NISTIR 8301

Blockchain Networks: *Token Design and Management Overview*

Loïc Lesavre
Priam Varin
Dylan Yaga

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8301

National Institute of Standards and Technology
U.S. Department of Commerce

NISTIR 8301

Blockchain Networks: *Token Design and Management Overview*

Loïc Lesavre
Priam Varin
Dylan Yaga
Computer Security Division
Information Technology Laboratory

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8301

February 2021

National Institute of Standards and Technology Interagency or Internal Report 8301
84 pages (February 2021)

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8301

Comments on this publication may be submitted to:

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: blockchain-token-paper@nist.gov

Reports on Computer Systems Technology

Abstract

Blockchain technology has enabled a new software paradigm for managing digital ownership in partial- or zero-trust environments. It uses tokens to conduct transactions, exchange verifiable data, and achieve coordination across organizations and on the web. Fundamental to this representation is that users can independently control token custody in digital wallets through public-key cryptography and interact with one another in a peer-to-peer manner. Blockchain networks provide secure transaction reconciliation, linkage, and storage in consolidated, integrity-protected distributed ledgers forming mutually operated record-keeping execution environments. Data models with varied capabilities and scopes have been defined to issue tokens, which additional protocols can help manage while enabling separation of concerns. Security and recovery mechanisms allow users to set up self-hosted, externally hosted, and hybrid account custody models. Scaling schemes have been developed to accommodate transactions off-chain with deferred on-chain settlement, as well as deposit contracts with built-in, self-enforceable conditions to exchange tokens without intermediaries, transaction submission rules to fit in with different deployment scenarios, and privacy-enhancing techniques to protect user confidentiality. Software design patterns and infrastructure tools can also make it easier to integrate blockchain networks, wallets, and external resources in user interfaces. This document provides a high-level technical overview and conceptual framework of token designs and management methods. It is built around five views: the token view, wallet view, transaction view, user interface view, and protocol view. The purpose is to lower the barriers to study, prototype, and integrate token-related standards and protocols by helping readers understand the building blocks involved both on-chain and off-chain.

Keywords

blockchain; cryptoasset; cryptocurrency; data portability; decentralized governance; digital asset; digital token; distributed ledger; fintech; off-chain scaling; self-hosting; smart contract; state channel; tokenization; transaction confidentiality; verifiable data; wallet; zero-knowledge proof.

Acknowledgments

The authors wish to thank all contributors to this publication and their colleagues who reviewed this report's drafts and contributed technical and editorial additions. This includes NIST staff Michael Davidson, Frédéric de Vaulx, Katya Delak, Aurélien Delaitre, Eric Trapnell, Mark Trapnell, and James Dray. Additional thanks to all the people and organizations who submitted comments during the public comment period. In particular, the authors would like to thank William Entriken.

Audience

This publication is designed for readers with prior knowledge of blockchain technology, including consensus models, smart contract development, and related cryptographic primitives, who wish to learn more about blockchain-based tokens and management methods that help support them. Readers who have little or no knowledge of blockchain technology and wish to understand the fundamentals are invited to read National Institute of Standards and Technology Internal Report (NISTIR) 8202, *Blockchain Technology Overview* [1]. Additionally, some general knowledge in web application architectures, financial technology, and identity management systems can make the paper easier to read. This publication is not intended to be a technical guide but to provide a conceptual understanding of the technology.

Executive Summary

Traditional data and operations management across organizations and on the web can involve inefficient transaction reconciliation between siloed databases, password fatigue, and single points of failure. This can lead to massive data leaks and abusive data collection for users and businesses.

Blockchain technology has enabled a new software paradigm for managing digital ownership in partial- or zero-trust environments. It uses tokens to conduct transactions, exchange verifiable data, and achieve coordination across organizations and on the web. Fundamental to this representation is that users can independently control token custody in digital wallets through public-key cryptography and interact with one another in a peer-to-peer manner. Blockchain networks provide secure transaction reconciliation, linkage, and storage in consolidated, integrity-protected distributed ledgers. They form mutually operated record-keeping execution environments, or virtual machines, for smart contracts, which are built with application-specific instruction sets or general-purpose programming languages.

These environments make it possible to issue programmable digital assets or tokens, the ownership of which is cryptographically verifiable, and to develop services to help manage them. Tokens are either native to a blockchain protocol or deployed on top of an existing blockchain protocol via user-generated logic at the smart contract layer. Fungible tokens are meant to be completely interchangeable, serving as digital coins and enabling payment systems. They are primarily used to build incentive and governance models for permissionless peer-to-peer networks, represent existing fungible assets, or derive financial instruments. On the other hand, tokens associated with unique identifiers are meant to uniquely identify things or data. They can be implemented on-chain (i.e., nonfungible tokens) or as self-contained tokens that use blockchain-based storage for status updates. Self-contained tokens enable authentication and authorization methods that can provide additional features for blockchain-based tokens or help build identity or supply chain management systems. These two types of uniquely identifiable tokens can either be native to a blockchain-based protocol or represent existing assets (e.g., physical objects).

Open standards for token data models have been developed that specify operations at the protocol level for token creation and supply/lifecycle management and at the account level for individual token transfers. These models have different capabilities and scopes, which additional token management protocols can complement while allowing for separation of concerns.

Users can securely store the private keys associated with the accounts that hold their tokens in their own wallets or entrust key storage to third-party custodians independent of token issuers. Smart contract vaults can serve as proxies and enable tailored account management models with additional security and recovery features while externally maintaining persistent blockchain addresses.

Operations modify the ledger's state by way of transactions submitted to the blockchain, which provides reconciliation but requires making trade-offs between decentralization, scalability, and security. Parallel transaction processing and off-chain scaling schemes have been developed to increase transaction throughput. State channels, commit-chains, and rollups allow transaction processing to be offloaded away from the root blockchain without relinquishing token custody. By attaching agreed-upon and self-enforceable conditions to deposit contracts, tokens can be

exchanged with one another while users still remain in control of the private keys at all times. Blockchain bridging schemes allow token and data portability across blockchains as well as hub-and-spoke architectures using different types of intermediary systems. Permissions and viewability restrictions may be put into place to help build narrowly defined environments, though privacy-enhancing technologies or separate private transaction execution are still needed to protect the confidentiality of user data.

Additionally, software design patterns and infrastructure tools make it easier to integrate blockchain networks, wallets, and external resources (e.g., user account data, external data feeds) with user interfaces and facilitate token interactions. The unbundling of user interfaces, application data, and logic results in a user-centric system architecture and requires re-examining approaches to break down and evaluate the security risks entailed by individual configurations.

While token-based protocols can integrate and transform existing organizations and web services with efficiency and interoperability gains, the parties involved must establish common purposes and rules to form secure and sustainable governance models. More generally, blockchain networks face multi-dimensional challenges that range from scalability and privacy obstacles to educational and regulatory needs (e.g., understanding of cryptoeconomics, assimilation of legal infrastructures) as well as standard- and product-related requirements (e.g., data format interoperability). The literature that has emerged on these challenges is rich, and substantial efforts are being made to address them publicly and across organizations.

In that way, blockchain-enabled tokens can be integrated into web and mobile applications to provide different types of embedded services, especially related to finance, identity, authentication, payments, and supply chains. A key driver is that tokens can act as tools with built-in usage and governance features to facilitate business-making online with increased efficiency and transparency, benefiting both users and businesses.

This document provides a high-level technical overview and conceptual framework of token designs and management methods. It is built around five views: the token view, wallet view, transaction view, user interface view, and protocol view. The purpose is to lower the barriers to study, prototype, and integrate token-related standards and protocols by helping readers understand the building blocks involved both on-chain and off-chain.

Table of Contents

List of Appendices

List of Figures

List of Tables

1 Introduction

Traditional data and operations management across organizations and on the web can involve inefficient transaction reconciliation between siloed databases, password fatigue, and single points of failure. This can lead to massive data leaks and abusive data collection for users and businesses.

Blockchain technology has enabled a new software paradigm for managing digital ownership in partial- or zero-trust environments. It uses tokens to conduct transactions, exchange verifiable data, and achieve coordination across organizations and on the web. Fundamental to this representation is that users can independently control token custody in digital wallets through public-key cryptography and interact with one another in a peer-to-peer manner. Blockchain networks provide secure transaction reconciliation, linkage, and storage in consolidated, integrity-protected distributed ledgers. They form mutually operated record-keeping execution environments or virtual machines. Combined with off-chain resources, blockchain-based tokens can allow for faster and cheaper transaction settlement while bolstering user-centric ownership models and interoperable data representations. Blockchain networks and the tokens that they form or support are also interchangeably referred to as *cryptonetworks*.

1.1 Background

Bitcoin and Ethereum introduced the technologies of blockchains and smart contracts as well as new types of global, web-native social constructs with decentralized governance. Anyone with an internet connection can view the blockchain, operate a publishing node, and submit transactions. Blockchain addresses are derived from public keys generated directly by the users who control the associated private keys' custody. Transactions are signed using these private keys before being validated and reconciled by the blockchain nodes, providing data integrity and public verifiability.

More generally, the domain of permissionless blockchains brought about protocol-native tokens, or *cryptocurrencies*. With the potential to provide alternatives to existing institutions and market discipline due to their decentralized and non-sovereign nature, cryptocurrencies could have long-term implications for financial inclusion and stability [2]. Moreover, permissionless blockchains brought about tamper-evident and tamper-resistant computing platforms that maintain a common state for all participants (*Base Protocol Layer* in Table 1). Those platforms allow *smart contract* execution with application-specific instruction sets, such as Bitcoin Script [3], or general-purpose execution environments, such as the Ethereum Virtual Machine [4]. Smart contracts make it possible to integrate off-chain schemes (*Second Layer Execution* in Table 1), deploy tokens, and more generally build blockchain applications (*Smart Contract Layer* in Table 1). Tokens represent digital assets and serve as instruments for exchanging verifiable data. *Fungible tokens* are meant to be completely interchangeable, acting as digital coins and enabling payment systems. When they are native to a protocol and used to decentralize its governance, they are also largely but inconsistently referred to as *platform* or *utility tokens*. Tokens associated with unique identifiers, *nonfungible tokens* and *stateful tokens*, are meant to identify things or data uniquely. They can be part of wider supply chain or traceability frameworks. Blockchain-based services have emerged to help manage account custody and individual token ownership as well as to increase transaction throughput, protect user privacy, and provide infrastructure tools. As shown in Table 1, these tokens and services are integrated into user interfaces at the *Application Layer* via middleware and/or off-chain schemes.

Table 1: Emerging Blockchain Computer Stack

Application Layer	User Interfaces	
Integration Layer	Middleware	***Second Layer Execution*** Off-Chain Schemes
Blockchain Layer	***Smart Contract Layer*** Custom Bytecodes (Compiled Smart Contracts)	
	Base Protocol Layer[1] Execution Environment	
	Consensus Service	Global State Storage
Network Layer	Peer-to-Peer Communication	
Physical Layer	Node Hardware	

Consortium blockchains have been derived from these technologies and support similar token data models and emerging stack but do not share the same type of social construct described above for permissionless blockchains. Instead, they aim to build systems that integrate and transform existing organizations by mutualizing data and operations management infrastructures. The scope of consortium blockchain networks is narrowly defined through granular access control systems for submitting transactions and/or publishing new blocks. This provides scalability gains and the ability for different types of governance models, user privacy frameworks, and data integrity levels to be developed according to consortium-specific policies.

By promoting open standards for token design and management with peer-to-peer user interactions, blockchain networks could serve as foundational infrastructures for *open finance/banking* and *user-centric identity*. These two notions are also referred to as *decentralized finance* and *self-sovereign identity*, especially when protocols are meant to work without any accounts being given special privileges, though public bootstrapping periods may occur.

1.2 Purpose and Scope

This document provides a high-level technical overview and conceptual framework of token designs and management methods. It is built around five views: the token view, wallet view, transaction view, user interface view, and protocol view.

The purpose is to lower the barriers to study, prototype, and integrate token-related standards and protocols by helping readers understand the building blocks both on-chain and off-chain. This publication assists with the characterization of token-related developments to fill some of the knowledge gaps between the various technologies that are being built and their intended roles. Note that this paper is not meant to provide any regulatory consideration or financial advice.

[1] Blockchain protocols perform multiple functions, primarily consensus, compute, and storage. The protocol components enabling these functions may be decoupled, with distinct roles and tasks assigned to different groups of nodes or subnetworks. For example, in Hyperledger Fabric transaction execution takes place before transaction ordering and consensus, handled by different nodes.

First, the paper looks at different types of tokens and blockchain implementations before discussing the fundamentals of how tokens are held in custody and owned. Then, it examines how transactions are validated, submitted, and viewed with blockchain networks serving as transaction reconciliation providers complemented by off-chain scaling and privacy schemes. Finally, it presents infrastructure components before concluding with deployment scenarios and use cases for all of the types of tokens discussed in the paper to help readers further understand their reach.

1.3 Disclaimers and Clarifications

Although *blockchains*, *smart contracts*, and related concepts and technologies are referred to and examined throughout the paper, no recommendation or endorsement of any particular protocol is provided. Any products or protocols mentioned are for explanatory purposes only and do not imply any endorsement or suitability for use. How these mentions are distributed across platforms or ecosystems, or the absence of mentions, does not imply any preference or disapproval for use. Furthermore, this work may be extended to other types of distributed ledger technologies (DLT) and databases. For the purpose of this paper, smart contracts are by and large assumed to be deployed on top of account-based blockchains with general-purpose smart contract capabilities, but alternative techniques and cryptographic schemes may be used instead on a case-by-case basis and the level of programming language expressiveness actually required may vary. In general, blockchain technologies come with their own sets of trade-offs and maturity levels. They are evolving to cope with the various challenges that stem from their cross-domain or public deployment purposes. This is a rich, multidisciplinary, and emerging domain with many approaches being studied and experimented with. This paper does not attempt to provide in-depth coverage of all developments, answer all questions, or judge between the different techniques, architectures, and models. It instead seeks to describe key concepts, explore fundamental issues, and highlight important differences to support its previously stated non-normative purpose. Regulatory compliance is also out of scope for this paper.

Notes on Terms: For the purpose of this paper, cryptographic *digital tokens* (or *cryptoassets*) will be referred to as *tokens*, with *tokenization* designating the concept of representing assets as tokens using blockchain networks. Certain terms used in this paper do not have a fixed and well-established meaning. While all terms used in this paper aim to adequately characterize the concepts and technologies discussed here, certain semantics may likely require further scrutiny.

Notes on References: References include scientific papers, academic research, standard proposals, along with source-code repositories, white papers, or other project resources that are available publicly due to the open-source and community-led nature of most blockchain- and token-related developments. Documentations and discussion threads within public source-code repositories can provide readers with additional information regarding certain technologies, open protocols, or open standards, though at different levels of maturity.

1.4 Results of the Public Comment Period

This document has seen revisions in response to the comments received during the public comment period that took place from September 29, 2020 to October 30, 2020. The names of six sections were changed: Sections 2.1.3, 4.3.3, 5.1.3, 5.1.3.1, 5.1.3.2, and 5.1.3.3. Section 4.1.1.4 was removed and discussions where the term "sidechain" appeared were refined with more specific vocabulary. The figure in Section 2.1.1 was also removed. Other clarifications, add-ons, and/or corrections were incorporated throughout the document.

1.5 Document Structure

The rest of this document is organized as follows:

- **Section 2** categorizes tokens to help make sense of where blockchain-based tokens fit within the wider landscape of digital tokens.
- **Section 3** discusses several options and considerations for managing wallets, public-private key pairs, and accounts.
- **Section 4** discusses schemes to execute transactions off-chain and across blockchains, exchange tokens, and manage transaction submission and viewability.
- **Section 5** introduces software design patterns and infrastructure tools to integrate blockchain networks, wallets, and off-chain resources in user interfaces before providing high-level architectural considerations.
- **Section 6** provides deployment scenarios and use cases for tokens before presenting potential breakthroughs for privacy-preserving verifiable data exchange.
- **Section 7** is the conclusion.
- **Appendix A** provides a high-level overview of the different types of consensus services and computing environments that blockchain protocols provide.
- **Appendix B** provides a list of acronyms and abbreviations used in the document.
- **Appendix C** contains a glossary for selected terms used in the document.

2 Token Categorization

Digital tokens serve as instruments that enable users to exchange verifiable data in different ways. Blockchain-based tokens allow programmable representations of digital assets. Self-contained tokens permit fixed representations of digital documents or certificates. This section presents these two general categories of tokens and discusses the data models to support them.

2.1 Blockchain-Based Tokens

This section examines token data models, protocol-level operations, and account-level operations.

2.1.1 Token Data Models

At their core, tokens are entries in distributed ledgers that are assigned to blockchain accounts and for which transactions require authorization, ensuring authenticity and preventing modification and tampering without consent. Once authorized by the associated accounts, token transactions are validated, encapsulated, and published to the ledger into blocks by blockchain nodes. In general-purpose smart contract platforms, accounts are either internal to the blockchain (i.e., smart contracts) or external and owned through public-key cryptography (i.e., end user-controlled). In the latter case, authorizations are provided by signing the blockchain transactions with the owner's private key(s), which are generally held in custody in digital wallets (see Section 3).

Blockchains keep records of token transactions according to two models: the *unspent transaction output-based* (UTXO) model and the *account-based* model. Furthermore, tokens are either native to a blockchain protocol (e.g., used to incentivize publishing full nodes) or custom and deployed on top of an existing blockchain protocol via user-generated logic at the smart contract layer. Table 2 summarizes the four resulting token representation types.

Table 2: Token Representation Types

	Blockchain-Native (Base Layer)	**On Top of an Existing Blockchain (Smart Contract Layer)**
UTXO-Based	System account balances are encoded as the sums of unspent transaction outputs of past transactions. Spending a token results in new, unspent transaction outputs. For example, bitcoin is Bitcoin protocol's native token.	A separate protocol, sometimes called *colored coin* method, encodes custom account balances or unique identifiers into extra metadata included in unspent transaction outputs of past transactions.
Account-Based	Variables in the blockchain's global state store system account balances assigned to blockchain addresses. For example, ether is Ethereum protocol's native token.	Variables in the blockchain's global state store custom account balances or unique identifiers assigned to blockchain addresses either centrally, within *token factory contracts*, or at the account level (i.e., data values and code are decoupled).

There are two types of blockchain-based tokens to represent existing assets or create new natively digital assets: fungible tokens and nonfungible tokens. Blockchain-native tokens are meant to be fungible, while tokens deployed on top of existing blockchains are meant to be either fungible or nonfungible. In both cases, they enable granular representations of provable digital ownership claims. In general-purpose smart contract platforms, issuers are externally owned accounts or other contracts. Some tokens have known issuers while others do not or have relinquished ownership.

Fungible Tokens:

A *fungible token* is a data representation or abstraction that assigns interchangeable units, or account balances, to blockchain addresses with programmable supply management. Token units are meant to serve as *digital coins*. Once in circulation, transferring token units means removing or debiting funds from the sending account balance and adding or crediting them to the receiving account balance (see Section 2.1.3). Token units cannot be copied or accidentally removed from a given account. This concept of resource-boundedness is enforced by the blockchain's transaction processing capabilities, which aim to prevent double-spending without a trusted intermediary. Note that token units may be rendered nonfungible if they can be uniquely identified by analyzing the transfer history. Data structures for fungible tokens may include fields for protocol-specific metadata, such as the token name, symbol, issuer address, total supply, and precision decimals.

Fungible tokens can represent both new and existing interchangeable assets (or bundles of assets) as well as derivatives. Their value can be intended to be intrinsic and floating, allowing for designing protocol-specific economic games or voting rights to decentralize protocol governance. For example, some community-controlled networks use an incentive token to distribute rewards to users when they contribute to the protocol or follow certain rules or behaviors. On the other hand, when redeemable for, pegged to, or derived from underlying assets, the value of fungible tokens is intended to be extrinsic. These main motives for using fungible tokens are further discussed in Section 6, *Deployment Scenarios and Use Cases*.

In general-purpose smart contract platforms, open standards that provide interfaces for token factory contracts are often followed, such as those introduced as Ethereum Requests for Comments (ERCs). They provide function and event declarations, constraints and requirements, and sometimes references to implementations. ERC-20 [5] is currently the most commonly used de facto standard, while ERC-777 [6] allows passing arbitrary data in token transfers to trigger external function calls. The draft ERC-1410 [7] and ERC-1404 [8] proposals support compliance requirements (e.g., withdrawal restrictions). Data models for fungible tokens may also be supported natively as part of the programing language, as in Move's *resource types* [9] where the constraints (or protections) for guaranteeing resource-boundedness are enforced statically by Move's type system. These constraints can thus be shared for all fungible tokens on the network, while additional constraints may be placed by token issuers on a case-by-case basis.

Nonfungible Tokens:

A *nonfungible token* is a data representation or abstraction that assigns uniquely identified and uniformly formatted qualitative data objects to blockchain addresses with programmable lifecycle management. Token objects are meant to serve as *unique digital assets*. Once in circulation, transferring a nonfungible token object means reassigning the owner's account (see Section 2.1.3). Like fungible tokens, the underlying blockchain ensures that nonfungible tokens cannot be duplicated and are bound to exactly one owner at any given point in time. Token objects may be accompanied by off-chain metadata, the integrity of which can be verified through on-chain cryptographic hashes (see Section 5.3). Data structures for nonfungible tokens may include fields for protocol-specific metadata, either fixed or dynamic (e.g., linked to on-chain data). Nonfungible tokens are often colloquially referred to by their acronym, NFTs. Motives for using them are further discussed in Section 6.3, *Tokenizing Uniquely Identifiable Things and Supply Chains*.

In general-purpose smart contract platforms, open standards that provide interfaces for token factory contracts are often followed, such as ERC-721 [10]. ERC-1155 [11] grants rights for both fungible and nonfungible tokens from the same interface, also referred to as a "hybrid" token interface, and is intended to help streamline implementations when the tokens are closely related. Like fungible tokens, core elements of data models for nonfungible tokens may also be directly built into the programing language.

Emerging Taxonomies and Frameworks:

The Token Taxonomy Initiative (TTI) has published a draft framework called the *Token Taxonomy Framework* [12]. It characterizes tokens using base types, behaviors, and property sets, which serve as the bases to generate common, blockchain-agnostic specifications [13]. The Token Taxonomy Framework was absorbed by the InterWork Alliance [14] and is meant to be developed further and complemented by a contractual definition framework, called the InterWork Framework, and an analysis definition framework, called the Analytics Framework. *To Token or not to Token: Tools for Understanding Blockchain Tokens* [15] provides another token taxonomy based on four attribute types: purpose, governance, functional, and technical. The reader may find relevant information about token regulation—which is out of scope for this paper—in *Considerations and Guidelines for Securities and Non-Securities Tokens* [16], published by the Token Alliance of the Chamber of Digital Commerce.

2.1.2 Protocol Management

Depending on the token representation type used, token operations are implemented as natively present functions in blockchain protocols and token factory contracts or built as separate protocols. These operations can be distinguished between *protocol-level operations*, discussed in this section, and *account-level operations* for individual tokens, discussed in Section 2.1.3. Note that for nonfungible tokens, more advanced account-level operations vary depending on use cases.

Supply/Lifecycle Management:

A *mint* operation creates and distributes new token units or objects. Once assigned to blockchain accounts, they become available for circulation. Each use case or specific implementation has its own token distribution and supply or lifecycle management model, which can be created as a new independent system (see Section 6.1 and *Freestanding Tokens* in Section 6.3) or built upon existing data on-chain or off-chain, via oracles (see Section 6.2 and *Companion Tokens* and *Redeemable Tokens* in Section 6.3). These models may imply having an initial supply that can be empty (i.e., all the tokens in circulation will be distributed through the mint operation) or pre-set (i.e., tokens are pre-allocated to particular accounts at launch). The supply can also be flexible or fixed (i.e., there is no mint operation). Finally, for fungible tokens, sets of conditions part of the distribution model can result in a supply that has a pre-set maximum or that is uncapped.

