages. Here, an add-on should allow users to decrypt all stored data using the current key and enter a new encryption key to put the data back on the server.
- **Chatting Allowlists**. Just as the chatting credential is stored in the profile, it could be extended with allowlists so only selected people can write into the user's inbox. Services utilizing the framework could provide a separate user interface to enter inbox requirements. These allowlists could be uploaded as a simple JSON file to a storage solution, which can act as an additional filter on the front end.

11 Concluding Remarks

Blockchain technology has brought a disruptive era of account and data management, overcoming the current way of navigating the internet through external custody. Within the last two years, the development of identity-based or social communities and their infrastructural demand has heavily increased, creating fair relationships and secure identities as known from real life.

11.1 Summative Insights

With pioneers like Lens Protocol, Farcaster, and Common Ground, social media applications are arousing interest and demonstrating how public blockchain-based identity integration can look like. While they deliver exceptional benefits such as self-sovereignty, individual feeds, open-source code, interchangeable frontends, and censorship-resistant base layers, they are generally limited and hindered by the widespread EOAs and storage networks still in their initial stage. Security and convenience are outsourced to tokens, external registries, or centrally governed environments to counteract the restraints. This phenomenon leads to a mixed nest of different identity systems for each service, similar to Web 2.0, as every ecosystem generates and manages its profile setup.

The social development front runs in parallel to the almost decade-long evolution of abstract smart accounts, trying to remove the limitations and unleash the full potential of blockchain for the next wave of adoption. The main focus points are modular transaction verification, indirect payment options, and updatable permission keys, making accounts operation centers for entities of any kind. Abstraction is divided into two main areas: the protocol update to the regular account frame and the account management systems sitting on top. The most diverse management project evolved out of Ethereum and is based on likewise open Solidity standardizations called LSPs. By utilizing them, an on-chain account can become a universal profile with rich social context and convenience features. Contract-based computation can be simplified by seamlessly merging the asset interactions into the core center. Both help to move beyond previous plutocratic and anonymous governance systems.

With the rapid account changes, new gates are opening to embed user data. In the process, various token and claim-based solutions are emerging to allow for identity-restricted assets. The extended analyses of the current token restriction standards have shown the importance of reconsidering data sharing through blockchains to navigate risks.

If managed improperly, multiple previously anonymous or pseudonymous blockchain accounts could quickly be linked in undesirable ways to marginalize or misuse social groups for attacks, as decentralized social circles do not rule out manipulative communities. Conversely, too many private SBTs and connections could lead to hidden communication channels off-chain. This dichotomy can hardly be contained due to the lack of national regulation within global blockchain realms, so developers should strictly follow presented protection rules.

Society-wise, a balance of transparency and privacy is needed to build trust and integrity. Activities of powerful institutions should remain transparent and accountable while the privacy of individuals is strictly protected. While SBTs offer programmable privacy features and great potential to navigate in-group dynamics, they are no panacea. Flexibility should not lead to an environment devoid of legal and ethical boundaries. As the thesis discussed different approaches for data sharing, it is often enough to use signed claims like VCs instead of SBTs and allocate rights directly within DOAs, alternatively to the assets themselves.

11.2 Implementation Recap

Observing the wide assortment of topics, new data economies must be pinned down to multiple software pieces. It is a mix of architectural matters in Web 2.0, blockchain account management, and smart contract developments to add further functions and assets. In the upcoming years, numerous projects must undergo a public evaluation to determine how elements can effectively counterbalance each other. What is particularly important are unresolved community structures to solve recovery and revocability. These features are closely intertwined, as poor governance could undermine the trust and functionality of the entire social ecosystem [73].

The major accomplishment of this thesis is the in-depth analysis of related puzzle pieces of the decentralized data economic subject and their combined practical integration. The designed restriction standard adds a modular restriction component to the already feature-rich LSP ecosystem and is a pioneer tool for consensual bound assets for abstracted accounts. On top of that, the outlined architecture of the chatting dApp demonstrated how well user communication could work together using on-chain anchor points. Following this master thesis, the outlined concept will be implemented as a web service showing the possibilities for a new blockchain era.

By examining the currently available data networks and standards, the architectural framework of the dApp was already able to show the importance of efficiency and minimal data sharing, shrinking down the complexity many contract-based services face. In this regard, it is crucial to effectively integrate traditional server storage with node networks to ensure security and privacy. While there is often an inclination to address issues with complete decentralization, it is only sometimes feasible or necessary. Here, the blockchain privilege should continue to be primarily utilized for ownership settlement and identity purposes rather than as a data store.

11.3 Future of the Social Data Economy

In conclusion, the paramount challenge for decentralized societies remains to empower individuals with complete control over their content while maintaining the user-friendliness they have come to expect from conventional web services. The emerging nexus of abstract profiles with intertwined digital goods is fundamental to achieving these goals. However, projects must first counteract the additional operation costs of heavy smart contract usage to build up network effects. Therefore, economies will likely spread within separate or subordinate networks. Nevertheless, more significant engagement opportunities across identities will lead to a renewed upswing of DAO and NFT technology and increased user interaction. As interaction amplifies, it will attract more customers outside the regular blockchain field. By gaining considerable adoption, the prevailing decentralization of accounts and new data-sharing schemes lead to more democracy on the Internet and in society. The rise of decentralized organizations, profiles, and associated SBTs represents a positive shift, transforming the financially driven focus of the blockchain industry into a domain that prioritizes social engagement and community building. In this context, decentralized networks and societies empower equality and pluralism by retaining control over their data exchange and shifting focus to what is truly important: individuals and the authentic, unique relationships they foster.

Bibliography

[1] P. B. Nath and Md. M. Uddin, “TCP-IP Model in Data Communication and Networking,” Am. J. Eng. Res., vol. 4, no. 10, pp. 102–107, 2015.

[2] A. Kaushik, “Web Analytics 2.0: The Art of Online Accountability and Science of Customer Centricity,” Sybex, 2009, pp. 1ff, 241ff.

[3] S. Voshmgir, “Token Economy: How the Web3 reinvents the Internet,” 2nd ed., BlockchainHub Berlin, 2020, pp. 30f, 39f.

[4] A. Preukschat, D. Reed, and D. Searls, “Self-Sovereign Identity: Decentralized Digital Identity and Verifiable Credentials,” Manning Publications, 2021, pp. 3–5.

[5] G. Gilder, “Life After Google: The Fall of Big Data and the Rise of the Blockchain Economy,” Simon and Schuster, 2018, p. 7.

[6] Proxyrack, “Social Media Security Report.” Accessed: Oct. 27, 2023. [Online]. Available: *https://www.proxyrack.com/blog/social-media-security-report/*

[7] I. Atik, “Investigation of Facebook-Cambridge Analytica Data Privacy Scandal.” Instituto Politécnico do Cávado, 2020. Accessed: Oct. 27, 2023. [Online]. Available: *https://www.academia.edu/41701131*

[8] H. Berghel, “Malice Domestic: The Cambridge Analytica Dystopia,” Inst. Electr. Electron. Eng., vol. 51, no. 5, pp. 84–89, 2018, doi: 10.1109/MC.2018.2381135.

