

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-draft>

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-draft>

September 2020



U.S. Department of Commerce
Wilbur L. Ross, Jr., Secretary

National Institute of Standards and Technology
Walter Copan, NIST Director and Under Secretary of Commerce for Standards and Technology

National Institute of Standards and Technology Interagency or Internal Report 8301
82 pages (September 2020)

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

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at <https://csrc.nist.gov/publications>.

Public comment period: *September 29, 2020 through October 30, 2020*

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

All comments are subject to release under the Freedom of Information Act (FOIA).

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems.

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 have the ability to directly control token custody in digital wallets through public-key cryptography and to 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 or virtual machines. Data models with varied capabilities and scopes have been defined to issue tokens, which additional protocols can help manage while allowing for separation of concerns. Security and recovery mechanisms make it possible for 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.

108 **Acknowledgments**

109 The authors wish to thank all contributors to this publication and their colleagues who reviewed
110 drafts of this report and contributed technical and editorial additions. This includes NIST staff
111 Michael Davidson, Aurélien Delaitre, Katya Delak, Frédéric De Vault, and James Dray.

112 **Audience**

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

122 **Trademark Information**

123 All registered trademarks and trademarks belong to their respective organizations.

124 **Note to Reviewers**

125 Suggestions for improvements are welcomed on any aspects of this draft and across relevant
126 domains, with special appreciation for those that can help ensure it adequately and appropriately:

- 127 • Defines and differentiates the main types of token data models,
- 128 • Explains how following common token data models allows for tokens to be composed with
129 one another and for protocols that enable more advanced operations to be built,
- 130 • Helps make sense of where blockchain-based tokenization on top of mutualized
131 infrastructures fits in the wider landscape of web applications, especially in terms of how
132 new types of network and protocol governance can be implemented, and
- 133 • Identifies and describes the main differences between transaction management techniques
134 at the second layer as well as key components and approaches for infrastructure
135 management.

136

Call for Patent Claims

137 This public review includes a call for information on essential patent claims (claims whose use
138 would be required for compliance with the guidance or requirements in this Information
139 Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be
140 directly stated in this ITL Publication or by reference to another publication. This call also includes
141 disclosure, where known, of the existence of pending U.S. or foreign patent applications relating
142 to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

143 ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in
144 written or electronic form, either:

145 a) assurance in the form of a general disclaimer to the effect that such party does not hold and
146 does not currently intend holding any essential patent claim(s); or

147 b) assurance that a license to such essential patent claim(s) will be made available to
148 applicants desiring to utilize the license for the purpose of complying with the guidance or
149 requirements in this ITL draft publication either:

150 i. under reasonable terms and conditions that are demonstrably free of any unfair
151 discrimination; or

152 ii. without compensation and under reasonable terms and conditions that are
153 demonstrably free of any unfair discrimination.

154 Such assurance shall indicate that the patent holder (or third party authorized to make assurances
155 on its behalf) will include in any documents transferring ownership of patents subject to the
156 assurance, provisions sufficient to ensure that the commitments in the assurance are binding on
157 the transferee, and that the transferee will similarly include appropriate provisions in the event of
158 future transfers with the goal of binding each successor-in-interest.

159 The assurance shall also indicate that it is intended to be binding on successors-in-interest
160 regardless of whether such provisions are included in the relevant transfer documents.

161 Such statements should be addressed to: blockchain-token-paper@nist.gov

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 have the ability to directly control token custody in digital wallets through public-key cryptography and to 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 that are either application-specific, offering limited instruction sets, or general-purpose, allowing smart contract execution.

These programming environments make it possible to issue tokens that represent programmable digital assets, the ownership of which is cryptographically verifiable, and to develop services to help manage them. Tokens meant to act as interchangeable units represent digital coins. Those meant to act as uniquely identifiable objects represent nonfungible assets. Protocols primarily use fungible tokens (i.e., digital coins) to build incentive and governance models for permissionless peer-to-peer networks, represent existing fungible assets, or derive new ones based on them. Tokens can also be self-contained and use blockchain-based storage for status updates. They enable authentication and authorization methods that can be used to provide additional features for blockchain-based tokens as well as to build identity and supply chain management systems.

Open standards for token data models have been developed that define operations at the protocol level for token creation and supply/lifecycle management and at the user 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 that are independent from token issuers. Smart contract vaults can enable tailored account management models with additional security and recovery features while externally maintaining persistent blockchain addresses.

Operations modify the state of the ledger by way of transactions submitted to the blockchain, which provides reconciliation but requires making tradeoffs between decentralization, scalability, and security. Parallel transaction processing and off-chain scaling schemes have been developed to increase transaction throughput. State channels and sidechains allow transaction processing to be offloaded away from the root blockchain. By attaching agreed-upon and self-enforceable conditions to deposit contracts, tokens can be exchanged with one another while users remain in control of the private keys at all times. Blockchain bridging schemes allow for the portability of tokens and oracles 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 the use of privacy-enhancing technologies and cryptographic primitives is still needed to protect the confidentiality of user data.

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

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

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

Table of Contents

221			
222			
223	Executive Summary	v	
224	1 Introduction	1	
225	1.1 Background.....	1	
226	1.2 Purpose and Scope	2	
227	1.3 Notes on Terms	3	
228	1.4 Disclaimers and Clarifications	3	
229	1.5 Document Structure	3	
230	2 Token Categorization.....	4	
231	2.1 Blockchain-Based Tokens	4	
232	2.1.1 Token Data Models	4	
233	2.1.2 Protocol Management	7	
234	2.1.3 User-Level Operations.....	8	
235	2.2 Self-Contained Tokens	8	
236	3 Wallet and Key Management.....	10	
237	3.1 Self-Hosted Wallets	11	
238	3.2 Custodial Wallets	13	
239	3.3 Account Origination and Recovery.....	14	
240	3.4 Smart Contract Vaults.....	15	
241	4 Transaction Management.....	17	
242	4.1 Transaction Validation	17	
243	4.1.1 Off-Chain Scaling	17	
244	4.1.2 Token Exchange	22	
245	4.1.3 Blockchain Bridging	26	
246	4.2 Transaction Submission.....	28	
247	4.2.1 Meta Transactions.....	29	
248	4.2.2 Smart Contract-Based Access Control	29	
249	4.2.3 Blockchain Node Permissioning	30	
250	4.3 Transaction Viewability	30	
251	4.3.1 Monitoring and Analysis Tools.....	31	
252	4.3.2 Privacy-Enhancing Techniques	31	
253	4.3.3 Computation on Encrypted Data	33	

254	5 Infrastructure Management	36
255	5.1 Blockchain Networks Integration	36
256	5.1.1 Base Layer	36
257	5.1.2 Second Layer	37
258	5.1.3 Open Connectors and Interfaces	38
259	5.2 Wallet Integration	40
260	5.3 User Account Data Integration	40
261	5.4 External Data Feeds Integration	41
262	5.5 Architectural Considerations	42
263	6 Deployment Scenarios and Use Cases	44
264	6.1 Decentralizing Protocol Governance	44
265	6.2 Tokenizing Money and Financial Products	46
266	6.2.1 Stablecoins	46
267	6.2.2 Lending and Borrowing	48
268	6.2.3 Fundraising and Derivatives	48
269	6.3 Tokenizing Uniquely Identifiable Things and Supply Chains	49
270	6.4 Towards Privacy-Preserving Verifiable Data Exchange	52
271	7 Conclusion	54
272	References	55

273
274

List of Appendices

275	Appendix A— Base Layer Consensus and Compute	66
276	Appendix B— Acronyms	68
277	Appendix C— Glossary	69

278

279

List of Figures

280	Figure 1: Blockchain-Wallet Coupling	5
281	Figure 2: State-Dependent Storage Methods	10
282	Figure 3: Payment Channel Phases	19
283	Figure 4: Hashed Timelock Contract Transfer Flow	23
284	Figure 5: Hub-and-Spoke Architecture	26
285	Figure 6: Relay Blockchain	28

286	Figure 7: Private Transaction Execution.....	34
287	Figure 8: Blockchain Node Types	36
288	Figure 9: Smart Contract Multi-Sided Platforms	45
289	Figure 10: Verifiable Proof-Based Decision-Making	52

290

291

List of Tables

292	Table 1: Emerging Blockchain Computer Stack	2
293	Table 2: Token Representation Types	6
294	Table 3: Off-Chain Scaling Schemes Comparison	18

295

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 that affect both 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 have the ability to directly control token custody in digital wallets through public-key cryptography and to 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, act as a publishing node, and submit transactions. Blockchain addresses are derived from public keys generated directly by the users who control the custody of the associated private keys. Transactions are signed using these private keys before being validated and reconciled by the blockchain nodes, thereby providing integrity and verifiability.

Permissionless blockchains brought about protocol-native tokens—*cryptocurrencies*—as well as tamper-evident and tamper-resistant computing platforms, or virtual machines (Layer 1 in Table 1). With the potential to provide alternatives to existing institutions and market discipline, ownerless/non-sovereign cryptocurrencies could have long-term implications for financial inclusion and stability [2]. Blockchain-enabled virtual machines offer limited instruction sets, such as Bitcoin Script, or provide general-purpose programming environments, such as the Ethereum Virtual Machine, allowing *smart contract* execution. This forms Layer 2 in Table 1, which makes it possible to deploy different types of tokens and management services. Tokens represent digital assets and serve as instruments for exchanging verifiable data. *Fungible tokens* are meant to be completely interchangeable (i.e., *digital coins*, 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 tokens* and *utility tokens*. Tokens associated with unique identifiers, *nonfungible tokens* and *stateful tokens*, are meant to uniquely identify things or data. 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 then integrated into user interfaces or middleware at Layer 3.

337

Table 1: Emerging Blockchain Computer Stack

<i>Layer 3</i>	User Interfaces
<i>Layer 2</i>	Smart Contracts and Off-Chain Resources
<i>Layer 1</i>	Consensus and Compute ¹
<i>Layer 0</i>	Hardware and Networking

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

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

351 **1.2 Purpose and Scope**

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

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

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

¹ Consensus and compute are decoupled in some blockchain protocols where distinct roles and tasks are assigned to different groups of nodes or subnetworks.

1.3 Notes on Terms

For the purposes 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.

1.4 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, and for concepts where smart contracts are mentioned, alternatives techniques and cryptographic schemes may be used instead. In general, blockchain technologies come with their own sets of tradeoffs and levels of maturity. 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. This paper does not attempt to provide an in-depth coverage of all developments, answer all questions, or judge between the different techniques, architectures, and models. It instead describes key concepts and highlights important differences to support its stated previously purpose. Regulatory compliance is also out of scope for this paper.

1.5 Document Structure

The rest of this document is organized as follows:

- **Section 2** provides a categorization of 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 the management of wallets, public-private key pairs, and accounts.
- **Section 4** discusses schemes to execute transactions off-chain and across blockchains, to exchange tokens, and to manage their 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 data models, protocol-level operations, and user-level operations for blockchain-based tokens.

2.1.1 Token Data Models

There are two main types of blockchain-based tokens that represent digital assets: fungible tokens and nonfungible tokens.

Fungible Tokens:

A *fungible token* is a data representation that assigns balances to blockchain addresses (e.g., user accounts) through public-key cryptography with programmable supply management. Token units are meant to serve as *digital coins*. They are interchangeable quantitative data with a double spending prevention mechanism. Note that token units may be rendered nonfungible if they can be uniquely identified through an analysis of the transfer history. 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). Data structures for fungible tokens also include fields for protocol-specific metadata, such as the token name, symbol, issuer address, total supply, and number of decimals of precision. Issuers are either externally owned accounts or other smart contracts.

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

Open standards that provide interfaces for token factory contracts (i.e., defining operations and events) are often followed, such as those introduced as Ethereum Requests for Comments (ERCs). ERC-20 [3] is currently the most commonly used de facto standard, while ERC-777 [4] allows the passing of arbitrary data in token transfers to trigger external function calls. ERC-1410 [5] and ERC-1404 [6] support compliance requirements (e.g., withdrawal restrictions).

Nonfungible Tokens:

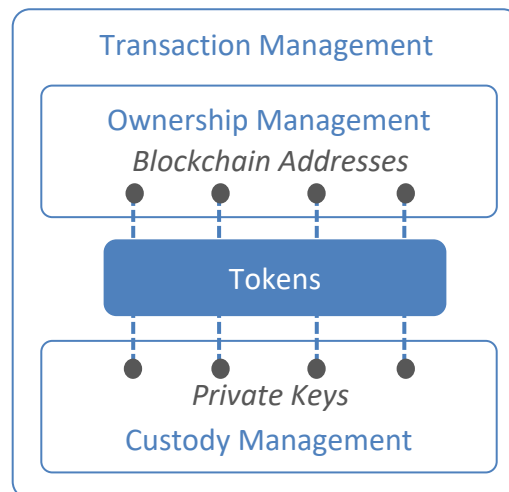
A *nonfungible token* is a data representation that assigns uniquely identified and uniformly formatted qualitative data objects to blockchain addresses (e.g., user accounts) through public-key cryptography with programmable lifecycle management. Token objects are usually accompanied by off-chain metadata, the integrity of which can be verified through on-chain cryptographic hashes (see Section 5.3). Transferring a nonfungible token object means reassigning the owner's account (see Section 2.1.3). Data structures for nonfungible tokens also include fields for protocol-specific metadata.

Motives for using nonfungible tokens vary depending on use cases (further discussed in Section 6.3, *Tokenizing Uniquely Identifiable Things and Supply Chains*). They are often colloquially referred to by their acronym, NFTs.

Open standards that provide interfaces for token factory contracts are often followed, such as ERC-721 [7]. ERC-1155 [8] grants rights for both fungible and nonfungible tokens from the same interface.

Representation Types:

At their core, tokens are entries in distributed ledgers that can be owned through public-key cryptography, ensuring authenticity and preventing modification and tampering without consent. Token transactions must be signed with the owner's private keys to be validated, encapsulated, and published to the ledger as blocks by blockchain nodes. Private keys are held in custody in digital wallets (see Section 3). In that way, tokens can be seen as a mapping between blockchain addresses and private keys, as shown in Figure 1. They enable granular representations of provable digital ownership claims.

Blockchain-Side Data and Operations Management**Wallet-Side Data and Operations Management****Figure 1: Blockchain-Wallet Coupling**

There are two record-keeping models that blockchain protocols use to represent tokens: 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 deployed on top of an existing blockchain protocol (e.g., via a smart contract). Table 2 summarizes the four resulting token representation types. Blockchain-native tokens are meant to be fungible while tokens implemented on top of existing blockchains are meant to be either fungible or nonfungible.