Minting, if there is any, can be conducted individually or in batch and according to two approaches:

- **Push-Based**: For fungible tokens, the mint operation increases account balances (i.e., credits account balances without debit in return). The total token supply variable, if there is any, must also be increased. For nonfungible tokens, the mint operation instantiates and assigns new token objects with unique identifiers to the receiving accounts. In general, push-based minting does not involve any action or approval from the receiving accounts.
- **Pull-Based**: The mint operation gives individual accounts minting rights purely on-chain or generates off-chain "mint request" tokens, giving minting rights on-chain to those who hold them (see *Authorization Methods* in Section 2.2, *Self-Contained Tokens*). This can provide scalability gains as issuers can give the right to claim tokens to multiple accounts in a single transaction, which can keep that right until it is exercised. For example, consider periodic fungible token minting (e.g., interests, dividends). An account could skip claiming tokens between period X and period X + Y before withdrawing all of the tokens earned between periods X and X + Y + 1 in a single transaction at period X + Y + 1.

A *burn* operation intentionally destroys existing token units or objects as part of protocol-defined conditions (e.g., when redeemed for the underlying assets, tokens are taken out of circulation) or transaction fees (e.g., to prevent denial-of-service attacks), as discussed in Section 6.1.

Update (or *amend*) and *revoke* operations for nonfungible tokens are specific to use cases.

Protocol Restrictions:

Partially decentralized or centralized governance models may involve restrictions on account-level operations (see Section 4.2.2). With additional protocol governance ramifications, a *pausable* operation may also exist that privileged accounts can call to enforce an emergency shutdown to disable transfers or other account-level operations temporarily. It is usually meant to provide a backstop for handling severe stress scenarios, sometimes while mitigation mechanisms are being implemented as part of a protocol governance model upgrade (i.e., progressive decentralization). Transparency requirements, protocol-defined rules, and multi-signature schemes can be used to narrowly define and limit the scope and capabilities of trusted intermediaries, if there are any, as well as to provide public auditability.

2.1.3 Account-Level Operations

Transfer and delegation are base operations implemented into token data models (see Section 2.1.1). More advanced account-level operations can be built as separate protocols, though the exact scope of token data models varies. These include token exchange (see Section 4.1.2), lending and borrowing (see Section 6.2.2), and fundraising and derivatives (see Section 6.2.3).

Transfer:

Token transfers can be achieved without any permission being required or with protocol-defined rules and allowed lists that restrict the pool of eligible receiving addresses (see Section 4.2.2). For example, it can be possible to have third parties place transfer restrictions or holds. Sending fungible tokens to other blockchain addresses (i.e., push payments) allows one-time payments, recurring payments, or streams/continuous payments (e.g., subscriptions). Manually transferring tokens to well-known burner addresses, which are computationally near impossible for anyone to own (e.g., the address "0"), is equivalent to renouncing or destroying the ownership of the tokens in a verifiable way as the transaction provides proof of ownership renunciation. Tokens can also be transferred to smart contracts for collateralization and staking with redemption value (see Section 3.4), though the smart contracts may need to implement functions that allow the tokens to be transferred out or else they may be permanently locked up (this may be seen as another way to renounce ownership). The reduction in circulating supply is known to other participants only if they know of the renunciation methods used and query the blockchain accordingly. When the same token factory contract issues multiple tokens, it may be possible to transfer them in a single transaction.

Delegation:

Single-use, conditioned, or permanent authorizations can be granted by an account owner to delegate access of certain account-level operations to third parties and on a per-token basis. For example, the *allowance*, *approve*, and *transferFrom* operations in the ERC-20 standard make it possible to request payments, share accounts, and transfer tokens out of smart contracts as discussed earlier. Note that authorizations may be implemented as part of smart contract vaults, as discussed in Section 3.4.

2.2 Self-Contained Tokens

A self-contained token is a data object that can be digitally signed and encrypted using a cryptographic secret or a public-private key pair. The most common format followed is the JavaScript Object Notation (JSON) Web Token format (often referred to by the acronym, JWT), standardized under Request for Comments (RFC) 7519 [17].

In particular, self-contained tokens that are signed using a public-private key pair provide a self-contained way to exchange verifiable certificates or documents containing fixed qualitative data. Thus, they can be used for authentications and authorizations. However, they cannot serve as digital coins without a mechanism to prevent double-spending. Blockchain technology provides this mechanism. The self-contained tokens can then be effectively viewed as signed transactions for spending blockchain-based fungible tokens, as discussed in the previous section.

There are two categories of self-contained tokens:

- **Stateless Tokens**: Stateless tokens do not involve any external system. They generally do not fit well with long-lived authentication or authorization methods since they cannot be updated or revoked. Thus, they are often used for short-lived verifiable data exchange.
- **Stateful Tokens**: Stateful tokens are meant to be used jointly with an external data store for status querying. Unlike stateless tokens, they generally fit well with long-lived authentication or authorization methods since their status can be updated or revoked.

Stateful tokens are of particular interest in this paper. By building their data stores upon blockchain networks, it is possible to implement authentication methods that inherit some of their security and governance properties, both as part of general-purpose, off-chain messaging or document exchange and for allowing additional blockchain-based token management schemes.

Authentication Methods:

Stateful tokens allow lightweight blockchain-based authentication methods to be built. The blockchain is essential but thinly used since only reading access is required for status verification.

Blockchain-based data stores for status querying can have multiple architectures (see Section 4.4 in [18]). They can be built as smart contract registries that are user-controlled, issuer- or consortium-controlled, or ownerless. These differ from nonfungible token factory contracts in that token ownership is not meant to be reassigned. Alternatively, blockchain-based data stores can be built on UTXO-based blockchains. Finally, the querying logic may involve additional components, such as status update batching protocols[2] or cryptographic accumulators.[3] The *verifiable credential* standard [19] published by the World Wide Web Consortium (W3C) specifies data field formats that provide the location and querying logic of data stores for stateful token status.

Additionally, blockchain-based identifier systems can make it easier to resolve and authenticate the digital signatures of cryptographically signed content. In particular, they can be used to identify the owners and issuers of stateful tokens (and any other entities whose public key is present in the token). These systems may follow the *decentralized identifier* (DID) standard [20] (see Section 5.1.3.2). Their data stores can have blockchain-based architectures similar to the ones discussed above (further examined in [18]), which may be complemented by additional smart contracts for public credentials registries, such as those introduced in the draft ERC-780 [21] proposal.

Authorization Methods:

Stateful tokens make it possible to preauthorize accounts to submit transactions. They can serve as vouchers that give an account the right to transfer blockchain-based token units or objects that belong to another account or to mint new ones. For example, this makes it possible to create "checks" (similar to paper checks) to withdraw funds from another account if funds are available. Payment channels (see Section 4.1.1.1) build on that by introducing an on-chain collateral with off-chain messages that give authorizations to withdraw from that collateral. Stateful tokens can also represent mint requests redeemable for blockchain-based tokens, which can be sent to users before they create a wallet, as in [22] for nonfungible tokens using Merkle proofs.

[2] Second layer protocols can be used to batch status updates and, thus, increase scalability (as in ION and Element for blockchain-based identifiers) using the SideTree protocol [23].

[3] Cryptographic accumulators can be used to prove whether the unique identifier of a given stateful token is included in a registry without revealing other entries of that registry [24], allowing for confidential status querying.

3 Wallet and Key Management

This section discusses token custody, which means the access to the accounts that tokens are assigned to, themselves secured by one or more private keys when the accounts are owned externally. It is of extreme importance to secure these private keys since if one has access to that account, one can prove ownership of the associated tokens and sign transactions that affect them (e.g., transfers). Private keys are owned and used by entities, such as individuals, organizations, devices, and processes.

Credential management and custody have traditionally been performed by institutions and organizations who own or operate services on behalf of users (e.g., banks manage their accounts, web service providers manage their user credentials) without always providing them with options or explanations about how (or by whom) their credentials were managed. However, users can now independently manage their own private keys used to transact on blockchain and DLT systems. They can choose to secure and store their private keys themselves or utilize a third-party custodian to manage private keys on their behalf. Furthermore, account custody models have been developed that provide users with abstractions of key management without relinquishing control.

International Organization for Standardization (ISO) / Technical Committee (TC) 307's *ISO 22739:2020 Blockchain and distributed ledger technologies - Vocabulary* [25] defines a wallet as "an application used to generate, manage, store or use private and public keys," which "can be implemented as a software or hardware module." They are often categorized in the two types below, which are usually intended to be used as shown in Figure 1:

- *Hot Wallet*: A wallet that is connected to the internet and intended to be highly accessible (containing low-value/in-transit tokens).
- *Cold Wallet*: A wallet that is not connected to the internet (e.g., generated on an air-gapped, general-purpose computer or special-purpose hardware wallet) and intended to be highly secure (containing high-value/at-rest tokens). Physical human intervention or authentication is required to sign transactions, such as pushing a button, entering a local pin, or scanning a quick response (QR) code. It is sometimes used jointly with a *proxy/warm wallet* to further secure withdrawals (e.g., through time delays, multi-signature, amount limits implemented in a smart contract, as well as admin and firewall restrictions).

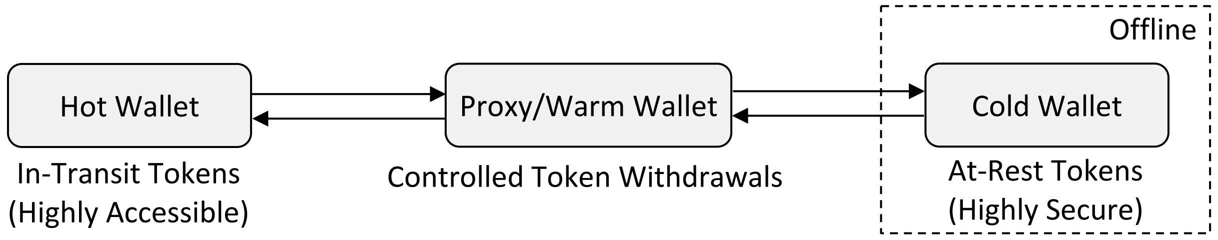
Offline
Hot Wallet
Proxy/Warm Wallet
Cold Wallet
In-Transit Tokens (Highly Accessible)
Controlled Token Withdrawals
At-Rest Tokens (Highly Secure)

Figure 1: State-Dependent Storage Methods

Traditional key management systems, alongside external modules for transaction signing and submission to the blockchain, may serve as wallets to avoid scattering private keys across systems.

3.1 Self-Hosted Wallets

In a fully user-controlled, self-hosted wallet scenario, key generation and management, including secure storage, backup, and restore functions, as well as transaction reviewing and signing lie completely with the user, locally. A popular phrase with users who run their own self-hosted wallets is "not your keys, not your coins." These wallets are also called *non-custodial wallets*. This scenario has the benefit of allowing the user to provide as much (or as little) security as they desire and of having their consent required by default before transaction execution. However, it has the drawback that the user is completely responsible for their keys. If proper systems are not in place for key backup and restore, the loss of a private key results in the loss of all associated tokens. The user must also review the details of every blockchain transaction to avoid irreversibly signing and submitting fraudulent or undesired transactions. Some self-hosted wallets are specialized for a particular type of blockchain protocol or deployment, while others can work with multiple blockchains. They may also integrate with multiple types of tokens and second layer protocols.

Software Wallets:

Users can choose to store their keys in software wallets for signing transactions securely. Software wallets are applications (e.g., a browser, a messaging application) or built-in operating system functionalities that provide secure storage for private keys and any data potentially associated with the tokens. Transactions may be initiated from software wallets or separate applications, on the same devices or distinct ones. For example, a smartphone application may be used to sign transactions initiated on a website accessed with a separate computer. Depending on what hardware the software wallet is being run on, it may also utilize security features present in hardware, such as a *hardware security module* (HSM) or a *trusted execution environment* (TEE)[4] (also called a *secure enclave*), to provide enhanced security. They can be built with open-source firmware code to make it easier to verify how private keys are generated. A software wallet can also support direct, ad-hoc communication and messaging protocols (depending on its hardware integration), allowing device-to-device authentication and exchange of value (see *Device-to-Device Communications* below). Additional in-wallet management features may be implemented, such as to organize accounts, examine transaction histories, or help users verify transaction details during transaction reviewing.

Dedicated Hardware Wallets:

Users can choose to store their keys in dedicated hardware wallets that are distinct from their primary devices. Hardware wallets are devices, such as small USB-based devices or smart cards, that store private keys in a secure enclave and do not allow them to be exported. Hardware wallets provide functions to allow the use of the private key without ever revealing the private key to applications. This prevents malware from attempting to steal the private key while still allowing a user to sign transactions or other message types. Although they are called hardware wallets, companion software is necessary to use them.

[4] A TEE is an isolated processing and memory enclave that is only accessible through restricted application programming interfaces (APIs) (i.e., it cannot be accessed by the operating process or any user process). It has a built-in private key that remains unknown to the device's owner and is used to decrypt data in the TEE.

There exists a variety of integrity mechanisms, such as secure inputs and secure display (i.e., "what you see is what you sign"). Validating the integrity of the device itself is essential (e.g., through verifying the digital signatures of the suppliers and security updates). Some hardware wallets come with two-factor authentication (2FA), biometrics authentication, and companion applications.

Device-to-Device Communications:

Device-to-device (D2D) communications are used to exchange data with wallets or devices that belong to other entities and to synchronize one's own wallets or devices.

Commonly used communication protocols range from wireless channels, such as Bluetooth, near-field communication (NFC), Wi-Fi, 5G, to physical media that require in-person operation, such as a USB connection or QR codes. D2D communications for local synchronization include mutually authenticating wallets on two different mobile devices via NFC and setting up a wireless local area network (WLAN) server to securely keep tokens up to date between an owner's devices (i.e., synchronizing the private keys of newly acquired tokens with a lightweight, self-hosted infrastructure). Since they are converted to human-readable data when scanned, QR codes make it easier to conduct air-gapped transmissions of verifiable data over devices (e.g., through JSON web tokens). Additionally, QR codes allow a printed medium, such as a piece of paper, to be converted to a digital medium, such as a newly minted token.

Multi-layered communication specifications and frameworks have been developed to provide interoperable and agnostic data transport methods for blockchains networks. For example, DIDComm [26] and DID Auth [27] use DIDs to exchange verifiable, machine-readable messages and handle authentication processes.

3.2 Custodial Wallets

Account custody and, therefore, the custody of the associated tokens can be delegated to institutional third-party custodians who hold and safeguard private keys on behalf of users. They provide different degrees of custody services and risk management. Custodial wallets are also known as *externally hosted wallets* or *managed wallets*.

Partial/Collaborative Custody:

Users can partially entrust third-party custodians to hold and safeguard their accounts by having only select private keys placed under their control as part of multi-signature security models (as discussed in Section 3.3). As those private keys alone are not sufficient by themselves to sign transactions on behalf of said accounts, this can provide users some balance between account control and recoverability. For example, consider an account management model in which two out of three keys are necessary for account recovery. A user can choose to store two keys themselves—for instance one on their phone and another in a bank vault—and appoint a third-party custodian to store the remaining key. It is also possible to rely on a network of nodes tasked with generating and storing private keys on behalf of users, as in DirectAuth [28].

Full Custody:

Users can fully entrust third-party custodians to hold and safeguard their accounts by having all private keys placed under their control. Alternatively, users may not have accounts on the blockchain, with token ownership and transactions instead recorded in book-keeping ledgers owned by the third-party custodians. With full custody, account security and recoverability are entirely managed by the provider.

By providing security, internal record-keeping/settlement, and gateway services for tokens that represent financial assets, the role of those custodians resembles that of a bank offering retail or wholesale services. This is why third-party custodians have historically been subject to know-your-customer and anti-money laundering laws and often require significantly more identity proofing than other private key custody methods. For users who value privacy and anonymity (or pseudonymity), the required amount of identifying information may be too much and may discourage them from utilizing such services. Additionally, since the third-party custodian holds the private keys, any data breach that occurs may result in the loss of user tokens. Third-party custodians, however, also have similarities to email providers, which most people choose to use for convenience and the quality of service provided by domain experts rather than hosting their own email servers. Similarly, protocol complexities can be abstracted away for users who interact with underlying blockchain-based services through managed services.

3.3 Account Origination and Recovery

Wallets allow users to create externally owned accounts by generating public-private key pairs. Optionally, accounts can be registered on a domain naming service to enable discovery and resolution through a more human-readable username. Wallets also offer methods for securing and recovering these key pairs in case of loss.

Key Generation:

With a *hierarchical deterministic* (HD) wallet, specified under Bitcoin Improvement Protocol BIP-32 [29], an unlimited number of key pairs can be generated from a single seed, in the same wallet. This enables users to maintain special-purpose identifiers decoupled from their primary identifier, providing some anonymity without the potential hassle of managing multiple key pairs separately. *Pairwise-pseudonymous identifiers* allow each and every relationship between users and third parties to have its own unique identifier. *Single-use identifiers* are immediately discarded after being used by the subject for a given transaction with a relying party.

Some wallets can generate key pairs for *stealth addresses*. A stealth address is an account that gives its owner the ability to compute the private keys of proxy accounts used by other entities to send payments to the stealth address owner. Key generation follows the elliptic-curve Diffie-Hellman protocol, wherein interested parties first generate a cryptographic nonce (sent to the stealth address owner) before creating a public key and an account that they can send tokens to. The stealth address owner then computes the private keys of these identifiers to retrieve the tokens. The stealth address itself does not receive transactions, making payments unlinkable by third parties. The concept of stealth addresses was introduced in Bitcoin development discussions [30].

Finally, *multi-signature wallets* distribute the key generation and signature processes among a set of participants, avoiding the need to rely on a single private key. Their function is similar to that of a lockbox with multiple keys: one cannot access the lockbox without the necessary keys. With a multi-signature wallet, multiple parties—each with their own public-private key pair—jointly produce a signature. One cannot access the tokens and submit transactions without the requisite number of signatures from an m-of-n quorum of private keys. Multi-signature wallets can rely on threshold cryptography[5] [31] to compute the aggregate signature.

Key Recovery:

Traditional means of private key backup for tokens involve mechanisms such as the generation and storage of seed words or seed phrases. This results in a mechanism by which anyone able to find the seed words or phrases can restore the associated tokens to the device of their choice. For example, all accounts managed in an HD wallet can be recovered at once by using the recovery phrase set during wallet creation. However, both paper and digital backups of the seed phrases can be lost, stolen, or destroyed.

Multi-signature wallets can also restore access to tokens when losing one or more private keys as long as access remains to the requisite number of private keys necessary to transfer the tokens. Moreover, if a wallet supports secret sharing and one or more shares of a secret key become compromised, new random shares may be computed—a process known as *resharing*. Resharing occurs either in response to a compromised share or periodically.

A wallet can implement a *dead man's switch* to recover a private key; after a certain period without activity, the wallet distributes a private key to select entities. This places a burden on the owner of the private key to continuously reset the timer with activity. Otherwise, the private key or tokens will automatically be sent to their specified entities.

Domain Naming Services:

Domain naming services make it possible to choose and register unique domain names that owners can link to blockchain accounts or other information. Domain names can be represented by nonfungible tokens, as discussed in Section 6.3.

3.4 Smart Contract Vaults

Smart contracts are accounts that are internal to the blockchain and can serve as programmable vaults to receive token deposits with built-in rules and automated agreements, allowing on-chain custody or reserve. Smart contract vaults are also interchangeably called deposit, escrow, multi-signature contracts, or simply smart contracts. Note that, as discussed in Section 2.1.3, to hold tokens in custody, smart contracts must implement the necessary logic for handling the corresponding types of tokens.

[5] Cryptographic schemes where a threshold of secret shares of data—distributed across a set of participants—must be computed together to produce a meaningful result [32][33]. Some threshold signature schemes do not reveal which individual entities participated in the signature process.

Smart Contract Wallets:

For individual users, smart contract vaults or wallets can make it easier to develop tailored account management models with additional security and recovery features while externally maintaining persistent identifiers, for interactions with other accounts. They act as proxies allowing for separation of concerns with respect to regular user-controlled wallets holding private keys, which, however, may still be subject to transaction submission fees or permissioning schemes depending on the base layer protocol. These account management models can be implemented directly as features within smart contract vaults or through a collection of authorized modules. They can permit multi-signature schemes and security and recoverability rules, such as security periods, thresholds requirements, and emergency modes. In delegated account recovery models, the wallet owner defines a set of recovery addresses, the private keys of which can be held in custody across the owner's devices, by trusted intermediaries, or social connections. Notably, such account recovery models do not require the use of seed phrases or full account custody by a third party. Wallet applications may abstract away features supported by smart contract vaults by deploying the smart contracts internally on behalf of users when creating new accounts.

Since smart contracts can hold tokens on behalf of users, schemes can be built that allow tokens to be sent to users even before they create a wallet. Instead, a smart contract is deployed with a mechanism that allows it to be claimed later by the new user via a known trusted attribute (e.g., phone number).

Joint or "Treasury" Vaults:

Groups of users or organizations can share the ownership and management of an account and the tokens it contains by using a multi-signature or "treasury" smart contract vault that enables trustees to propose token withdrawals and requires a certain number of them to approve each proposal before conducting a withdrawal. Such joint vaults may also implement additional agreed-upon clauses.

Protocol Collaterals/Deposit Vaults:

Protocols can use smart contract vaults (or deposit contracts) to receive tokens and condition their release. Token collateralization is used to secure individual transactions, such as atomic swaps (see Section 4.1.2.1), through self-enforceable rules. On a wider scale, token collateralization—or *staking*—is also used to build *cryptoeconomic incentives* for community-controlled networks, either at the base layer or at the smart contract layer. When staked, tokens can earn yields or provide benefits and privileges within the protocol (see Section 6.1). In proof-of-stake consensus models (see Appendix A), staking serves as the basis for coordinating participants and supporting blockchain network operations (e.g., rewards for participants staking tokens). Certain staking models involve a period during which tokens are locked up (i.e., they cannot be transferred) and subject to penalties, acting as programmable security deposits. On-chain collaterals are also used to enable financial instruments (see Section 6.2), such as for lending and borrowing. The rules that condition the release of smart contract collateral funds can be based on internal blockchain data or external data sources (see Section 5.4). Collateralized token owners can build proofs based on the collateral to authenticate with third parties (see Section 6.4).

Smart contracts can record deposits as either an "internal" non-transferable balance or as new units of a separate token (or *wrapper*) that represent transferable withdrawal rights (i.e., "tokenized" collateral). Since there may be regulatory implications for users, protocols may let users opt in on whether to allow transfers of the withdrawal rights or not. A proxy smart contract may also allow adding, verifiably suspending, or conditioning the ability to transfer collateral withdrawal right tokens or giving a third party the ability to place a hold without modifying the issuing protocols.

Key collateralization schemes, particularly useful for securing individual transactions, include cryptographic timelocks to place time conditions and hashlocks to require the knowledge of a secret. A *hashed timelock contract* (HTLC) is a smart contract that implements these two techniques, allowing for deposit contracts to be built with vesting periods and conditioned refunds.

4 Transaction Management

This section discusses how transactions for protocol-level and account-level operations are managed by blockchain and second layer protocols. It breaks down the analysis into three aspects: transaction validation, transaction submission, and transaction viewability. The building blocks examined allow different trade-offs between decentralization, scalability, and security.

4.1 Transaction Validation

Transaction validation can take place on-chain, off-chain, or across different blockchains:

- *On-chain transactions* are settled and stored in the blockchain's global state for tamper-evident and tamper-resistant record-keeping at the base layer. The blockchain provides reconciliation through its consensus service and cryptographic linking of blocks replicated across the network, forming a distributed ledger. On-chain transactions may be processed in parallel with sharding. See Appendix A for more information.
- *Off-chain transactions* act as "bar tab" record-keeping, deferring settlement on the root blockchain via *state updates*. The root blockchain enables anchoring and settlement at the base layer for off-chain scaling and privacy schemes so that the amount of data stored on the root blockchain is minimized. Developers may have varying needs for off-chain scaling and privacy depending on the performance and properties at the base layer. Note that the same schemes may provide both some privacy and scalability, though schemes may also be designed specifically for one or the other in priority. Off-chain scaling schemes are discussed in Section 4.1.1, and off-chain privacy schemes, in Section 4.3.3.
- *Cross-chain transactions* allow updates for global states from two or more distinct blockchain networks in concert, enabling token and data portability across them. Transactions may take place across different types of blockchain protocols.