[9] W. Christl and S. Spiekermann, “Networks of Control: A Report on Corporate Surveillance, Digital Tracking, Big Data & Privacy,” Facultas, 2016, pp. 15–21. Accessed: Oct. 27, 2023. [Online]. Available: *https://crackedlabs.org/dl/Christl_Spiekermann_Networks_Of_Control.pdf*

[10] General Data Protection Regulation, GDPR-EU. 2023, p. Article 4. Accessed: Oct. 27, 2023. [Online]. Available: *https://www.privacy-regulation.eu/en/article-4-definitions-GDPR.htm*

[11] General Data Protection Regulation, GDPR-EU. 2023, p. Article 13. Accessed: Oct. 27, 2023. [Online]. Available: *https://www.privacy-regulation.eu/en/article-13-information-to-be-provided-where-personal-data-are-collected-from-the-data-subject-GDPR.htm*

[12] A. Gandomi and M. Haider, “Beyond the hype: Big data concepts, methods, and analytics,” Int. J. Inf. Manag., vol. 35, no. 2, pp. 137–144, 2015, doi: *https://doi.org/10.1016/j.ijinfomgt.2014.10.007.*

[13] A. Voss, “Fixing the GDPR: Torwards Version 2.0.” EPP - European People’s Party, 2021. Accessed: Oct. 27, 2023. [Online]. Available: *https://www.axel-voss-europa.de/wp-content/uploads/2021/05/GDPR-2.0-ENG.pdf*

[14] U. Feige and A. Shamir, “Zero-knowledge proofs of identity,” J. Cryptol., no. 1, pp. 77–94, 1988.

[15] “Decentralized Identifiers (DIDs v1.0.” Jul. 19, 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://www.w3.org/TR/did-core/*

[16] “Verifiable Credentials Data Model v1.1.” Mar. 03, 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://www.w3.org/TR/vc-data-model/*

[17] A. Antonopoulos and G. Wood, “Mastering Ethereum: Building Smart Contracts and Dapps,” 1. Edition., O’Reilly Media, 2018, pp. 61–76.

[18] A. Antonopoulos and G. Wood, “Mastering Ethereum: Building Smart Contracts and Dapps,” 1. Edition., O’Reilly Media, 2018, pp. 127–129.

[19] “Safe Wallet, Web3 Account Abstraction Developer Stack.” Gnosis. Accessed: Oct. 27, 2023. [EVM Smart Contracts]. Available: *https://safe.global/*

[20] A. Fauvre-Willis, “Hiding suspicious NFT transfers on OpenSea,” OpenSea. Accessed: Oct. 27, 2023. [Online]. Available: *https://opensea.io/blog/articles/hiding-suspicious-nft-transfers-on-opensea*

[21] S. Casale-Brunet, M. Zichichi, L. Hutchinson, M. Mattavelli, and S. Ferretti, “The impact of NFT profile pictures within social network communities,” in GoodIT: Proceedings of the ACM Conference on Information Technology for Social Good, New York: Association for Computing Machinery, 2020, pp. 283–291. Accessed: Oct. 27, 2023. [Online]. Available: *https://doi.org/10.1145/3524458.3547230*

[22] “Ethereum Name Service.” ENS. Accessed: Oct. 27, 2023. [EVM Smart Contracts]. Available: *https://ens.domains/*

[23] “Etherscan.” Etherscan. Accessed: Oct. 27, 2023. [Ethereum Block Explorer]. Available: *https://etherscan.io/*

[24] V. Buterin, “A History of Account Abstraction,” presented at the EthCC[6], Paris, Jul. 18, 2023. Accessed: Oct. 27, 2023. [Online]. Available: *https://www.youtube.com/live/iLf8qpOmxQc?si=_C5fOLPH84DZe5qo*

[25] V. Buterin, “EIP-86: Abstraction of transaction origin and signature.” 2017. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-86*

[26] F. Vogelsteller, “ERC-734: Key Manager.” 2017. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/ethereum/eips/issues/734*

[27] F. Vogelsteller, “ERC-735: Claim Holder.” 2017. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/ethereum/eips/issues/735*

[28] F. Vogelsteller and T. Yasaka, “ERC-725: General data key/value store and execution.” 2017. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/ethereum/eips/issues/735*

[29] ERC-725 Alliance, “Ethereum Identity Standard.” Accessed: Oct. 27, 2023. [Online]. Available: *https://erc725alliance.org/*

[30] F. Giordano, M. Condon, P. Castonguay, A. Bandeali, J. Izquierdo, and B. Masius, “ERC-1271: Standard Signature Validation Method for Contracts.” 2018. Accessed: Oct. 27, 2023. [Online]. *Available: https://eips.ethereum.org/EIPS/eip-1271*

[31] V. Buterin, A. Dietrichs, M. Garnett, W. Villanueva, and S. Wilson, “EIP-2938: Account Abstraction.” 2020. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-2938*

[32] S. Wilson, A. Dietrichs, M. Garnett, and M. Zoltu, “EIP-3074: AUTH and AUTHCALL opcodes.” 2020. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-3074*

[33] D. Finlay and S. Wilson, “EIP-5003: Insert Code into EOAs with AUTHUSURP.” 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-5003*

[34] V. Buterin et al., “ERC-4337: Account Abstraction Using Alt Mempool.” 2021. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-4337*

[35] Privacy & Scaling Explorations, “BLS Wallet: Bundling up data,” Medium. Accessed: Oct. 27, 2023. [Online]. Available: *https://medium.com/privacy-scaling-explorations/bls-wallet-bundling-up-data-fb5424d3bdd3*

[36] Ethereum Foundation, “ERC-4337 Documentation.” Accessed: Oct. 27, 2023. [Online]. Available: *https://www.erc4337.io/docs*

[37] F. Vogelsteller and V. Buterin, “ERC-20: Token Standard.” 2015. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-20*

[38] F. Vogelsteller, “LSP0 ERC725 Account.” 2021. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-0-ERC725Account.md*

[39] J. Carvalho and F. Vogelsteller, “LSP1 Universal Receiver.” 2019. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-1-UniversalReceiver.md*

[40] F. Vogelsteller, “LSP5 Received Assets.” 2021. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-5-ReceivedAssets.md*

[41] F. Vogelsteller, “LSP12 Issued Assets.” 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-12-IssuedAssets.md*

[42] F. Vogelsteller and J. Cavallera, “LSP6 Key Manager.” 2021. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-6-KeyManager.md*

[43] F. Vogelsteller, “LSP2 ERC725Y JSON Schema.” 2020. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md*

[44] F. Vogelsteller, “LSP3 Profile Metadata.” 2020. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-3-Profile-Metadata.md*

[45] D. Silverman, “Building on Lens Protocol,” Amsterdam, Mar. 26, 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://youtu.be/usgYL-KYd7I*

[46] F. Vogelsteller, “LSP4 Digital Asset Metadata.” 2020. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-4-DigitalAsset-Metadata.md*

[47] F. Vogelsteller, C. Weck, M. Stevens, and A. Kumar, “LSP7 Digital Asset.” 2021. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-7-DigitalAsset.md*