Table 2: Token Representation Types

	Blockchain-Native (Layer 1)	On Top of an Existing Blockchain (Layer 2)
UTXO-Based	Account balances are encoded as the sums of unspent transaction outputs of past transactions. Spending a token results in new, unspent transaction outputs. As an example, bitcoin is the native token in the Bitcoin protocol.	Account balances or unique identifiers are encoded by a separate protocol into extra metadata included in unspent transaction outputs of past transactions.
Account-Based	Account balances are stored as variables assigned to blockchain addresses in the blockchain's global state. As an example, ether is the native token in the Ethereum protocol.	Account balances or unique identifiers are stored as variables assigned to blockchain addresses in token factory contracts. This paper particularly focuses on this more general-purpose representation.

Emerging Taxonomies and Frameworks:

The Token Taxonomy Initiative (TTI) has published a draft framework called the *Token Taxonomy Framework* [9]. It characterizes tokens using base types, behaviors, and property sets, which serve as the bases to generate common, blockchain-agnostic specifications [10]. The Token Taxonomy Framework was absorbed by the InterWork Alliance [11] 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 [12] provides another token taxonomy based on four types of attributes: 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* [13], 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 *user-level operations* for individual tokens, which are discussed in Section 2.1.3. Note that for nonfungible tokens, more advanced user-level operations vary depending on use cases.

Supply/Lifecycle Management:

The *mint* operation creates and distributes new token units or objects to users. Once assigned to users, they become available for circulation. Each use case or specific implementation has its own token distribution and supply or lifecycle management model. Minting can be conducted individually or in batch and according to two approaches:

- **Push-Based:** For fungible tokens, the mint operation increases account balances (along with the total token supply, depending on the token data model). For nonfungible tokens, the mint operation instantiates and assigns new token objects with unique identifiers to users. In general, push-based minting does not involve any action or approval from users.
- **Pull-Based:** The mint operation gives individual accounts minting rights or generates “mint request” tokens, which give minting rights 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 users in a single transaction, and users can keep that right until it is exercised. For example, consider the periodic minting of fungible tokens (e.g., interests, dividends). A user could skip claiming tokens between period X and period $X + Y$ before withdrawing all the tokens earned between periods X and $X + Y + 1$ in a single transaction at period $X + Y + 1$.

The *burn* operation destroys existing token units or objects as part of protocol-defined conditions (e.g., when redeemed for underlying assets, tokens are taken out of circulation). Protocols may also involve transaction fees that burn tokens (e.g., to prevent denial-of-service attacks).

The *update* and *revoke* operations for nonfungible tokens are specific to use cases and out of scope for this paper.

Protocol Restrictions:

Partially decentralized or centralized governance models may involve restrictions on user-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 temporarily disable transfers or other user-level operations. 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 User-Level Operations

Transfer and delegation are base operations directly built as functions implemented into token data models (see Section 2.1.1). Although the exact scope of token data models varies, more advanced user-level operations can be built as separate protocols. 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:

Tokens 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). Push payments consist of sending fungible tokens to other blockchain addresses. They include one-time payments, recurring payments, or streams/continuous payments (e.g., for subscriptions, memberships, payrolls). 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 said tokens in a verifiable way (i.e., the transaction that processed the transfer provides proof of ownership renunciation). Tokens can also be transferred to smart contract vaults (see Section 3.4) for collateralization and staking. They can then be redeemed later on.

Delegation:

Single-use, conditioned, or permanent authorizations can be granted by an owner to delegate access of certain user-level operations to third parties and on a per-token basis. For example, the “allowance” and “approval” operations in the ERC-20 standard make it possible to request payments (i.e., pull payments) and share an account with another entity. 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 JSON Web Token format (often referred to by the acronym, JWT), standardized under RFC 7519 [14].

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 having their data stores built 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 [15]). 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 protocols² or cryptographic accumulators.³ The *verifiable credential* standard [16] 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 [17], and the blockchain-based data stores that they use can have architectures similar to the ones discussed above and further examined in [15]. They may also be complemented by smart contracts for public credentials registries, such as those introduced in ERC-780 [18].

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 personal paper checks) that allow the withdrawal of 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 be used to create mint requests used to claim blockchain-based tokens, as in [19] 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 [165].

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 [166], allowing for confidential status querying.

3 Wallet and Key Management

This section discusses the custody of tokens, which means the access to the accounts that they are assigned to, themselves secured by one or more private keys. 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).

The management and custody of credentials 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 providing them with any options or explanations about how (or by whom) their credentials were managed. However, users are now able to independently manage their own private keys used to transact on blockchain and DLT systems. They can choose to fully secure and store their private keys themselves or to utilize a third-party custodian to manage their private keys on their behalf. Furthermore, account custody models have been developed that enable key management abstractions for users without relinquishing control.

ISO/TC 307's *ISO 22739:2020 Blockchain and distributed ledger technologies - Vocabulary* [20] 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 meant to be used as shown in Figure 2:

- *Hot Wallet*: A wallet that is connected to the internet and meant 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 meant to be highly secure (containing high-value/at-rest tokens). Physical human intervention or authentication is required to sign transactions (e.g., pushing a button, entering a local pin, scanning a 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).

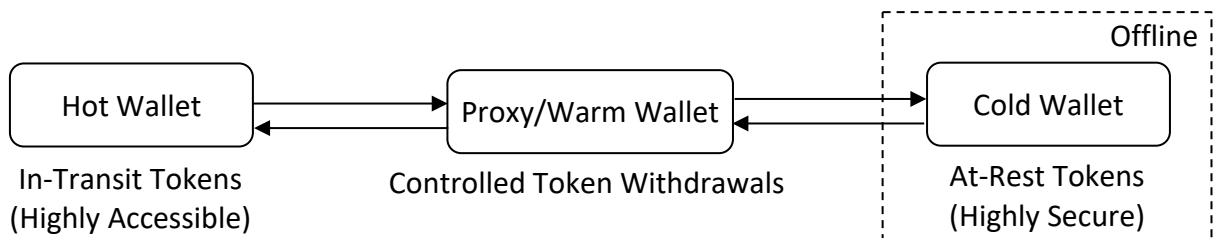


Figure 2: State-Dependent Storage Methods

3.1 Self-Hosted Wallets

In a fully user-controlled, self-hosted wallet scenario, key generation and management, secure storage, backup, and restore functions lie completely with the user. 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; 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 key results in the loss of all associated tokens. 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 that allow the user to securely sign transactions. 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 that is potentially associated with the tokens. Transactions may be initiated directly from software wallets or from separate applications on the same device or another device. For example, a smartphone application may be used to sign transactions initiated on a website accessed with a 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).

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, 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. There exists a variety of integrity and genuineness mechanisms, such as secure inputs and secure display (i.e., “what you see is what you sign”). Being able to validate the integrity of those devices is essential and consists of verifying the digital signatures of the suppliers and security updates. Some hardware wallets come with two-factor authentication (2FA), biometrics authentication, and companion applications.

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 owner of the device and is used to decrypt data in the TEE.

651 Device-to-Device Communications:

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

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

664 Multi-layered communication specifications and frameworks have been developed to provide
665 interoperable and agnostic data transport methods for blockchains networks. For instance,
666 DIDComm [21] and DID Auth [22] use DIDs to exchange verifiable, machine-readable messages
667 and handle authentication processes.

668 Local Processing:

669 In a self-hosted environment, wallets can use local processing techniques to provide insights and
670 assistance or even automation features for key, account, and token management. Those features
671 can involve querying blockchain networks to view token transfer histories and status updates, as
672 well as signing and submitting transactions. *On-device machine learning* techniques make it
673 possible to perform on-device training and import curated models from external galleries to
674 compute predictions and suggestions directly on the users' personal devices without requiring
675 access to external cloud providers. Wallets may also be controlled by machines to provide them
676 with the ability to store, receive, and send tokens.

677 At the custody management level, predictions and suggestions can assist in partial synchronization
678 to store keys in separate locations across devices or in transfers between wallets with different
679 security levels (e.g., cold, warm, and hot). Additionally, if irregular activities are detected on a
680 particular device and other devices are available via multi-signature (see Section 3.3), mechanisms
681 can be put in place for self-destruction of the private key on the compromised device.
682 Alternatively, if a recovery withdrawal address was designated beforehand, tokens could be
683 automatically sent to it.

684 At the token ownership management level, predictions and suggestions can provide financial or
685 portfolio management assistance and capture knowledge of user preferences to recommend or even
686 directly accept transactions on behalf of users (e.g., for repeated micropayments that have
687 previously been approved), as in the Fetch.AI protocol [23]. They may also help build proofs to
688 authenticate with third parties (see Section 6.4).

This can be complemented by *asynchronous privacy-preserving data mining* techniques (e.g., differential privacy, federated learning), which make it possible to extract trends from collective user data without compromising individual user privacy (e.g., automatically hide tokens considered spams). Note that computation on encrypted data is discussed in Section 4.3.3.

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 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 three out of five keys are necessary for account recovery. A user can choose to store three keys themselves and appoint third-party custodians to store the remaining two keys. 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 [24].

Full Custody:

Users can fully entrust third-party custodians to hold and safeguard their accounts by having all private keys placed under their control, though users remain legal owners of the associated tokens. Alternatively, users may choose not to 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 that offers 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.

3.3 Account Origination and Recovery

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

Key Generation:

Hierarchical deterministic (HD) wallets, specified under Bitcoin Improvement Protocol BIP-32 [25], allow an unlimited number of key pairs to be generated in the same wallet from a single seed. Users can therefore maintain special-purpose identifiers decoupled from their primary identifier, providing a certain level of anonymity without having to manage 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 also enable the generation of key pairs for *stealth addresses*. A stealth address is an account that gives its owner the ability to compute the private keys of accounts created by other entities. Key generation follows the elliptic-curve Diffie-Hellman protocol, wherein interested parties first generate a cryptographic nonce (sent to the owner of the stealth address) before creating a public key and an account that they can send tokens to. The owner of the stealth address then computes the private keys of these identifiers to take possession of the tokens. The stealth address itself does not receive transactions, making payments untraceable by third parties. The concept of stealth addresses was first introduced in Bitcoin development discussions [26].

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 cryptography⁵ [27] 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/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 the creation of the wallet. However, this method is not as secure as it could be; both paper and digital backups of the seed phrases can be lost, stolen, or destroyed.

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 [28][29]. Some threshold signature schemes do not reveal which individual entities participated in the signature process.

Multi-signature wallets can also restore access to tokens in the case of a loss of 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, in the event that one or more shares of a secret key become compromised over time, new random shares may be computed—a process known as *resharing*. Resharing occurs either reactively (e.g., as a response to the detection of a compromised share) or proactively (e.g., at periodic time intervals).

A wallet can implement a *dead man's switch* to recover a private key; after a certain period of time 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 (and other information). Domain names can be represented by nonfungible tokens, as discussed in Section 6.3.

3.4 Smart Contract Vaults

Smart contracts can serve as programmable vaults to receive token deposits (i.e., “on-chain custody”). Depending on the use case or configuration, they are also called deposit or multi-signature contracts.

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 maintaining persistent identifiers for their interactions with other users. They are tied to private keys held in regular user-controlled wallets, acting as a proxy and allowing for separation of concerns so that each layer can focus solely on its role. These account management models can be implemented directly within smart contract vaults or through authorized modules. They can permit multi-signature schemes and security and recoverability rules, such as security periods, thresholds requirements, and emergency modes. This can make it easier to define trustee addresses for partial account custody as well as cross-device and social account recovery methods. Notably, these account recovery methods do not require 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 under the hood on behalf of users when they create 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 through a known trusted attribute (e.g., phone number).

796 Joint Vaults:

797 Groups of users who wish to share the ownership and management of an account can use smart
798 contract vaults that implement the supporting access control and rules.

799 Protocol Collaterals/Deposit Vaults:

800 Protocols can use smart contract vaults (or deposit contracts) to receive tokens and condition their
801 release. Token collateralization is used to secure individual transactions, such as atomic swaps (see
802 Section 4.1.2.1), through self-enforceable rules. On a wider scale, token collateralization—or
803 *staking*—is also used to build *cryptoeconomic incentives* for community-controlled networks,
804 either at the base layer or at the second layer. When staked, tokens earn yields and/or provide
805 privileges within the protocol (see Section 6.1). In proof-of-stake consensus models (see Appendix
806 A), staking serves as the basis for supporting the operations of the blockchain network (i.e.,
807 rewards for participants staking tokens). Certain staking models involve a time period during
808 which tokens are locked up (i.e., they cannot be transferred) and subject to penalties, acting as
809 programmable security deposits. The rules that condition the release of smart contract collateral
810 funds can be based on both internal blockchain data and external data sources (see Section 5.4).
811 Owners of collateralized tokens can build proofs based on the collateral to authenticate with third
812 parties (see Section 6.4).

813 Key collateralization schemes, particularly useful for securing individual transactions, include
814 cryptographic timelocks to place time conditions and hashlocks to require the knowledge of a
815 secret. *Hashed timelock contracts* (HTLC) are smart contracts that implement these two
816 techniques, allowing for deposit contracts to be built with vesting periods and conditioned refunds.

4 Transaction Management

This section discusses how transactions associated with both protocol-level and user-level operations are managed by blockchain and second layer protocols. It breaks down the analysis in three aspects: transaction validation, transaction submission, and transaction viewability. Different tradeoffs between decentralization, scalability, and security can be made from the building blocks examined.

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 – *Base Layer Consensus and Compute* 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 scaling schemes at the second layer so that the amount of data stored on the root blockchain is minimized. Note that developers may have varying needs for off-chain scaling depending on performance at the base layer, though the same schemes may also provide privacy and usability properties on a case-by-case basis.
- *Cross-chain transactions* allow updates for global states from two or more distinct blockchain networks in concert, enabling token and oracle portability across them. It can be possible for cross-chain transactions to take place across different blockchain protocols.

The following sections 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.

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 to enable more efficient and suitable solutions.

4.1.1 Off-Chain Scaling

This section presents schemes to enable faster and cheaper transactions through secure off-chain processing. 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 protocol at the base layer and have primarily been studied for deployment on top of permissionless blockchains.

Some scaling schemes involve periodic updates on the root blockchain as part of the protocol. 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 settlement is necessary to prevent transactions from being reverted. Thus, only transaction finality at the base layer ultimately matters (see Appendix A – *Base Layer Consensus and Compute*). Scaling schemes are designed to offer guarantees that capture a certain degree of transaction finality at the base layer and that depend on the particular structure, security deposits (collaterals at the base layer), state update method, and dispute resolution mechanism used. Table 3 compares these aspects for different types of scaling schemes at a high level.