The following subsections discuss schemes to record and execute transactions off-chain, exchange tokens, and represent tokens on other blockchains through bridging schemes. These schemes may be used to build hub-and-spoke architectures composed of multiple blockchains, such as sharding and architectures where general-purpose and application-specific blockchains complement each other or span across permissionless and permissioned environments. A blockchain connected to another blockchain through a two-way bridge is sometimes called *sidechain*. Note that schemes for off-chain and cross-chain transactions, as well as the standards and tooling to support them, are actively being researched and developed, representing open research challenges.

4.1.1 Off-Chain Scaling

This section presents schemes to enable faster and cheaper transactions through secure off-chain processing without relinquishing token custody. They aim to improve scalability and help minimize on-chain transactions, allowing transactions such as micropayments that would otherwise be too expensive or too slow. Scaling schemes do not require modifying the base layer protocol, though the level of programming language expressiveness needed varies. They have primarily been studied for deployment on top of permissionless blockchains, which often prioritize decentralization over scalability.

Some scaling schemes trigger periodic updates on the root blockchain. Others rely on users initiating the updates themselves, verifying the correctness of on-chain transactions, and having the ability to dispute fraudulent transactions. In both cases, deferred on-chain validation of these updates is necessary to prevent transactions from being reverted. Although, for that reason, the prospect of transaction finality at the base layer is essential, scaling schemes are designed to offer cryptographic and game-theoretic guarantees with their own degree or type of transaction finality. These guarantees depend on the types of off-chain structures, state update methods, on-chain security deposits or collaterals, and dispute resolution mechanisms used. Scaling schemes may be operated by either the users themselves or external entities that are intended to be non-custodial but may be trusted for liveness (e.g., when they are centrally administered). In addition, some schemes require monitoring the root blockchain for fraud prevention and have data availability constraints (when transaction data is stored by the users themselves or by external storage providers). Other schemes are based on more advanced cryptographic primitives, such as zero-knowledge proofs (ZKP). Table 3 compares these aspects for the different scaling schemes presented in this section.

Table 3: Off-Chain Scaling Schemes Comparison

	State/Payment Channels	**Plasma**	**ZK-Rollups**	**Optimistic Rollups**
Structure	Group of users	Commit-chain operator(s)	Rollup operator(s)	Rollup operator(s)
Transaction History	Stored by user or external provider	Stored by user or external provider	"Compressed" on-chain	"Compressed" on-chain
State Update Content	Most recent state	Hash of the most recent state	Hash of the most recent state and "compressed" transaction data	Hash of the most recent state and "compressed" transaction data
Disputes	Requires on-chain monitoring	Requires on-chain monitoring	Prevented by validity proofs	Requires on-chain monitoring

4.1.1.1 State/Payment Channels

A *state channel* is a scheme that enables a group of participants to independently authorize and record transactions off-chain that are fully backed with on-chain collateral. This system architecture allows any participant in the group to push state updates on-chain, at their own initiative. It is intended to provide strong transaction finality guarantees, instant at the second layer, without external operators. A *payment channel* is a state channel specialized for sending payments.

Three phases occur during a state channel's lifecycle [34] (though implementations vary):

1. **Establishment**: A channel is established after participants agree to lock up a portion of the blockchain's current state. For example, a fixed number of tokens are locked up by a set of participants in a smart contract, as discussed in Section 3.4. This locked up state, also known as *state deposit* (or security deposit), may only be released once unanimous agreement is reached among the channel participants. Multi-signature smart contracts are used for holding the state deposit among channel participants.
2. **Transitions**: Once the state deposit is locked up, channel partners can begin sending off-chain transactions to one another, which must be approved and signed by all channel partners.
3. **Closure (or Dispute)**: After an arbitrary number of transactions have taken place off-chain and all participants have reached a channel closure agreement (i.e., signed the latest transaction), the new state is pushed on-chain, and the channel is terminated. If channel partners are non-responsive, a participant can always claim their funds on-chain, subject to a waiting period. A separate dispute mechanism exists such that if a channel partner submits an outdated or incorrect state on the root blockchain, honest participants can prove this and re-claim their funds. To do so, they must submit a challenge on-chain with the latest state and go through the dispute resolution mechanism, which redistributes the channel's collateral accordingly.

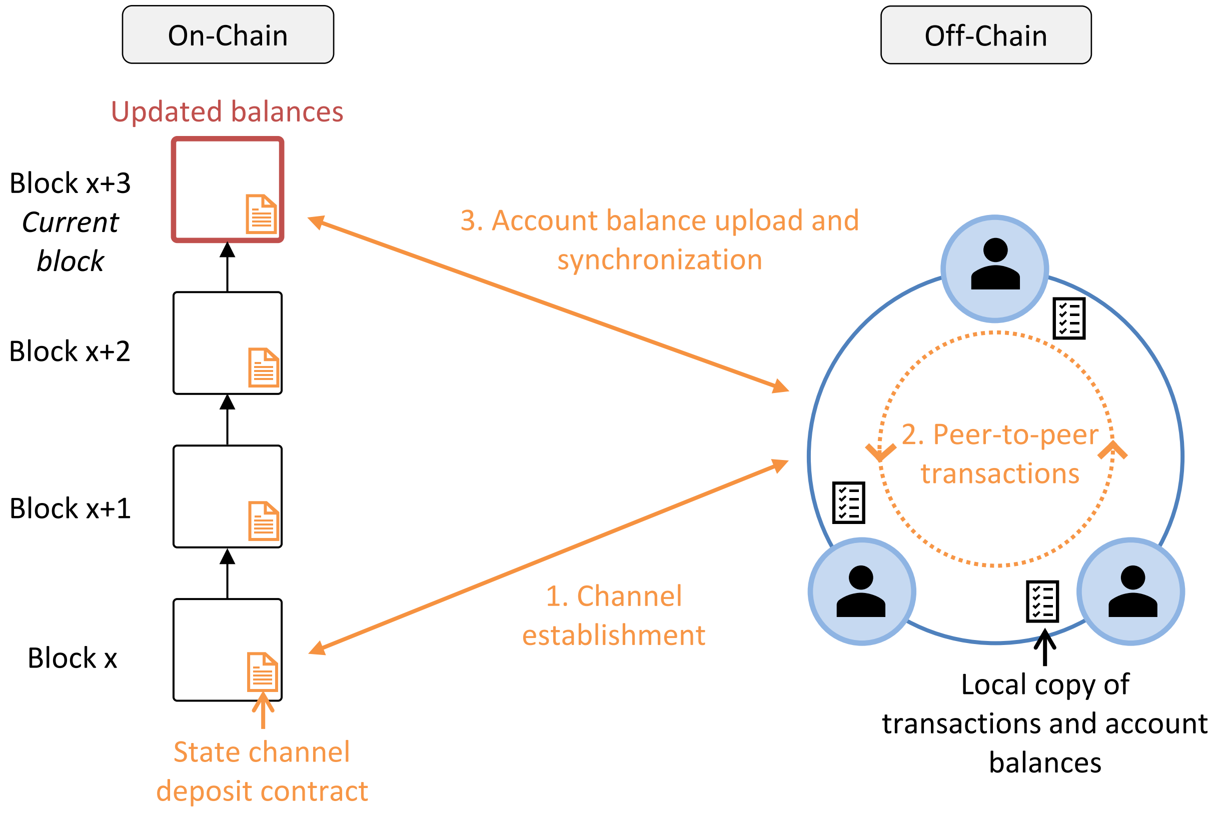
On-Chain
Off-Chain
Updated balances
Block x+3
Current block
Block x+2
Block x+1
Block x
State channel deposit contract
3. Account balance upload and synchronization
2. Peer-to-peer transactions
1. Channel establishment
Local copy of transactions and account balances

Figure 2: Payment Channel Phases

Figure 2 shows that only the establishment and closure (or dispute) phases require transactions on the root blockchain. State channel transactions take place directly between channel participants, with only final account balances being broadcast on-chain publicly. However, participants must remain connected to one another to approve transactions and maintain access to the root blockchain to verify that the closure state published did not exclude any transactions. Additionally, state channels involve defined sets of participants since all channel participants' addresses must be registered on-chain during Phase 1 prior to transacting in the state channel.

4.1.1.2 Payment Channel Networks

Cryptographic hashlocks, timelocks, and multi-signature schemes allow a set of payment channels to be combined into a network. These techniques are used to create off-chain *payment paths* for multi-hop payments, wherein a single payment operation affects balances from multiple channels. The network relies on a set of intermediary nodes, also referred to as *payment channel hubs*, that provide access to channels, incentivized by a transaction fee in permissionless systems. This access permits participants that do not directly share a payment channel to transact with one another and avoid the cost of setting up new channels. A transaction routing algorithm determines the particular sets of intermediary nodes used to relay transactions from the sender to the receiver (i.e., payment paths). They can be characterized by their effectiveness (i.e., maximizing payment success probability), efficiency (i.e., ensuring low computational overhead for path discovery), cost (i.e., transaction fees for the paths used), scalability, and privacy implications [34].

An example of a payment channel network on top of the public Bitcoin network is the Lightning Network [35]. Transactions across two different blockchains using the Lightning network can be possible if the blockchains share the same hash function and if it is possible to create timelocks (e.g., Bitcoin and Litecoin). Another example is Raiden [36], on top of the public Ethereum network, which uses smart contracts to enable off-chain transactions for the protocol-native token and ERC-20 tokens. In both of these protocols, pre-computed transaction routes can use an onion routing communication protocol,[6] preventing intermediary nodes from reading payment destinations. Note that some approaches do not require intermediary nodes to have access to the target destinations of the transfers they relay.

Schemes to synchronize payments across channels using TEEs hardware-enforced consensus rules (i.e., hardcoded rules that cannot be reprogrammed or tampered with) have also been developed, eliminating the need for on-chain collaterals and dispute periods after channel closure [37].

4.1.1.3 Commit-Chains

A commit-chain [38] is a scheme that enables one or more operators to collect transactions from users, add them into state updates (e.g., cryptographically linked blocks), and periodically push commitments for those state updates onto the root blockchain, usually as Merkle root hashes. Prior to sending payments using the commit-chain, users must deposit funds in its smart contract vault on the root blockchain. Commit-chain operator(s) may do so as well to increase transaction finality depending on the collateral and dispute resolution mechanism (e.g., to enable instant payments). Users do not need to remain online to receive payments, but secure storage of their transaction

[6] Onion routing encapsulates a message in multiple layers of encryption; these layers are gradually decrypted by the different nodes routing the message.

history is necessary to be able to retrieve their tokens independently. To do so, users must generate Merkle proofs from their transaction history and submit them to the commit-chain's smart contract vault on the root blockchain. The proofs may also be used to report dishonest operators and invalid transactions, in which case the collateral is redistributed according to the rules implemented by the dispute resolution mechanism. In comparison, state channels have higher bootstrapping costs and deposit requirements but do not require on-chain transactions during normal use. Both commit-chains and state channels have data availability constraints since transaction data, which is stored by users or external storage providers, must remain accessible for generating fraud proofs. Commit-chains can take the form of block-producing ledgers centrally operated by a single entity or a group via multi-signature, or blockchains. In both cases, they remain non-custodial, with funds secured on the root blockchain.

Plasma [39] is an example of protocol to create commit-chains called *plasma-chains* on top of the public Ethereum network. Different protocol variants have been developed, such as Plasma Cash, Plasma Debit, and Minimum Viable Plasma. For each token deposit on a Plasma Cash chain [40], a new nonfungible token is issued using a *sparse Merkle tree* (a Merkle tree wherein data is indexed). The tree is divided into slots that store a fixed token amount and the owner's public key. Every transaction in that slot updates the public key associated with the slot. This allows Plasma Cash participants to build proofs of non-spending for a given nonfungible token. This indexed data structure also enables token holders to only store a copy of their own transaction history instead of a copy of the whole plasma-chain. Building on Plasma Cash, Plasma Debit chains [41] create payment channels between users and operators when nonfungible tokens are issued to new users. Finally, the Minimal Viable Plasma [42] protocol follows the UTXO model and is focused on payments. In all three variants, users need to periodically download the plasma-chain to verify its integrity and provide withdrawal proofs.

4.1.1.4 Rollups

Like commit-chains, *rollups* [43] are schemes that batch transactions off-chain and periodically push state update commitments onto the root blockchain, but they do so along with "compressed" per-transaction data. This data allows the rollup's current state to be retrieved from the root blockchain, independently of the rollup. Note that it is stored in *calldata* storage in Ethereum-based networks. This system architecture eliminates the data availability constraints found in state channels and commit-chains, and thus enables more general-purpose applications, though it may not provide the same degree of scalability. Two rollup types have been proposed with different approaches to ensure transaction correctness in each state update and are discussed below.

ZK-Rollup:

ZK-rollups are rollups where a zero-knowledge proof is attached to each state update commitment to ensure the state update's correctness. An incorrect state update will by design not be processed by the rollup smart contract on the root blockchain. This eliminates the need for invalid state update monitoring as found in the other scaling schemes described in this paper. On the other hand, zero-knowledge proofs can be cumbersome and computationally expensive to produce (though, once computed, their size generally does not depend on the number of transactions contained in the state update). Setting up a ZK-rollup may require an initial trusted setup.

Optimistic Rollup:

Unlike ZK-rollups, state update commitments in optimistic rollups are tentatively accepted, meaning they are assumed to be valid by default without providing a validity proof or requiring an initial trusted setup. However, if a user or operator notices that an incorrect state update commitment (e.g., one that reports invalid transactions) has been published on-chain, they can produce a challenge by posting the valid state and a Merkle proof. This defers transaction finality at the base layer as a prolonged fraud-proof challenge period may be necessary for each state update, but eliminates the computational limitations found in ZK-rollups.

4.1.2 Token Exchange

Tokens can be exchanged without the need for intermediaries in an operation referred to as *atomic swap*. It comprises a set of instructions for executing a token exchange with only two possible results: either the swap succeeds, and all participants receive the desired tokens, or the swap fails, and the state remains at its starting point [44]. Atomic swaps can occur directly when the parties involved already know each other and agree on the exchange. Otherwise, one must use an exchange, providing price discovery. Techniques allow the building of non-custodial exchanges, some of which are based on off-chain order messaging and relaying.

4.1.2.1 Atomic Swaps

Atomic swaps allow two participants to exchange tokens directly, in a non-custodial manner. Each participant deploys a smart contract vault wherein they transfer the tokens they would like to exchange. A set of rules is used to specify the exchange's terms, such as initial deposit requirements and expiration times. If the terms are followed, each participant obtains the ability to withdraw the tokens present in the other participant's smart contract vault. Those rules are implemented in the smart contract vaults themselves or in a separate orchestration contract.

Cryptographic hashlocks and timelocks are some of the most prominent primitives to conduct atomic swaps. A simple structure for this involves deploying two HTLCs (see Section 3.4), one for each participant in the exchange, Alice and Bob. The process is illustrated in Figure 3:

1. After creating a secret s, Alice publishes an HTLC with hashlock $h=hash(s)$ and timelock t on the blockchain. Alice then transfers to it the tokens that she intends to exchange with Bob under the condition that they will be transferred to Bob only if he sends a transaction containing the secret s prior to t expiring. If no transaction is sent and t expires, they will be transferred back to Alice.

2. Once aware that Alice's HTLC has been published on the blockchain, Bob publishes another HTLC on the blockchain with a timelock $t' < t$, the same hashlock h, and reciprocal token collateralization conditions. Bob then transfers to it the tokens that he intends to exchange with Alice under the opposite conditions as in the previous step.

3. Then, for Alice to receive Bob's tokens, Alice sends a transaction containing the secret s to the HTLC that Bob published before t' expires.

4. Bob is hence made aware of s and can, in turn, send a transaction to the HTLC that Alice published to receive Alice's tokens before t expires.

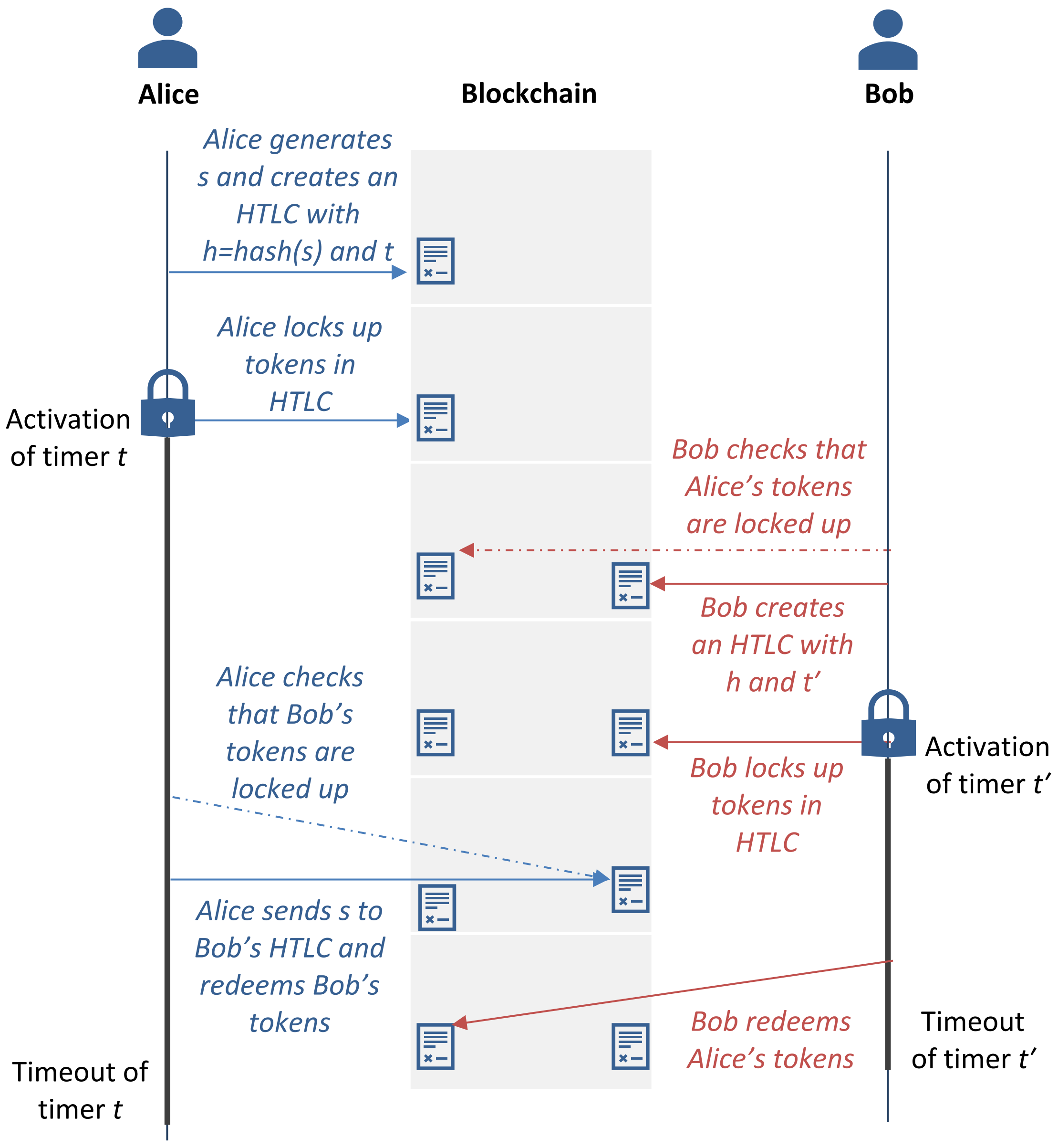
Alice
Blockchain
Bob
Alice generates s and creates an HTLC with h=hash(s) and t
Alice locks up tokens in HTLC
Activation of timer t
Bob checks that Alice's tokens are locked up
Bob creates an HTLC with h and t'
Alice checks that Bob's tokens are locked up
Bob locks up tokens in HTLC
Activation of timer t'
Alice sends s to Bob's HTLC and redeems Bob's tokens
Bob redeems Alice's tokens
Timeout of timer t'
Timeout of timer t

Figure 3: Hashed Timelock Contract Transfer Flow

For Bob to have enough time to receive Alice's tokens, it is preferable to set the condition $t > t'$ with a reasonable margin. Alice and Bob must choose the amounts to transfer (and possibly extra rules) at the beginning of the process since it is not possible to restrict the transfer to a portion of the tokens locked up in the HTLCs afterward (or to change the rules). Hashlocking is an interactive process. Participants must have access to the blockchain and communicate with each other throughout the steps described.

Alternatively, the rules of the atomic swap can be implemented in an orchestration smart contract that handles order matching, signature verifications, and transfer requests between the two smart contract vaults taking part in the exchange.

For example, in the Wyvern Protocol [45], participants signal their intent to exchange tokens by sending a transaction to the orchestration smart contract. This transaction grants the orchestration smart contract the ability to transfer tokens from the participant's smart contract vault if a matching buy order is found. Until this occurs, participants can withdraw their intent by transferring the tokens out of the smart contract vault. Implementations of the Wyvern Protocol include OpenSea [46], a non-custodial marketplace for ERC-721 and ERC-1155 nonfungible tokens.

Some systems conduct atomic swaps through off-chain coordination rather than through rules implemented in smart contract vaults, such as Algorand [47]. Once participants reach an exchange agreement off-chain, the transactions necessary for the exchange to occur are bundled into an aggregate transaction signed by all parties before being submitted to the blockchain.

HTLCs also allow atomic swaps between two distinct blockchain networks that support the same hashing function, enabling point-to-point communication. For example, Decred [48] provides a repository of implementations that leverage HTLCs to support cross-chain transactions across different types of blockchain protocols. Another locking-based technique for cross-chain transactions uses discrete log-based signature locks (the blockchains must also support the same hashing function). In this structure, the participants lock tokens in multi-signature contracts deployed on each respective blockchain. Instead of solving a hash pre-image problem, as in hashlocking, a discrete logarithm problem serves as the basis for the exchange. The *Scriptless Scripts* [49] project from Elements provides an example based on discrete log-based signature locks for cross-chain transactions.

4.1.2.2 Non-Custodial Exchanges

Non-custodial exchanges allow users to trade tokens with one another while remaining in control of the private keys at all times. They are also known as *decentralized exchanges* or by the acronym DEX. By design, user onboarding may not be required since users bring their own accounts, the users themselves may have the ability to add new token pairs, and trades are recorded directly on the blockchain or using a second layer protocol. In comparison, traditional exchanges act as third-party custodians that record trades in their own ledgers on top of their own infrastructures, which integrate the underlying network(s).

The attack surface of non-custodial exchanges is lessened due to account security risks being shifted towards users, but it is not eliminated. In particular, mechanisms are usually needed to mitigate *front-running attacks* wherein miners (or sometimes other entities, indirectly) learn about an order's information and attempt to submit transactions that take advantage of that information before the order is executed and published onto a block.

The exact security model, scope, architecture, and reliance on off-chain resources of non-custodial exchanges vary. Several types of architectures for fungible token exchanges are described below:

- **Orderbook Market Making**: An *orderbook* contains a list of buy and sell orders for a given token pair, called *bids* and *asks*. The highest bid and lowest ask are referred to as *the top of the book*, and the difference between them as the *spread*. Two main order types can be submitted: market orders and limit orders. When submitting a market order, tokens are immediately bought or sold for the best available price by pairing buyers and sellers with orders currently at the top of the book. On the other hand, limit orders are placed on the orderbook upon submission at the specified price and remain unfilled until the top of the book moves to the specified price. Orderbooks are implemented fully on-chain or as a hybrid—orders being collected and matched by non-custodial liquidity providers (or relayers) off-chain before being settled on-chain. They may do so in exchange for a fee and may form an incentivized peer-to-peer network. Note that this architecture is the closest to that of most traditional exchanges.