[48] F. Vogelsteller, C. Weck, M. Stevens, and A. Kumar, “LSP8 Identifiable Digital Asset.” 2021. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-8-IdentifiableDigitalAsset.md*

[49] “LSP9 Vault.” 2021. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-9-Vault.md*

[50] “LSP10 Received Vaults.” 2021. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-10-ReceivedVaults.md*

[51] “Argent X Browser Wallet.” StarkNet, Oct. 17, 2023. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/argentlabs/argent-x*

[52] “Lens Protocol.” Aave. Accessed: Oct. 27, 2023. [Social Network Protocol]. Available: *https://docs.lens.xyz*

[53] “Lens Dispatcher.” Aave. Accessed: Oct. 27, 2023. [Social Network Protocol]. Available: *https://docs.lens.xyz/docs/dispatcher*

[54] “StarkNet.” StarkNet. Accessed: Oct. 27, 2023. [L2 Blockchain]. Available: *https://docs.starknet.io/documentation/*

[55] “Polygon PoS Network.” Polygon, 2017. Accessed: Oct. 27, 2023. [L2 Blockchain]. Available: *https://polygon.technology/polygon-pos*

[56] Lens Protocol, “Introducing Lens V2.” Accessed: Oct. 27, 2023. [Online]. Available: *https://mirror.xyz/lensprotocol.eth/-hJH-2IYSe56rK7IEdwSI17hUWt-paTyAs1r4Zes0uQ*

[57] Ethereum Organization, “Layer 2: Ethereum for Everyone.” Accessed: Oct. 27, 2023. [Online]. Available: *https://ethereum.org/en/layer-2/*

[58] “StarkNet JS Account Creation.” StarkNet. Accessed: Oct. 27, 2023. [Online]. Available: *https://www.starknetjs.com/docs/guides/create_account/*

[59] “Starknet Account Interface.” StarkNet. Accessed: Oct. 27, 2023. [Online]. Available: *https://docs.starknet.io/documentation/architecture_and_concepts/Accounts/approach/*

[60] M. Köppelmann, “Ethereum’s L2 Limitations: Need for More!,” Denver, 2023. Accessed: Oct. 27, 2023. [Online]. Available: *https://youtu.be/dimUOYzhkmk*

[61] V. Buterin, “Ethereum in 30 minutes,” Bogota, 2023. Accessed: Oct. 27, 2023. [Online]. Available: *https://youtu.be/UihMqcj-cqc*

[62] “CryptoPunks.” Yuga Labs. Accessed: Oct. 27, 2023. [Online]. Available: *https://cryptopunks.app/*

[63] “Bored Ape Yacht Club.” Yuga Labs. Accessed: Oct. 27, 2023. [Online]. Available: *https://boredapeyachtclub.com/*

[64] G. E. Weyl, P. Ohlhaver, and V. Buterin, “Decentralized Society: Finding Web3’s Soul,” 2022, pp. 15–19. Accessed: Oct. 27, 2023. [Online]. Available: *https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4105763*

[65] W. Entriken, D. Shirley, J. Evans, and N. Sachs, “ERC-721: Non-Fungible Token Standard.” 2018. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-721*

[66] “Farcaster: A Decentralized Social Network.” Farcaster. Accessed: Oct. 27, 2023. [Peer to Peer Network]. Available: *https://github.com/farcasterxyz/protocol/blob/main/docs/OVERVIEW.md*

[67] V. Buterin, “Soulbound,” Vitalik Buterin’s website. Accessed: Oct. 27, 2023. [Online]. Available: *https://vitalik.ca/general/2022/01/26/soulbound.html*

[68] L. Shin, G. E. Weyl, and P. Ohlhaver, “How Soulbound Tokens Could Reduce Speculation and Improve DAO Voting.” Accessed: Oct. 27, 2023. [Online]. Available: *https://youtu.be/lKKgP2wS39U*

[69] “Proof of Attendance Protocol.” POAP Inc. [Ethereum]. Available: *https://poap.xyz/*

[70] K. Sills, “Soulbound Tokens (SBTs) Should Be Signed Claims,” Kate Sills’ website. Accessed: Oct. 27, 2023. [Online]. Available: *https://katelynsills.com/blockchain/soulbound-tokens/*

[71] V. Buterin, “Where to use a blockchain in non-financial applications?,” Vitalik Buterin’s website. Accessed: Oct. 27, 2023. [Online]. Available: *https://vitalik.ca/general/2022/06/12/nonfin.html*

[72] D. Allen, E. Frankel, W. Lim, D. Siddarth, J. Simons, and G. E. Weyl, “Ethics of Decentralized Social Technologies: Lessons from Web3, the Fediverse, and Beyond.” The Justice, Health, and Democracy Impact Initiative, 2023. Accessed: Oct. 27, 2023. [Online]. Available: *https://veri-media.io/wp-content/uploads/2023/03/ethics-decentralized-social-tech.pdf*

[73] “Soulbound Tokens (SBTs) Study Report Part 1: Building and Embracing a New Social Identity Layer?” Blockchain Governance Initiative Network, 2023. Accessed: Oct. 27, 2023. [Online]. Available: *https://bgin-global.org/documents/20230201_SBT.pdf*

[74] T. Daubenschütz, “What are Account-bound tokens?,” Proof In Progress. Accessed: Oct. 23, 2027. [Online]. Available: *https://proofinprogress.com/posts/2022-05-30/what-are-account-bound-tokens.html*

[75] C. Allen and S. Appelcine, “A Primer on Self-Sovereign Identity.” Web Of Trust Info. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/WebOfTrustInfo/rwot5-boston/blob/master/topics-and-advance-readings/self-sovereign-identity-primer.md*

[76] K. Cameron, “The Laws of Identity,” Kim Cameron’s Identity Weblog. Accessed: Oct. 27, 2023. [Online]. Available: *https://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf*

[77] S. Natsuhiko, “Seven Principles of Digital Being,” .Nat Zone Digital Identity and Privacy. Accessed: Oct. 27, 2023. [Online]. Available: *https://nat.sakimura.org/2023/01/19/seven-principles-of-digital-being/*

[78] “Metadata Standards: Disable trading for staked or locked tokens.” in Developer Tutorials. OpenSea. Accessed: Oct. 27, 2023. [Online]. Available: *https://docs.opensea.io/docs/metadata-standards#disable-trading-for-staked-or-locked-tokens*

[79] E. Bottazzi and S. Jain, “Zero-Knowledge Soul-Bound-Token (ZK SBT).” Accessed: Oct. 27, 2023. [Online]. *Available: https://github.com/enricobottazzi/ZK-SBT*

[80] N. Goel, “ERC-1132: Extending ERC20 with token locking capability.” 2018. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-1132*

[81] J. Lebrun, T. Malghem, K. Thizy, L. Falempin, and A. Boudjemaa, “ERC-3643: T-REX - Token for Regulated EXchanges.” 2021. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-3643*

[82] M. Onila, N. Zeman, and N. Cotaie, “ERC-6808: Fungible Key Bound Token.” 2023. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-6808*