Table 3: Off-Chain Scaling Schemes Comparison

	State/Payment Channels	Commit Chains	Sidechains		
			Plasma	ZK-Rollups	Optimistic Rollups
Structure	Fixed group	Single operator	Blockchain	Blockchain	Blockchain
Security Deposits	Required at establishment for all users	Not required for recipients	Not required for recipients	Not required for recipients	Not required for recipients
State Updates	User-initiated	Periodical	Periodical	Periodical	Periodical
Disputes	User-initiated	User-initiated	User-initiated	Periodical	User-initiated

4.1.1.1 State/Payment Channels

A *state channel* is a scheme that enables a group of participants to sign and process transactions directly with one another. It is backed by full on-chain collateralization, and transactions are signed and stored by all participants. This is meant to provide strong transaction finality guarantees, instant at the second layer. These signatures allow for state updates to be pushed on-chain. A *payment channel* is a state channel specialized for sending payments.

Three phases occur during the lifecycle of a state channel [30] (though implementations vary):

1. **Establishment:** A channel is established after participants agree to lock up a portion of the current state of the blockchain. For instance, 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), can 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 participants can begin sending off-chain transactions to one another. Any participant can propose a state update, which then has to be approved and signed by all of the channel participants.

3. **Closure (or Dispute):** Once all participants have reached an agreement (i.e., signed the latest state update), the new state is pushed on-chain, and the channel is terminated. If a participant that submitted a state update does not receive enough signatures after a certain period of time, they may initiate a dispute procedure—conducted on-chain—during which participants submit evidence (a hash of the latest off-chain state) to the blockchain.

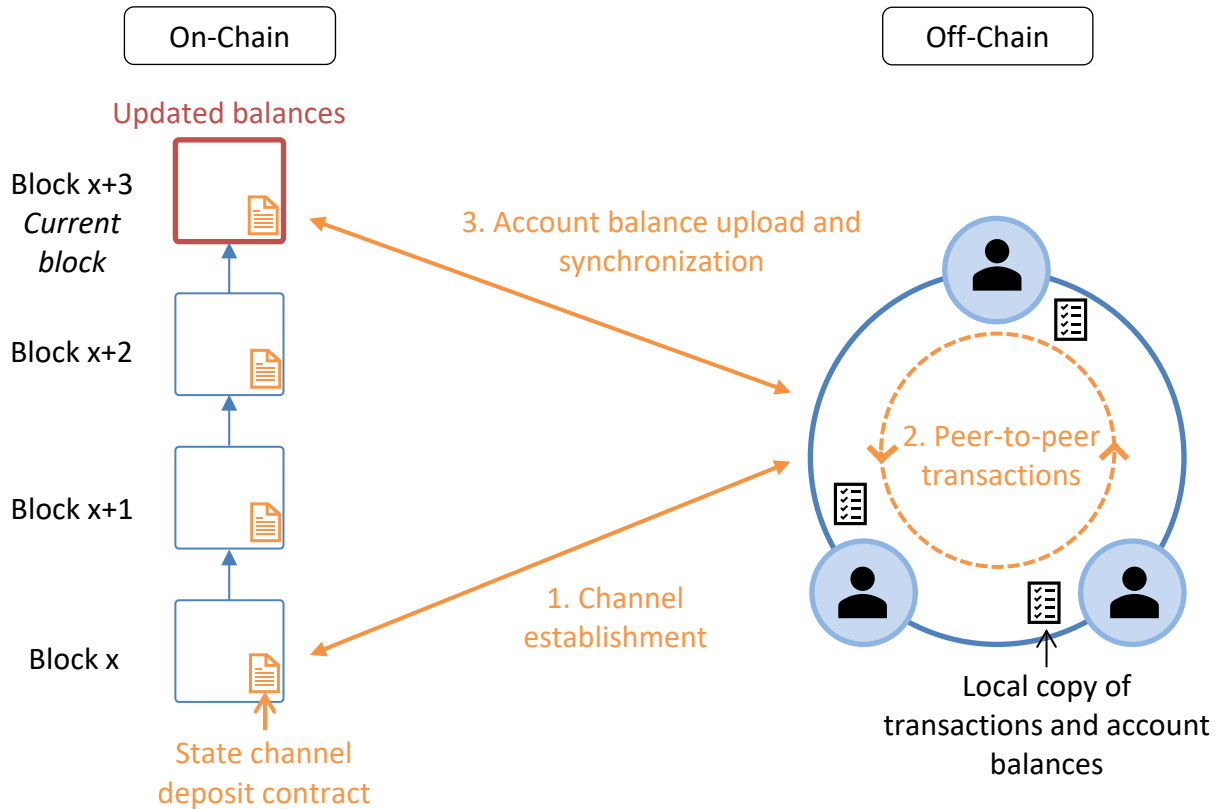


Figure 3: Payment Channel Phases

As shown in Figure 3, 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 publicly on-chain. 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 the addresses of all channel participants 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* wherein *multi-hop payments* can take place, allowing balances on multiple channels to be used during the same payment operation. 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 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 payment paths (i.e., the particular sets of intermediary nodes used to relay transactions from the sender to the receiver). They can be characterized by their effectiveness (i.e., maximizing the probability of payment success), efficiency (i.e., ensuring low computational overhead for path discovery), cost-effectiveness (i.e., finding paths with low transaction fees), scalability, and privacy implications [30].

An example of a payment channel network on top of the public Bitcoin network is the Lightning Network [31]. 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 [32], 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 nodes to validate all of the transactions routed across their channels.

It is also possible to prevent double-spending through hardware-enforced consensus rules (i.e., hardcoded rules that cannot be reprogrammed or tampered with). By using TEEs to synchronize payments across channels, the need for on-chain collaterals and dispute periods after channel closure is eliminated [33].

4.1.1.3 Commit-Chains

A *commit-chain* [34] is a ledger maintained by a non-custodial operator that collects transactions from users and periodically pushes them on-chain as cryptographic commitments, either as a Merkle root hash or zero-knowledge proof. Senders lock up the amount they wish to send in the smart contract, and no deposit is required for the recipients. To increase the expectation of transaction finality, the operator itself may also deposit funds in an on-chain collateral. If the operator pushes invalid transactions, users can halt the execution of the commit-chain and recover their funds. Moreover, since the operator acts as a middleman, users do not need to remain online to receive payments, though they are expected to store their transaction history to exit the commit-chain. In comparison, state channels have higher bootstrapping costs but require less on-chain transactions during normal use.

4.1.1.4 Sidechains

A *sidechain* is a blockchain that is connected to another blockchain (i.e., the root blockchain) through a bridge (see Section 4.1.3). A token that is deposited on the root blockchain (e.g., locked up in a deposit or bridge smart contract) is represented as a separate token on the sidechain. Reciprocally, these sidechain tokens may be redeemed on the root blockchain.

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

Being separated from the root blockchain, sidechains have their own consensus mechanism, incentive structure, and network topology. Thus, their security, scalability, and transaction finality properties vary. To prevent malicious blockchain forks, a sidechain can periodically post snapshots of block headers onto the root blockchain, as in commit-chains.

An example of protocol to create sidechains is Plasma [35], on top of the public Ethereum network, where sidechains are called *plasma-chains*. A new blockchain can be derived from an existing plasma-chain, resulting in a tree-like structure of sidechains. Like commit-chains, plasma-chains periodically send data hashes of their blocks to the root blockchain. Smart contracts called *fraud proofs* and deployed on the root blockchain allow users to exit plasma-chains (i.e., unlock the tokens held in the bridge smart contract), as well as report fraudulent transactions or plasma nodes by submitting cryptographic proofs. Since the architecture of plasma-chains does not rely on trusting one particular set of sidechain operators, they are sometimes referred to as non-custodial sidechains. Note that users may have to maintain access to a copy of their past transactions.

Different protocol variants have been developed, such as Plasma Cash, Plasma Debit, and Minimum Viable Plasma. Each token deposited onto a Plasma Cash chain [36] results in the issuance of a nonfungible token 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 the transaction history of their own tokens instead of a copy of the whole plasma-chain. Building on Plasma Cash, Plasma Debit chains [37] create payment channels between users and nodes when nonfungible tokens are issued to new users. Finally, the Minimal Viable Plasma [38] protocol follows a UTXO design. Users need to periodically download the plasma-chain to verify its integrity and provide withdrawal proofs.

4.1.1.5 Rollups

A *rollup* is a type of sidechain that periodically pushes transaction data onto the root blockchain. This consists of a Merkle root hash of the current state of the sidechain as well as some data for each of the transactions included in the latest sidechain block (stored in calldata storage in Ethereum-based networks). Rollups enable anyone to verify the validity of all sidechain transactions, unlike Plasma sidechains, which publish the current state of the sidechain but do not provide the needed transaction data. Two types of rollups have been proposed, discussed below.

ZK-Rollup:

In addition to transaction data, ZK-rollup sidechains publish a zero-knowledge proof used to verify the validity of any transaction [39]. It is, therefore, impossible for the sidechain operators to commit a falsified update to the root blockchain. Setting up a ZK-rollup sidechain, however, may require an initial trusted setup in order to enable the zero-knowledge proof. Zero-knowledge proofs are also usually computationally expensive to produce, though their size when used in ZK-rollups generally does not depend on the number of transactions it is built from. Note that, in ZK-rollups, the zero-knowledge proofs are generally used as a verifiable data compression mechanism and do not necessarily provide more privacy properties than other off-chain scaling schemes.

974 **Optimistic Rollup:**

975 Unlike in ZK-rollup sidechains, optimistic rollup transactions that get anchored onto the root
 976 blockchain are assumed to be valid by default without having to provide a validity proof or
 977 requiring an initial trusted setup. However, if a user notices that an incorrect state or invalid
 978 transaction has been published on-chain, they can produce a challenge by posting the valid state
 979 and a Merkle proof. This defers transaction finality at the base layer as a prolonged fraud proof
 980 challenge period may be necessary. Optimistic rollup chains usually offer the ability to implement
 981 arbitrary smart contract logic, which may be harder to achieve in ZK-rollup chains due to the
 982 limitations of zero-knowledge proofs discussed previously.

983 **4.1.2 Token Exchange**

984 Tokens can be exchanged with one another without the need for any intermediary in an operation
 985 referred to as *atomic swap*. It is composed of a set of instructions that allows the execution of a
 986 token exchange with only two possible results: either the transaction succeeds, and all participants
 987 receive the desired tokens, or the transaction fails, and the state remains at its starting point [40].
 988 This can be done in a direct fashion when the parties involved already know each other and agree
 989 on the exchange. Otherwise, one must use an exchange, which provides price discovery.
 990 Techniques allow the building of non-custodial exchanges, some of which are based on off-chain
 991 order messaging and relaying.

992 **4.1.2.1 Atomic Swaps**

993 Atomic swaps allow two participants to exchange tokens directly. They rely on the deployment of
 994 a smart contract vault by each participant wherein they transfer the tokens that they would like to
 995 exchange. A set of rules is used to specify the terms of the exchange, such as initial deposit
 996 requirements and expiration times. If the terms are followed, each participant obtains the ability to
 997 withdraw the tokens present in the other participant's smart contract vault. Those rules are
 998 implemented in the smart contract vaults themselves or in a separate orchestration contract.

999 Cryptographic hashlocks and timelocks are some of the most prominent primitives to conduct
 1000 atomic swaps. Within smart contract vaults, a simple structure for this involves the deployment of
 1001 two HTLCs, one for each participant in the exchange, Alice and Bob. The process, illustrated in
 1002 Figure 4, unfolds as follows:

- 1003 1. After creating a secret s , Alice publishes an HTLC with hashlock $h = \text{hash}(s)$ and timelock
 1004 t on the blockchain. Alice then transfers to it the tokens that she intends to exchange with
 1005 Bob under the condition that they will be transferred to Bob only if he sends a transaction
 1006 containing the secret s prior to t expiring. If no transaction is sent and t expires, they will
 1007 be transferred back to Alice.
- 1008 2. Once aware that Alice's HTLC has been published on the blockchain, Bob publishes
 1009 another HTLC on the blockchain with a timelock $t' < t$, the same hashlock h , and reciprocal
 1010 token collateralization conditions. Bob then transfers to it the tokens that he intends to
 1011 exchange with Alice under the opposite conditions as in the previous step.

1012 3. Then, in order for Alice to receive Bob's tokens, Alice sends a transaction containing the
1013 secret s to the HTLC that Bob published before t' expires.

1014 4. Bob is hence made aware of s and can, in turn, send a transaction to the HTLC that Alice
1015 published in order to receive Alice's tokens. This must be done before t expires.

1016 In order for Bob to have enough time to receive Alice's tokens, it is preferable to set the condition
1017 $t > t'$ with a reasonable margin. Alice and Bob must choose the amounts to transfer (and possibly
1018 extra rules) at the beginning of the process since it is not possible to restrict the transfer to a portion
1019 of the tokens locked up in the HTLCs afterwards (or to change the rules). Hashlocking is an
1020 interactive process. Participants must have access to the blockchain and be able to communicate
1021 with each other throughout the steps described.