- **Automated Market Making**: *Automated market making* is another type of non-custodial exchange based on *liquidity pools* rather than orderbooks. Liquidity pools are smart contract vaults that hold funds for both tokens of a given token pair and act as automated market makers by determining the exchange rate between the two tokens using a formula that considers the relative quantities of each token in the pool. As long as a liquidity pool for a particular token pair exists, the liquidity problem found in markets that lack buyers is eliminated: the liquidity pool acts as a counterparty that maintains continuous availability to execute non-custodial token exchanges. Protocols may incentivize liquidity providers by distributing rewards for adding a new liquidity pool or contributing to existing ones. When contributing liquidity to a pool, deposits are recorded as a non-transferable balance or pool-specific staking tokens, allowing composability with other protocols (see Section 3.4). Liquidity pools from multiple sources may be aggregated with the exchange rate of token pairs calculated across them.
- **Dutch Auction Market Making**: In this type of non-custodial exchange, *Dutch auctions* are continually conducted. Sellers can submit orders at any moment, but those orders are only executed during the next auction. Auctions may start with an initial price set to twice the previous auction's final closing price for the same token pair, which then gradually decreases until the price clears the buy and sell orders. During auctions, buyers submit bid orders when they are satisfied with the current price, knowing that every buyer will receive the tokens for the same price in a batch settlement at the end of the auction. Like automated market making, Dutch auction market making has especially been used for more illiquid tokens.
- **Ring Trade Market Making**: Like Dutch auction market making but with shared liquidity across token pairs, ring trades enable non-custodial exchanges based on periodic batch auctions and settlements. Users place limit sell orders that are processed at the next available auction. Auctions consist of open competitions where order settlement propositions are selected by the protocol to maximize traders' profits while providing single clearing prices.

4.1.3 Blockchain Bridging

This section discusses bridging schemes to support cross-chain transactions, enabling token and data portability and architectures composed of multiple blockchains.

Deposit or bridge smart contracts are core primitives used to connect one blockchain to another. They enable users to receive proof of collateral for locking up tokens, which is then used to claim a representation of those tokens on the other blockchain. In two-way bridge contracts, that representation provides redemption value for the original token. On the other hand, one-way bridge contracts are used to enable the permanent migration of tokens from one blockchain to another, meaning the tokens do not have redemption value on the original blockchain. Deposit smart contracts with different locking techniques are coupled with intermediary coordination systems for enabling cross-chain communications, more particularly, for recognizing deposits across blockchain networks. Without smart contracts on both sides, bridges may also be built from similar coordination systems by verifying transactions or payments across blockchain networks. Coordination systems include notaries, relays, or separate blockchains. Some are trusted off-chain while others use on-chain orchestration artifacts. Communications follow a common data format, which can be supported by messaging protocols that structure, queue, and route messages between blockchains, such as the Cross-Chain Message Passing protocol developed by the Web3 Foundation [50] and the Inter-Blockchain Communication protocol developed by the Interchain Foundation [51]. As shown in Figure 4, the intermediary system (in dashed grey) creates a hub-and-spoke architecture that connects two or more blockchains.

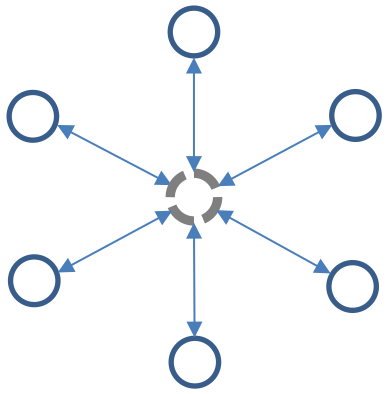

Figure 4: Hub-and-Spoke Architecture

4.1.3.1 Sharding

The requirement for every blockchain node to store and process all transactions in the network creates a bottleneck and limits transaction throughput, especially for permissionless blockchains.

Sharding increases transaction throughput via parallel processing. The blockchain's global state is split among blockchain subnetworks, which have their own transaction history and set of nodes. A separate hub blockchain coordinates these subnetworks by assigning nodes to them, processing a shared snapshot history of the state updates or metadata it periodically receives from them, and enabling mechanisms to mitigate fraudulent activities. Transactions are considered valid once added to a block by a blockchain subnetwork and do not require any additional validation from the root blockchain. Consequently, the validity of the whole system is compromised when a single blockchain subnetwork is tampered with. This notion, called *tight coupling*, differentiates sharding from relayed blockchains (see Section 4.1.3.3) [52].

For example, Polkadot [53] uses sharding to allow application-specific or specialized blockchain subnetworks, called *parachains*, to communicate with one another, with shared security. Ethereum 2.0 is also expected to use sharding but with identical blockchain subnetworks, called *shard chains* [54]. The former is also designated heterogenous sharding and the latter homogenous sharding.

4.1.3.2 Notaries

A *notary* is a trusted entity (or a set of trusted entities with a multi-signature contract) tasked with reading a blockchain's global state and submitting transactions to another blockchain on behalf of users with respect to tokens they previously deposited in a smart contract vault. Notaries act proactively, automatically responding to events that occur on a blockchain, or reactively, when prompted to do so by users. While reducing the number of bridges necessary to connect new blockchains, notaries may also act as a bottleneck for transaction throughput and introduce a centralized attack surface.

Two categories of notaries can be distinguished [55]:

- Custodians, who generally receive full control over a user's tokens locked in the smart contract vault (see Section 2.1.3) and are trusted to release them when asked to do so. Custodians may be subject to collaterals and penalties to disincentivize fraudulent activities. As an example of a notary scheme, [56] links the IOTA protocol to the Hyperledger Fabric protocol.
- External escrows, which receive conditional control over a user's tokens locked in the smart contract vault generally via a multi-signature contract in which the signature of the user and that of the escrows are required before a transaction is executed. Wanchain [57] uses a set of dedicated nodes as notaries tasked with verifying cross-chain transactions and creating external escrows through secure multi-party computation (see Section 4.3.3).

4.1.3.3 Relays

A *relay* is a system, often a smart contract, deployed on top of a given blockchain network that receives data from one or more distinct blockchain networks (e.g., block headers) via external entities called *relayers* and allows verifying the validity of transactions from these networks. Unlike notaries, which rely on trusted entities to conduct cross-chain transactions on behalf of users, relays allow users to remain in control of their tokens and independently verify (or prove) that funds have been locked on another blockchain. An example of relay is BTC Relay [58], a one-way bridge where a smart contract deployed on top of the public Ethereum blockchain receives block headers of the public Bitcoin blockchain from incentivized relayers and allows users to verify Bitcoin transactions by submitting Merkle proofs, following Bitcoin's simple payment verification technique.

When a single blockchain is linked to more than one distinct blockchain and serves as a coordinating entity, it is also referred to as a *relay blockchain*, as illustrated in Figure 5.

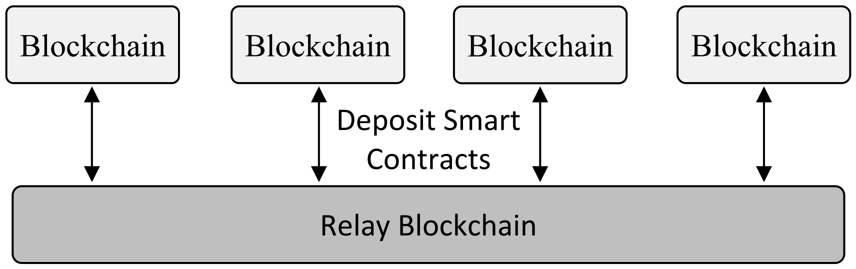

Figure 5: Relay Blockchain

Relay blockchains use the same hub-and-spoke architecture as sharding but are not tightly coupled, as in Cosmos [59]. Each "subnetwork" has its own separate consensus model and transaction finality properties. The blockchain hub network aims to enable cross-chain transactions between subnetworks but does not play any role in their security. By locking up tokens on the relay blockchain, proofs of collaterals can be generated to complete transactions on relayed blockchains.

Relays and notaries can be combined into hybrid schemes. For example, a relay blockchain could be set up to connect blockchain A to blockchain B. If the relay blockchain is unable to issue or validate transactions on blockchain B, a notary compatible with this particular blockchain could be used instead. For example, Rootstock [60] features a two-way bridge with Bitcoin to lock and unlock tokens using a federation of notaries with a multi-signature contract.

4.2 Transaction Submission

Transaction submission is either open to anyone or restricted to particular privileged users or roles via multilevel permissions. Access control can be implemented at the smart contract layer using conditions on function calls to accept or reject transactions according to protocol-defined rules or directly in the blockchain protocol at the base layer.

As discussed in Section 2.1.3, token owners can give approvals and special authorizations to let other accounts submit transactions that directly affect their tokens. These authorizations can take the form of capability-based delegations to particular accounts or keys issued with specific usage conditions (e.g., spending limits). Such approval and authorization mechanisms can be used to implement payment requests, such as for *pull payments* and *split payments*, in which a group of users accepts a common payment request. Additionally, payment protocols can be designed to send payment acknowledgments and provide refund addresses [61]. Note that off-chain messaging schemes are generally needed to enable incoming and outgoing payment notifications.

4.2.1 Meta Transactions

Signing transactions is free but submitting and broadcasting them on the blockchain can involve fees or tips with users bidding for storage (i.e., to get their transactions included on the blockchain). Those fees can depend on how data intensive transactions are. There may also be transaction fees at the smart contract layer, depending on the protocol. *Meta transactions* (or *fee delegation*) make it possible for transaction fees to be paid by third parties on behalf of users. Similar to prepaid "transaction envelopes," this enables users to submit *fee-free transactions*. Thus, users do not have to own protocol-native tokens of a given token-based network to interact with it. Note that some permissionless blockchain networks have native fee-free transaction submission models that rely on staking rather than pay-per-use fees.

Meta transactions can be used to subsidize transaction costs (e.g., promotional incentive periods, subscriptions). They can also help improve user onboarding and usability in multiple ways. The risk of token value depreciation can be transferred to third parties (between the moment tokens are acquired and the moment they are used to pay for submitting a transaction). Thus, protocols can be used without having to go through the regulations that may otherwise be associated with the act of spending tokens. Note that fee subsidizing can be conducted as a one-time operation or for all operations (e.g., a company paying for gas on behalf of employees or users).

Examples of protocols that implement meta transactions for the public Ethereum network (also called *gasless* transactions) include draft ERC-1613 [62] and the Gas Station Network [63]. The latter allows the cost of using a smart contract to be borne by the protocol itself (or paid by users in ERC-20 token denominations) through relayers that pay for and submit transactions on-chain.

4.2.2 Smart Contract-Based Access Control

Users may be able to initiate operations, particularly token transfers, in a pure peer-to-peer manner. Alternatively, protocol-level restrictions may be placed to implement conditional transactions and permissions through role-based access control, attribute-based access control, or hybrid/fine-grained access control (e.g., a list of authorized users that enables token transfers with different identity verification tiers and transfer limits). For fungible tokens, those restrictions may apply to a user's entire account balance or a portion of it [7]. Sending an unauthorized transaction means that the transaction will fail to execute and the state of the blockchain will remain at its starting point.

Note that some publications categorize tokens in two types—"token-based" and "account-based"—depending on whether transfers are subject to controls or the representation type used (as discussed in Section 2.1.1) [64].

Role-Based Access Control:

In *role-based access control* (see NIST definition [65]), role assignation follows either a top-down approach, where privileged entities act as system owners and directly manage the roles, or a bottom-up approach, where roles are self-assigned by users with predefined conditions and time delays during which system owners may be allowed to cancel new role assignations.

Roles are implemented through role manager smart contracts that are integrated with token factory contracts. Smart contract libraries that offer role-based access control have been developed, such as the Open-Zeppelin library [66]. It is also possible to implement roles in blockchains that follow the UTXO model by modifying the input and output parameters in the transaction format [67].

Attribute-Based Access Control:

In *attribute-based access control*, an identity management system provides users with token-based attributes or credentials generated either on-chain or off-chain, which they can then use to authenticate themselves and be authorized to call certain account-level operations. For example, it can be used to restrict the transfer of a given token type to accounts that passed a predefined test or met certain requirements, as implemented in the Transaction Permission Layer Protocol [68].

4.2.3 Blockchain Node Permissioning

The two main types of permissioning schemes to control which blockchain nodes can join the network are described below. The choice of node permissioning scheme, if there is any, can indirectly affect the users' ability to submit token transactions (e.g., by forcing them to send transactions to a specific node).

Local Permissioning:

Each node maintains a configuration file that contains a list of nodes from which to accept connections. This enables two types of governance models:

- In consortium blockchains, each node maintains its own configuration file.
- In private blockchains, each node maintains the same configuration file, digitally signed and provided by a trusted central authority (i.e., a system owner).

On-Chain Permissioning:

Smart contract-based access control can also be used as a permissioning scheme for blockchain nodes and accounts at the protocol level. Full nodes access a landing smart contract, the address of which is provided in the network configuration, that itself contains the permissioning smart contract address once it is deployed, as implemented in Hyperledger Besu [69].

This access control type provides another way to develop governance models for adding and removing nodes and accounts that do not necessarily require trusting a central authority. For example, a voting system that provides equal governance rights to all nodes in a consortium blockchain network could be developed. It is also possible to deploy smart contract-based access control for node permissioning on a blockchain distinct from the one that it is intended for.

4.3 Transaction Viewability

This section discusses monitoring and analysis tools, privacy-enhancing techniques, and computation on encrypted data.

Blockchain protocols are generally meant to provide correctness but not privacy. By design, all on-chain transactions are at least visible to all of the blockchain nodes so that they can verify their correctness and publish them onto new blocks. These blocks are intended to provide immutable records in a consolidated, integrity-protected view or bulletin board: the global state. In public networks, on-chain transactions are visible to anyone. This enables public auditability, encourages transparency, and puts everyone on an equal footing. At the same time, this has critical ramifications for privacy as transactions may be linked to known identities. That is why privacy-enhancing techniques and off-chain privacy schemes are fundamental components of enabling user privacy or preventing unintended enterprise data dissemination in a blockchain setting while maintaining transaction correctness. Note that, in permissioned blockchain settings, preventing users from viewing on-chain transactions does not by itself guarantee privacy since any node can share transactions externally. For users, it also entails trusting the entities that operate the nodes for the integrity of the network, directly or via auditors.[7]

Depending on their architecture and deployment characteristics, off-chain transactions (see Section 4.1.1) may provide some degree of data isolation and user privacy. In particular, the off-chain privacy schemes discussed in Section 4.3.3 are specifically designed to provide confidentiality.

[7] Alternatively, some permissioned blockchain protocols allow architectures where users can join groups and operate their own nodes within networks specific to those groups (see Section 4.3.3). Having multiple isolated states or ledgers shared with subsets of participants rather than a single global state shared with all participants may help define the scope of what each user can see.

4.3.1 Monitoring and Analysis Tools

Being able to view the transactions means that data can be collected and processed with monitoring and analytics tools to provide insights that can help manage tokens. A *blockchain explorer*, or *network monitor*, is software that allows users to browse and visualize blocks and transactions and provides network activity metrics, such as average transaction fees, hashrates, block size, and block difficulty. More advanced tools that take into account special-purpose protocol logic or off-chain transactions can inform on other types of global insights, such as token activity (e.g., transfer volume and data, active addresses, pending transactions, top token holders), non-custodial exchange activity (e.g., number of traders, trading volume), lending protocol activity (e.g., collateral amount, interest rates), and more generally, smart contract activity (e.g., account balances and event logs).

These tools can be coupled with actionable alerts, integrated with real-world activity, and offered as data feeds externally through platform application programming interfaces (APIs). Services can also offer real-time monitoring of unconfirmed transactions by collecting data from the pending transaction pool of select nodes in the network, enabling events or push notifications for transaction status updates.

Depending on whether data is encrypted or privacy-enhancing techniques are used, insights into individual accounts may also be obtained (e.g., tokens owned, transaction tracing, participant identification, interaction visualization, and payload correlation). With many blockchain networks involving public transactions, tools that check regulatory compliance and monitor fraudulent activities may be implemented. System designs for embedded privacy-preserving compliance are being researched and developed on a case-by-case basis. They aim to enable the creation of fine-grained viewing or audit keys that let authorized accounts confirm compliance by minimally revealing information from transactions.

4.3.2 Privacy-Enhancing Techniques

A high-level review of key privacy-enhancing techniques to shield transaction data is provided below. These techniques can enable private transactions without a separate state, including in public permissionless blockchains, but present open research challenges. Readers who want to learn more are invited to access additional resources, such as [70]. *Single-use identifiers* (see Section 3.3), *transaction mixers,* and *ring signatures* provide anonymity through transaction unlinkability (transaction data remains visible). *Zero-knowledge proofs* and *Pedersen commitments* are primarily used to keep transactions confidential.

Zero-Knowledge Proofs:

A *zero-knowledge proof* is a cryptographic scheme where a prover can convince a verifier that a statement is true without providing any additional information. Zero-knowledge proofs can be embedded into encrypted transactions so that users can verify transaction correctness without learning the content. This allows payments where only the sender and recipient can decrypt the amounts and account addresses involved. Mechanisms for zero-knowledge proofs can be built directly into protocols at the base layer or implemented as second layer protocols using smart contracts (or additional cryptographic schemes).

For example, EY's Nightfall project [71] is an open-source suite of tools and smart contracts that enables ZK-SNARKs-based[8] private transactions on Ethereum-based networks and is compliant with the ERC-20 and ERC-721 token standards. Users generate ZK-SNARKs by using ZoKrates [72] (which provides a high-level language for writing code before converting it into a ZK-SNARK) and send them along with their tokens to a smart contract vault that creates a cryptographic commitment for each deposit (see paragraph on *Commitment Schemes* below). This commitment can then be transferred under zero-knowledge to other users within the same smart contract vault. Another smart contract is tasked with verifying the cryptographic proofs submitted to the smart contract vault; it uses the elliptic pairing curve functions specified in the Ethereum Improvement Proposal (EIP)196 [73] and EIP-197 [74] standards. The Aztec Protocol [75] follows a similar architecture: zero-knowledge proofs received by the smart contract vaults are sent to and independently verified by a central smart contract. It supports different types of zero-knowledge proofs to convince that the amount transferred is within a given interval, such as range proofs.

Transaction Mixers:

Transaction mixers are meant to provide transaction untraceability. A first approach consists of users sending equal amounts of a given token to an intermediary who, in return, sends the funds back to other addresses owned by the same users [76]. The intermediary can be a trusted custodian or a non-custodial smart contract vault. Another approach to transaction mixing aggregates transactions to obscure the linkage between senders and recipients, as in CoinJoin [77].

Blind Signatures:

A *blind signature* is a digital signature for which the message's content is not visible to the signer (*blinded*) [79]. Once the message's content is revealed (*un-blinded*), the signer may not be able to recognize it from the blinded version of the message that they previously signed, providing unlinkability between the blinded and un-blinded versions of the message. Applications of blind signatures include on-chain anonymous voting [80].

Ring Signatures:

A *ring signature* is a digital signature produced indistinguishably by the private key of any members of a group [81]. This allows members to sign transactions under the group's identity without revealing which particular member originally created the signature.

Commitment Schemes:

A *commitment scheme* is a cryptographic algorithm where an encoded message is sent to the receiver with a condition on when it can be decoded. Commitments (Pedersen commitments are among the most common ones [82]) can be used in blockchain transactions to keep their content private, such as in the Confidential Transactions scheme [83].

[8] Zero-knowledge succinct non-interactive arguments of knowledge (ZK-SNARK) is a form of non-interactive zero-knowledge proof and, thus, requires an initial trusted setup. On the other hand, it does not involve multiple cycles of information exchange between the prover and the receiver [78].

4.3.3 Off-Chain Privacy

Zero-knowledge proofs enable user privacy features for any data, including for data that is stored in a blockchain's global state and shared with untrusted participants (see Section 4.3.2). With its own execution environment, an off-chain private state can provide additional user privacy features while still offering integrity protections through proofs periodically submitted on-chain. Those proofs may be zero-knowledge proofs or contain encrypted data and may be used for on-chain dispute resolution. In addition, the integrity protections may also require the use of TEEs or computation on encrypted data.

Off-chain privacy schemes have been studied and developed for both permissioned and permissionless blockchains when transactions and smart contract data are intended to remain confidential. This can help implement tailored user privacy frameworks and data integrity levels, following consortium-specific policies. Off-chain privacy schemes are similar to off-chain scaling schemes (see Section 4.1.1) but are specially intended for privacy through the use of encryption mechanisms or cryptographic protocols (as they mature, off-chain scaling schemes may implement privacy features, and vice-versa). Implementations consist of second layer protocols that are deployed on top of either a general-purpose or an application-specific blockchain.

This section discusses private state execution networks that are operated by a privacy group, those that are operated by a community, and secure computation.

Privacy Group Networks:

Private state execution environments on top of permissioned blockchains are usually operated by privacy groups in which transactions are either visible to all group members or only to the members directly involved in a given transaction. Privacy group members must run a *private transaction manager* node client, forming a group-specific network wherein members privately exchange off-chain information or private transactions. Users can join privacy groups by registering existing or dedicated blockchain addresses that they can prove ownership of using the associated private key(s) and then serve as identifiers within the groups.

The Enterprise Ethereum Client specification [84] published by the Enterprise Ethereum Alliance (EEA) specifies a private transaction model for permissioned blockchains. Hyperledger Besu [85] is an example of a protocol that follows this specification with the private states managed by an open-source private transaction manager called Orion [86]. Each Hyperledger Besu node that sends or receives private transactions requires an associated Orion node. Private transactions are passed from Hyperledger Besu nodes to the associated Orion nodes, which encrypt and broadcast them to the other Orion nodes participating in the transaction. Quorum [87] follows a similar approach with a private transaction manager called Tessera [88]. Figure 6 below provides a simplified flow-chart of a private transaction execution.

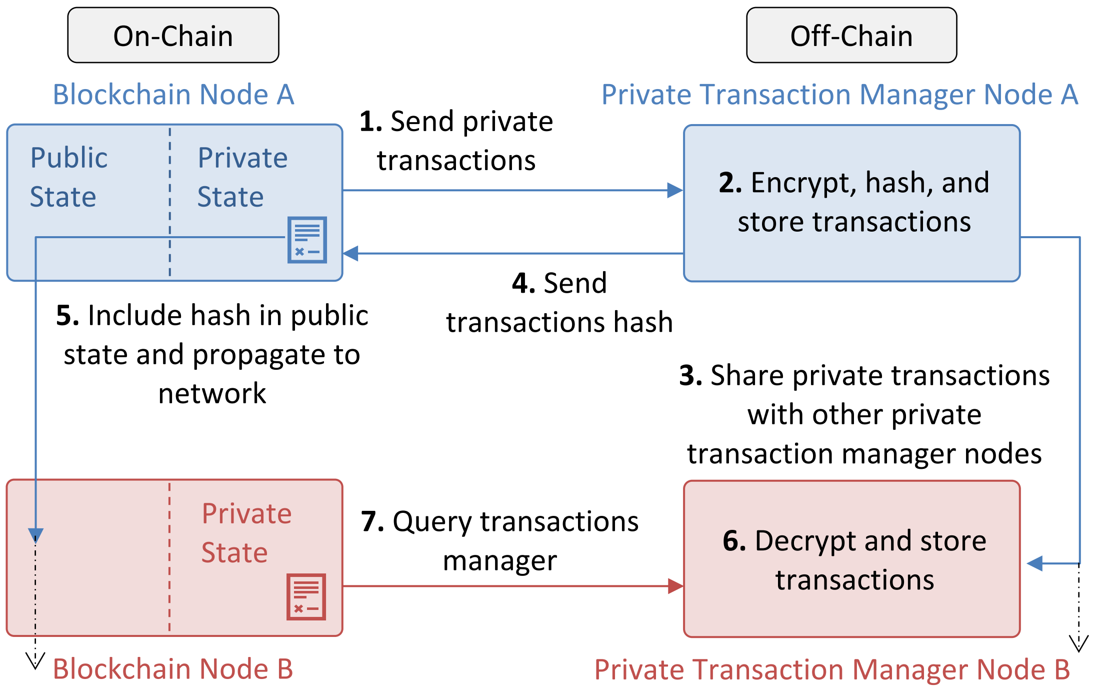
On-Chain
Off-Chain
Blockchain Node A
Public State
Private State
Private Transaction Manager Node A
1. Send private transactions
2. Encrypt, hash, and store transactions
4. Send transactions hash
5. Include hash in public state and propagate to network
3. Share private transactions with other private transaction manager nodes
Private State
7. Query transactions manager
6. Decrypt and store transactions
Blockchain Node B
Private Transaction Manager Node B

Figure 6: Private Transaction Execution

Community-Controlled Networks:

Private state execution environments on top of permissionless blockchains can be controlled and operated by the community. In Enigma's Secret Network [89], nodes are meant to both participate in the blockchain's consensus model and compute private transactions. Developers submit smart contracts to the blockchain, the code of which is publicly visible on-chain. Users then deposit tokens or submit encrypted data to serve as input. All nodes perform the secret contract execution, and if more than two-thirds of the nodes agree on the same encrypted output, that output gets published on-chain. During secret contract execution, the encrypted inputs are decrypted and processed inside a TEE (see Section 3.1).