[83] O. Aflak, P.-M. Le Bris, and M. Martin, “ERC-4671: Non-Tradable Tokens Standard.” 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-4671*

[84] T. Daubenschütz, “ERC-4973: Account-bound Tokens.” 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-4973*

[85] Tyler, Alex, and John, “ERC-5058: Lockable Non-Fungible Tokens.” 2022. Accessed: Oct. 27, 2023. [Online]. *Available: https://eips.ethereum.org/EIPS/eip-5058*

[86] M. Onila, N. Zeman, and N. Cotaie, “ERC-6809: Non-Fungible Key Bound Token.” 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-6809*

[87] T. Daubenschütz, “ERC-5192: Minimal Soulbound NFTs.” 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-5192*

[88] B. Cai, “ERC-5484: Consensual Soulbound Tokens.” 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-5484*

[89] F. Makarov, “ERC-5753: Lockable Extension for EIP-721.” 2022. Accessed: Oct. 27, 2023. [Online]. *Available: https://eips.ethereum.org/EIPS/eip-5753*

[90] 5660-eth and Wizard Wang, “ERC-6147: Guard of NFT/SBT, an Extension of ERC-721.” 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-6147*

[91] B. Škvorc, F. Sullo, S. Pineda, S. Bogosavljevic, and J. Turk, “ERC-6454: Minimal Transferable NFT detection interface.” 2023. Accessed: Oct. 27, 2023. [Online]. *Available: https://eips.ethereum.org/EIPS/eip-6454*

[92] F. Sullo and A. Spataru, “ERC-6982: Efficient Default Lockable Tokens.” 2023. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-6982*

[93] P. Chittara, StreamNFT, and S. Joshi, “ERC-7066: Lockable Extension for ERC-721.” 2023. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-7066*

[94] R. Roullet, “PR for ERC-1238: Non-transferable Token Standard.” 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/ethereum/EIPs/pull/5617/files*

[95] N. Greco, “ERC-1238: Non-transferrable Non-Fungible Tokens (badges).” 2018. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/ethereum/EIPs/issues/1238*

[96] W. Radomski, A. Cooke, P. Castonguay, J. Therien, E. Binet, and R. Sandford, “ERC-1155: Multi Token Standard.” 2018. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-1155*

[97] R. Roullet, “ERC1238 Implementation.” iolet Protocol, 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/violetprotocol/ERC1238-token/blob/main/contracts/ERC1238/ERC1238.sol*

[98] R. Roullet, “ERC1238 Receiver Implementation.” iolet Protocol, 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/violetprotocol/ERC1238-token/blob/main/contracts/ERC1238/IERC1238Receiver.sol*

[99] R. Bloemen, L. Logvinov, and J. Evans, “EIP-712: Typed structured data hashing and signing.” 2017. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-712*

[100] R. Roullet, N. Greco, and C. Chung, “ERC-1238: Non-transferable Token (NTT) Standard.” 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://erc1238.notion.site/*

[101] W. Wang, M. Meng, Y. Cai, R. Chow, Z. Wu, and AlvisDu, “ERC-3525: Semi-Fungible Token.” 2020. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-3525*

[102] L. M. G. Ramos and M. Arazi, “ERC-5516: Soulbound Multi-owner Tokens.” 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-5516*

[103] HonorLabs, "ERC-5633: Composable Soulbound NFT, EIP-1155 Extension." 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-5633*

[104] A. Zhu and T. Chen, "ERC-5727: Semi-Fungible Soulbound Token." 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-5727*

[105] Y. Aoki, "ERC-6268: Untransferability Indicator for EIP-1155." 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-6268*

[106] T. Daubenschütz, "ERC-5107: Initial draft for Name-bound tokens." 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/ethereum/EIPs/pull/5107*

[107] T. Daubenschütz, T. Cohen, and E. Bottazzi, "ERC-5107: Name-bound Tokens." 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/ethereum/EIPs/blob/ad528431af47054626b468edabecc0dcec91bd54/EIPS/eip-xxxx.md*

[108] M. Zoltu, "ERC-5114: Soulbound Badge." 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-5114*

[109] H. Kang and V. Pernjek, "ERC-5252: Account-bound Finance." 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-5252*

[110] J. Chang, "ERC-6239: Semantic Soulbound Tokens." 2022. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-6239*

[111] J. Windle et al., "ERC-6551: Non-fungible Token Bound Accounts." 2023. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-6551*

[112] T. Bergmueller and L. Meyer, "ERC-6956: Asset-bound Non-Fungible Tokens." 2023. Accessed: Oct. 27, 2023. [Online]. Available: *https://eips.ethereum.org/EIPS/eip-6956*

[113] J. Monegro, "Fat Protocols," Union Square Ventures. Accessed: Oct. 27, 2023. [Online]. Available: *https://www.usv.com/writing/2016/08/fat-protocols/*

[114] "Architecture Overview of Push." Push Protocol. Accessed: Oct. 27, 2023. [Online]. Available: *https://push.org/docs/chat/concepts/push-chat-architecture*

[115] "Status Specification." Status Research & Development GmbH. Accessed: Oct. 27, 2023. [Online]. Available: *https://specs.status.im/spec/1*

[116] "Architectural Overview of XMTP." XMTP Labs. Accessed: Oct. 27, 2023. [Online]. Available: *https://xmtp.org/docs/concepts/architectural-overview*

[117] "Common Ground: The Web3 Community Platform." Common Ground Association. Accessed: Oct. 27, 2023. [Online]. Available: *https://www.commonground.cg/*

[118] "Nftychat: Chat, discuss, and connect." NftyChat Inc. Accessed: Oct. 27, 2023. [Online]. Available: *https://nftychat.xyz/*

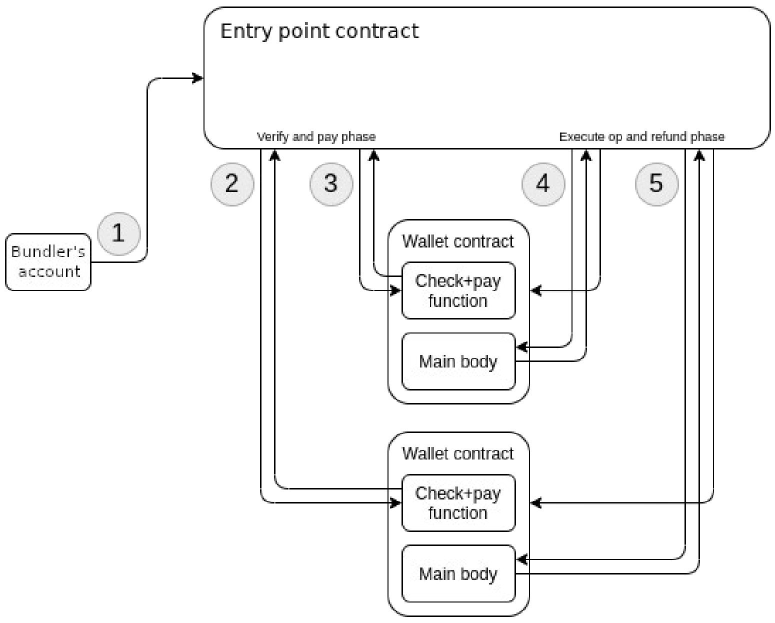
[119] “Ethereum Name Service Statistics,” ENS. Accessed: Oct. 27, 2023. [Online]. Available: *https://ens.domains/de/#home-statistics*