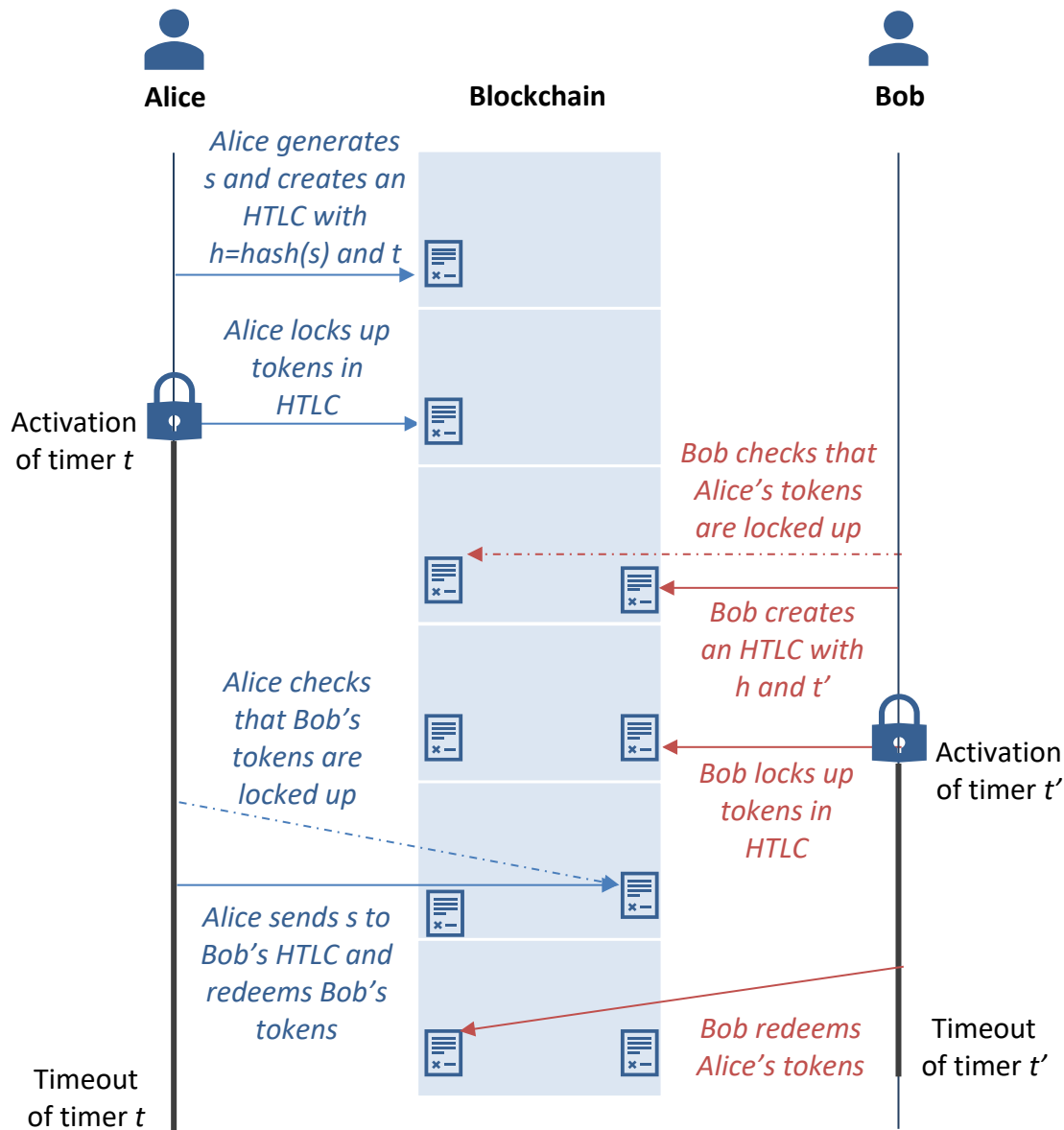


Figure 4: Hashed Timelock Contract Transfer Flow

1022 Alternatively, the rules of the atomic swap can be implemented in an orchestration smart contract.
1023 It handles order matching, signature verifications, and transfer requests between the different smart
1024 contract vaults engaged in an exchange.

1025 As an example, in the Wyvern Protocol [41], participants signal their intent to exchange tokens by
1026 sending a transaction to the orchestration smart contract. This transaction grants the orchestration
1027 smart contract the ability to transfer the specified token from the user's smart contract vault if a
1028 matching buy order is found. Until this occurs, users can withdraw their intent by transferring the
1029 tokens out of the smart contract vault. Implementations of the Wyvern Protocol include OpenSea
1030 [42], a non-custodial marketplace for ERC-721 and ERC-1155 nonfungible tokens.

1031 Some systems conduct atomic swaps through off-chain coordination rather than through rules
1032 implemented in smart contract vaults, such as Algorand [43]. Once participants reach an exchange
1033 agreement off-chain, the transactions necessary for the exchange to occur are bundled into an
1034 aggregate transaction that is signed by all parties before being submitted to the blockchain.

1035 HTLCs also allow atomic swaps between two distinct blockchain networks that support the same
1036 hashing function, enabling point-to-point communication. As an example, Decred [44] provides a
1037 repository of implementations that leverage HTLCs to support cross-chain transactions across
1038 different blockchains protocols.

1039 Another locking-based technique for cross-chain transactions uses discrete log-based signature
1040 locks (the blockchains must also support the same hashing function). In this structure, the
1041 participants lock tokens in multi-signature contracts deployed on each respective blockchain.
1042 Instead of solving a hash pre-image problem, as in hashlocking, a discrete logarithm problem
1043 serves as the basis for the exchange. The "Scriptless Scripts" [45] project from Elements, which
1044 aims to design cryptographic protocols that run on top of Bitcoin, gives an example of the
1045 implementation of discrete log-based signature locks for cross-chain transactions.

1046 4.1.2.2 Non-Custodial Exchanges

1047 Non-custodial exchanges allow users to trade tokens with one another while remaining in control
1048 of the private keys at all times. By design, user onboarding may not be required since users bring
1049 their own accounts, and token pairs may be added directly by users. In comparison, traditional
1050 exchanges act as third-party custodians, recording trades in their own ledgers. Non-custodial
1051 exchanges are also known as *decentralized exchanges* or by the acronym DEX.

1052 The attack surface of non-custodial exchanges is lessened due to account security risks being
1053 shifted towards users, but it is not entirely eliminated. In particular, mechanisms are usually needed
1054 to mitigate *front-running attacks* wherein miners (or sometimes other entities, indirectly) learn
1055 about an order's information and attempt to submit transactions that take advantage of that
1056 information before the order is executed and published onto a block.

1057 The exact security model, scope, architecture, and reliance on off-chain resources of non-custodial
1058 exchanges vary. Several types of architectures for fungible token exchanges are described below:

- 1059 • **Orderbook Market Making:** An *orderbook* contains a list of buy and sell orders for a
1060 given token pair, also called *bids* and *asks*. The highest bid and the lowest ask are referred
1061 to as *the top of the book*, and the difference between them is the *spread*. Two main types
1062 of orders can be submitted: market orders and limit orders. When submitting a market
1063 order, tokens are immediately bought or sold for the best available price by pairing buyers
1064 and sellers with orders currently at the top of the book. On the other hand, limit orders are
1065 placed on the orderbook upon submission at the specified price and remain unfilled until
1066 the top of the book moves to the specified price. Orderbooks are implemented fully on-
1067 chain or as a hybrid—orders being collected and matched by non-custodial liquidity
1068 providers (or relayers) off-chain before being settled on-chain. They may do so in exchange
1069 for a fee and may form an incentivized peer-to-peer network. Note that this architecture is
1070 the closest to that of most traditional exchanges.
- 1071 • **Automated Market Making:** *Automated market making* is another type of non-custodial
1072 exchange based on *liquidity pools* rather than orderbooks. Liquidity pools are smart
1073 contract vaults deployed for every token pair that hold funds for both tokens of the pair.
1074 They act as automated market makers by determining the exchange rate between the two
1075 tokens using a formula that takes into account the relative quantities of each token in the
1076 pool. As long as a liquidity pool for a particular token pair exists, the liquidity problem
1077 found in illiquid markets is eliminated (i.e., lack of buyers). Protocols may incentivize
1078 liquidity providers by distributing rewards for the addition of a new liquidity pool or for
1079 contributing to existing liquidity pools. When contributing liquidity to a pool, pool-specific
1080 staking tokens may, in return, be minted and distributed to record the contribution, allowing
1081 composability with other protocols. Liquidity pools from multiple sources may be
1082 aggregated with the exchange rate of token pairs calculated across them.
- 1083 • **Dutch Auction Market Making:** In this type of non-custodial exchange, *Dutch auctions*
1084 are continually conducted. Sellers can submit orders at any moment, but those orders are
1085 only executed during the next auction. Auctions start with an initial price set to twice the
1086 final closing price of the previous auction for the same token pair, which then gradually
1087 decreases until the price clears the buy and sell orders. During auctions, buyers submit bid
1088 orders when they are satisfied with the current price, knowing that at the end of the auction,
1089 every buyer will receive the tokens for the same price. Like automated marker making,
1090 Dutch auction market making is especially used for more illiquid tokens.
- 1091 • **Ring Trade Market Making:** Like Dutch auction market making but with shared liquidity
1092 across token pairs, ring trades enable non-custodial exchanges based on periodic batch
1093 auctions. Users place limit sell orders that are processed at the next available auction.
1094 Auctions consist of open competitions where order settlement propositions are submitted
1095 and end with the protocol selecting the proposition that maximizes traders' profit while
1096 providing single clearing prices.

4.1.3 Blockchain Bridging

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

Deposit smart contracts are core primitives used to bridge a blockchain to another. They enable users to receive a 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 bridges, that representation provides redemption value for the original token. One-way bridges are used to support the permanent migration of tokens from one blockchain to another without redemption value on the original blockchain. Deposit smart contracts can implement different locking techniques. To enable cross-chain communications, especially to provably recognize one another's deposits, blockchain networks are coordinated via intermediary systems: notaries, relays, or separate blockchains. They are trusted off-chain or 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 [46] and the Inter-Blockchain Communication protocol developed by the Interchain Foundation [47]. As shown in Figure 5, the intermediary system (in dashed grey) creates a hub-and-spoke architecture that connect two or more blockchains together.

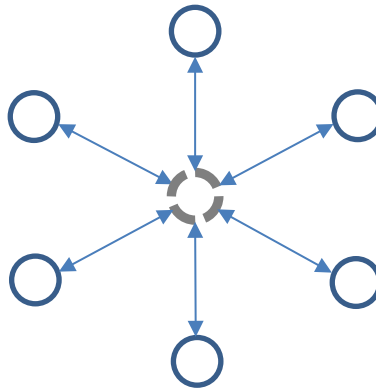


Figure 5: Hub-and-Spoke Architecture

4.1.3.1 Sharding

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

Sharding increases transaction throughput via parallel processing. The entire state of the blockchain is split among blockchain subnetworks, which have their own transaction history and set of nodes. A separate hub blockchain coordinates these subnetworks by affecting 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. Note that sharding can have fundamental security ramifications for the consensus model used by the blockchain network.

1124 Transactions are considered valid once they are added to a block by a blockchain subnetwork and
1125 do not require any additional validation from the root blockchain. Consequently, the validity of
1126 the whole system is compromised when a single blockchain subnetwork is tampered with. This
1127 notion, called *tight coupling*, differentiates sharding from relayed sidechains (see Section 4.1.3.3)
1128 [48].

1129 For example, Polkadot [49] uses sharding to allow application-specific or specialized blockchain
1130 subnetworks, called *parachains*, to communicate with one another with shared security. Ethereum
1131 2.0 is also expected to use sharding but with identical blockchain subnetworks, called *shard chains*
1132 [50]. The former is designated heterogenous sharding and the latter homogenous sharding.

1133 4.1.3.2 Notaries

1134 A *notary* is a trusted entity (or a set of trusted entities with a multi-signature contract) tasked with
1135 reading and sometimes validating a blockchain's transactions to report them to another blockchain
1136 and potentially vice-versa. Notaries act either proactively (i.e., automatically responding to events
1137 that occur on a blockchain) or reactively when prompted to do so. While reducing the number of
1138 bridges necessary to connect new blockchains together, notaries may also act as a bottleneck for
1139 transaction throughput and introduce a centralized attack surface.

1140 Two categories of notaries can be distinguished [51]:

- 1141 • Custodians generally receive full control over a user's tokens (see Section 2.1.3) and are
1142 trusted to release them when asked to do so. Custodians may be subject to collaterals and
1143 penalties to disincentivize fraudulent activities. As an example of a notary scheme, [52]
1144 links the IOTA protocol to the Hyperledger Fabric protocol.
- 1145 • External escrows only receive conditional control over a user's tokens. These escrows often
1146 take the form of a multi-signature contract in which the signature of the user and that of
1147 the escrows are required before a transaction is executed. Wanchain [53] uses a set of
1148 dedicated nodes as notaries that are tasked with verifying cross-chain transactions and
1149 creating external escrows through secure multi-party computation (see Section 4.3.3).

1150 4.1.3.3 Relays

1151 A *relay* is a system implemented in a given blockchain network that can read and validate events
1152 and states from distinct blockchain networks (sidechains, as discussed in Section 4.1.1.4). It
1153 usually replicates part of the state of a blockchain (e.g., a block header) onto another blockchain;
1154 the latter blockchain can then verify the existence of some information on the first blockchain.
1155 Relays generally take the form of smart contracts deployed on one or more blockchains. When a
1156 single (central) blockchain is linked to more than one sidechain and serves as a coordinating entity,
1157 it is also referred to as a *relay blockchain*, as illustrated in Figure 6. By allowing tokens to be
1158 locked up on the relay blockchain, proofs of collaterals can be generated to complete transactions
1159 on sidechains. Unlike notaries, relays do not rely on a trusted external entity to conduct cross-chain
1160 transactions. Relays are notably efficient on blockchains that have rapid finality.

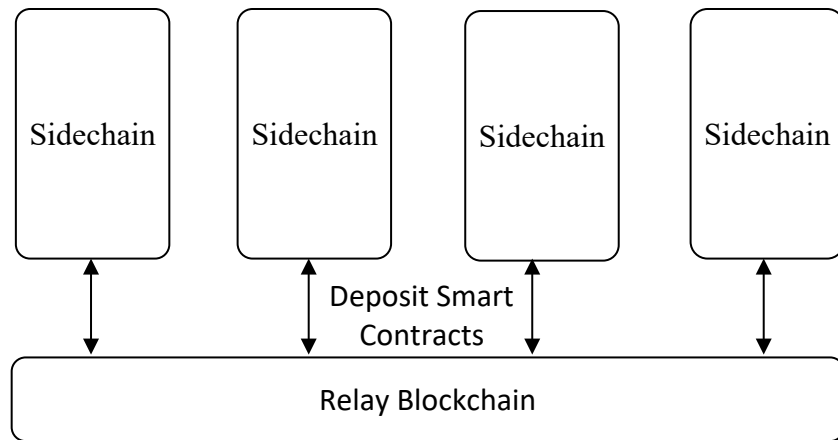


Figure 6: Relay Blockchain

1161 Cosmos [54] uses the same hub-and-spoke architecture as sharding with the relay blockchain as
 1162 the root blockchain, but it is not tightly coupled. Each subnetwork has its own separate consensus
 1163 model. The blockchain hub network aims to enable cross-chain transactions between subnetworks
 1164 but does not play any role in their security. Another example is BTC Relay [55], a one-way bridge
 1165 that takes the form of a smart contract that receives Bitcoin block headers and allows Ethereum
 1166 users to confirm Bitcoin transactions.

1167 Relays and notaries can be combined into hybrid schemes. For instance, a relay blockchain could
 1168 be set up to connect sidechain A to sidechain B. If, for some reason, the relay blockchain is unable
 1169 to issue or validate transactions on sidechain B, a notary compatible with this particular sidechain
 1170 could be used instead. For example, Rootstock [56] features a two-way bridge with Bitcoin to lock
 1171 and unlock tokens using a federation of notaries with a multi-signature contract.

1172 4.2 Transaction Submission

1173 Transaction submission is either open to anyone or restricted to particular privileged users or roles
 1174 via multilevel permissions. Access control can be implemented at the smart contract layer using
 1175 conditions on function calls to accept or reject transactions according to protocol-defined rules or
 1176 directly in the blockchain protocol at the base layer.