In Hawk [90], a blockchain-agnostic smart contract protocol, a smart contract's private state is executed by a third-party intermediary, a TEE, or the users themselves through multi-party computation. The system uses ZK-SNARKs to obfuscate the amounts and identifiers involved in token transfers. Each smart contract requires its own initial trusted setup. In NuCypher Network [91], users can delegate decryption rights over private data. In exchange for fees and protocol-native rewards, nodes are tasked with re-encrypting private data provided by users to enable the designated delegate to decrypt it. Nodes are also required to stake protocol-native tokens to disincentivize negative behaviors. Finally, in Arbitrum [92], smart contracts take the form of virtual machines that are implemented and executed off-chain. A committee of managers designated by the smart contract creator is charged with monitoring the progress of the virtual machine and posting state updates on-chain. Staking tokens into the protocol enables managers to challenge the correctness of a particular on-chain state update. The dispute is then resolved on-chain, and the stake is returned to the challenger if they are successful.

Secure Computation:

Computation on encrypted data, such as *secure multi-party computation*[9] and *homomorphic encryption,*[10] makes it possible to perform confidential and distributed computations in zero-trust environments such that no one can use or read the data being computed. Expanding on applications for financial services [93], a key usage for blockchain networks rests in enabling a separate network of nodes that process transactions blindly, meaning the nodes can verify the transactions' correctness without viewing their content. This differs from the privacy group networks discussed previously where the members generally decrypt transactions before processing them, which is also less computationally intensive. As the integration of secure computation in blockchain settings matures, it has the potential to mitigate front-running attacks while enabling privacy-preserving features in systems such as auctions, voting, auditing, and data sharing.

Hyperledger Avalon [94] (formerly known as the Trusted Compute Framework) is an open-source, blockchain-agnostic framework for selective disclosure and private transactions. It uses trusted compute resources based on zero-knowledge proofs, secure multi-party computation, and hardware-based TEEs. This makes it possible to maintain on-chain auditability and policy enforcement, following EEA's *Off-Chain Trusted Compute Specification* [95]. Microsoft's Confidential Consortium Framework [96] is another example of an open-source secure computation framework. It is designed for consortium settings and relies on a network of TEEs to host the ledger and execute blockchain operations. All consortium governance updates (e.g., node removal, software upgrade) are recorded on the ledger. Additionally, the Linux Foundation has launched the Confidential Computing Consortium [97] to accelerate the adoption of TEE technologies and the development of open supporting standards.

[9] Secure multi-party computation allows datasets to be processed by fragmenting and distributing them among a network of nodes so that each node only has access to its own data share and cannot gain knowledge about the other shares. Once the computation is completed, the output is known by all of the nodes. Thus, secure multi-party computation allows multiple parties, often mutually distrustful, to compute some functionality of their inputs as if they were computed by a trusted third party [31].

[10] Homomorphic encryption allows encrypted data to be processed without having to be decrypted beforehand. Fully homomorphic encryption is a more efficient variant of homomorphic encryption that allows using and combining more functions with encrypted data.

5 Infrastructure Management

This section discusses software design patterns and infrastructure tools that make it easier to integrate token-based protocols in user interfaces. They include browsers, wallets, exchanges, dashboards, service aggregators, and other types of web or mobile applications that make calls to the components involved on-chain and off-chain to let users access their tokens and initiate operations.

Transactions are constructed by user interfaces or middleware, signed by wallets, and recorded by blockchain networks and second layer protocols. Anyone can integrate permissionless protocols in existing or new user interfaces (and other blockchain-based protocols). Self-hosted, cloud-based, and community-controlled methods are examined.

5.1 Blockchain Networks Integration

Blockchain networks can be integrated directly at the base layer and the second layer or through open connectors and interfaces designed to facilitate interoperability.

5.1.1 Base Layer

To interact directly with a blockchain network, an application must make calls (e.g., using the JSON Remote Procedure Call (RPC) API integrated via protocol-specific libraries) to a client running a node to read transactions, blocks, or balances and submit transactions.

Blockchain node clients are generally categorized into two types [1]:

- *Full nodes* download new transactions and blocks for verification and, if they are valid, broadcast them to other nodes. To verify transactions and blocks, full nodes must synchronize with the entire blockchain. There are two types of full nodes based on whether they propose new blocks for publication: *publishing full nodes* (also known as *miners, validators,* or *block producers*) and *non-publishing full nodes* (also known as *verifiers*). The security of the network comes from both the publication and the propagation of valid new blocks. Thus, full nodes form the backbone of the network (in solid grey in Figure 7). There may be multiple full node client software available for the same blockchain protocol.

- *Lightweight nodes* do not synchronize with the entire blockchain but instead only download and verify block headers. When a lightweight node receives a new transaction, it passes it to a full node. Only full nodes are trusted for the verification of transactions correctness (securing the network). Thus, lightweight nodes sit exclusively at the edge of the network (in dotted grey in Figure 7). Incentivization schemes are being researched to compensate full nodes for their verification work via transaction fees as, in general, only the publication of new blocks is rewarded.

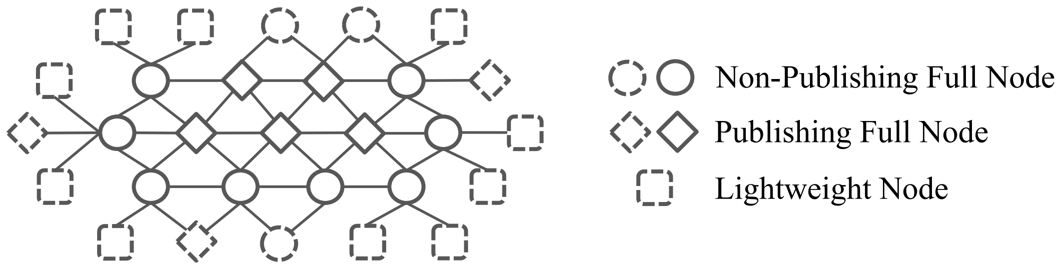

Figure 7: Blockchain Node Types

User-Controlled Full Nodes:

In addition to helping secure the whole network, running a full node is the most secure way to interact with a blockchain since its integrity is verified independently. A popular phrase with users who run their own full nodes is "don't trust, verify." Thus, when an application needs to access a blockchain network, a user can obtain maximal security by directly running their own full node and pointing the application to it. This requires that no node permissioning scheme prevents the user from doing so and that the user interface allows connections to custom nodes. Running a full node on the same device as the one accessing the application may be unsuitable, depending on the circumstances, as it requires synchronization with the entire blockchain (e.g., memory-limited mobile devices, reduced internet bandwidth causing high latency). Solutions include running local lightweight nodes or connecting to remote full nodes that are controlled and operated by the users themselves (like self-hosted virtual private network servers). These can take the form of dedicated preconfigured full node boxes (i.e., plug-and-play, headless computers) that are always powered on and connected to the internet. Depending on the throughput of the blockchain network, whether they produce blocks, and the type and configuration of the consensus model, full nodes may run on computers with relatively low computing power. The same computer may support full nodes for multiple blockchain networks. Alternatively, nodes can be provided by vendors and blockchain infrastructure service providers.

Blockchain Infrastructure Service Providers:

Blockchain infrastructure service providers primarily offer node provisioning and orchestration for organizations and application developers with guaranteed reliability and speed, as well as other services such as transactions and queries analytics. Applications usually interact with these cloud-based nodes through standard JSON-RPC calls or proprietary APIs that provide higher-level abstractions, depending on the services provided (e.g., hosted software, platform, infrastructure).

Service providers may offer access to nodes shared among customers (with load balancing), deployments of new dedicated nodes to a single customer, and "bring-your-own-node" models where customers use node orchestration tools and APIs but provision their own nodes. Cloud-based, on-premises, and hybrid architectures can thus be designed, allowing organizations and application developers to share or outsource some of the deployment and maintenance work needed to operate nodes for particular blockchain networks (and sometimes, second layer protocols). They can also bootstrap new blockchain networks by providing custom genesis files.

5.1.2 Second Layer

Applications do not always interact directly with blockchain networks. Integrating second layer protocols can provide scalability and privacy gains. These protocols act as off-chain interfaces to perform verifiable computation and submit transactions on-chain. Key architectures and integration options are discussed at a high level below.

External Providers and Trusted Intermediaries:

Applications can integrate services provisioned by external providers with varying functions and integration requirements. Commit-chains users (see Section 4.1.1.3) may delegate transaction verification and storage to proxies and monitoring of the blocks published on-chain by the commit-chain node(s) to watching services. Watching services may also be used for payment or state channels and payment channel networks to monitor channels and issue challenges on behalf of the participants so that they do not have to stay online themselves. In cross-chain custodial schemes (see Section 4.1.3.2), notaries are given either partial or full custody of a user's tokens. Notaries may operate blockchain nodes themselves.

Closed Groups of Participants:

Applications can integrate protocols that enable off-chain transaction processing among a closed group of participants. State or payment channel (see Section 4.1.1.1) clients allow users to manage channel deposits on-chain (i.e., create, top up, and settle or dispute) as well as send, store, and relay off-chain transactions to one another through peer-to-peer communication protocols. Each participant must store all participants' transactions. Nodes in privacy group networks (see Section 4.3.3) are usually coupled with the underlying blockchain nodes that maintain the on-chain public state.

Community-Controlled Networks:

Another option is to integrate open protocols for incentivized peer-to-peer networks that enable off-chain transaction processing. Since these networks are generally open to the public, they are often expected to provide a higher level of resilience against malicious behaviors (e.g., byzantine resistance). Before joining a payment channel network (see Section 4.1.1.2), a node must usually synchronize with the underlying blockchain network. Like the base layer, applications can have their own second layer nodes or rely on external blockchain infrastructure service providers. Peer-to-peer incentivized networks are also used to operate watching services, non-custodial transaction relays (see Section 4.1.2.2), and community-controlled off-chain privacy schemes (see Section 4.3.3).

5.1.3 Application-Interface Layer

Applications can integrate blockchain networks and protocols built upon them through open connectors and interfaces, which can help build more interoperable systems but do not provide one-size-fits-all abstractions.

5.1.3.1 Data Indexing and Transaction Constructing

Open-source software can instantiate blockchain data retrieval and indexing APIs. They aim to make it easier for user interfaces or middleware to detect on-chain events and process queries that involve filtering or sorting blockchain data and are otherwise more difficult to perform with direct blockchain calls (e.g., accessing all transactions submitted by a particular account during a given period). For example, The Graph [98] is a protocol for running a server or node to process blockchain data queries formatted using GraphQL, which may be used locally, as a hosted service, or via a community-controlled network. It requires specifying which smart contracts and which events within these smart contracts should be indexed and how queries should be structured. Note that some blockchain protocols natively maintain a database of the most up-to-date state of the blockchain in addition to the blockchain data itself (e.g., Hyperledger Fabric).

In addition to retrieving and indexing blockchain data, open-source software may also allow for constructing and submitting transactions in a standard format, such as Rosetta [99], which also requires defining artefacts to handle specific contract logic. Key generation and transaction signing must be performed by separate modules (e.g., an external wallet).

5.1.3.2 Identifier Resolving

Tokens are assigned to blockchain addresses, which do not necessarily provide applications with general-purpose identifiers. When needed, applications must make calls to external identifier management systems to resolve identifiers and help manage service endpoints, as described below.

DIDs are unique, persistent, cryptographically verifiable identifiers that do not need a central registration authority and resolve to some JSON-formatted metadata, called *DID document*. DID documents usually contain information about the owner (e.g., blockchain address, payout account). DIDs are generated directly by users, with some architectures requiring registration on-chain [18].

The Universal Resolver developed by the Decentralized Identity Foundation (DIF) [100] allows identifiers to be resolved to their associated metadata for multiple blockchain-based identifier management systems (DID methods[11]) from a single interface, without managing system-specific calls. However, this requires these blockchain-based identifier management systems to be included in the Universal Resolver configuration information and to provide node endpoints. The DIF deployed a publicly available instantiation of the Universal Resolver and published the open-source code for anyone to deploy their own instantiation. The PayString protocol [101] developed by the Open Payments Coalition also aims to provide a cross-domain identifier system focused on making it easier to send and receive payments using persistent, human-readable domains. It plans to integrate W3C's *Payment Request* standard [102].

By design, these systems allow users to provide applications with standardized payout account information. This can facilitate operating models wherein tokens are distributed in exchange for contributions (e.g., to pay users to follow a tutorial or complete a survey). Note that Section 6.3 provides more details on tokenized credentials and domain names.

[11] Note that Rebooting the Web of Trust published a paper called *A DID for Everything – Attribution, Verification and Provenance for Entities and Data Items* [103], which introduces the concept of data objects associated with DIDs [104]—called decentralized autonomic data (DAD) items—to provide authentication for data provenance.

5.1.3.3 Payment Routing

Applications can integrate open protocols and standards that interface with payment systems built upon different traditional and blockchain technologies and thus allow for a certain degree of interoperability. For example, Interledger [105] enables a peer-to-peer network wherein nodes relay payments between participants by following a request/response protocol similar to notaries, as discussed in Section 4.1.3.2. Before sending payments, every pair of participants must choose a settlement method and set amount thresholds (e.g., credit line before settlement). The Interledger protocol defines a standardized HTTP API meant to abstract the differences between different settlement techniques (e.g., HTLCs, real-time gross settlement systems) and payment systems.

5.2 Wallet Integration

To interact with a blockchain network or second layer protocol, an application must make calls to a wallet to generate new blockchain addresses, pull existing blockchain addresses, and access the keys necessary to sign transactions. Transaction constructing, signing, and/or submission may be performed by wallets, user interfaces, or external modules (see Section 3, Section 4.2, and Section 5.1.3.1). By default, transactions are signed on a per-action basis upon user approval. Standards, such as draft EIP-2255 [106], are emerging to enable users to give granular per-session or per-application permissions.

Wallet-agnostic, general-purpose APIs can enable application developers to reach more users without integrating the specific APIs of each wallet, vendor, or third-party custodian. Both users and application developers can benefit from enhanced token portability across reusable wallets. It can also offer easier access to increased security and innovation since newly developed wallets can have immediate compatibility with pre-existing applications [107][108]. This makes it easier for a user's wallet and token activity to be embedded in existing user interfaces rather than being offered as a standalone product.

Examples of open protocols that enable servers to connect mobile wallets with web applications using end-to-end encryption include WalletLink [109] and WalletConnect [110]. Users can sign in to blockchain-based web applications by scanning QR codes from their smartphones without creating a new account for each application. Web3Modal [111] also offers a library to help application developers add support for multiple Ethereum wallet providers and enable end-users to choose their wallet.

5.3 User Account Data Integration

Blockchains enable shared, verifiable accounting but are not designed for general-purpose data storage and management. Applications are usually built with most if not all of the user account or profile data, if there is any, stored off-chain and, when needed, select hashes referenced on-chain for integrity validation. Architectures can rely on traditional cloud providers and on-premises infrastructures, but notably, applications need not necessarily store off-chain user account data themselves. Alternative storage options are discussed below. Trust in the overall system, however, could be impacted depending on how exposure to data withholding attacks and data availability issues are handled. In particular, it may be important to identify who controls the servers that host the data and to establish the level of data redundancy at play.

User-Controlled Storage:

Since data integrity is verified with on-chain hashes, users themselves can store and bring their own data to the applications that they are using without eliminating data integrity expectations. Users can do so directly on the wallet or device that they are using with the application. Protocols for local storage networks can also enable data synchronization across all of the devices and machines that users own locally, such as Identity Hubs [112].

Community-Controlled Cloud Storage:

Another option is to integrate open protocols for incentivized peer-to-peer networks that enable *distributed file storage systems* (i.e., community-controlled cloud storage).

At a high level, these protocols encrypt, split, distribute, replicate, relay, and address files. Cryptographic hashes are computed for each file and used as unique persistent identifiers for indexing. Blockchain infrastructure service providers may offer node provisioning and orchestration for distributed file storage systems. In addition to user account data, distributed file storage systems may also host user interfaces themselves. Examples of such storage protocols include Filecoin [113], based on the Inter-Planetary File System (IPFS) protocol [114], and StorJ [115].

Based on distributed file storage systems, protocols have also been developed for *encrypted data vaults* (or *containers*). Although users do not store their own data, they can control which applications have access to it using custom access control schemes that are themselves based on blockchain-based identifiers. For example, 3Box [116] makes it possible to store user account data using an OrbitDB key-value datastore that is controlled by the user.

5.4 External Data Feeds Integration

Blockchain networks are designed to provide transaction determinism but not data input verification. Yet, some applications rely upon external or real-world data being fed or provided—by *oracles*—to the blockchain, either directly (e.g., event results) or via the integration of tokens that are themselves built upon oracle data (e.g., price feeds). Thus, dedicated schemes with different trust and operating models are necessary to verify oracle integrity, as discussed below. Note that applications may access on-chain oracle data through blockchain data indexing APIs (see Section 5.1.3.1). Oracles may enable data feeds across blockchains and traditional systems.

Data Providers:

Individual data providers may operate their own signed data feeds and push them directly on-chain themselves or rely on intermediaries to do so on their behalf. It is possible to verify the authenticity and/or provenance of the data using the associated public keys. Frameworks are emerging to improve the interoperability of signed data feeds, such as the Open Oracle System [117]. Additionally, a data feed can be run and signed from within a TEE to provide a higher degree of trustworthiness [118].

Community-Controlled Oracle Networks:

To reduce or eliminate single points of failure, another option is to integrate incentivized peer-to-peer oracle networks. They aim to provide smart contracts with tamper-proof, off-chain data inputs by aggregating multiple data reporting sources that redundantly verify the same data points without trusted intermediaries. Different architectures and incentivization schemes that allow general-purpose frameworks are emerging, such as ChainLink [119], where a collection of independent oracle networks report on individual data feed types, and Band Protocol [120], where a consolidated oracle network is built upon a dedicated blockchain and token-curated registries.

5.5 Architectural Considerations

This section discusses high-level architectural considerations of applications that are based on blockchain networks and second layer protocols. It first examines trust assumptions before exploring some of the implications for protocol control and liability, data governance, and security. Note that this paper does not aim to provide any architectural or policy recommendations.

Trust Assumptions:

As discussed previously in Section 5, applications can integrate infrastructure components that are external to blockchain networks and do not necessarily adhere to the same peer-to-peer model and security or trust model. Thus, the degree of trustworthiness of an application does not solely depend on the characteristics of the underlying blockchain network and could rather be seen as that of the system's overall weakest component. Server-based infrastructure components reintroduce trusted intermediaries that must be evaluated to ensure that the overall system properties continue to match expectations (e.g., that data withholding attacks and data availability issues are mitigated). When using an externally hosted interface, users should be able to verify that the transactions that they sign match the transactions presented on the interface (e.g., through a signing phrase displayed in their wallets). On the other side, serverless blockchain-based infrastructure components can have decentralized governance, where users must trust game-theoretical incentive alignment mechanisms and cryptography rather than intermediaries, though the degree of decentralization is not always fixed. Token issuance and management at the smart contract layer comes with its own governance and security or trust model distinct from that of the underlying blockchain network. In particular, architectures can involve both immutable and upgradable protocol modules and implement different types of upgrade mechanisms. As such, an end-to-end assessment of risk factors across all of the individual components involved is generally needed, with particular attention on identifying centralization risks and trust assumptions [121].

Each entity or user may also have their own needs and preferences about the level of administration over their data and tokens that they are comfortable controlling themselves or delegating to custodial applications that provide professional services and manage security and blockchain network integration on their behalf.

Protocol Control and Liability:

The distributed and cryptographic nature of blockchain technology provides resilience and verifiability but does not eliminate the notions of control and liability. Permissions, roles, and backstops may be placed at the base layer and the smart contract layer, potentially giving some degree of protocol control to privileged entities that could be held accountable for their actions. This entails making policy decisions that can have wide-ranging implications, especially if nodes span across jurisdictions. Isolated legal actions from a small subset of those jurisdictions may, by design, not be able to affect a blockchain's global state. Note that data may also be isolated and kept within certain region(s) via the use of separate blockchain networks, states for subsets of participants, or off-chain schemes. International regulation, survey, and standardization efforts have been developed to help address adverse outcomes and rethink oversight approaches in such distributed and digital environments [122][123].

Data Governance and Security:

The decoupling between intermediaries, digital asset custody, and users' capability to control custody themselves results in a user-centric system architecture, where user interfaces are unbundled from data and application logic. This has fundamental data governance and security ramifications that could benefit both users and businesses but need to be carefully evaluated:

- User agency and privacy can be improved by letting users have more control over the dissemination and flow of the data they generate throughout the applications used. Disintermediation can reduce gatekeeping and increase access to digital services. Unlike credit card numbers, which allow online payments without built-in mechanism to prevent unauthorized transactions, token transactions can by default only be processed when authorized. Coupled with privacy-enhancing techniques, disintermediation can also limit user tracking and profiling. In return, users are in charge of their own protection against security risks, which they can choose to delegate at different degrees (see Section 3.2).
- Businesses' exposure to security and liability risks, with the potential associated insurance and regulatory compliance costs, can be reduced by integrating non-custodial protocols that offload some degree of control over customer data management [124]. By being available across jurisdictions or without attachment to any particular jurisdictions, blockchain networks enable global access. Integrating tokens can facilitate business development by offloading the handling of associated regulatory and infrastructure implications, with on- and off-ramps provided by separate, specialized organizations on a per-jurisdiction basis. It has the potential to make it easier for businesses to hold account balances of the currencies of their choice themselves and execute cross-border transfers with reduced costs and delays compared to the traditional correspondent banking model.

This more user-centric model for the web—with applications built as user interfaces that integrate blockchain networks and second layer protocols rather than traditional databases—is often referred to as *web3*.

The security risks entailed are multilevel:

- At the base layer, blockchain networks and consensus models have varying security, immutability, scalability, and functionality levels.
- At the custody management level, users must hold and manage their private keys securely to avoid tokens being lost or stolen as well as carefully review transactions before signing them. Recoverability techniques must be assessed to meet individual needs [125].
- At the smart contract layer, external data sources can be attacked or add inaccurate data to the blockchain (this is referred to as *oracle risk*). Smart contracts can also be subject to different types of security bugs [126] and transaction-ordering attacks, which can lead to the loss of staked tokens, fraudulent transactions, manipulation of protocol governance, and freezing of some key protocol components. Security analysis, audits, and formal verification by reputable entities and public bug bounty programs can help identify and mitigate smart contract security risks.
- Protocol governance itself also involves risks, such as those related to administrative privileges being stolen or misused or, more generally, those related to behaviors that undermine confidence and game-theoretic attacks. Multi-signature schemes and delays are often used as preventive measures.
- Protocols that enable financial instruments have specific risks, such as liquidation risk and, more generally, collateral management risks. Re-collateralization schemes at the protocol level (e.g., protocol-native tokens serving as backstops), as well as hedging and mutualized insurance schemes at the user level, may help mitigate these risks.

The security risks mentioned here are not exhaustive and present open challenges in finding ways to mitigate them appropriately.

6 Deployment Scenarios and Use Cases

This section provides deployment scenarios and use cases for issuing and distributing tokens. Protocols can involve the issuance of multiple tokens or the derivation of new tokens from existing tokens or deposits that are part of other protocols. Facilitated by following common token data models and building on top of shared or public infrastructure, composability is one of the key drivers for token-based protocols. Tokens are generally meant to be integrated into third-party wallets and applications; as new tokens are issued, built-in methods for token curation (e.g., reference lists) and standardized identification could be needed. This section concludes with potential resulting breakthroughs for privacy-preserving verifiable data exchange based on tokens. Note that this section illustrates emerging blockchain-based tokenization use cases with notions described in this paper but is not meant to provide an exhaustive list nor to judge their viability.