[120] “dm3 Specifications.” corpus.io GmbH. Accessed: Oct. 27, 2023. [Online]. Available: *https://dm3.readthedocs.io/en/doc-latest/*

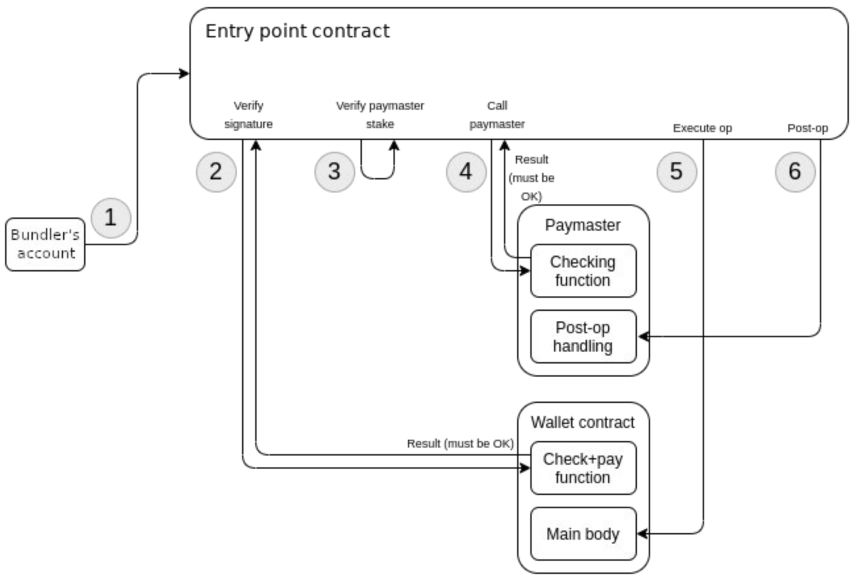
[121] “dm3 Protocol.” corpus.io GmbH. Accessed: Oct. 27, 2023. [Online]. Available: *https://github.com/corpus-io/dm3*