1177 As discussed in Section 2.1.3, token owners can give approvals and special authorizations to let
 1178 other accounts submit transactions that directly affect their tokens. This permits multiple keys to
 1179 be issued to interact with protocols with specific usage and spending limits. This also allows for
 1180 systems to be developed that support *pull payments* (i.e., that involve requests to pay) as well as
 1181 *push payments*. These delegations can also be used to implement *split payments*, in which a group
 1182 of users accepts a common payment request. Additionally, payment protocols can be designed to
 1183 send payment acknowledgements and provide refund addresses [57]. Note that off-chain
 1184 messaging schemes are generally needed to enable incoming and outgoing payment notifications.

4.2.1 Meta Transactions

Many protocols, both at the base layer and the second layer, involve transaction fees or tips. *Meta transactions* (or *fee delegation*) make it possible for these transaction fees to be paid by third parties on behalf of users. Similar to prepaid “transaction envelopes,” this enables users to sign and submit *fee-free transactions*. Thus, users do not have to own protocol-native tokens of a given token-based network in order 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). Protocols can be used without having to go through the regulations associated with spending tokens, depending on the jurisdictions.

Examples of protocols that implement meta transactions for the public Ethereum network (also called “gasless” transactions) include EIP-1613 [58] and the Gas Station Network [59]. 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 relay servers.

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 between users with identity verification tiers and transfer limits). 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) [60].

Role-Based Access Control:

In *role-based access control* (see NIST definition [61]), role assignment 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 assignments.

Roles are implemented by deploying 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 [62]. 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 [63].

1221 Attribute-Based Access Control:

1222 In *attribute-based access control*, an identity management system provides users with token-based
1223 attributes or credentials that they can then use to authenticate themselves and be authorized to call
1224 certain user-level operations. For example, it can be used to restrict the transfer of a given type of
1225 token to people who passed a predefined test or met certain requirements, as implemented in the
1226 Transaction Permission Layer Protocol [64]. These blockchain-based credentials may be stored
1227 directly in the user's wallet.

1228 4.2.3 Blockchain Node Permissioning

1229 The two main types of permissioning schemes to control which blockchain nodes can join the
1230 network are described below. The choice of node permissioning scheme, if there is any, can
1231 indirectly affect the ability of users to submit token transactions (e.g., by forcing them to send
1232 transactions to a specific node).

1233 Local Permissioning:

1234 Each node maintains a configuration file that contains a list of nodes from which to accept
1235 connections. This enables two types of governance models:

- 1236 • In consortium blockchains, each node maintains its own configuration file.
- 1237 • In private blockchains, each node maintains the same configuration file, digitally signed
1238 and provided by a trusted central authority (i.e., a system owner).

1239 On-Chain Permissioning:

1240 Smart contract-based access control can also be used as a permissioning scheme for blockchain
1241 nodes and accounts at the protocol level. It is enforced by full nodes that access the landing
1242 permissioning smart contract at the address provided in the network configuration, as implemented
1243 in Hyperledger Besu [65].

1244 This type of access control provides another way to develop governance models for adding and
1245 removing nodes and accounts that do not necessarily require trusting a central authority. For
1246 example, a voting system that provides equal governance rights to all nodes in a consortium
1247 blockchain network could be developed. It is also possible to deploy smart contract-based access
1248 control for node permissioning on a blockchain distinct from the one that it is intended for (which
1249 could thus be seen as being implemented off-chain).

1250 4.3 Transaction Viewability

1251 This section discusses monitoring and analysis tools, privacy-enhancing techniques, and
1252 computation on encrypted data.

1253 Blockchain protocols are generally meant to provide correctness but not privacy. By design, all
1254 on-chain transactions are at least visible to all of the nodes of the blockchain network so that they
1255 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 are fundamental components of enabling user privacy in a blockchain setting. Note that preventing users from being able to view on-chain transactions does not by itself guarantee privacy since any node can share transactions externally, besides entailing users to blindly trust the integrity of the network.

Depending on their architecture and deployment characteristics, off-chain transactions (see Section 4.1.1) may provide some degree of user privacy. As discussed in Section 4.3.3, encrypted computations provide confidentiality by design.

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., event logs). These tools can be coupled with actionable alerts, integrated with real-world activity, and offered as data feeds externally through platform APIs.

Depending on whether data is encrypted or privacy-enhancing techniques are used, insights on 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 private transactions.

4.3.2 Privacy-Enhancing Techniques

A high-level review of key privacy-enhancing techniques to shield transaction data is provided below. Readers who want to learn more are invited to access additional resources, such as [66]. *Single-use identifiers* (see Section 3.3), *transactions mixers*, and *ring signatures* provide anonymity through transaction unlinkability. *Zero-knowledge proofs* and *Pedersen commitments* are primarily used to keep transactions confidential.

1293 **Zero-Knowledge Proofs:**

1294 A *zero-knowledge proof* is a cryptographic scheme where a prover is able to convince a verifier
1295 that a statement is true without providing any more information than that single bit. Zero-
1296 knowledge proofs can be embedded into encrypted transactions so that users can verify transaction
1297 correctness without learning the content. This allows payments to be sent where the amounts and
1298 account addresses involved can only be decrypted by the sender and recipient. Mechanisms for
1299 zero-knowledge proofs can be built directly into protocols at the base layer or implemented as
1300 second layer protocols using smart contracts (or additional cryptographic schemes).

1301 As an example, EY's Nightfall project [67] is an open-source suite of tools and smart contracts
1302 that enables ZK-SNARKs-based⁷ private transactions on Ethereum-based networks and is
1303 compliant with the ERC-20 and ERC-721 token standards. Users generate ZK-SNARKs by using
1304 ZoKrates [68] (which provides a high-level language for writing code before converting it into a
1305 ZK-SNARK) and send them along with their tokens to a smart contract vault that creates a
1306 cryptographic commitment for every deposit (see paragraph on *Commitment Schemes* below). This
1307 commitment can then be transferred under zero-knowledge to other users within the same smart
1308 contract vault. Another smart contract is tasked with verifying the cryptographic proofs submitted
1309 to the smart contract vault; it uses the elliptic pairing curve functions specified in the EIP-196 [69]
1310 and EIP-197 [70] standards. The Aztec Protocol [71] follows a similar architecture: zero-
1311 knowledge proofs received by the smart contract vaults are sent to and independently verified by
1312 a central smart contract. It supports multiple formats of zero-knowledge proofs, such as range
1313 proofs, used to convince that the amount transferred is within a given interval.

1314 **Transaction Mixers:**

1315 *Transaction mixers* are meant to provide transaction untraceability. A first approach consists of
1316 users sending equal amounts of a given token to an intermediary, who in return sends the funds
1317 back to other addresses owned by the same users [72]. The intermediary can be a trusted custodian
1318 or a non-custodial smart contract vault. Another approach to transaction mixing aggregates
1319 transactions to obscure the linkage between senders and recipients, as in CoinJoin [73].

1320 **Blind Signatures:**

1321 A *blind signature* is a digital signature for which the content of the message is not visible to the
1322 signer (*blinded*) [74]. Once the content of the message is revealed (*un-blinded*), the signer may not
1323 be able to recognize it from the blinded version of the message that they previously signed,
1324 providing unlinkability between the blinded and un-blinded versions of the message. Applications
1325 of blind signatures include on-chain anonymous voting [75].

⁷ 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. However, it does not involve multiple cycles of information exchange between the prover and the receiver [76].

1326 Ring Signatures:

1327 A *ring signature* is a digital signature produced indistinguishably by the private key of any
1328 members of a group [77]. This allows members to sign transactions under the identity of the group
1329 without revealing which particular member originally created the signature.

1330 Commitment Schemes:

1331 A *commitment scheme* is a cryptographic algorithm where an encoded message is sent to the
1332 receiver with a condition on when it can be decoded. Commitments (Pedersen commitments are
1333 among the most common ones [78]) can be used in blockchain transactions to keep their content
1334 private, such as in the Confidential Transactions scheme [79].

1335 4.3.3 Computation on Encrypted Data

1336 *Computation on encrypted data*, such as *secure multi-party computation*⁸ and *homomorphic*
1337 *encryption*,⁹ makes it possible to perform confidential and distributed computations in zero-trust
1338 environments such that no one can use or read the data being computed. Expanding on applications
1339 for financial services [80], a key usage for blockchain networks rests in the outsourcing of private
1340 transaction processing for token operations and, more generally, for data associated with tokens to
1341 separate networks. This section discusses confidential smart contracts, the networks that support
1342 them, and consortium efforts to help develop the domain of secure computation. As it matures,
1343 this technology has the potential to mitigate front-running attacks and enable privacy-preserving
1344 features, such as auctions, voting, auditing, and data sharing.

1345 Confidential Smart Contracts:

1346 *Confidential smart contracts* (or *secret contracts*) are composed of an on-chain state and an off-
1347 chain private state that implements a cryptographic protocol for private transaction execution. The
1348 on-chain state is updated through regular on-chain transactions that are visible to and executed by
1349 all of the nodes in the blockchain network. On the other hand, private encrypted transactions that
1350 update the private state, such as token balances and operation arguments, are executed off-chain
1351 and remain encrypted. This computation on encrypted data can be operated by a distinct network
1352 or natively implemented in the protocol used by the underlying blockchain network. Hashes of
1353 private transactions are usually anchored onto the public state once they have been processed off-
1354 chain. This allows users to transact in a privacy-preserving manner without impacting the integrity
1355 and correctness of the smart contract execution.

⁸ 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 [27].

⁹ 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.

1356 Privacy Group Networks:

1357 Within a permissioned blockchain, the separate network that enables the private state is usually
 1358 operated by a privacy group in which transactions are either visible to all group members or only
 1359 to the members directly involved in a transaction. Users can join privacy groups by registering
 1360 existing or dedicated blockchain addresses that they can prove ownership of using the associated
 1361 private key(s) and then serve as identifiers within the groups that are visible to all members. The
 1362 members of a privacy group must run a *private transaction manager* node client, forming group-
 1363 specific peer-to-peer networks wherein members can privately exchange off-chain information.

1364 The Enterprise Ethereum Client specification [81] published by the Enterprise Ethereum Alliance
 1365 (EEA) specifies a private transaction model for permissioned blockchains. Hyperledger Besu [82]
 1366 is an example of a protocol that follows this specification with the private states managed by an
 1367 open-source private transaction manager called Orion [83]. Each Hyperledger Besu node that
 1368 sends or receives private transactions requires an associated Orion node. Private transactions are
 1369 passed from Hyperledger Besu nodes to the associated Orion nodes, which encrypt and broadcast
 1370 them to the other Orion nodes participating in the transaction. Quorum [84] also follows a similar
 1371 approach with a private transaction manager called Tessera [85]. Figure 7 below provides a
 1372 simplified flow-chart of a private transaction execution.

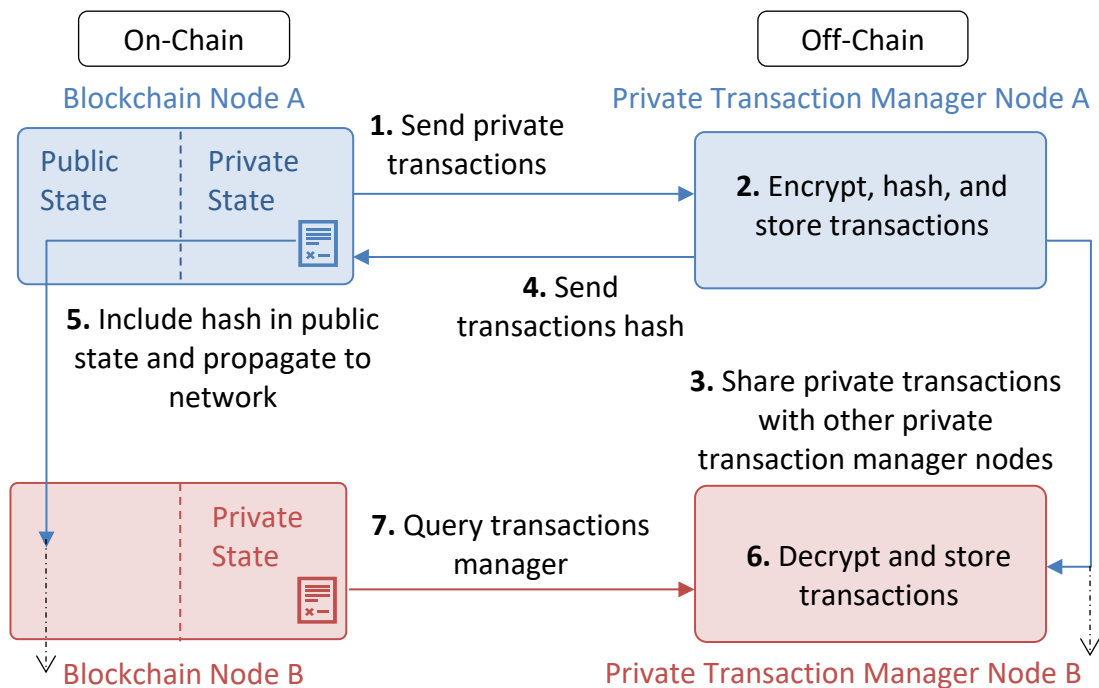


Figure 7: Private Transaction Execution

1373 Community-Controlled Networks:

1374 With a permissionless blockchain, the separate network that enables the private state can be
1375 controlled and operated by the community, as in Enigma [86]. In Enigma's Secret Network [87],
1376 nodes are meant to both participate in the blockchain's consensus model and compute private
1377 transactions. Developers submit smart contracts to the blockchain, the code of which is publicly
1378 visible on-chain. Users then deposit tokens or submit encrypted data to serve as input. All nodes
1379 perform the secret contract execution, and if more than two-thirds of the nodes agree on the same
1380 encrypted output, that output gets published on-chain. During secret contract execution, the
1381 encrypted inputs are decrypted and processed inside a TEE (as described in Section 3.1).

1382 In Hawk [88], a blockchain-agnostic smart contract protocol, the private state of a smart contract
1383 is executed by a third-party intermediary, a TEE, or the users themselves through multi-party
1384 computation. The system uses ZK-SNARKs to obfuscate the amounts and identifiers involved in
1385 token transfers. Each smart contract requires its own initial trusted setup. In NuCypher Network
1386 [89], users can delegate decryption rights over private data. In exchange for fees and protocol-
1387 native rewards, nodes are tasked with re-encrypting private data provided by users to enable the
1388 designated delegate to decrypt it. To disincentivize negative behaviors, nodes are required to stake
1389 protocol-native tokens. Finally, in Arbitrum [90], smart contracts take the form of virtual machines
1390 that are implemented and executed off-chain. A committee of managers designated by the creator
1391 of the smart contract is charged with monitoring the progress of the virtual machine and posting
1392 state updates on-chain. Staking tokens into the protocol enables managers to challenge the
1393 correctness of a particular on-chain state update. The dispute is then resolved on-chain, and the
1394 stake is returned to the challenger if they are successful.