6.1 Decentralizing Protocol Governance

Tokens can represent programmable, protocol-native digital assets[12] that enable coordination and network effects without central enforcement through built-in economic games, utility purposes, usage rules, and/or protocol governance rights. Trust is meant to be minimized with respect to particular entities distributed among a self-governing community and trusted in aggregate. When successfully designed, this model empowers users rather than individual system owners to operate and control networks and capture the value that they generate. Cryptoeconomic models aim to encourage and discourage particular behaviors or activities as part of protocol governance and security frameworks, especially for incentivizing participants to sustainably align self and common interests, thus reinforcing the protocol's network effects and preventing Sybil attacks. The Bitcoin network has stimulated the study of decentralized governance and open-source development for permissionless peer-to-peer networks for more than a decade. It is a rich, multidisciplinary domain that involves conceptualizing notions that combine computer science, distributed systems, cryptography, and protocol engineering with economics, game theory, mechanism design, and organization theory. Thus, well-established methods, theories, and tools related to those fields may be reused, and discussions are often open-ended. Definitions have emerged for terms such as "tokenized ecosystems" [127], "token engineering" [128], and "cryptoeconomic primitives" [129].

Acting as algorithmically programmed inflation, new tokens can be minted to distribute rewards and compensate contributions. Users can be required to have a minimum account balance, stake tokens for some time, or meet other account conditions. Their stake may be exposed to penalties and subject to withdrawal delays or rules. Transactions can involve fees that either burn tokens (e.g., to prevent denial-of-service attacks and indirectly reward token holders) or transfer tokens to other accounts (e.g., non-inflationary rewards) or a protocol treasury in exchange for some services (i.e., the funds are held in custody by the smart contract(s) of the protocol). Staking tokens may earn yields or deferred rewards and grant privileges such as being authorized as a node or as a voter for self-enforceable protocol upgrades. The voting power relative to the amount staked, if there is any, varies and may be delegated, depending on protocols. On the other hand, staking

[12] Protocol-native tokens can simultaneously have characteristics of currencies, securities, and commodities as well as characteristics related to serving as protocol incentives/rewards or offering utility value. Those characteristics may evolve depending on protocol upgrades, token usage, or distribution models.

tokens may entail assuming exposure to loss of stake value while having responsibilities such as setting and maintaining system risk parameters or, in more formalized participatory structures, assigning privileges to reputable users or governance bodies. Thus, protocol-native tokens may grant access to different forms of participation as well as claims of revenue or cashflow streams and redistributions allocated to particular sets of protocol members or market participants. Token supply management policies, such as token emission schedules, can be meant to follow fixed, predetermined rules or be subject to change as part of intentionally evolving incentive models. Community-controlled networks are also called incentivized or user-owned peer-to-peer networks, with the line between network participants and network owners sometimes blurred.

A key factor affecting the level of decentralization in staking-based governance models is the extent to which the community of stakeholders is considered large and distributed enough. If too uneven, skin-in-the-game models can centralize governance power to a single entity or a small group of closely aligned entities and potentially reintroduce single points of failure. Thus, mechanisms for wide distribution of protocol-native tokens among independent actors can be of fundamental importance to ensure broad participation, though there is no one-size-fits-all model. While still emerging, different token distribution mechanisms have been followed where tokens are purchased, converted, earned (when utilizing particular protocol features, in exchange for services, or following a lottery approach, as in proof-of-work), distributed to accounts that meet certain criteria, or allocated to particular accounts with vesting periods. A key approach that makes it possible to distribute tokens through automated market making (see Section 4.1.2) rather than fixed-price sales administered by protocol founders (see Section 6.2.3) is *bonding curves*. The cost to buy the tokens is determined by their supply. Generally, the more tokens in circulation, the higher the cost. The mathematical formula that supports this is implemented directly in the token factory contract as buy and sell functions that mint and burn tokens accordingly with a pre-existing token used as a reserve token and denomination unit for pricing. Users can call those functions at any time, providing deterministic pricing and instant liquidity. There are multiple shapes of bonding curves (e.g., linear, quadratic), and the choice of one particular curve (and associated formula) predetermines price discovery. A key characteristic of this token distribution model is that it does not give any special fund withdrawal rights to protocol founders and early adopters or contributors but still can incentivize them to develop the token's utility and the network's value proposition with the token used as a tool to bootstrap adoption.

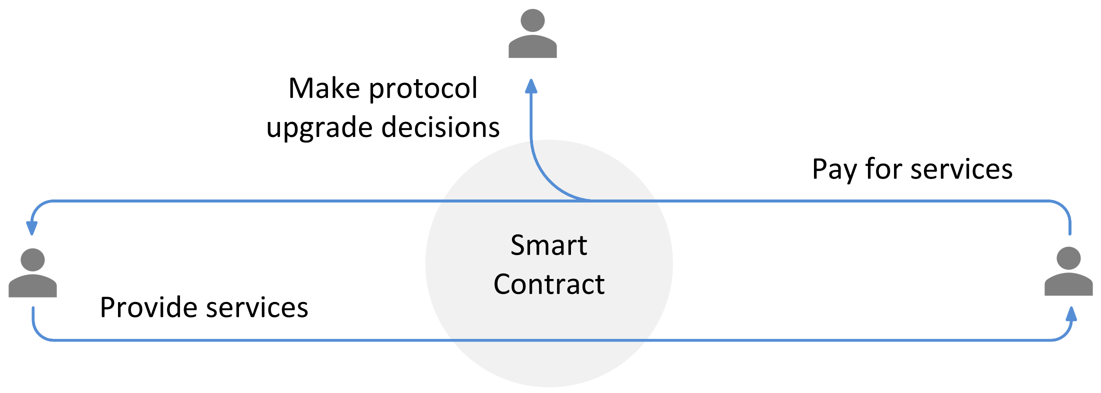

Figure 8: Smart Contract Multi-Sided Platforms

Incentive and governance tokens can be built at the base layer, as part of the consensus model, and at the smart contract layer, with some of them being used to provision a separate shared or public infrastructure (e.g., order relaying, distributed file storage). Operating models for community-controlled services at the smart contract layer often take the form of multi-sided platforms, as presented in Figure 8. They are still being experimented with, are likely diverse, and deploying distinct protocol-native tokens is often seen as potentially beneficial only when their design creates decoupled economic value. Note that schemes may enable pay-per-use fee structures (e.g., transaction fees for function calls) with a choice of unit of denomination, which is converted into the denomination needed during the function call. To avoid harmful or unintended effects, token distribution, incentive, and governance models must be designed carefully. Some protocols use backstops as a temporary risk mitigation mechanism, as described in Section 2.1.2. In general, the understanding of the governance properties of the different mechanisms that have been developed up until now remain at an early stage.

6.2 Tokenizing Money and Financial Products

This section discusses protocols that issue tokens to represent existing assets, enable lending and borrowing, and support fundraising and derivatives. Unlike protocol-native tokens discussed in Section 6.1, these tokens are built upon existing data on-chain or off-chain, via oracles.

6.2.1 Stablecoins

A *stablecoin* is a fungible token that is pegged to or redeemable for one or more underlying assets (e.g., a fiat currency). *Redeemable* or *convertible stablecoins* are backed by a collateral or reserve and grant redemption value for the underlying assets. Alternatively, *synthetic stablecoins* are designed to track the price of one or more underlying assets but do not grant any redemption value for these assets. Their reserve may comprise multiple assets distinct from the tracked assets, and their peg is maintained with a specific responsiveness level. In general, protocol-level mechanisms may be used to stabilize the value by dynamically managing the token supply and maintaining the collateralization ratio at a certain rate or range target with either partial, full, or surplus reserve. System-specific counterparty risks and the permanent loss of peg risks may affect a token's perceived valuation depending on the nature, configuration, and governance model of the underlying blockchain network and the protocol that issues the token. The token's mint and burn operations (see Section 2.1.2) are usually controlled by a trusted intermediary, a consortium, or users themselves through rules directly built into the protocol (e.g., algorithmic stablecoins). On-chain collaterals involve tokens being provably locked up using cryptographic schemes or deposit smart contracts (see Section 3.4). Funds can also be collateralized off-chain by a trusted central authority (e.g., in a traditional bank account with periodic audits). Finally, some synthetic stablecoins rely on algorithmically issuing debt on-chain to regulate the supply.

This section discusses different types of stablecoin designs at a high level. *A Classification Framework for Stablecoin Designs* [130] provides more in-depth details. Projects have been studied or developed for tokens pegged to fiat currencies and cryptocurrencies as well as tokens that directly represent ownership claims for bank deposits, securities, and central bank reserves. Stablecoins may be composable with other blockchain-based protocols and tools to form more advanced financial instruments and exchange platforms. They can make it possible to borrow and lend tokens issued by distinct protocols, earning yields on deposits. Thus, they may be seen as portable and programmable alternatives to traditional bank accounts, savings accounts, and brokerage accounts, with integrated payment systems.

Cryptocurrencies:

As discussed in Section 4.1.3, blockchain-native tokens (i.e., ownerless or non-sovereign cryptocurrencies) can be represented on a separate blockchain through a bridge that collateralizes them and provides proof of that collateral. They are also referred to as *wrapped cryptocurrencies*.

WBTC [131] is meant to represent bitcoins as tokens that follow the ERC-20 standard on the public Ethereum network through a consortium, the members of which act as notaries. WBTC tokens' minting involves two different roles: 1) merchants, who sign mint requests and provide liquidity for the WBTC/BTC pair, and 2) custodians, who process these requests. The consortium members jointly own the token factory contract via multi-signature. tBTC [132] (associated with the Cross-Chain Group [133]) allows a similar representation using a relay and incentives coupled with an overcollateralized reserve rather than trusted intermediaries. Each bitcoin deposit is represented as a nonfungible token redeemable for fungible tBTC tokens.

Fiat Currencies and Bank Deposits:

Trusted intermediaries and consortiums can issue tokens that represent commercial bank deposits on top of their own supporting blockchain networks. For example, JPM Coin [134], currently designed for wholesale customers, has been deployed on top of a private blockchain controlled by JP Morgan, and Libra Coin (and the associated LibraUSD, LibraEUR, and LibraGDP) [135], meant for retail customers, on top of a public consortium blockchain controlled by the members of the Libra Association, later rebranded Diem Association. Alternatively, trusted intermediaries and consortiums can use existing blockchains that they do not control themselves, as in the CENTRE protocol [136] deployed on top of several public blockchain networks.

Unlike the previous examples, where the tokens have known issuers and represent ownership claims on bank deposits, there have been tokens issued by community-controlled protocols to represent fiat currencies that are backed by a surplus reserve of separate tokens on-chain. For example, MakerDAO [137] uses a decentralized price feed to manage the supply of a synthetic asset intended to track the price of the U.S. dollar, a protocol collateral, and a protocol governance token that lets the community adjust system parameters while serving as backstop in case of collateral deficit. Proof-of-stake blockchains where a portion of the block rewards is used to fund a built-in reserve have also been used to issue tokens that track the value of external assets and fiat currencies, as in Celo [138].

Note that the term "stablecoin" is sometimes used to refer specifically to tokenized representations of fiat currencies and bank deposits, as described in this section.

Central Bank Reserves:

Tokens are also being studied to build central bank digital currencies (CBDC), which would represent central bank reserves, for reasons such as reinforcing the transmission of monetary policies, establishing new transmission channels, or just in response to a decline in cash. Deployments on top of dedicated consortium or private blockchains for use by the private sector or the general public have been considered, though most projects are at early stages. Note that tokenizing central bank reserves would have substantial ramifications for financial inclusion, economic policymaking, and the scope of central bank missions. At the same time, it would introduce new system counterparty, security, and privacy risks that do not exist with self-contained banknotes that circulate freely in society today. This paper is not meant to provide any considerations on those ramifications, risks, or potential benefits. In the architectures introduced in [139], a dedicated permissioned blockchain is bootstrapped with role-based access control and voting systems for blockchain nodes administration, token supply management, and system security operations. Account providers allow identity verification tiers with different permissioning structures (e.g., transfer amount per period). Several CBDC design choices have also been identified to add controls to transfers between non-custodial wallets [140]. Initiatives have emerged that aim to advance CBDC research more generally, such as the Digital Dollar Project initiative [141].

6.2.2 Lending and Borrowing

Tokens can be used to record what is owned but also what is owed. They can be lent by being deposited in a smart contract vault, which either records the deposits as a non-transferable balance or as new units of its own token (see Section 3.4), acting as tokenized debt or liabilities. These deposits can be redeemed for the underlying tokens plus interest or, if the deposits are tokenized, be used, transferred, or collateralized again separately. To mitigate default risks in overcollateralized lending, borrowers are required to provide a collateral that is greater than the amount that they intend to borrow. This aims to eliminate the need for assessment of individual user profiles that is otherwise found in non- or under-collateralized lending. Rules can also be implemented directly within the deposit contracts to automate fund transfers in case of a default. Schemes that aim to determine the collateralization ratio and interest rate (floating or fixed) algorithmically, using oracles and community-controlled governance, have been developed. It can also be possible to receive and pay back a non-collateralized loan in a single transaction wherein token units are borrowed from a smart contract vault, used to facilitate a separate transaction, and transferred back with interest to the smart contract by the end of the transaction. If the loan is not paid off, the transaction is reverted (atomic execution). Finally, borrowers may be able to receive loans that are collateralized with nonfungible tokens.

Lending protocols have several types of risk, as mentioned in Section 5.5. Note that composing tokens with one another to create lending and borrowing instruments introduces a chain of trust and, thus, new types of interdependencies and systemic risks. Stablecoin designs that represent ownership claims on the underlying asset, as discussed in the previous section, can also be seen as a form of tokenized liability. In lending protocols, yields (interests) are usually earned passively, unlike staking yields where active participation in protocol governance may be expected (see Section 6.1).

6.2.3 Fundraising and Derivatives

Tokens may be used for fundraising (e.g., multi-round, fixed-price sales), including some protocol-native tokens that are also meant to have utility purposes (see Section 6.1). However, they can be subject to external governance and regulatory frameworks, often depending on the exact token sales or distribution model and its framing to the public. Projects may attempt to issue protocol-native tokens and claim that they have or will have utility when they, in fact, primarily serve as a fundraising mechanism, are unnecessary for the protocol, or may even burden its usability. At the same time, some protocol-native tokens may have no or low utility at the time of issuance but gain utility purposes later on by becoming integral to the operations of the network that they support as it becomes adopted and decentralized. Thus, it is generally essential for those who administer the distribution of tokens to clearly identify, justify, and communicate the approach followed, the rationale behind it, and the timing (e.g., whether the token is intended to be issued or become available for circulation only once the network that it is designed to support is built out). External platforms can facilitate token issuance and initial distribution and assist with handling regulatory and compliance requirements. Although regulatory aspects are out of scope for this paper (more information can be found in [16]), token distribution models are key aspects of token designs since rules and conditions to mint or release tokens are implemented on-chain and can form the basis of protocol governance.

Tokens can also represent existing equities, commodities, and derivatives. Synthetic assets are based on price feeds that either come from trusted sources or use decentralized oracle networks (see Section 6.2.1). Put and call options are issued as tokens that provide rights to a collateral deposited in smart contract vaults with parameters that specify the terms to exercise options. Tokens have also been built to represent bundled assets, automated trading strategies or portfolio rebalancing, and mutualized insurance schemes.

6.3 Tokenizing Uniquely Identifiable Things and Supply Chains

Nonfungible tokens and stateful tokens allow the representation of uniquely identifiable things on top of blockchain networks (see Section 2). Nonfungible tokens are often used when the purpose is to represent assets that are public-facing and exchangeable (e.g., cryptocollectibles). On the other hand, stateful tokens—associated with on-chain registries for status querying—are often meant to represent personal or interpersonal assets, the content of which is assigned to particular entities (e.g., identity documents, user credentials). Note that the choice between nonfungible and stateful tokens is dependent on systems and issuers; for example, a voucher may either be assigned to a particular person or unassigned and exchangeable. Tokenizing uniquely identifiable things makes it possible to create public or cross-domain systems for user-centric identity management and supply chain management. They can provide access rights or authentication factors and enable verifiable data provenance across organizations and end-to-end asset traceability. Uniquely identifiable tokens can follow the Verifiable Credentials [19] standard, as discussed in [18].

Three main motives for deploying nonfungible tokens can be identified:

- **Companion Tokens**: Companion tokens represent digital twins tracking physical things or external assets. These tokens are useful only when presented in tandem with a corresponding object. For example, they can be used for food inventory management (e.g., IBM's Food Trust [142]).
- **Redeemable Tokens**: Redeemable tokens represent transferable rights to claim underlying assets from trusted custodians, be it physical or immaterial things. Non-transferable rights may be implemented using stateful tokens (see Section 2.2).
- **Freestanding Tokens**: Freestanding tokens represent uniquely identifiable assets that have no prior meaning, are not linked to any external assets or systems, and exist independently of the issuer after issuance.

Unlike freestanding tokens, companion and redeemable tokens are built upon existing data on-chain or off-chain, via oracles. Note that a nonfungible token may be simultaneously companion and redeemable. Ownership of a nonfungible token can be subdivided into a set of tokens held by different owners (e.g., through a deposit contract). This can be repeated with that partial ownership being further subdivided, allowing for nested data structures. Note that a nonfungible token subdivided into a set of fungible tokens is sometimes called a *re-fungible token*.

High-level guidelines for blockchain-based supply chain management have been developed, such as the NIST Advanced Manufacturing Series 300-6, *Securing the Digital Threat for Smart Manufacturing: A Reference Model for Blockchain-Based Product Data Traceability* [143], and the Department of Homeland Security's *Blockchain and Suitability for Government Applications* [144]. The Hyperledger Supply Chain Special Interest Group [145] aims to provide reference architectures and frameworks for the blockchain logistics and supply chain industry [146].

Financial, Business, and Legal Documents:

Both nonfungible and stateful tokens can represent financial, business, and legal documents or agreements, such as purchase orders, invoices, letters of credit, loans, grants, deposits, and certain securities. Nonfungible tokens may be fractionalized into fungible tokens or deployed as part of a smart contract that manages multiple token types and instantiations to allow dependent payments and conditions (e.g., expiration dates, limited pool of recipient accounts). As new assets get tokenized and exchanges and marketplaces integrated into third-party applications, regular users may increasingly act as market participants and internally interact with financial services.

Centrifuge [147] features factory contracts for minting nonfungible tokens that incorporate verified attributes from stateful tokens. The mint operation requires submitting a set of Merkle root hashes stored on-chain for specific fields of the concerned credential. OpenLaw [148] offers a markup language and templates to create factory contracts for nonfungible tokens that represent binding legal agreements. The Baseline protocol [149] aims to enable synchronized business logic and confidential data processing between organizations on top of the public Ethereum network using zero-knowledge proofs based on Nightfall (see Section 4.3.2).

Identity Documents, User Credentials, and Domain Names:

Uniquely identifiable tokens can also represent user credentials and identity documents, such as digital driver licenses, passports, employee or student badges, and other types of membership rights. For example, a company can decide to issue digital employee badges as uniquely identifiable tokens on a public or consortium blockchain (shared with business partners and other stakeholders) to speed up staff authentication with external organizations and web services while facilitating the posting of verifiably endorsed and timestamped data. Some personally identifiable data attached to a nonfungible or stateful token may be publicly viewable on-chain, allowing entities to publicly share information about themselves, such as a service endpoint at which they can be reached.

Examples of systems for user credential tokenization using stateful tokens include uPort [150], Blockcerts [151], and Hyperledger Indy [152]. In uPort, stateful tokens take the form of JWTs (see Section 2.2) and are coupled with a smart contract issuer registry deployed on the public Ethereum network that follows the draft ERC-780 [21] proposal, the ownership of which was relinquished.

In Blockcerts, stateful tokens take the form of JSON Linked Data (LD) objects coupled with an issuer-controlled smart contract registry containing a list of addresses authorized to revoke tokens and a list of revoked tokens. In Hyperledger Indy, each stateful token issuer must publish a revocation registry containing a cryptographic accumulator that allows users to verify whether a given credential was revoked by the issuer without compromising the registry's privacy.

Domain names can be represented as nonfungible tokens too, as in the Ethereum Name Service (ENS). ENS allows the registration and linkage of blockchain addresses (or other hashes) to human-readable identifiers (e.g., following the format "username.eth") through auctions and nominal fees. The token factory contract follows the ERC-721 standard and maps blockchain addresses to readable identifiers as well as for referencing domain resolvers.

These tokenized identity documents, user credentials, and domain names may be used as user-centric reusable identifiers, log-in, and user onboarding methods for online services, providing alternatives to traditional password-based authentication and verification methods, such as emails, phone numbers, and single sign-on solutions.

Licenses, Vouchers, and Event Tickets:

As mentioned in the section introduction, licenses or memberships, vouchers, and event tickets can either be assigned to a particular person (see previous paragraph) or unassigned and exchangeable (see *Exchanging Event Tickets and Coupons* in Section 7 on *Use Case*s of [18]). Stateful tokens and nonfungible tokens can be used and grant one-off, repeated, or prolonged use.

Academic Certificates:

Several projects [153][154] have enabled the issuance of tokens that represent diplomas and other academic certificates. Research and standardization efforts in the domain are currently being spearheaded by initiatives such as the Digital Credentials Consortium [155].

Public Organization Registrations:

Uniquely identifiable tokens can serve as proofs of registration for businesses and nonprofits (see *Verifying Business Identity* in Section 7 on *Use Case*s of [18]). This could enable a local government or chamber of commerce to maintain a smart contract registry upon which stateful tokens reflect legal organization filings. For example, this structure has been deployed by the governments of British Columbia and Ontario through the Verifiable Organizations Network [156] using Hyperledger Indy.

Cryptocollectibles:

Cryptocollectibles are nonfungible tokens that represent digital goods, such as trading cards (e.g., CryptoKitties [157]), artwork (Maecenas [158]), as well as video game artifacts and virtual land (e.g., Decentraland [159]).

Dataset Ownership and Sharing:

Nonfungible tokens can be used to uniquely identify content or datasets, enabling novel monetization and data sharing frameworks. For example, the Unlock protocol [160] uses non-fungible tokens compatible with the ERC-721 standard to offer membership access to specific resources (e.g., websites) and drive web content monetization. Tokens that identify datasets can also lay the foundations for incentive structures that encourage collaboration by rewarding contributors for access to their datasets or models. For example, the Ocean Protocol [161] offers privacy-preserving and role-based access control features for data marketplaces, where users can trade dataset ownership, as well as for *data commons*, where users can share datasets publicly.

6.4 Towards Privacy-Preserving Verifiable Data Exchange

Blockchain networks, second layer protocols, and privacy-enhancing techniques have the potential to enable *privacy-preserving verifiable data exchange* across organizations and on the web. As these ecosystems mature and new tokens are put in circulation, it could become easier for users to build tailored, verifiable proofs based on their tokens by way of aggregating otherwise fragmented identity artifacts directly from their wallet. Building on traditional token-based authorizations, this could ease user onboarding (see *Identity Documents, User Credentials, and Domain Names* in Section 6.3) and enable new peer-to-peer authentication methods.

Different types of proofs and bundles of proofs can be created independently of how the underlying tokens were originated and on a per-relying party or per-session basis, including:

- *Proofs of ownership* (e.g., account balances, licenses, certificates, property passes)
- *Proofs of collateral or stake* (e.g., security deposits)
- *Proofs of transfer* (e.g., payment receipts)
- *Proofs of participation* (e.g., voter stickers)
- *Proofs of origin or existence* (e.g., endorsed and timestamped documents)

The disclosure of these proofs can be achieved in a highly controlled manner where the data shared is minimized to what is strictly necessary without compromising integrity (see Section 4.3 on *Presentation Disclosure* and *Renting a Vehicle* in Section 7 on *Use Cases* in [18]). The proofs themselves can take the form of self-contained tokens for off-chain document exchange (see Section 2.2).