Appendix

Appendix, Part A: ERC-4337

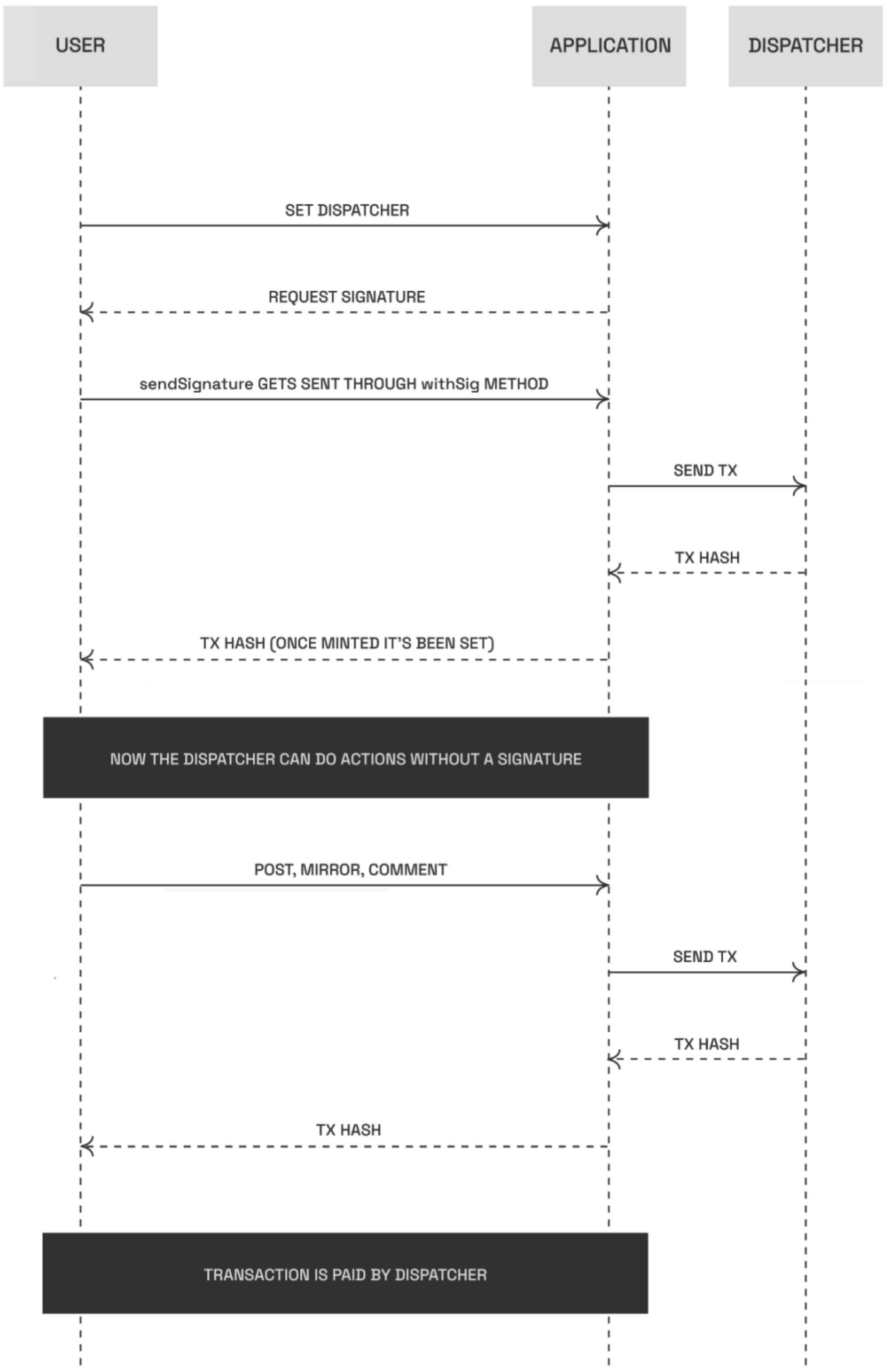


A1: Bundler flow for Abstracted Wallet Execution



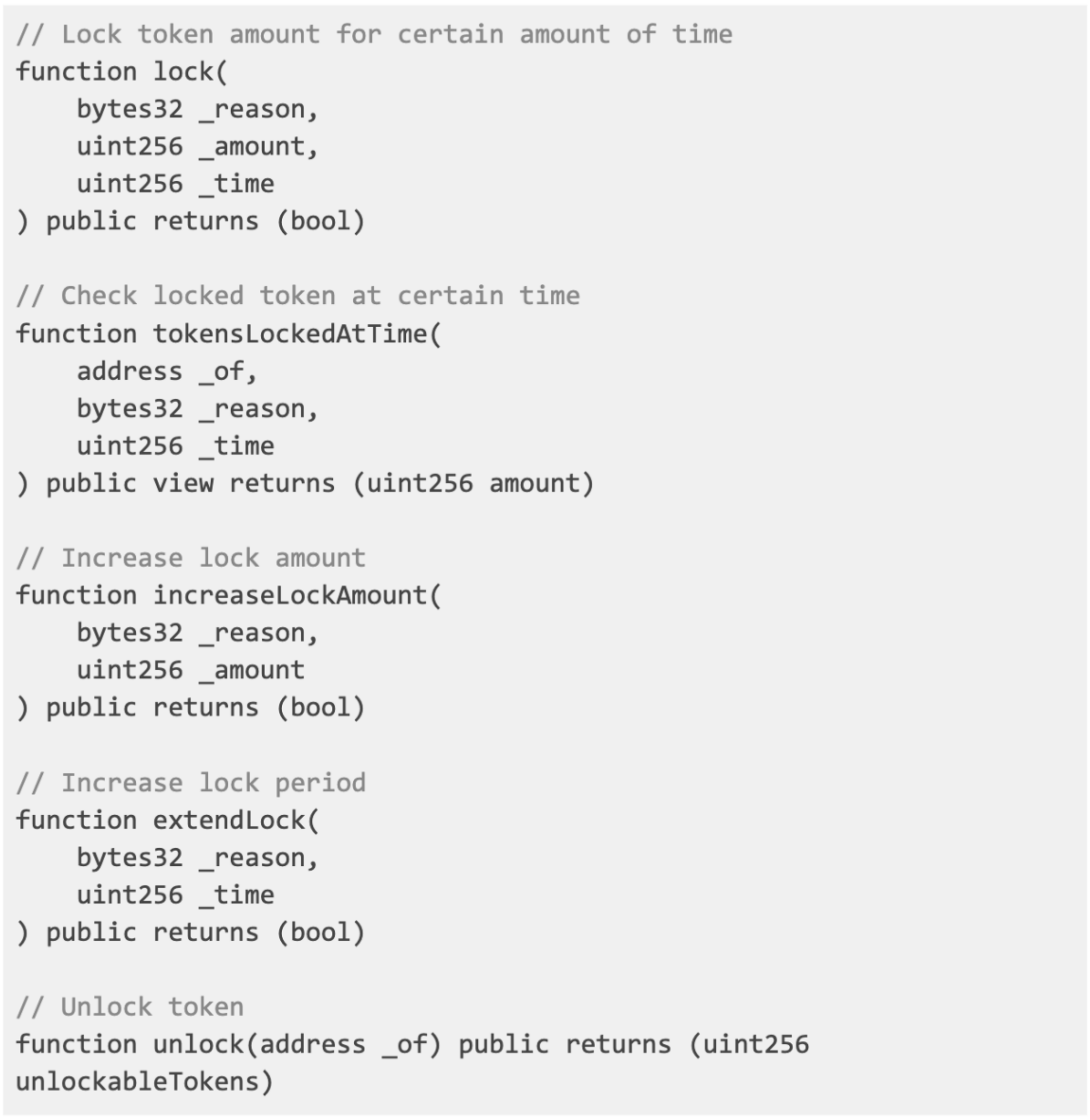
A2: Paymaster flow for Abstracted Wallet Execution

Appendix, Part B: Lens Dispatcher


USER
APPLICATION
DISPATCHER
SET DISPATCHER
REQUEST SIGNATURE
sendSignature GETS SENT THROUGH withSig METHOD
SEND TX
TX HASH
TX HASH (ONCE MINTED IT'S BEEN SET)
NOW THE DISPATCHER CAN DO ACTIONS WITHOUT A SIGNATURE
POST, MIRROR, COMMENT
SEND TX
TX HASH
TX HASH
TRANSACTION IS PAID BY DISPATCHER

B: Lens Proxy Wallet Payment

Appendix, Part C: ERC Restriction Standards



```
// Lock token amount for certain amount of time
function lock(
    bytes32 _reason,
    uint256 _amount,
    uint256 _time
) public returns (bool)

// Check locked token at certain time
function tokensLockedAtTime(
    address _of,
    bytes32 _reason,
    uint256 _time
) public view returns (uint256 amount)

// Increase lock amount
function increaseLockAmount(
    bytes32 _reason,
    uint256 _amount
) public returns (bool)

// Increase lock period
function extendLock(
    bytes32 _reason,
    uint256 _time
) public returns (bool)

// Unlock token
function unlock(address _of) public returns (uint256
unlockableTokens)
```

C1: ERC-1132 Fungible Time Lock

```
// Typed data of a mint transaction needing approval
struct MintApproval {
    address recipient;
    uint256 id;
    uint256 amount;
    uint256 approvalExpiry;
}

// Mint single non-tradable token
event MintSingle(
    address indexed minter,
    address indexed to,
    uint256 indexed id,
    uint256 amount);

// Mint multiple non-tradable token
event MintBatch(
    address indexed minter,
    address indexed to,
    uint256[] ids,
    uint256[] amounts);

// Burn single non-tradable token
event BurnSingle(
    address indexed burner,
    address indexed owner,
    uint256 indexed id,
    uint256 amount);

// Burn multiple non-tradable token
event BurnBatch(
    address indexed burner,
    address indexed owner,
    uint256[] ids,
    uint256[] amounts);
```

C2: ERC-1238 Minting Approval Data

```
// Set token approval for subordinate token category
function setApprovalForSlot(
    address _owner,
    uint256 _slot,
    address _operator,
    bool _approved
) external payable;
```

C3: ERC-3525 Slot Approval Stack

```
// Set authority role
function addAgent(address _agent) external;

// Force transfer of assets from authority
function forcedTransfer(
    address _from,
    address _to,
    uint256 _amount
) external returns (bool);

// Recover tokens to new wallet
function recoveryAddress(
    address _lostWallet,
    address _newWallet,
    address _investorOnchainID
) external returns (bool);

// Freeze transfers from a certain wallet
function freezePartialTokens(
    address _userAddress,
    uint256 _amount
) external;

// Bind agreement rules to token
function bindToken(address _token) external;

// Check transfer approvals based on linked agreement
function canTransfer(
     address _from,
     address _to,
     uint256 _amount
) external view returns (bool);
```

C4: ERC-3643 Token Regulation Concept

```
// Give one-time minting right to issuer
function delegate(address operator, address owner) external;

// Agree to approve minting a specific token
function approveMint(address owner) external;

// Agree to revoke
function approveRevoke(uint256 tokenId) external;

// Recover tokens to new wallet
function pull(
    uint256 tokenId,
    address owner,
    bytes memory signature
) external;

// Check transfer approvals
function isValid(uint256 tokenId) external view returns (bool);
```

C5: ERC-4671 Token Lifecycle

```
// Create and transfer token to an EOA address
function give(
    address to,
    bytes calldata metadata,
    bytes calldata signature
) external returns (uint256);

// Create and transfer token to caller
function take(
    address from,
    bytes calldata metadata,
    bytes calldata signature
) external returns (uint256);

// Disassociate token from account
function unequip(uint256 tokenId) external;
```