1395 Secure Computation Frameworks:

1396 Hyperledger Avalon [91] (formerly known as the Trusted Compute Framework) is an open-source
1397 blockchain-agnostic framework for selective disclosure and private transactions. It uses trusted
1398 compute resources based on zero-knowledge proofs, secure multi-party computation, and
1399 hardware-based TEEs. This makes it possible to maintain on-chain auditability and policy
1400 enforcement, following EEA's *Off-Chain Trusted Compute Specification* [92]. Microsoft's
1401 Confidential Consortium Framework [93] is another example of open-source secure computation
1402 framework. Is it designed for consortium settings and relies on a network of TEEs to host the
1403 ledger and execute blockchain operations. All consortium governance updates (e.g., removal of a
1404 node, software upgrade) are recorded on the ledger.

1405 Additionally, the Linux Foundation has launched the Confidential Computing Consortium [94] to
1406 accelerate the adoption of TEE technologies and the development of open standards to support
1407 them.

5 Infrastructure Management

This section discusses software design patterns and infrastructure tools that make it easier to integrate token-based protocols in user interfaces or middleware. They include browsers, wallets, exchanges, dashboards, service aggregators, and other types of web or smartphone 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, 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 at the base layer, the second layer, or through open connectors and interfaces.

5.1.1 Base Layer

To interact directly with a blockchain network, an application must make calls (e.g., using the JSON-RPC API integrated via protocol-specific libraries) to a client running a node in order 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. Based on whether they propose new blocks for publication themselves, there exist two types: *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 8). There may be multiple node client software available for the same blockchain protocol.
- *Lightweight nodes* download and verify new block headers only. Full nodes are trusted for the verification of transactions correctness (securing the network). Incentivization schemes are being researched to compensate full nodes for their verification work via transaction fees (only the publication of new blocks is generally rewarded).

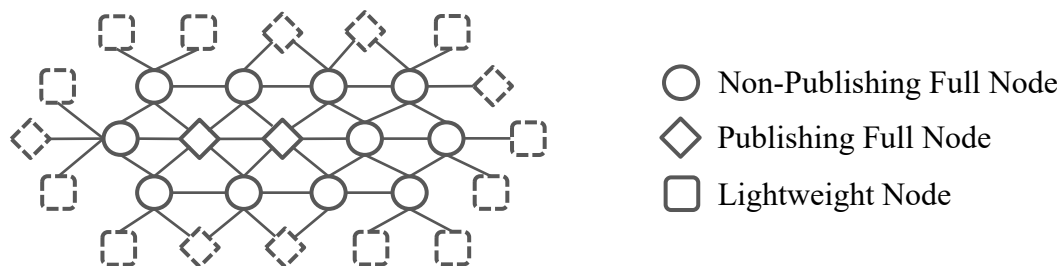


Figure 8: Blockchain Node Types

1437 User-Controlled Full Nodes:

1438 In addition to helping secure the whole network, running a full node is the most secure way to
1439 interact with a blockchain since its integrity is verified independently. A popular phrase with users
1440 who run their own full nodes is “don’t trust, verify.” Thus, when an application needs to access a
1441 blockchain network, a user can obtain maximal security by directly running their own full node
1442 and pointing the application to it. This requires that no node permissioning scheme prevents the
1443 user from doing so and that the user interface allows connections to custom nodes. Running a full
1444 node on the same device as the one accessing the application may be unsuitable, depending on the
1445 circumstances, as it requires synchronization with the entire blockchain (e.g., memory-limited
1446 mobile devices, reduced internet bandwidth causing high latency). Solutions include running local
1447 lightweight nodes or connecting to remote full nodes that are controlled and operated by the users
1448 themselves (like self-hosted VPN servers). These can take the form of dedicated preconfigured
1449 full node boxes (i.e., plug-and-play, headless computers) that are always powered on and
1450 connected to the internet. Depending on the throughput of the blockchain network, whether they
1451 produce blocks, and the type and configuration of the consensus model, full nodes may run on
1452 computers with relatively low computing power. The same computer may support full nodes for
1453 multiple blockchain networks. Alternatively, nodes can be provided by vendors and blockchain
1454 infrastructure service providers.

1455 Blockchain Infrastructure Service Providers:

1456 Blockchain infrastructure service providers are primarily meant to provision and orchestrate nodes
1457 for organizations and application developers with guaranteed reliability and speed, as well as to
1458 provide other services such as transactions and queries analytics. Applications usually interact with
1459 these cloud-based nodes through standard JSON-RPC calls or proprietary APIs that provide
1460 higher-level abstractions, depending on the services provided (e.g., hosted software, platform,
1461 infrastructure).

1462 Service providers may offer access to nodes shared among customers (with load balancing),
1463 deployments of new dedicated nodes to a single customer, and “bring-your-own-node” models
1464 where customers use node orchestration tools and APIs but provision their own nodes. Cloud-
1465 based, on-premises, and hybrid architectures can thus be designed, allowing organizations and
1466 application developers to share or outsource some of the deployment and maintenance work
1467 needed to operate nodes for particular blockchain networks (and sometimes, second layer
1468 protocols). They can also bootstrap new blockchain networks by providing custom genesis files.

1469 5.1.2 Second Layer

1470 Applications do not always interact directly with blockchain networks. Integrating second layer
1471 protocols can provide scalability and privacy gains. These protocols act as off-chain interfaces to
1472 perform verifiable computation and submit transactions on-chain. Key architectures and
1473 integration options are discussed at a high level below.

1474 Trusted Intermediaries:

1475 Applications can integrate services provisioned by external providers with varying functions and
1476 integration requirements. In commit-chains (see Section 4.1.1.3), trusted intermediaries (e.g.,
1477 watching services or proxies) may be used to ease the verification and transaction storage burden
1478 for users by relaying transactions and monitoring the blocks published on-chain by the commit-
1479 chain operator. Watching services may also be used for payment or state channels and payment
1480 channel networks to monitor channels and issue challenges on behalf of the participants so that
1481 they do not have to stay online themselves. In cross-chain custodial notary schemes (see Section
1482 4.1.3.2), the operator is given direct custody over a user's token. This occurs through either partial
1483 or full custody of the user's private key (see Section 3.2). The notaries may choose to operate their
1484 own set of blockchain nodes if necessary.

1485 Closed Groups of Participants:

1486 Applications can integrate protocols that enable off-chain transaction processing among a closed
1487 group of participants. State or payment channel (see Section 4.1.1.1) clients allow users to manage
1488 channel deposits on-chain (i.e., create, top up, and settle or dispute) as well as send, store, and
1489 relay off-chain transactions to one another through peer-to-peer communication protocols. Each
1490 participant must store the transactions of all participants. Nodes in permissioned encrypted
1491 computation networks (see Section 4.3.3) used by privacy groups are usually coupled one-to-one
1492 with the underlying blockchain nodes that maintain the on-chain public state.

1493 Community-Controlled Networks:

1494 Another option is to integrate open protocols for incentivized peer-to-peer networks that enable
1495 off-chain transaction processing. Since these networks are generally open to the public, they are
1496 often expected to provide a higher level of resilience against malicious behaviors (e.g., byzantine
1497 resistance). Prior to joining a payment channel network (see Section 4.1.1.2), a node must usually
1498 synchronize with the underlying blockchain network and act as a full node. As for base layer
1499 integration, applications can have their own nodes or choose to rely on external blockchain
1500 infrastructure service providers. Peer-to-peer incentivized networks are also used to operate
1501 watching services, non-custodial transaction relays (see Section 4.1.2.2), and off-chain verifiable
1502 computation frameworks (see Section 4.3.3).

1503 5.1.3 Open Connectors and Interfaces

1504 Applications can integrate blockchain networks, blockchain-based payment systems, and
1505 blockchain-based identifier systems through open connectors and interfaces. Although these open
1506 standards help build more interoperable systems, there is no one-size-fits-all abstraction.

1507 5.1.3.1 Blockchain Read and Write APIs

1508 Some clients offered as open-source software, such as Rosetta [95], make it possible to instantiate
1509 standardized, general-purpose blockchain read/write APIs that connect to trusted remote full nodes
1510 with varying limitations.

Another option is to integrate open protocols for incentivized peer-to-peer networks that enable community-controlled blockchain-querying APIs. They consist of a network of nodes that index on-chain events and process blockchain data queries, as in The Graph [96], based on GraphQL.

5.1.3.2 Identifier Resolving APIs

The blockchain addresses that tokens are assigned to do not provide applications with general-purpose user 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 an initial registration (e.g., on some on-chain registry) [15].

The Universal Resolver developed by the Decentralized Identity Foundation (DIF) [97] allows identifiers to be resolved to their associated metadata for multiple blockchain-based identifier management systems (DID methods¹⁰) without having to manage system-specific calls. However, this requires these blockchain-based identifier management systems to be included in the Universal Resolver configuration information and 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 PayID protocol [100] 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 [101].

By design, these systems allow users to provide applications with standardized payout account information. This can facilitate operating models wherein tokens are sent to users (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.

5.1.3.3 Payment Routing APIs

Applications can integrate open protocols and standards that interface payment systems built upon different traditional and blockchain technologies and allow for a certain degree of interoperability. As an example, Interledger [102] 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. Prior to 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 system) and payment systems.

¹⁰ Note that Rebooting the Web of Trust published a paper called *A DID for Everything - Attribution, Verification and Provenance for Entities and Data Items* [98] that introduces the concept of data objects associated with DIDs [99]—called decentralized autonomic data (DAD) items—to provide authentication for data provenance.

5.2 Wallet Integration

To interact with a blockchain network (as well as second layer protocols and blockchain-based identifier systems), an application must make calls to a wallet to generate new blockchain addresses, pull existing blockchain addresses, and sign transactions. This requires user approval or preapproval if preferences were set beforehand. Standards, such as EIP-2255 [103], are emerging to enable users to give granular permissions for the different applications that they interact with.

There are potential interoperability gains in the development of open standards that allow for decoupling wallets from user interfaces. Wallet-agnostic, general-purpose APIs can enable application developers to reach more users without having to integrate 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 could have immediate compatibility with pre-existing applications [104][105]. 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 [106] and WalletConnect [107]. Users can sign in to blockchain-based web applications by scanning QR codes with their smartphones without having to create an account. Web3Modal [108] 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 are not designed for general 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 (e.g., for some nonfungible tokens), only select hashes referenced on-chain to enable data integrity. Notably, applications do not necessarily have to store off-chain user account data themselves. Alternative storage options are discussed below.

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 [109].

Cloud Infrastructure Service Providers:

Architectures can rely on traditional cloud providers and on-premises infrastructures. 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.

1583 **Community-Controlled Cloud Storage:**

1584 Another option is to integrate open protocols for incentivized peer-to-peer networks that enable
1585 *distributed file storage systems* (i.e., community-controlled cloud storage).

1586 At a high level, these protocols encrypt, split, distribute, replicate, relay, and address files.
1587 Cryptographic hashes are computed for each file and used as unique persistent identifiers for
1588 indexing. Blockchain infrastructure service providers may offer node provisioning and
1589 orchestration for distributed file storage systems. In addition to user account data, they may also
1590 be used to host the user interfaces themselves. Examples of such storage protocols include Filecoin
1591 [110], based on the Inter-Planetary File System (IPFS) protocol [111], and StorJ [112].

1592 Based on distributed file storage systems, protocols have also been developed for *encrypted data*
1593 *vaults* (or *containers*). Although users do not store their own data, they can control which
1594 applications have access to it using custom access control schemes that are themselves based on
1595 blockchain-based identifiers. As an example, 3Box [113] makes it possible to store user account
1596 data using an OrbitDB key-value datastore that is controlled by the user.

1597 **5.4 External Data Feeds Integration**

1598 Applications must often interact with protocols that involve external or real-world data feeds (e.g.,
1599 price feeds, event results), also called *oracles*. By design, blockchain networks are meant to
1600 provide transaction determinism but not data input verification. Thus, dedicated schemes with
1601 different trust models can be used to verify the integrity of external data feeds or providers, as
1602 discussed below.

1603 **Trusted Intermediaries:**

1604 Signed data feeds operated by trusted intermediaries can be provided as APIs. It is possible to
1605 verify the authenticity of the data using the associated public keys. Frameworks are emerging to
1606 improve interoperability of signed data feeds, such as the Open Oracle System [114]. Additionally,
1607 a data feed can be run and signed from within a TEE to provide a higher degree of trustworthiness
1608 [115].

1609 **Community-Controlled Oracle Networks:**

1610 To reduce or even eliminate single points of failure, another option is to integrate incentivized
1611 peer-to-peer oracle networks. They aim to provide tamper-proof, off-chain data inputs for
1612 blockchain networks and smart contracts on top of them by aggregating multiple data reporting
1613 sources. Different architectures and incentivization schemes are emerging allowing general-
1614 purpose frameworks, such as ChainLink [116], where a collection of independent oracle networks
1615 report on individual data feed types, and Band Protocol [117], where a consolidated oracle network
1616 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 hybrid architecture models 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.

Hybrid Architecture Models:

As discussed previously in Section 5, applications can integrate infrastructure components that are external to blockchain networks and do not necessarily follow the same peer-to-peer 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 overall weakest component in the system. 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 infrastructure components built upon second layer protocols can have decentralized governance, reducing or even eliminating the need for trusted intermediaries, though the degree of decentralization is not always fixed.

As such, an end-to-end assessment of risk factors across all of the individual components involved is generally needed, with a particular attention on centralization risks. Coming with their own governance and security models distinct from those of the underlying blockchain networks, second layer protocols for token issuance and management must also be central to that assessment. In particular, some architectures involve protocol modules that are meant to be immutable as well as modules that are upgradable with their own protocol upgrade mechanisms.

Each user may 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 manage security and blockchain networks 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 both at the base layer and at the smart contract layer, giving some degree of protocol control to privileged entities, which could thus be held accountable for their actions. This entails having to make 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 the state of a blockchain.

Data Governance and Security:

The decoupling between intermediaries, the custody of digital assets, and the capability for users to control custody themselves result 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 the 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 are only processed when they are cryptographically signed. 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 [118]. 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*. Additionally, user-controlled wallets can be seen as forming an *edge computing infrastructure* since on-device data processing can take place closer to the source of the data and data exchange can be conducted over peer-to-peer communication channels (see Section 3.1).

The security risks entailed by this user-centric system architecture are multilevel. First and foremost, blockchain networks and consensus models have varying levels of security and immutability. Additionally, users must be able to hold and manage their private keys securely to avoid tokens being lost or stolen. Recoverability techniques must be assessed to meet individual needs [119]. At the smart contract layer, external data sources can be attacked or add inaccurate data to the blockchain (this is also referred to as *oracle risk*). Protocols and smart contracts can also be subject to different types of bugs [120], 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 can help mitigate these 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, risks related to collateral management. 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.