On-device processing techniques can be used to build smart wallets that help users offload the complexity of building multi-token proofs adapted to specific relying parties, user preferences, and contexts. Reciprocally, methods can be built on the relying party side to request specific information or advertise the types of proof that they expect to receive voluntarily from users. As shown in Figure 9, open standards and protocols for security and credit risk evaluation could help build new types of adaptive, passwordless access control systems, lending, or insurance services based on verifiable proofs. For example, consider delegated or uncollateralized loans. Users could convince lenders of their creditworthiness using token-based proofs acting as verifiable personal data sources (or "tokenized reputation" [162]). Verifiable device data could also be exchanged for devices equipped with HSM or TEE (see Section 3.1).

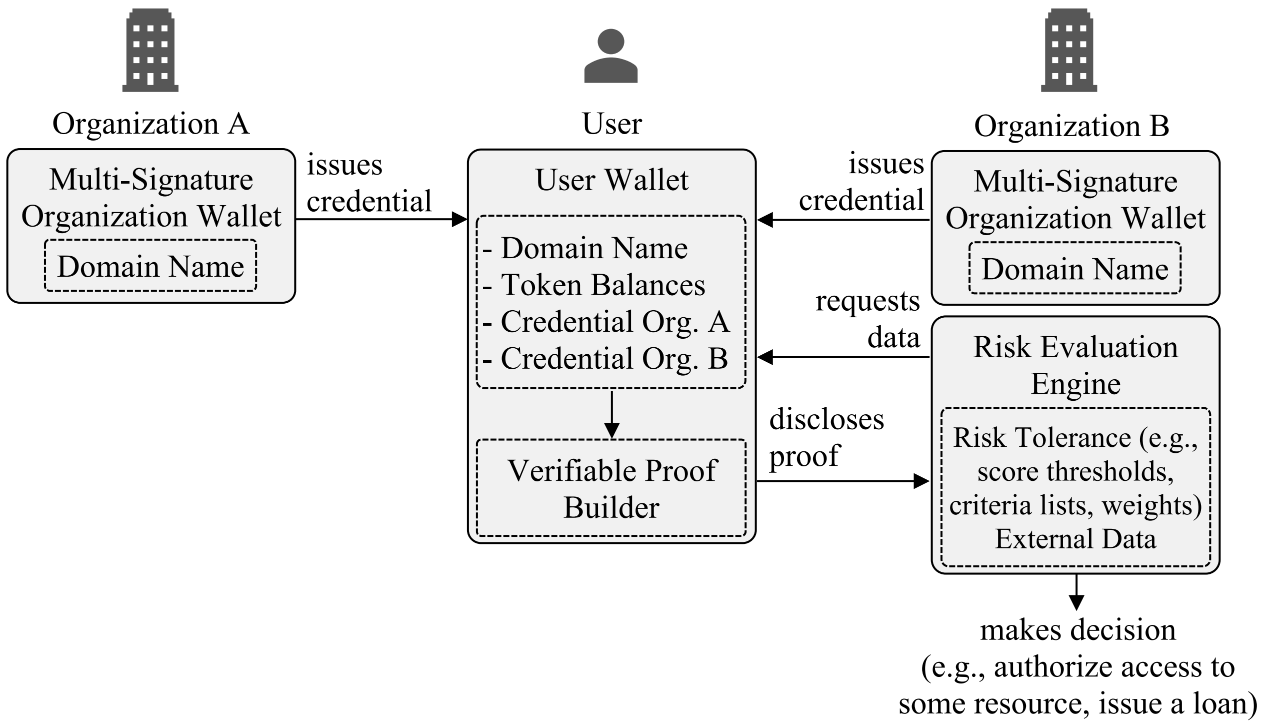

Figure 9: Verifiable Proof-Based Decision-Making

The development of open protocols for verifiable data exchange has a compounding effect that stems from each new protocol's ability to reuse verifiable data feeds provided by pre-existing protocols. For example, a collateral in one protocol can be reused to condition transactions in other systems.

More generally, the standardization of the components involved in token-based protocols, privacy-preserving verifiable data exchange methods, and cryptographically signed data feeds could lead to the emergence of a new interoperable digital framework to help reach agreements, control access to digital resources, and conduct business on the web. Giving data property rights to users that allow for trustworthy and privacy-preserving data sharing could benefit society through reduced data hoarding and more efficient data allocation [163]. In this context, the Linux Foundation has recently launched the Trust over IP Foundation (ToIP) [164] to build open standards and software for the trustworthy exchange and verification of data between any two parties on the web that encompass both technical and policy interoperability frameworks.

7 Conclusion

Tokens allow for the design of programmable digital assets that can represent different forms of ownership to enable users to store, move, and create value on top of shared or public digital infrastructures. With the ability to implement self-enforceable, built-in usage and governance features, tokens can act as coordination tools to achieve community objectives. By increasing reconciliation efficiency and providing verifiable data feeds across organizations and on the web, blockchain-based tokenization can serve as a foundation for new types of embedded services. They include but are not limited to payment systems, financial services, peer-to-peer authentication methods, shared business processes, and provable audit trails. This document is meant to share knowledge on current token design and management approaches and help the reader identify the logical components they comprise, both on-chain and off-chain.

This paper has provided a high-level technical overview of blockchain-based tokens by identifying key models, representations, and architectures. It first highlighted the different types of tokens and how they are held in custody. Then, it examined transaction management under three fundamental aspects: validation, submission, and viewability. Infrastructure components to help develop applications that integrate blockchain networks and second layer protocols were also reviewed. Finally, the paper presented deployment scenarios and use cases for tokens before concluding on potential breakthroughs for privacy-preserving verifiable data exchange.

The security, scalability, and privacy of token-based protocols are paired with the ability to sustainably deliver the necessary team or public efforts across organizational boundaries and to clearly articulate the vision and mission statements, trust assumptions, and supporting governance models. Data and process standardization is needed to provide clarity for building more interoperable protocols, developing supporting regulatory infrastructures for token ownership, and implementing software that handles complex and overwhelming tasks for users. The literature that has emerged on these challenges is rich, and efforts are being made to address them at an increasing pace. By relying on peer-to-peer networks and open standards instead of domain-specific and heterogeneous ecosystems, blockchain-enabled digital assets could bolster the accessibility and interoperability of financial, identity, authentication, and supply chain services. They have the potential to be integrated into third-party applications while maintaining data integrity and user control directly within their devices. This can facilitate online data exchange and transform business-making in partial- or zero-trust environments. Enabling more user-centric data security and privacy models can benefit both users and businesses. With many blockchain projects being explored or developed, organizations should consider what specific needs issuing tokenized representations of existing assets or creating new ones could help meet, who the parties involved are, which desirable features and processes the tokens should implement internally, and how they should be distributed and managed. In some cases, this pushes organizations to rethink their structures and approaches for identifying and managing risks. This includes finding alignments between individual and collective incentives and organizational design principles that allow for new efficiencies and joint opportunities.

References

[1] Yaga D, Mell P, Roby N, Scarfone K (2018) Blockchain Technology Overview. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) 8202. https://doi.org/10.6028/NIST.IR.8202

[2] Raskin M, Saleh F, Yermack D (2019) How Do Private Digital Currencies Affect Government Policy? *NYU Stern School of Business; NYU Law and Economics Research Paper No. 20-05.* Available at https://ssrn.com/abstract=3437529

[3] Bitcoin Wiki (2020) *Script*. Available at https://en.bitcoin.it/wiki/Script

[4] Ethereum.org (2020) *Ethereum Virtual Machine (EVM)*. Available at https://ethereum.org/en/developers/docs/evm

[5] Buterin V, Vogelsteller F (2015) *EIP 20: ERC-20 Token Standard*. Available at https://eips.ethereum.org/EIPS/eip-20

[6] Dafflon J, Baylina J, Shababi T (2017) *EIP 777: ERC-777 Token Standard*. Available at https://eips.ethereum.org/EIPS/eip-777

[7] Dossa A, Ruiz P, Vogelsteller F, Gosselin S (2018) *ERC-1410 Partially Fungible Token Standard (draft proposal)*. Available at https://github.com/ethereum/eips/issues/1410

[8] Gierlach R, Poole J, Borda M, Baker L (2018) *ERC-1404 Simple Restricted Token Standard (draft proposal)*. Available at https://github.com/ethereum/eips/issues/1404

[9] Diem (2020) *Move – Technical Paper*. Available at https://developers.diem.com/docs/technical-papers/move-paper

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

[11] Radomski W, Cooke A, Castonguay P, Therien J, Binet E, Sandford R (2018) *EIP 1155: ERC-1155 Multi Token Standard*. Available at https://eips.ethereum.org/EIPS/eip-1155

[12] InterWork Alliance (2020) *Token Taxonomy Framework*. Available at https://github.com/InterWorkAlliance/TokenTaxonomyFramework/blob/master/token-taxonomy.md

[13] InterWork Alliance (2020) *Token Taxonomy Framework - Book*. Available at https://github.com/InterWorkAlliance/TokenTaxonomyFramework/blob/master/TTF-Book.pdf

[14] InterWork Alliance (2020) *InterWork Alliance*. Available at https://interwork.org

[15] Luis O, Liudmila Z, Ingrid B, Gerhard S (2018) To Token or not to Token: Tools for Understanding Blockchain Tokens. *Zurich Open Repository and Archive* (University of Zurich, Zurich, Switzerland). Available at https://www.zora.uzh.ch/id/eprint/157908/1/To%20Token%20or%20not%20to%20Token_%20Tools%20for%20Understanding%20Blockchain%20Toke.pdf

[16] The Token Alliance of the Chamber of Digital Commerce (2019) *Considerations and Guidelines for Securities and Non-Securities Tokens*. Available at

https://digitalchamber.org/wp-content/uploads/2019/08/Security-Non-Security-Tokens.pdf

[17] Jones M, Bradley J, Sakimura N (2015) JSON Web Token (JWT) (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) 7519. https://doi.org/10.17487/RFC7519

[18] Lesavre L, Varin P, Mell P, Davidson M, Shook J (2020) A Taxonomic Approach to Understanding Emerging Blockchain Identity Management Systems. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Cybersecurity Whitepaper. Available at https://csrc.nist.gov/publications/detail/white-paper/2020/01/14/a-taxonomic-approach-to-understanding-emerging-blockchain-idms/final

[19] Sporny M, Longley D, Chadwick D (2019) *Verifiable credentials data model 1.0 – Expressing verifiable information on the Web* (W3C). Available at https://www.w3.org/TR/vc-data-model

[20] Reed D, Sporny M, Longley D, Allen C, Grant R, Sabadello M (2019) *Decentralized Identifiers (DIDs) v1.0 – Data Model and Syntaxes for Decentralized Identifiers* (W3C Credentials Community Group). Available at https://www.w3.org/TR/did-core

[21] Torstensson J (2017) *ERC-780 Ethereum Claims Registry (draft proposal)*. Available at https://github.com/ethereum/EIPs/issues/780

[22] Stehlik P, Vogelsang L (2018) *Privacy-Enabled NFTs: Self-Mintable Non-Fungible Tokens With Private Off-Chain Data*. Available at https://github.com/centrifuge/paper-privacy-enabled-nfts/releases/download/v1.01/paper-privacy-enabled-nfts.pdf

[23] Decentralized Identity Foundation (2020) *SideTree Protocol*. Available at https://identity.foundation/sidetree/spec

[24] Christos P, Samari K, Kiayias A, Roussopoulos M (2019) On the Practicality of a Smart Contract PKI. *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, pp 109-118. Available at https://arxiv.org/pdf/1902.00878.pdf

[25] International Organization for Standardization (2020) *ISO 22739:2020 - Blockchain and distributed ledger technologies — Vocabulary* (Technical Committee 307). Available at https://www.iso.org/standard/73771.html

[26] Hardman D (2019) *Aries RFC 0005: DID Communication*. Available at https://github.com/hyperledger/aries-rfcs/tree/master/concepts/0005-didcomm

[27] Sabadello M, Den Hartog K, Lundkvist C, Franz C, Elias A, Hughes A, Jordan J, Zagidulin D (2018) Introduction to DID Auth. *Rebooting the Web of Trust VI*. Available at https://github.com/WebOfTrustInfo/rwot6-santabarbara/blob/master/final-documents/did-auth.md

[28] Torus (2020) *DirectAuth*. Available at https://directauth.io

[29] Wuille P (2012) *BIP-32: Hierarchical Deterministic Wallets*. Available at https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

[30] User ByteCoin (2011) *Untraceable transactions which can contain a secure message are*

inevitable. Bitcoin Forum. Available at https://bitcointalk.org/index.php?topic=5965.0

[31] Brandão L, Mouha N, Vassilev A (2019) Threshold Schemes for Cryptographic Primitives: Challenges and Opportunities in Standardization and Validation of Threshold Cryptography," (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) 8214. https://doi.org/10.6028/NIST.IR.8214

[32] Shamir A (1979) How to share a secret. Communications of the ACM, Volume 22, Issue 11. https://doi.org/10.1145/359168.359176

[33] Blakley GR (1979) Safeguarding cryptographic keys. *1979 International Workshop on Managing Requirements Knowledge (MARK)* (IEEE), pp 313-318. Available at https://www.computer.org/csdl/proceedings-article/afips/1979/50870313/12OmNCeK2a1

[34] Gudgeon L, Moreno-Sanchez P, Roos S, McCorry P, Gervais A (2019) SoK: Off The Chain Transactions. *Cryptology ePrint Archive*, *Report 2019/360*. Available at https://ia.cr/2019/360

[35] Poon J, Dryja T (2016) *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*. Available at https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf

[36] Brainbot Labs Establishment (2020) *The Raiden Network*. Available at https://raiden.network

[37] Lind J, Naor O, Eyal I, Kelbert F, Gün Sirer E, Pietzuch P (2019) Teechain: A Secure Payment Network with Asynchronous Blockchain Access. *Proceedings of the 27th ACM Symposium on Operating Systems Principles,* pp 63-79. Available at https://arxiv.org/pdf/1707.05454.pdf

[38] Khalil R, Zamyatin A, Felley G, Moreno-Sanchez P, Gervais A (2018) Commit-Chains: Secure, Scalable Off-Chain Payments. *Cryptology ePrint Archive*, *Report 2018/642*. Available at https://eprint.iacr.org/2018/642.pdf

[39] Poon J, Buterin V (2017) *Plasma: Scalable Autonomous Smart Contracts*. Available at https://plasma.io/plasma.pdf

[40] Buterin V (2018) *Plasma Cash: Plasma with much less per-user data checking*. Available at https://ethresear.ch/t/plasma-cash-plasma-with-much-less-per-user-data-checking/1298

[41] Robinson D (2018) *Plasma Debit: Arbitrary-denomination payments in Plasma Cash*. Available at https://ethresear.ch/t/plasma-debit-arbitrary-denomination-payments-in-plasma-cash/2198

[42] Buterin V (2018) *Minimal Viable Plasma*. Available at https://ethresear.ch/t/minimal-viable-plasma/426

[43] Buterin V (2021) *An Incomplete Guide to Rollups*. Available at https://vitalik.ca/general/2021/01/05/rollup.html

[44] Herlihy M (2018) Atomic Cross-Chain Swaps. *Proceedings of the 2018 ACM symposium on principles of distributed computing*, pp 245-254. Available at https://arxiv.org/pdf/1801.09515.pdf

[45] Wyvern Protocol (2020) *Wyvern Protocol*. Available at https://wyvernprotocol.com

[46] OpenSea Inc (2020) *OpenSea*. Available at https://opensea.io

[47] Algorand (2020) *Algorand - Atomic Transfers*. Available at https://developer.algorand.org/docs/features/atomic_transfers

[48] Decred (2020) *Decred-compatible cross-chain atomic swapping*. Available at https://github.com/decred/atomicswap

[49] Elements Project (2019) *Adaptor Signatures and Atomic Swaps from Scriptless Scripts*. Available at https://github.com/ElementsProject/scriptless-scripts/blob/master/md/atomic-swap.md

[50] Web3 Foundation (2020) *XCMP – Relay chain light client design*. Available at https://research.web3.foundation/en/latest/polkadot/XCMP.html

[51] Tendermint Inc – Interchain Foundation (2020) *Cosmos - Inter-Blockchain Communication*. Available at https://cosmos.network/ibc

[52] Buterin V (2019) *Sidechains vs Plasma vs Sharding*. Available at https://vitalik.ca/general/2019/06/12/plasma_vs_sharding.html

[53] Web3 Foundation (2020) *Polkadot*. Available at https://polkadot.network

[54] Ethereum Foundation (2020) *Sharding-FAQs*. Available at https://eth.wiki/sharding/Sharding-FAQs

[55] Kannengießer N, Pfister M, Lins S, Sunyaev A (2020) Bridges between Islands: Cross-Chain Technology for Distributed Ledger Technology. *Proceedings of the 53rd Hawaii International Conference on System Sciences*. Available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3452714

[56] Jiang Y, Wang C, Wang Y, Gao L (2019) A Cross-Chain Solution to Integrating Multiple Blockchains for IoT Data Management. *Sensors, Volume 19* (Basel, Switzerland). Available at https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6539637

[57] Wanchain Foundation Ltd (2020) *Wanchain*. Available at https://www.wanchain.org

[58] Ethereum (2016) *BTC Relay*. Available at http://btcrelay.org

[59] Tendermint Inc (2020) *Cosmos*. Available at https://cosmos.network

[60] RSK Labs (2020) *RSK*. Available at https://www.rsk.co

[61] Andresen G, Hearn M (2013) *BIP-70: Payment Protocol*. Available at https://github.com/bitcoin/bips/blob/master/bip-0070.mediawiki

[62] Weiss Y, Tirosh D, Forshtat A (2018) *EIP 1613: ERC-1613 Gas stations network (EIP draft)*. Available at https://eips.ethereum.org/EIPS/eip-1613

[63] GSN alliance (2020) *Ethereum Gas Station Network*. Available at https://www.opengsn.org

[64] Bank for International Settlements - Committee on Payments and Market Infrastructures (2018) *Central bank digital currencies*. Available at https://www.bis.org/cpmi/publ/d174.pdf

[65] Sandhu R, Ferraiolo D, Kuhn R (2000) The NIST Model for Role-Based Access Control:

Towards A Unified Standard. *Proceedings of the Fifth ACM Workshop on Role-Based Access Control* (*RBAC 2000*) (Berlin, Germany). Available at https://csrc.nist.gov/publications/detail/conference-paper/2000/07/26/nist-model-for-rbac-towards-a-unified-standard

[66] OpenZeppelin (2020) *OpenZeppelin docs - Access Control*. Available at https://docs.openzeppelin.com/contracts/3.x/access-control

[67] Mell P, Delaitre A, De Vaulx F, Dessauw P (2019) Implementing a Protocol Native Managed Cryptocurrency. *The Fourteenth International Conference on Software Engineering Advances (ICSEA 2019)*. Available at https://www.nist.gov/publications/implementing-protocol-native-managed-cryptocurrency

[68] OpenZeppelin (2020) *TPL - Transaction Permission Layer*. Available at https://tplprotocol.org

[69] PegaSys (2020) *Permissioning Smart Contracts*. Available at https://github.com/PegaSysEng/permissioning-smart-contracts

[70] NIST – Cryptographic Technology Group (2020) *Privacy-Enhancing Cryptography - Project Overview*. Available at https://csrc.nist.gov/projects/pec

[71] EY Blockchain (2020) *Nightfall*. Available at https://github.com/EYBlockchain/nightfall

[72] ZoKrates (2020) *ZoKrates*. Available at https://zokrates.github.io

[73] Reitwiessner C (2017) *EIP 196: Precompiled contracts for addition and scalar multiplication on the elliptic curve alt_bn128*. Available at https://eips.ethereum.org/EIPS/eip-196

[74] Buterin V, Reitwiessner C (2017) *EIP 197: Precompiled contracts for optimal ate pairing check on the elliptic curve alt_bn128*. Available at https://eips.ethereum.org/EIPS/eip-197

[75] Aztec Protocol (2020) *Aztec Protocol - 1.0.0 specification*. Available at https://github.com/AztecProtocol/specification

[76] Béres F, Seres I, Benczúr András, Quintyne-Collins M (2020) Blockchain is Watching You: Profiling and Deanonymizing Ethereum Users. *Eprint arXiv preprint arXiv:2005.14051*. Available at https://arxiv.org/pdf/2005.14051.pdf

[77] BitcoinWiki (2020) *CoinJoin*. Available at https://en.bitcoin.it/wiki/CoinJoin

[78] Buterin V (2017) *Zk-SNARKs: Under the Hood*. (Medium). Available at https://medium.com/@VitalikButerin/zk-snarks-under-the-hood-b33151a013f6

[79] Chaum D (1984) Blind Signature System. *Advances in cryptology (Springer, Boston, USA)*. https://doi.org/10.1007/978-1-4684-4730-9_14

[80] Liu Y, Wang Q (2017) An E-voting Protocol Based on Blockchain. *Cryptology ePrint Archive*, *Report 2017/1043*. Available at https://eprint.iacr.org/2017/1043.pdf

[81] Rivest R, Shamir A, Tauman Y (2001) How to Leak a Secret. *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2001) (Springer, Berlin, Germany)*, pp 552-565. Available at

https://people.csail.mit.edu/rivest/pubs/RST01.pdf

[82] Pedersen T (1991) Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. *Annual International Cryptology Conference (Crypto 1991)* (Springer, Berlin, Germany), pp 129-140. https://doi.org/10.1007/3-540-46766-1_9

[83] Poelstra A, Back A, Friedenbach M, Maxwell G, Wuille P (2018) Confidential Assets. *International Conference on Financial Cryptography and Data Security (FC 2018)*, pp 43-63. https://doi.org/10.1007/978-3-662-58820-8_4

[84] Enterprise Ethereum Alliance (2020) *Enterprise Ethereum Alliance Client Specification v6*. Available at https://entethalliance.github.io/client-spec/spec.html

[85] Hyperledger Foundation (2020) Hyperledger Besu. Available at https://www.hyperledger.org/projects/besu

[86] PegaSys (2020) *Orion Private Transaction Manager*. Available at https://docs.orion.pegasys.tech/en/latest

[87] J.P. Morgan (2020) *Quorum*. Available at https://github.com/jpmorganchase/quorum

[88] J.P. Morgan (2020) *Tessera*. Available at https://github.com/jpmorganchase/tessera

[89] Secret Network (2020) *Secret Network*. Available at https://scrt.network

[90] Kosba A, Miller A, Shi E, Wen Z, Papamanthou C (2016) Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. *2016 IEEE symposium on security and privacy*, pp 839-858. Available at https://eprint.iacr.org/2015/675.pdf

[91] NuCypher (2020) The NuCypher Network. Available at https://www.nucypher.com/network

[92] Kalodner H, Goldfeder S, Chen X, Weinberg M, Felten E (2018) Arbitrum: Scalable, private smart contracts. *Proceedings of the 27th USENIX Security Symposium (USENIX Security 2018)*, pp 1353-1370. Available at https://www.usenix.org/conference/usenixsecurity18/presentation/kalodner

[93] World Economic Forum, Deloitte (2019) *The Next Generation of Data-Sharing in Financial Services: Using Privacy Enhancing Techniques to Unlock New Value*. Available at https://www2.deloitte.com/content/dam/Deloitte/lu/Documents/financial-services/lu-next-generation-data-sharinging-financial-services.pdf

[94] Hyperledger Foundation (2020) *Hyperledger Avalon*. Available at https://www.hyperledger.org/projects/avalon

[95] Enterprise Ethereum Alliance (2020) *Enterprise Ethereum Alliance Off-Chain Trusted Compute Specification v1.1*. Available at https://entethalliance.github.io/trusted-computing/spec.html

[96] Russinovich M, Ashton E, Avanessians C, Castro M, Chamayou A, Clebsch S, Costa M, Fournet C, Kerner M, Krishna S, Maffre J, Moscibroda T, Nayak K, Ohrimenko O, Schuster F, Schuster R, Shamis A, Vrousgou O, Wintersteiger C (2019) *CCF: A Framework for Building Confidential Verifiable Replicated Services*. Available at https://github.com/microsoft/CCF/blob/master/CCF-TECHNICAL-REPORT.pdf

[97] The Linux Foundation (2020) *Confidential Computing Consortium*. Available at

https://confidentialcomputing.io

[98] Graph Protocol (2020) *The Graph - A Query Protocol for Blockchains*. Available at https://thegraph.com