C6: ERC-4973 Item Management

```
// Accept the locking of the token
function lockApprove(address to, uint256 tokenId) external;

// Lock the token for a fixed time period
function lock(uint256 tokenId, uint256 expired) external;

// Allowed operators can terminate locking earlier
function unlock(uint256 tokenId) external;
```

C7: ERC-5058 Time Locking

```
// Transfer locked to a byte entity instead of an address
event Transfer(
    bytes32 indexed _from,
    bytes32 indexed _to,
    uint256 indexed _id);
```

C8: ERC-5107 ENS Binding

```
// Reference the parent NFT address and its ID during mint time
event Mint(
    uint256 indexed badgeId,
    address indexed nftAddress,
    uint256 indexed nftTokenId);

// Check parent NFT ownership of the badge
function ownerOf(uint256 badgeId) external view returns (
    address nftAddress,
    uint256 nftTokenId);
```

C9: ERC-5114 Badge NFT Binding

```
// Emit events based on locking and unlocking occasions
event Locked(uint256 tokenId);
event Unlocked(uint256 tokenId);

// Check the status of the asset
function locked(uint256 tokenId) external view returns (bool);
```

C10: ERC-5192 Minimal Locking

```
// Set of addresses to remove the bound asset
enum BurnAuth { IssuerOnly, OwnerOnly, Both, Neither }

// Include permissions within on-chain event
event Issued (
    address indexed from,
    address indexed to,
    uint256 indexed tokenId,
    BurnAuth burnAuth);
```

C11: ERC-5484 Consensual Creation and Removal

```
// Transfer token to several addresses
event TransferMulti(
    address indexed operator,
    address indexed from,
    address[] to,
    uint256 amount,
    uint256 id);

// Interact with pending asset
function claimOrReject(
    address account,
    uint256 id,
    bool action
) external;
```

C12: ERC-5516 Multi-Address Claims

```
// Binding indicator for one token ID in collection
function isSoulbound(uint256 id) external view returns (bool);
```

C13: ERC-5633 Individual ID Binds

```
// Issue a token in a specified slot to an owner.
function issue(
    address to,
    uint256 tokenId,
    uint256 slot,
    BurnAuth burnAuth,
    address verifier,
    bytes calldata data
) external payable;

// Issue credit to a token ID
function issue(
     uint256 tokenId,
     uint256 amount,
     bytes calldata data
) external payable;

// Revoke credit from a token ID
function revoke(
    uint256 tokenId,
    uint256 amount,
    bytes calldata data
) external payable;

// Revoke a token from its owner
function revoke(
    uint256 tokenId,
    bytes calldata data
) external payable;

// Check if a token is valid
function verify(
    uint256 tokenId,
    bytes calldata data
) external returns (bool);
```

C14: ERC-5727 Issuer Verifier Split

```
// Lock asset with ability for single unlocker
function lock(address unlocker, uint256 id) external;

// Unlocking needs permissions
function unlock(uint256 id) external;
```

C15: ERC-5753 Simple Locker Restriction

```
// Add or update guard of the NFT
function changeGuard(
     uint256 tokenId,
     address newGuard,
     uint64 expires
) external;

// Transfer token and remove guard afterward
function transferAndRemove(
     address from,
     address to,
     uint256 tokenId
) external;
```

C16: ERC-6147 Temporary Guards

```
// Check single token lock
function locked(uint256 _id) external view returns (bool);

// Check token batch status
function lockedBatch(uint256[] _ids) external view returns (bool);
```

C17: ERC-6268 Minimal Multi-Type Locking

```
// Check mint, burn, or transfer for specific addresses
function isTransferable(
     uint256 tokenId,
     address from,
     address to
) external view returns (bool);
```

C18: ERC-6454 Address-Based Restriction

```
// Create registry account and bind it to a token address
function createAccount(
    address implementation,
    uint256 chainId,
    address tokenContract,
    uint256 tokenId,
    uint256 salt,
    bytes calldata initData
) external returns (address);

// Check if contract account can execute token actions
function isValidSigner(
     address signer,
     bytes calldata context
) external view returns (bytes4 magicValue);

// Return token-specific account from registry
function account(
    address implementation,
    uint256 chainId,
    address tokenContract,
    uint256 tokenId,
    uint256 salt
) external view returns (address);

// Return account-specific token from registry
function token() external view returns (
        uint256 chainId,
        address tokenContract,
        uint256 tokenId);
```

C19: ERC-6551 Token Bound Account

```
// Transfer rules for a token
struct TransferConditions {
    uint256 amount;
    uint256 time;
    address to;
    bool allFunds;
}

// Approval conditions for a token
struct ApprovalConditions {
    uint256 time;
    uint256 numberOfTransfers;
}

// Add binding restriction to a token
function addBindings(
    address _keyWallet1,
    address _keyWallet2
) external returns (bool);

// Reset binding restrictions for a token
function resetBindings() external returns (bool);

// Approve a specific token transfer
function allowTransfer(
    uint256 _amount,
    uint256 _time,
    address _to,
    bool _allFunds
) external returns (bool);
```

C20: ERC-6808 & ERC-6809 Security Locking

```
// Enumeration to define Burn Authorization
enum Authorization {
     NONE,
     OWNER,
     ISSUER,
     ASSET,
     OWNER_AND_ISSUER,
     OWNER_AND_ASSET,
     ASSET_AND_ISSUER,
     ALL}

// Change owner to comply with real world ownership
function transferAnchor(
     bytes memory attestation,
     bytes memory data
) external;

// Approve an attestation of an anchor
function approveAnchor(
    bytes memory attestation,
    bytes memory data
) external;

// Burn the token mapped to the anchor
function burnAnchor(
    bytes memory attestation,
    bytes memory data
) external;
```

C21: ERC-6956 Digital Twin Binding

```
// Initial and global lock
function defaultLocked() external view returns (bool);

// Individual token lock
function locked(uint256 tokenId) external view returns (bool);
```