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, 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 then 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; it 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 assets¹¹ 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 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 the conceptualization of notions that combine computer science, distributed systems, cryptography, and protocol engineering with economics, game theory, and mechanism design. 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” [121], “token engineering” [122], and “cryptoeconomic primitives” [123].

Acting as algorithmically programmed inflation, new tokens can be minted to distribute rewards. Users can be required to have a minimum account balance or to stake tokens for some period of time. They may be exposed to penalties during this period. 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 or a protocol treasury in exchange for some services. Holding or staking tokens may earn yields and give privileges such as being authorized as a node or as a voter for self-enforceable protocol upgrades. This may also give responsibilities such as the setting and maintenance of system risk parameters. Depending on protocols, voting power is not necessarily

¹¹ 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.

proportional to the amount staked and may be delegated. 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 market participants. Token supply management policies can be meant to follow pre-determined rules and schedules or be subject to change. Community-controlled networks are also called user-owned or incentivized peer-to-peer networks.

For networks built upon staking-based governance rights, the level of decentralization may be perceived as depending on if and when the community of stakeholders becomes 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 fair and wide distribution of protocol-native tokens can be of fundamental importance, though there is no one-size-fits-all governance model. While still emerging, different token distribution approaches have been followed where tokens are purchased, converted, earned in exchange for services (e.g., 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 these tokens is determined by its 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 unit of denomination 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) pre-determines 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 incentivizes them to develop the utility of the token and the value proposition of the network that it supports with the token used as a tool to bootstrap adoption.

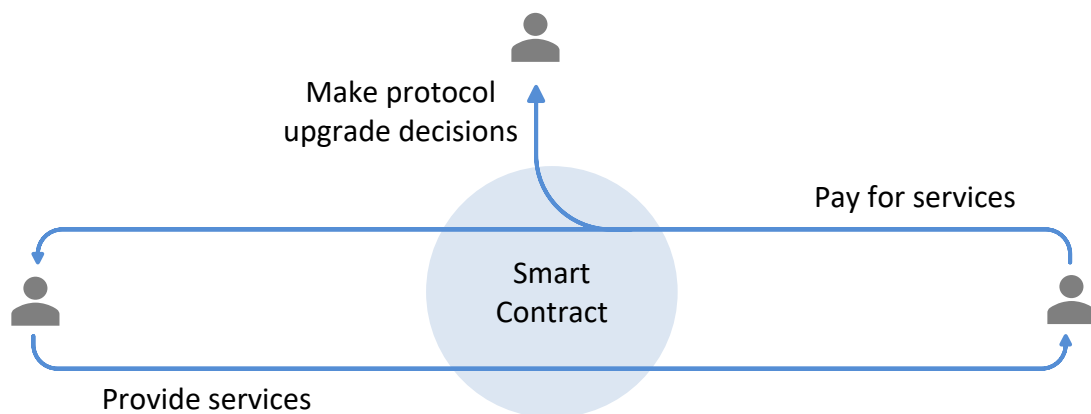


Figure 9: Smart Contract Multi-Sided Platforms

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 enable a separate, shared 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 9. They are still being experimented, are likely diverse, and do not always involve protocol-native tokens. 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 *Protocol Restrictions* in Section 2.1.2.

6.2 Tokenizing Money and Financial Products

This section discusses protocols that issue tokens to represent existing assets, enable lending and borrowing, and support token-based fundraising and derivatives.

6.2.1 Stablecoins

When pegged to or redeemable for underlying assets, the value of fungible tokens is meant to be extrinsic, backed by a collateral or reserve. These tokens are also called *stablecoins*. Some represent bearer instruments that provide ownership claims and grant redemption value for the underlying assets. Protocol-level mechanisms aim 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. In some cases, the reserve is composed of multiple, separate assets, and the peg is maintained with a specific responsiveness level. 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. On-chain collaterals involve tokens being provably locked up using cryptographic schemes or deposit smart contracts, providing auditability (see Section 3.4). Funds can also be collateralized off-chain by a trusted central authority (e.g., in a traditional bank account). 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).

This section discusses different types of stablecoin designs at a high level. *A Classification Framework for Stablecoin Designs* [124] 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.

1795 Cryptocurrencies:

1796 As discussed in Section 4.1.1.4 and Section 4.1.3, blockchain-native tokens (i.e., ownerless or non-
1797 sovereign cryptocurrencies) can be represented on a separate blockchain through a bridge that
1798 collateralizes them and provides proof of that collateral. They are also referred to as *wrapped*
1799 *cryptocurrencies* or *wrapped tokens*.

1800 WBTC [125] is meant to represent bitcoins as tokens that follow the ERC-20 standard on the public
1801 Ethereum network through a consortium, the members of which act as notaries. The minting of
1802 WBTC tokens involves two different roles: 1) merchants, who sign mint requests and provide
1803 liquidity for the WBTC/BTC pair, and 2) custodians, who process these requests. The token factory
1804 contract is jointly owned by the consortium members via multi-signature. tBTC [126] (associated
1805 with the Cross-Chain Group [127]) allows a similar representation using a relay and incentives
1806 coupled with an overcollateralized reserve rather than trusted intermediaries. Each bitcoin deposit
1807 is represented as a nonfungible token redeemable for fungible tBTC tokens.

1808 Fiat Currencies and Bank Deposits:

1809 Trusted intermediaries and consortiums can issue tokens that represent commercial bank deposits
1810 on top of their own supporting blockchain networks, as in JPM Coin [128] on top of a private
1811 blockchain controlled by JP Morgan and Libra Coin (and the associated LibraUSD, LibraEUR,
1812 and LibraGDP) [129] on top of a public consortium blockchain controlled by the members of the
1813 Libra Association. Alternatively, trusted intermediaries and consortiums can use existing
1814 blockchains that they do not control themselves, as in the CENTRE protocol [130] on top of the
1815 public Ethereum network.

1816 Unlike the previous examples where the tokens have known issuers and represent ownership
1817 claims, pegged tokens can also be issued by community-controlled protocols using on-chain
1818 overcollateralization, a separate floating token, and a decentralized price feed, as in MakerDAO
1819 [131]. In this example, both the token meant to be pegged to the U.S. dollar, called Dai, and the
1820 floating token used for protocol governance, called Maker, are deployed on the public Ethereum
1821 network. Characteristically, this design allows the representation of fiat currencies without
1822 involving deposits held in traditional bank accounts. A proof-of-stake blockchain with built-in
1823 reserve and non-custodial exchange can also be used to issue pegged tokens, as in Celo [132].

1824 Note that the term “stablecoin” is sometimes used to refer specifically to tokenized representations
1825 of fiat currencies and bank deposits, as described in this section.

1826 Central Bank Reserves:

1827 Tokens are also being studied to build central bank digital currencies (CBDC), which would
1828 represent central bank reserves. Deployment on top of dedicated consortium or private blockchains
1829 for use by the private sector or the general public have been considered, though most projects are
1830 at early stages. Note that tokenizing central bank reserves would have substantial ramifications for
1831 financial inclusion and economic policymaking. At the same time, it would introduce new system
1832 counterparty, security, and privacy risks that do not exist with self-contained banknotes that

1833 circulate freely in society today. This paper is not meant to provide any considerations on those
1834 ramifications, risks, or potential benefits.

1835 In the architectures introduced in [133], a dedicated permissioned blockchain is bootstrapped with
1836 role-based access control and voting systems for blockchain nodes administration, token supply
1837 management, and system security operations. Account providers allow identity verification tiers
1838 with different permissioning structures (e.g., transfer amount per period).

1839 The Digital Dollar Project initiative [134] aims to advance CBDC research. Several CBDC design
1840 choices have been identified to add controls to transfers between non-custodial wallets [135].

1841 **6.2.2 Lending and Borrowing**

1842 Tokens can be used to record what is owned but also what is owed. They can be lent by being
1843 deposited in a smart contract vault that mints new units of its own token to represent these deposits
1844 (i.e., tokenized collateral, debt, or liabilities). These newly minted tokens can be redeemed for the
1845 underlying ones plus interest or be used, transferred, or collateralized again separately. To mitigate
1846 default risks, borrowers are usually required to provide a collateral that is greater than the amount
1847 that they intend to borrow. This eliminates the need for assessment of individual user profiles.
1848 Rules can also be implemented directly within the deposit contracts to automate fund transfers in
1849 case of a default. The collateralization ratio and interest rate can be determined algorithmically
1850 using oracles and community-controlled governance. It can also be possible to receive and pay
1851 back a loan in a single transaction wherein token units are borrowed from a smart contract vault,
1852 used to facilitate a separate transaction, and transferred back with interest to the smart contract. If
1853 the loan is not paid off, the transaction is reverted. Borrowers may also be able to receive loans
1854 that are collateralized with nonfungible tokens.

1855 Lending protocols have several types of risk, as mentioned in Section 5.5. Note that composing
1856 tokens with one another to create lending and borrowing instruments introduces a chain of trust
1857 and, thus, new types of systemic risks. Stablecoin designs that represent ownership claims on the
1858 underlying asset, as discussed in the previous section, can also be seen as a form of tokenized
1859 liability. In lending protocols, yields (interests) are usually earned passively, unlike staking yields
1860 where active participation in protocol governance may be expected (see Section 6.1).

1861 **6.2.3 Fundraising and Derivatives**

1862 Tokens may be used for fundraising (e.g., multi-round, fixed-price sales), including some protocol-
1863 native tokens that are also meant to have utility purposes (see Section 6.1). However, they can be
1864 subject to external governance and regulatory frameworks, often depending on the exact token
1865 sales or distribution model and its framing to the public. Projects may attempt to issue protocol-
1866 native tokens and claim that they have or will have utility when they, in fact, primarily serve as a
1867 fundraising mechanism, are unnecessary for the protocol, or may even burden its usability. At the
1868 same time, some protocol-native tokens may have no or low utility at the time of issuance but gain
1869 utility purposes later on by becoming integral to the operations of the network that they support as
1870 it gets more adopted and decentralized. Thus, it is generally essential for protocol founders who
1871 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 may be used to facilitate the issuance and initial distribution of the tokens.

Although regulatory aspects are out of scope for this paper (more information can be found in [13]), token distribution models are key aspects of token designs since rules and conditions to mint or release the 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. 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 tradable using exchanges (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 both identity and supply chain management. Used in identity management, these uniquely identifiable tokens promote user-centricity and interoperability, easing user onboarding through reusable credentials; they can follow the Verifiable Credentials [16] standard, as discussed in [15]. Used in supply chain management, uniquely identifiable tokens enable data provenance across organizations, end-to-end asset traceability throughout the lifecycle, and more integrated exchanges. These tokens can represent physical things or “digital twins,” such as for food inventory management (e.g., IBM’s Food Trust [136]), and access rights or immaterial things, such as software licenses and legal agreements. 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* [137] and the Department of Homeland Security’s *Blockchain and Suitability for Government Applications* [138]. The Hyperledger Supply Chain Special Interest Group [139] also aims to provide reference architectures and frameworks for the blockchain logistics and supply chain industry, such as Hyperledger Grid [140].

The ownership of nonfungible tokens can be subdivided into a set of tokens, which can be 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 that was subdivided into a set of fungible tokens is sometimes called a *re-fungible token*.

1912 Financial, Business, and Legal Documents:

1913 Both nonfungible and stateful tokens can represent financial, business, and legal documents or
1914 agreements, such as purchase orders, invoices, letters of credit, grants, deposits, and certain
1915 securities. Nonfungible tokens may be fractionalized into fungible tokens or deployed as part of a
1916 smart contract that manages multiple token types and instantiations to allow payments and
1917 conditions (e.g., expiration dates, limited pool of recipient accounts). As new assets get tokenized
1918 and exchanges and marketplaces integrated into third-party applications, regular users may
1919 increasingly act as market participants and internally interact with financial services.

1920 Centrifuge [141] features factory contracts for minting nonfungible tokens that incorporate verified
1921 attributes from stateful tokens. The mint operation requires the submission of a set of Merkle root
1922 hashes stored on-chain for specific fields of the concerned credential. OpenLaw [142] enables the
1923 creation of on-chain binding legal agreements and offers a markup language and templates to
1924 create factory contracts for nonfungible tokens. The Baseline protocol [143] aims to enable
1925 synchronized business logic and confidential data processing between organizations on top of the
1926 public Ethereum network using zero-knowledge proofs based on Nightfall (see Section 4.3.2).

1927 Identity Documents, User Credentials, and Domain Names:

1928 Uniquely identifiable tokens can also be used to represent user credentials and identity documents,
1929 such as digital driver licenses, passports, and employee or student badges. For example, a company
1930 can decide to issue digital employee badges as uniquely identifiable tokens on a public or
1931 consortium blockchain (shared with business partners and other stakeholders) to speed up
1932 authentication of staff members with external organizations and web services and permit the
1933 posting of endorsed and timestamped data. Some personally identifiable data attached to a
1934 nonfungible or stateful token may be publicly viewable on-chain, allowing entities to publicly
1935 share information about themselves, such as a service endpoint at which they can be reached.

1936 Examples of systems that enable the tokenization of user credentials through stateful tokens
1937 include uPort [144], Blockcerts [145], and Hyperledger Indy [146]. In uPort, stateful tokens take
1938 the form of JWTs (see Section 2.2) and are coupled with a smart contract issuer registry deployed
1939 on the public Ethereum network that follows the ERC-780 [18] standard, the ownership of which
1940 was relinquished. In Blockcerts, stateful tokens take the form of JSON-LD objects that are coupled
1941 with an issuer-controlled smart contract registry, which contains both a list of addresses authorized
1942 to revoke tokens and a list of revoked tokens. In Hyperledger Indy, each issuer of stateful tokens
1943 must publish a revocation registry containing a cryptographic accumulator that allows users to
1944 verify whether a given credential was revoked by the issuer without compromising the registry's
1945 privacy.

1946 Domain names can be represented as nonfungible tokens, as in the Ethereum Name Service (ENS).
1947 ENS allows the registration and linkage of blockchain addresses (or other hashes) to human-
1948 readable identifiers (e.g., following the format “username.eth”) through auctions and nominal fees.
1949 The token factory contract follows the ERC-721 standard and acts as registry for the mapping
1950 between blockchain addresses and readable identifiers, as well as for referencing domain resolvers.

1951 These tokenized identity documents, user credentials, and domain names may be used as identifiers
1952 and log-in services for users online, providing alternatives to traditional password-based
1953 authentication and verification methods, such as emails, phone numbers, and single sign-on
1954 solutions.