[99] Coinbase (2020) *Rosetta - Documentation*. Available at https://www.rosetta-api.org/docs/principles_introduction.html

[100] Decentralized Identity Foundation (2020) *Universal Resolver*. Available at https://github.com/decentralized-identity/universal-resolver

[101] Malhotra A, King D, Schwartz D, Zochowsli M (2020) *PayString Protocol*. Available at https://paystring.org/whitepaper.pdf

[102] Cáceres M, Denicola D, Koch Z, McElmurry R, Jacobs I, Solomakhin R (2019) *Payment Request API* (W3C). Available at https://www.w3.org/TR/payment-request

[103] Conway S, Hughes A, Ma M, Poole J, Riedel M, Smith S, Stöcker C (2019) *A DID for Everything - Attribution, Verification and Provenance for Entities and Data Items*. Available at https://nbviewer.jupyter.org/github/WebOfTrustInfo/rwot7-toronto/blob/master/final-documents/A_DID_for_everything.pdf

[104] Smith S, Gupta V (2018) *Decentralized Autonomic Data (DAD) and the three R's of Key Management* (Rebooting the Web of Trust VI). Available at https://github.com/WebOfTrustInfo/rwot6-santabarbara/blob/master/final-documents/DecentralizedAutonomicData.pdf

[105] Interledger Project (2020) *Interledger*. Available at https://interledger.org

[106] Finlay D, Marks E (2019) *EIP 2255: Wallet Permissions Standard (EIP draft)*. Available at https://eips.ethereum.org/EIPS/eip-2255

[107] Finlay D (2019) *Introducing Web3 Plugins*. (Medium - Metamask) Available at https://medium.com/metamask/introducing-the-next-evolution-of-the-web3-wallet-4abdf801a4ee

[108] Kinsley D (2019) *Kirby and the birth of wall-apps*. (Medium – Civil). Available at https://blog.joincivil.com/kirby-and-the-birth-of-wall-apps-bd6ce396e229

[109] WalletLink (2020) *WalletLink*. Available at https://www.walletlink.org

[110] WalletConnect (2020) *WalletConnect*. Available at https://www.walletconnect.org

[111] Web3Modal (2020) *Web3Modal*. Available at https://github.com/Web3Modal/web3modal

[112] Decentralized Identity Foundation (2019) *DIF Identity Hubs*. Available at https://github.com/decentralized-identity/identity-hub/blob/master/explainer.md

[113] Filecoin (2020) *Filecoin*. Available at https://filecoin.io

[114] Protocol Labs (2020) *IPFS*. Available at https://ipfs.io

[115] Storj Labs (2018) *Storj*. Available at https://storj.io

[116] 3Box (2020) *3Box*. Available at https://3box.io

[117] Berry C (2019) *The Open Oracle System*. (Medium - Compound) Available at

https://medium.com/compound-finance/announcing-compound-open-oracle-development-cff36f06aad3

[118] Zhang F, Cecchetti E, Croman K, Juels A, Shi E (2016) Town Crier: An Authenticated Data Feed for Smart Contracts. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security 2016*, pp 270-282. Available at https://eprint.iacr.org/2016/168.pdf

[119] Ellis S, Juels A, Nazarov S (2017) *ChainLink - A Decentralized Oracle Network*. Available at https://link.smartcontract.com/whitepaper

[120] Band Protocol (2020) *Band Protocol*. Available at https://bandprotocol.com

[121] Buterin V (2020) *Trust Models*. Available at https://vitalik.ca/general/2020/08/20/trust.html

[122] FATF (2019) *FATF Recommendation – Guidance for a Risk-Based Approach to Virtual Assets and Virtual Asset Service Provider*. Available at https://www.fatf-gafi.org/publications/fatfrecommendations/documents/guidance-rba-virtual-assets.html

[123] FSB (2019) *Crypto-assets: Work underway, regulatory approaches and potential gaps*. Available at https://www.fsb.org/2019/05/crypto-assets-work-underway-regulatory-approaches-and-potential-gaps/

[124] Buterin V (2019) *Control as Liability*. Available at https://vitalik.ca/general/2019/05/09/control_as_liability.html

[125] The Token Alliance - Chamber of Digital Commerce (2019) *Considerations and Guidelines for Advancing Cybersecurity in the Token Economy*. Available at https://digitalchamber.org/wp-content/uploads/2019/09/Cybersecurity-Understanding-Digital-Tokens.pdf

[126] Dingman W, Cohen A, Ferrara N, Lynch A, Jasinski P, Black P, Deng L (2019) Classification of Smart Contract Bugs Using the NIST Bugs Framework. *IEEE 17th International Conference on Software Engineering Research, Management and Applications* (SERA), pp 116-123. Available at https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8886793

[127] Dhaliwal E, Gurguc Z, Machoko A, Le Fevre G, Burke J (2018) *Token Ecosystem Creation*. (Outlier Ventures). Available at https://outlierventures.io/wp-content/uploads/2018/03/Token-Ecosystem-Creation-Outlier-Ventures-1.pdf

[128] McConaghy T (2018) *Towards a Practice of Token Engineering*. (Medium – Ocean Protocol). Available at https://blog.oceanprotocol.com/towards-a-practice-of-token-engineering-b02feeeff7ca

[129] Coinbase (2018) *The Emergence of Cryptoeconomic Primitives*. (Medium – The Coinbase Blog). Available at https://blog.coinbase.com/the-emergence-of-cryptoeconomic-primitives-14ef3300cc10

[130] Moin A, Gün Sirer E, Sekniqi K (2019) A Classification Framework for Stablecoin Designs. *Eprint arXiv preprint arXiv:1910.10098*. Available at https://arxiv.org/pdf/1910.10098.pdf

[131] Kyber Network, BitGo Inc, Republic Protocol (2019) *Wrapped Tokens*. Available at https://www.wbtc.network/assets/wrapped-tokens-whitepaper.pdf

[132] tBTC (2020) *tBTC Security Model*. Available at https://tbtc.network/developers/tbtc-security-model

[133] Cross-Chain Group (2020) *Introducing Cross-Chain Group*. Available at https://crosschain.group

[134] J.P. Morgan (2020) *J.P. Morgan Creates Digital Coin for Payments*. Available at https://www.jpmorgan.com/global/news/digital-coin-payments

[135] Libra Association Members (2020) *Libra - White Paper V2*. Available at https://libra.org/en-US/wp-content/uploads/sites/23/2020/04/Libra_WhitePaperV2_April2020.pdf

[136] Centre (2018) *Centre Whitepaper*. Available at https://www.centre.io/pdfs/centre-whitepaper.pdf

[137] Maker Foundation (2020) *The Maker Protocol: MakerDAO's Multi-Collateral Dai (MCD) System*. Available at https://makerdao.com/en/whitepaper

[138] cLabs Team (2020) *Celo: A Multi-Asset Cryptographic Protocol for Decentralized Social Payments*. Available at https://celo.org/papers/Celo__A_Multi_Asset_Cryptographic_Protocol_for_Decentralized_Social_Payments.pdf

[139] Bouchaud M, Lyons T, Saint Olive M, Timsit K (2020) *Central banks and the future of digital money*. Available at https://cdn2.hubspot.net/hubfs/4795067/Enterprise/ConsenSys-CBDC-White-Paper_final_2020-01-20.pdf?__hstc=148571112.4cd1f133b859c7d3248bcfb375378b8a.1580026541510.1580026541510.1580026541510.1&__hssc=148571112.2.1580026541511

[140] Koning J (2018) *Approaches to a Central Bank Digital Currency in Brazil*. Available at https://www.r3.com/wp-content/uploads/2018/11/CBDC_Brazil_R3.pdf

[141] The Digital Dollar Project (2020) *Digital Dollar Project*. Available at https://www.digitaldollarproject.org

[142] IBM (2020) *IBM Food Trust*. Available at https://www.ibm.com/blockchain/solutions/food-trust

[143] Krima S, Hedberg T, Barnard Feeney A (2019) Securing the Digital Threat for Smart Manufacturing: A Reference Model for Blockchain-Based Product Data Traceability. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Advanced Manufacturing Series 300-6. https://doi.org/10.6028/NIST.AMS.300-6

[144] Department of Homeland Security – 2018 Public-Private Analytic Exchange Program (2018) *Blockchain and Suitability for Government Applications*. Available at https://www.dhs.gov/sites/default/files/publications/2018_AEP_Blockchain_and_Suitability_for_Government_Applications.pdf

[145] Hyperledger Foundation (2020) *Supply Chain SIG*. Available at

https://wiki.hyperledger.org/display/SCSIG/Supply+Chain+SIG

[146] Hyperledger Foundation (2020) *Hyperledger Grid*. Available at https://wiki.hyperledger.org/display/GRID/Hyperledger+Grid

[147] Centrifuge (2020) *Centrifuge*. Available at https://centrifuge.io

[148] Wright A, Roon D, ConsenSys AG (2020) *OpenLaw*. Available at https://www.openlaw.io

[149] Oasis Open Projects (2020) *Baseline Protocol*. Available at https://docs.baseline-protocol.org

[150] uPort (2020) *uPort*. Available at https://www.uport.me

[151] Blockcerts (2020) *Blockcerts*. Available at https://www.blockcerts.org

[152] Hyperledger Foundation (2020) *Hyperledger Indy*. Available at https://www.hyperledger.org/use/hyperledger-indy

[153] Durant E, Trachy A (2017) *Digital Diploma debuts at MIT*. (MIT News). Available at https://news.mit.edu/2017/mit-debuts-secure-digital-diploma-using-bitcoin-blockchain-technology-1017

[154] Government of Malta (2019) *Press Release by the Office of the Prime Minister*. (doi.gov.mt). Available at https://www.gov.mt/en/Government/DOI/Press%20Releases/Pages/2019/February/21/pr190340.aspx

[155] Digital Credentials Consortium (2020) *Building the digital credential infrastructure for the future*. Available at https://digitalcredentials.mit.edu/wp-content/uploads/2020/02/white-paper-building-digital-credential-infrastructure-future.pdf

[156] VON (2019) Verifiable Organizations Network. Available at https://vonx.io

[157] Dapper Labs (2020) *CryptoKitties*. Available at https://www.cryptokitties.co

[158] Maecenas Fine Art (2020) *Maecenas*. Available at https://www.maecenas.co/whats-maecenas

[159] Decentraland (2020) *Decentraland*. Available at https://decentraland.org

[160] Unlock (2020) *Unlock Protocol*. Available at https://unlock-protocol.com

[161] Ocean Protocol Foundation (2020) *Ocean Protocol*. Available at https://docs.oceanprotocol.com/concepts/introduction

[162] Born C (2019) *Tokenized Reputation*. (Medium - TheCapital). Available at https://medium.com/the-capital/tokenized-reputation-dee463fbc631

[163] Jones C, Tonetti C (2020) Nonrivalry and the Economics of Data. *National Bureau of Economic Research*. Available at https://christophertonetti.com/files/papers/JonesTonetti_DataNonrivalry.pdf

[164] Joint Development Foundation Projects (2020) *Trust Over IP Foundation*. Available at https://trustoverip.org

[165] Nakamoto S (2008) *Bitcoin: A Peer-to-Peer Electronic Cash System*. Available at

https://bitcoin.org/bitcoin.pdf

[166] Proof of Stake Alliance (2020) *PoS Alliance*. Available at https://www.proofofstakealliance.org

[167] Buterin V (2015) *On Public and Private Blockchains*. Available at https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains

[168] ACT-IAC Emerging Technology Community of Interest - Blockchain Working Group (2018) *Blockchain Playbook for Federal U.S. Gover*nment. Available at https://www.actiac.org/act-iac-white-paper-blockchain-playbook-us-federal-government

[169] Solidity (2020) *Solidity Documentation*. Available at: https://docs.soliditylang.org

[170] WebAssembly Community Group (2020) *WebAssembly*. Available at https://webassembly.org

[171] National Institute of Standards and Technology (2006) Minimum Security Requirements for Federal Information and Information Systems. (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) 200. https://doi.org/10.6028/NIST.FIPS.200

Appendix A—Base Layer Consensus and Compute

This appendix provides a high-level overview of the different types of consensus services and computing environments in blockchain protocols. For more in-depth information, the reader is invited to read NISTIR 8202 [1].

Consensus models for blockchain are categorized into two types based on how they are meant to be used: permissionless and permissioned. Sybil attack resistance is achieved, respectively, through built-in cryptoeconomic incentives that enable nodes to work together in zero-trust environments and through access control, wherein nodes have to be authorized by system owners or consortium members. Note that consensus models provide a total ordering of all transactions but generally do not prevent nodes from choosing the order of transactions within the blocks they publish.

Permissionless consensus models are defined in NISTIR 8202 [1] as follows: "Since permissionless blockchain networks are open to all to participate, malicious users may attempt to publish blocks in a way that subverts the system [..]. To prevent this, permissionless blockchain networks often utilize a multiparty agreement or 'consensus' system [..] that requires users to expend or maintain resources when attempting to publish blocks. This prevents malicious users from easily subverting the system. Examples of such consensus models include proof-of-work and proof-of-stake methods. The consensus systems in permissionless blockchain networks usually promote non-malicious behavior through rewarding the publishers of protocol-conforming blocks with a native cryptocurrency."

In *proof-of-stake* consensus models, nodes compete for the right to publish a new block by staking tokens. While the node is active, the tokens are locked up and cannot be transferred. The greater the stake, the higher the chances of being designated block publisher during the next consensus round. The methodology used to designate the next block publisher varies from one system to another. For example, multiple nodes from the staking pool may propose blocks and vote for the winning block based on their respective stakes. Staking can take different forms, such as sending tokens to a deposit smart contract or holding them within a specific wallet software. Proof-of-stake consensus models consume fewer computational resources than their *proof-of-work* counterparts, based on solving computationally intensive problems, as in Bitcoin's Nakamoto consensus model [165]. Both of these types of consensus models enable transactions between untrusted participants in permissionless networks and are generally seen as providing a high immutability level when the network of nodes is sufficiently decentralized. Note that intentional blockchain forks can occur if the community comes to a governance impasse and splits up into two portions (see Section 5 in [1]). The reader may find relevant information about proof-of-stake regulation (out of scope for this paper) through the Proof of Stake Alliance [166].

Characteristically, permissioned consensus models do not rely on blockchain-native tokens. Scalability is generally increased, though this comes at the expense of reduced immutability expectations [167]. Rules may be changed and transactions reverted under certain circumstances and governance models. Based on how the list of authorized nodes is administered (see Section 4.2.3), permissioned consensus models are used in two types of blockchains: consortium blockchains and private blockchains. *Consortium blockchains* have a list of authorized nodes but do not involve exclusive governance by a central authority. They are sometimes referred to as partially decentralized since governance rights are shared among independent consortium members running nodes in the network. Unlike permissionless blockchains, the rules to become a node in the network are not deterministically and independently enforced by the blockchain protocol. There must generally be an explicitly defined external governance framework that organizes a community of known participants, though this framework can have on-chain features, such as access control or tokenized identity artifacts. Depending on business requirements and design characteristics, decentralizing governance is not always relevant. A central authority may be deemed trustworthy, a transition period preferable, or decentralization benefits not necessarily worth the costs. Blockchains with centralized governance (i.e., where a central authority selects the nodes or delegates that power), often referred to as *private blockchains,* have been used to build append-only distributed ledgers with fault tolerance and cryptographically verifiable transaction logs that are open for others to use without relinquishing control of the infrastructure. Note that a blockchain is not necessary to enable cryptographic authentication or obtain data integrity guarantees. For example, cryptographically signed data or self-contained tokens (see Section 2.2) can provide such guarantees on their own, and zero-knowledge proofs can add privacy features to them. Frameworks have been developed to assess whether a specific use case is suitable for blockchain [168]. Blockchain platforms can provide highly modular and configurable protocols (e.g., pluggable consensus models) that are intended to allow different types of use cases.

Per EEA's Client Specification [84], *transaction finality* "occurs when a transaction is definitely part of the blockchain and cannot be removed. A transaction reaches finality after some event defined for the relevant blockchain occurs. For example, an elapsed amount of time or a specific number of blocks added." In a consortium blockchain, where nodes are partially trusted to behave appropriately, transaction finality can usually be considered *deterministic* (or *absolute*). When deterministic, a transaction is deemed final as soon as it provably satisfies an explicit condition, such as being added to a block. In a permissionless blockchain, transaction finality is often *probabilistic*. The more blocks are added after a transaction is posted, the more final the transaction. This has fundamental ramifications for participants' expectations of data integrity and risks.

Additionally, blockchain protocols enable virtual machines for smart contracts that offer application-specific instruction sets (e.g., Bitcoin Script) or general-purpose programming languages (e.g., Solidity [169], Move [9]), which may be *Turing-complete*. Note that blockchain protocols' execution environments (e.g., Ethereum Virtual Machine [4], WebAssembly open standard (Wasm) [170]) do not interact with the smart contracts themselves but with the bytecodes that the smart contracts compile into. High expressiveness can enable a variety of use cases, but low expressiveness may be a desirable feature too depending on security models. Note that certain schemes described in this paper that are built upon smart contracts, such as HTLCs, do not require Turing-complete scripting support.

Appendix B—Acronyms

Selected acronyms and abbreviations used in this paper are defined below.

API	Application Programming Interface
CBDC	Central Bank Digital Currency
DAD	Decentralized Autonomic Data
DEX	Decentralized Exchange
DID	Decentralized Identifier
DLT	Distributed Ledger Technology
D2D	Device-To-Device
EEA	Enterprise Ethereum Alliance
ENS	Ethereum Name Service
ERC	Ethereum Request for Comment
HD	Hierarchical Deterministic
HSM	Hardware Security Module
HTLC	Hashed Timelock Contract
ISO	International Organization for Standardization
ITL	Information Technology Laboratory
JSON	JavaScript Object Notation
JWT	JSON Web Token
LTE	Long-Term Evolution
MPC	Multi-Party Computation
NIST	National Institute of Standards and Technology
NIST-IR	National Institute of Standards and Technology Internal Report
NFC	Near-Field Communication
NFT	Nonfungible Token
TEE	Trusted Execution Environment
TTI	Token Taxonomy Initiative
UTXO	Unspent Transaction Output
WLAN	Wireless Local Area Network
W3C	World Wide Web Consortium
ZKP	Zero-Knowledge Proof

Appendix C—Glossary

Account	An entity in a blockchain that is identified with an address and can send transactions to the blockchain.
Address [1]	A short, alphanumeric string derived from a user's public key using a hash function, with additional data to detect errors. Addresses are used to send and receive digital assets.
Airdrop [18]	A distribution of digital tokens to a list of blockchain addresses.
Atomic Swap	An exchange of tokens that does not involve the intervention of any trusted intermediary and automatically reverts if all of the provisions are not met.
Authentication [171]	Verifying the identity of a user, process, or device, often as a prerequisite for allowing access to resources in an information system.
Blockchain [1]	Blockchains are distributed digital ledgers of cryptographically signed transactions that are grouped into blocks. Each block is cryptographically linked to the previous one (making it tamper evident) after validation and undergoing a consensus decision. As new blocks are added, older blocks become more difficult to modify (creating tamper resistance). New blocks are replicated across copies of the ledger within the network, and any conflicts are resolved automatically using established rules.
Blockchain Explorer	A software for visualizing blocks, transactions, and blockchain network metrics (e.g., average transaction fees, hashrates, block size, block difficulty).
Blockchain Subnetwork	A blockchain network that is tightly coupled with one or more other blockchain networks, as found in sharding.
Commit-Chain	A scheme that enables the off-chain processing of transactions by one or more operators with on-chain state update commitments that do not contain per-transaction data.
Consensus Model [1]	A process to achieve agreement within a distributed system on the valid state.
Consortium	A group of organizations or individuals with the objective of mutualizing resources for achieving a common goal (e.g., operating a consortium blockchain).
Cryptocurrency [1]	A digital asset/credit/unit within the system, which is cryptographically sent from one blockchain network user to another. In the case of cryptocurrency creation (such as the reward for mining), the publishing node includes a transaction sending the newly created cryptocurrency to one or more blockchain network users.

Custodian	A third-party entity that holds and safeguards a user's private keys or digital assets on their behalf. Depending on the system, a custodian may act as an exchange and provide additional services, such as staking, lending, account recovery, or security features.
Fungible	Refers to something that is replaceable or interchangeable (i.e., not uniquely identifiable).
Hash [1, Adapted]	The output of a hash function (e.g., hash(data) = digest). Also known as a message digest, digest, hash digest, or hash value.
JSON Web Token [17, Adapted]	A data exchange format made of a header, payload, and signature where the header and the payload take the form of JSON objects. They are encoded and concatenated with the aggregate being signed to generate a signature.
Merkle Tree [1]	A data structure where the data is hashed and combined until there is a singular root hash that represents the entire structure.
Mint	A protocol-level operation that creates and distributes new tokens to blockchain addresses, either individually or in batch.
Multi-Signature	A cryptographic signature scheme where the process of signing information (e.g., a transaction) is distributed among multiple private keys.
Non-Custodial	Refers to an application or process that does not require users to relinquish any control over their data or private keys.
Nonfungible [18]	Refers to something that is uniquely identifiable (i.e., not replaceable or interchangeable).
Notary	A trusted entity that submits transactions across blockchains on behalf of users, often with respect to tokens the users have previously locked up.
Off-Chain [18]	Refers to data that is stored or a process that is implemented and executed outside of any blockchain system.
On-Chain [18]	Refers to data that is stored or a process that is implemented and executed within a blockchain system.
Oracle [18]	A source of data from outside a blockchain that serves as input for a smart contract.
Permissioned [1]	A system where every node, and every user must be granted permissions to utilize the system (generally assigned by an administrator or consortium).
Permissionless [1]	A system where all users' permissions are equal and not set by any administrator or consortium.
Permissions [1]	Allowable user actions (e.g., read, write, execute).

Relay	A scheme deployed on a given blockchain to receive and verify transactions from another blockchain.
Resolver [18]	Software that retrieves data associated with some identifier.
Rollup	A scheme that enables the off-chain processing of transactions by one or more operators with on-chain state update commitments that contain “compressed” per-transaction data.
Separation of Concerns	A design principle for breaking down an application into modules, layers, and encapsulations, the roles of which are independent of one another.
Sharding	A blockchain configuration and architecture that enables the processing of transactions in parallel. The blockchain’s global state is split among multiple blockchain subnetworks coordinated by a separate hub blockchain.
Sidechain	A blockchain with its own consensus mechanism and set of nodes that is connected to another blockchain through a two-way bridge.
Staking	Protocol-defined token collateralization earning yields and/or providing privileges, either at the base layer (in proof-of-stake consensus models) or at the smart contract layer.
State Channel	A scheme that enables the off-chain processing of transactions by a group of participants with instant second layer finality and deferred on-chain settlement via state updates.
State Update	An on-chain transaction used to anchor the current state of an external ledger onto the underlying blockchain.
Stateful	Refers to a data representation or a process that is dependent on an external data store.
Stateless	Refers to a data representation or a process that is self-contained and does not depend on any external data store.
Smart Contract [1]	A collection of code and data (sometimes referred to as functions and state) that is deployed using cryptographically signed transactions on the blockchain network. The smart contract is executed by nodes within the blockchain network; all nodes must derive the same results for the execution, and the results of execution are recorded on the blockchain.
Sybil Attack	A cybersecurity attack wherein an attacker creates multiple accounts and pretends to be many persons at once.
Token [18]	A representation of a particular asset that typically relies on a blockchain or other types of distributed ledgers.
Token Factory Contract	A smart contract that defines and issues a token.

Transaction [1, Adapted]	A recording of an event, such as the transfer of tokens between parties, or the creation of new assets.
Transaction Fee [1, Adapted]	An amount of cryptocurrency charged to process a blockchain transaction. Given to publishing nodes to include the transaction within a block.
Wallet [25]	An application used to generate, manage, store or use private and public keys. A wallet can be implemented as a software or hardware module.
Zero-Knowledge Proof [18]	A cryptographic scheme where a prover is able to convince a verifier that a statement is true, without providing any more information than that single bit (that is, that the statement is true rather than false).