C22: ERC-6982 Optimized Default Assignment

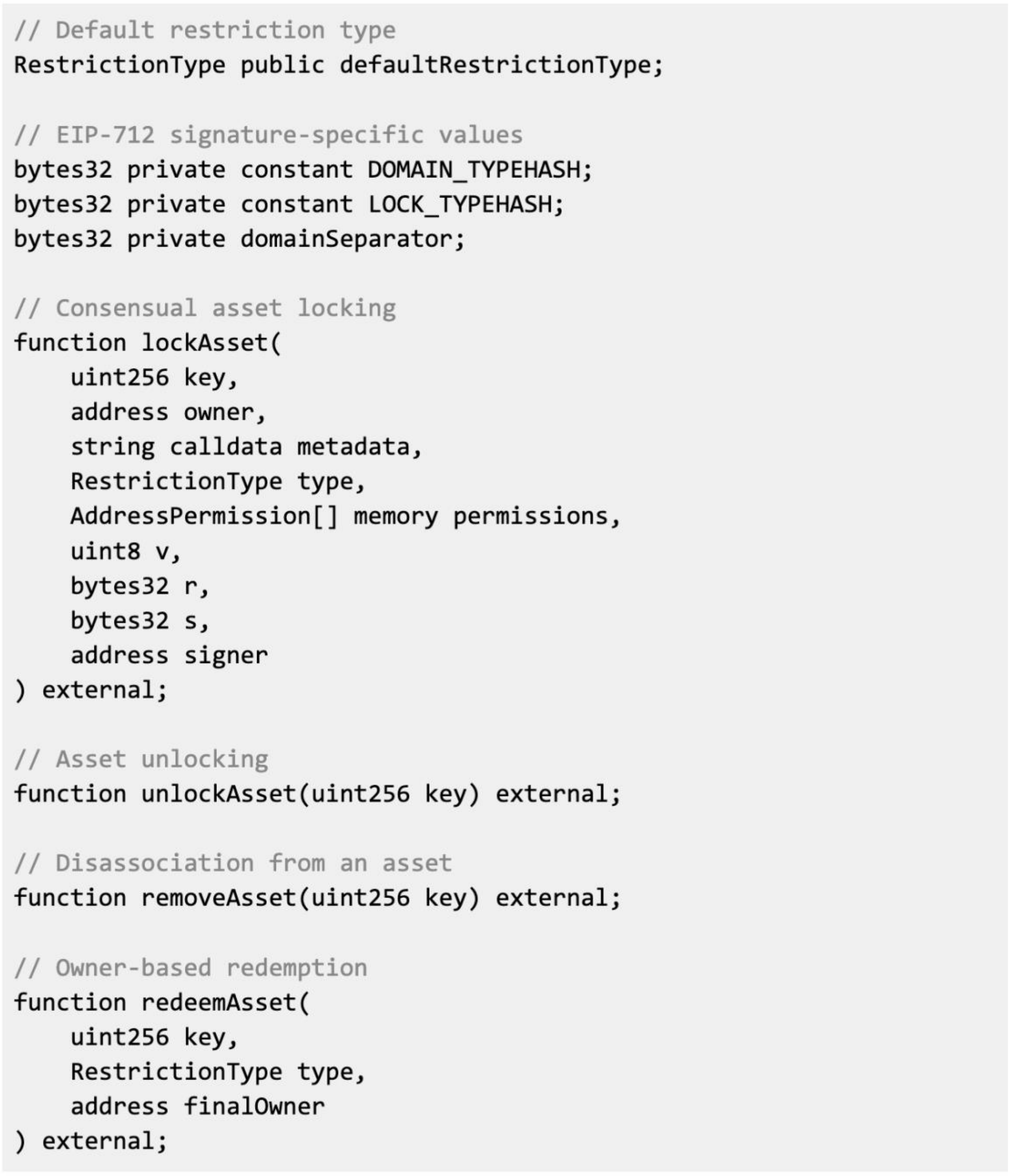
```
// Lock asset to address
function lock(uint256 tokenId, address _locker) external;

// Unlock asset from set address
function unlock(uint256 tokenId) external;

// Combined transfer and locking
function transferAndLock(
    uint256 tokenId,
    address from, address to,
    bool setApprove
) external;

// Check unlocker of assets
function lockerOf(uint256 tokenId) external view returns (address);
```

C23: ERC-7066 Single Hybrid Locking

Appendix, Part D: LSP22 Implementation

```
// Open asset restriction types
enum RestrictionType { None, TempLock, SoftLock, HardLock }

// Various transfer permissions
enum RestrictionPermission { None, CanRemove, IsOperator }

// Extended lock information
struct LockInfo {
    bool isLocked;
    address executedBy;
    uint lockedAt;
}

// Permission for each account
struct AddressPermission {
    address addr;
    RestrictionPermission permission;
}

// Type allocation for each asset or owner
mapping(uint256 => RestrictionType) public tokenRestriction;

// Role allocation for each asset or owner
mapping(uint256 => mapping(
    address => RestrictionPermission)
) public tokenPermissions;

// Retrievable status for each asset or owner
mapping(uint256 => LockInfo) public lockStatus;

// Signature nonce management
mapping(address => uint256) public nonces;
```

D1: LSP22 Enumerables and Mappings



```
// Default restriction type
RestrictionType public defaultRestrictionType;

// EIP-712 signature-specific values
bytes32 private constant DOMAIN_TYPEHASH;
bytes32 private constant LOCK_TYPEHASH;
bytes32 private domainSeparator;

// Consensual asset locking
function lockAsset(
    uint256 key,
    address owner,
    string calldata metadata,
    RestrictionType type,
    AddressPermission[] memory permissions,
    uint8 v,
    bytes32 r,
    bytes32 s,
    address signer
) external;

// Asset unlocking
function unlockAsset(uint256 key) external;

// Disassociation from an asset
function removeAsset(uint256 key) external;

// Owner-based redemption
function redeemAsset(
    uint256 key,
    RestrictionType type,
    address finalOwner
) external;
```

D2: LSP22 Variables and Functions

```
//// LSP22 contract identification
bytes32 private constant DOMAIN_TYPEHASH = keccak256("EIP712Domain(
        string name,
        string version,
        uint256 chainId,
        address contract
)");

// LSP22 signed data structure
bytes32 private constant LOCK_TYPEHASH = keccak256("lockAsset(
        address owner,
        string metadata,
        RestrictionType type,
        AddressPermission[] permissions,
        address signer,
        uint256 nonce
)");

// Signature hash value
bytes32 private domainSeparator;

// Signature nonce management
mapping(address => uint256) public nonces;

// Signature hash initialization
constructor() {
    domainSeparator = keccak256(abi.encode(
        DOMAIN_TYPEHASH,
        keccak256(bytes("<LockableAsset>")),
        keccak256(bytes("<1>")),
        block.chainid,
        address(this)
    ));
 ...
}
```

D3: LSP22 Data Signing

```
// Consensual asset locking
function lockAsset( ... ) external {
    // Hash the data structure
    bytes32 digest = keccak256(abi.encodePacked(
        "\x19\x01",
        domainSeparator,
        keccak256(abi.encode(
            LOCK_TYPEHASH,
            owner,
            metadata,
            type,
            permissions,
            signer nonces[signer]++
        ))
    ));

    // Get the operator address based on hash and signature
    address recovered_signer = ecrecover(digest, v, r, s);

    // Retrieve the key permission of the signer
    bytes32 lsp6key = keccak256(
        abi.encodePacked("0x4b80742de2bf866c29110000",
        recovered_signer));

    // Instantiate the LSP6 key manager
    ownerContract = IExternalContract(owner);

    // Retrive the permissions of the key
    bytes32 permissions = ownerContract.getData(lsp6key);

    // Validate the signer permission for the owner's address
    bytes32 signingThreshold = uint256(0x0...0200000);
    bytes32 signingPermission = uint256(permissions);
    require(signingPermission < signingThreshold, "Invalid signer");

    ...
}
```

D4: LSP22 Lock Signature Validation

Declaration of Authorship

I have prepared this thesis independently and only using the literature and resources indicated. Passages quoted literally or, in essence, from sources are marked as such.

This work has not been submitted to any other examination authority in the same or similar form.

Chemnitz, 27.10.2023 [redacted]

Location, Date Felix Hildebrandt, B.Sc.