1955 **Vouchers and Event Tickets:**

1956 As mentioned in the section introduction, vouchers and event tickets can either be assigned to a
1957 particular person or unassigned and tradable using exchanges (also see *Exchanging Event Tickets*
1958 *and Coupons* in Section 7 on *Use Cases* of [15]). Stateful tokens and nonfungible tokens can be
1959 used, respectively.

1960 **Academic Certificates:**

1961 Several projects [147][148] have enabled the issuance of tokens that represent diplomas and other
1962 academic certificates. Research and standardization efforts in the domain are currently being
1963 spearheaded by initiatives such as the Digital Credentials Consortium [149].

1964 **Public Organization Registrations:**

1965 Uniquely identifiable tokens can serve as proofs of registration for businesses and nonprofits (see
1966 *Verifying Business Identity* in Section 7 on *Use Cases* of [15]). This could enable a local
1967 government or chamber of commerce to maintain a smart contract registry upon which stateful
1968 tokens reflect legal organization filings. This structure has for instance been deployed by the
1969 governments of British Columbia and Ontario through the Verifiable Organization Network [150]
1970 using Hyperledger Indy.

1971 **Cryptocollectibles:**

1972 *Cryptocollectibles* are nonfungible tokens that represent digital goods, such as trading cards (e.g.,
1973 CryptoKitties [151]), video game artifacts, artwork (Maecenas [152]), and virtual land (e.g.,
1974 Decentraland [153]).

1975 **Dataset Ownership and Sharing:**

1976 Nonfungible tokens can be used to uniquely identify content or datasets, enabling novel
1977 monetization and data sharing frameworks, such as [154] to monetize web content through
1978 memberships and [155] to focus on scientific data sharing. Tokens that identify datasets can also
1979 lay the foundations for incentive structures that encourage collaboration by rewarding contributors
1980 for access to their datasets or models. For example, the Ocean Protocol [156] offers privacy-
1981 preserving and role-based access control features for data marketplaces, where users can trade
1982 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 [15]). The proofs themselves can take the form of self-contained tokens for off-chain document exchange (see Section 2.2).

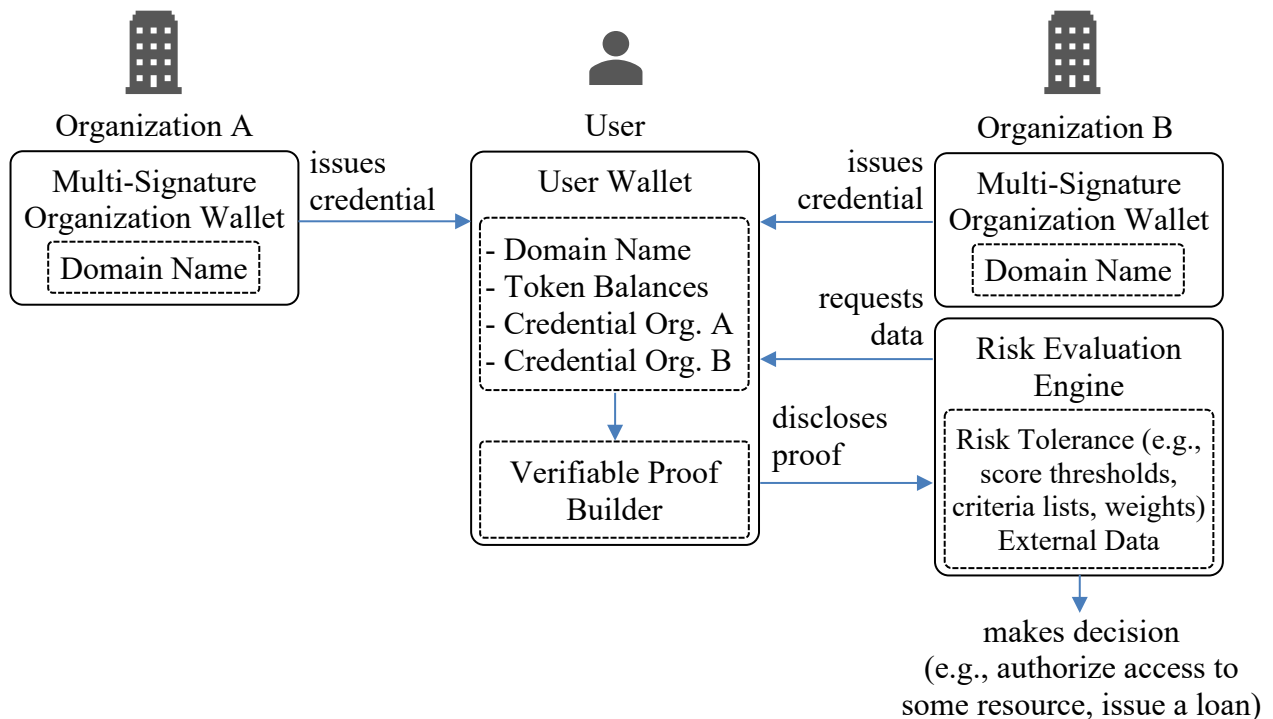


Figure 10: Verifiable Proof-Based Decision-Making

2003 On-device processing techniques (as discussed in Section 3.1) can be used to build smart wallets
2004 that help users offload the complexity of building multi-token proofs adapted to specific relying
2005 parties, user preferences, and contexts. Reciprocally, methods can be built on the relying party side
2006 to request specific information or help advertise the types of proofs that they expect to receive
2007 voluntarily from users. As shown in Figure 10, open standards and protocols for security and credit
2008 risk evaluation could help build new types of adaptive, passwordless access control systems,
2009 lending, or insurance services based on verifiable proofs. For example, consider delegated or
2010 uncollateralized loans. Users could convince lenders of their creditworthiness using token-based
2011 proofs acting as verifiable personal data sources (or “tokenized reputation” [157]). Verifiable
2012 device data could also be exchanged for devices equipped with HSM or TEE (see Section 3.1).
2013 The development of open protocols for verifiable data exchange has a compounding effect that
2014 stems from the ability of each new protocol to reuse verifiable data feeds provided by pre-existing
2015 protocols. For example, a collateral in one protocol can be reused to condition transactions in other
2016 systems. More generally, the standardization of the components involved in token-based protocols,
2017 privacy-preserving verifiable data exchange methods, and cryptographically signed data feeds
2018 could lead to the emergence of a new interoperable digital framework to help reach agreements,
2019 control access to digital resources, and conduct business on the web. Giving data property rights
2020 to users that allow for trustworthy and privacy-preserving data sharing could benefit society
2021 through reduced data hoarding and more efficient data allocation [158]. In this context, the Linux
2022 Foundation has recently launched the Trust over IP Foundation (ToIP) [159] to build open
2023 standards and software for the trustworthy exchange and verification of data between any two
2024 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 even 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 that they are composed of, 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 tools 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, this 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.

2064 **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*. <https://ssrn.com/abstract=3437529>
- [3] Buterin V, Vogelsteller F (2015) *EIP 20: ERC-20 Token Standard*. Available at <https://eips.ethereum.org/EIPS/eip-20>
- [4] Dafflon J, Baylina J, Shababi T (2017) *EIP 777: ERC-777 Token Standard*. Available at <https://eips.ethereum.org/EIPS/eip-777>
- [5] Dossa A, Ruiz P, Vogelsteller F, Gosselin S (2018) *ERC-1410 Partially Fungible Token Standard*. Available at <https://github.com/ethereum/eips/issues/1410>
- [6] Gierlach R, Poole J, Borda M, Baker L (2018) *ERC-1404 Simple Restricted Token Standard*. Available at <https://github.com/ethereum/eips/issues/1404>
- [7] 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>
- [8] 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>
- [9] InterWork Alliance (2020) *Token Taxonomy Framework*. Available at <https://github.com/InterWorkAlliance/TokenTaxonomyFramework/blob/master/token-taxonomy.md>
- [10] InterWork Alliance (2020) *Token Taxonomy Framework - Book*. Available at <https://github.com/InterWorkAlliance/TokenTaxonomyFramework/blob/master/TTF-Book.pdf>
- [11] InterWork Alliance (2020) *InterWork Alliance*. Available at <https://interwork.org>
- [12] 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 www.zora.uzh.ch/id/eprint/157908/1/To%20Token%20or%20not%20to%20Token_%20Tools%20for%20Understanding%20Blockchain%20Toke.pdf
- [13] 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>
- [14] 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>

- [15] 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. <https://csrc.nist.gov/publications/detail/white-paper/2020/01/14/a-taxonomic-approach-to-understanding-emerging-blockchain-idms/final>
- [16] 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>
- [17] 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>
- [18] Torstensson J (2017) *ERC-780 Ethereum Claims Registry*. Available at <https://github.com/ethereum/EIPs/issues/780>
- [19] 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>
- [20] International Organization for Standardization (2020) *ISO 22739:2020 - Blockchain and distributed ledger technologies — Vocabulary* (Technical Committee 307). <https://www.iso.org/standard/73771.html>
- [21] Hardman D (2019) *Aries RFC 0005: DID Communication*. Available at <https://github.com/hyperledger/aries-rfcs/tree/master/concepts/0005-didcomm>
- [22] 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*. <https://github.com/WebOfTrustInfo/rwot6-santabarbara/blob/master/final-documents/did-auth.md>
- [23] Fetch.ai (2019) *How autonomous agents use machine learning to transact on the Fetch.ai network*. Available at <https://fetch.ai/how-autonomous-agents-use-machine-learning-to-transact-on-the-fetch-ai-network>
- [24] Torus (2020) *DirectAuth*. Available at <https://directauth.io>
- [25] Wuille P (2012) *BIP-32: Hierarchical Deterministic Wallets*. Available at <https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki>
- [26] 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>
- [27] 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>
- [28] Shamir A (1979) How to share a secret. Communications of the ACM, Volume 22, Issue 11. <https://doi.org/10.1145/359168.359176>

- [29] Blakley GR (1979) Safeguarding cryptographic keys. *1979 International Workshop on Managing Requirements Knowledge (MARK)* (IEEE), pp 313-318.
<https://www.computer.org/csdl/proceedings-article/afips/1979/50870313/12OmNCeK2a1>
- [30] Gudgeon L, Moreno-Sanchez P, Roos S, McCorry P, Gervais A (2019) SoK: Off The Chain Transactions. *Cryptology ePrint Archive, Report 2019/360*. <https://ia.cr/2019/360>
- [31] 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>
- [32] Brainbot Labs Establishment (2020) *The Raiden Network*. Available at <https://raiden.network>
- [33] 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.
<https://arxiv.org/pdf/1707.05454.pdf>
- [34] 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*.
<https://eprint.iacr.org/2018/642.pdf>
- [35] Poon J, Buterin V (2017) *Plasma: Scalable Autonomous Smart Contracts*. Available at <https://plasma.io/plasma.pdf>
- [36] 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>
- [37] 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>
- [38] Buterin V (2018) *Minimal Viable Plasma*. Available at <https://ethresear.ch/t/minimal-viable-plasma/426>
- [39] Buterin V (2018) *On-chain scaling to potentially ~500 tx/sec through mass tx validation*. Available at <https://ethresear.ch/t/on-chain-scaling-to-potentially-500-tx-sec-through-mass-tx-validation/3477>
- [40] Herlihy M (2018) Atomic Cross-Chain Swaps. *Proceedings of the 2018 ACM symposium on principles of distributed computing*, pp 245-254. <https://arxiv.org/pdf/1801.09515.pdf>
- [41] Wyvern Protocol (2020) *Wyvern Protocol*. Available at <https://wyvernprotocol.com>
- [42] OpenSea Inc (2020) *OpenSea*. Available at <https://opensea.io>
- [43] Algorand (2020) *Algorand - Atomic Transfers*. Available at https://developer.algorand.org/docs/features/atomic_transfers
- [44] Decred (2020) *Decred-compatible cross-chain atomic swapping*. Available at <https://github.com/decred/atomicswap>
- [45] 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>
- [46] Web3 Foundation (2020) *XCMP – Relay chain light client design*. Available at <https://research.web3.foundation/en/latest/polkadot/XCMP.html>
 - [47] Tendermint Inc – Interchain Foundation (2020) *Cosmos - Inter-Blockchain Communication*. Available at <https://cosmos.network/ibc>
 - [48] Buterin V (2019) *Sidechains vs Plasma vs Sharding*. Available at https://vitalik.ca/general/2019/06/12/plasma_vs_sharding.html
 - [49] Web3 Foundation (2020) *Polkadot*. Available at <https://polkadot.network>
 - [50] Ethereum Foundation (2020) *Sharding-FAQs*. Available at <https://eth.wiki/sharding/Sharding-FAQs>
 - [51] 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*.
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3452714
 - [52] 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).
<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6539637>
 - [53] Wanchain Foundation Ltd (2020) *Wanchain*. Available at <https://www.wanchain.org>
 - [54] Tendermint Inc (2020) *Cosmos*. Available at <https://cosmos.network>
 - [55] Ethereum (2020) *BTC Relay*. Available at <http://btreelay.org>
 - [56] RSK Labs (2020) *RSK*. Available at <https://www.rsk.co>
 - [57] Andresen G, Hearn M (2013) *BIP-70: Payment Protocol*. Available at <https://github.com/bitcoin/bips/blob/master/bip-0070.mediawiki>
 - [58] Weiss Y, Tirosch D, Forshtat A (2018) *EIP 1613: ERC-1613 Gas stations network*. Available at <https://eips.ethereum.org/EIPS/eip-1613>
 - [59] GSN alliance (2020) *Ethereum Gas Station Network*. Available at <https://www.opengsn.org>
 - [60] 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>
 - [61] 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).
<https://csrc.nist.gov/publications/detail/conference-paper/2000/07/26/nist-model-for-rbac-towards-a-unified-standard>
 - [62] OpenZeppelin (2020) *OpenZeppelin docs - Access Control*. Available at <https://docs.openzeppelin.com/contracts/3.x/access-control>

- [63] Mell P, Delaitre A, De Vault F, Dessauw P (2019) Implementing a Protocol Native Managed Cryptocurrency. *The Fourteenth International Conference on Software Engineering Advances (ICSEA 2019)*. <https://www.nist.gov/publications/implementing-protocol-native-managed-cryptocurrency>
- [64] OpenZeppelin (2020) *TPL - Transaction Permission Layer*. Available at <https://tplprotocol.org>
- [65] PegaSys (2020) *Permissioning Smart Contracts*. Available at <https://github.com/PegaSysEng/permissioning-smart-contracts>
- [66] NIST – Cryptographic Technology Group (2020) *Privacy-Enhancing Cryptography - Project Overview*. Available at <https://csrc.nist.gov/projects/pec>
- [67] EY Blockchain (2020) *Nightfall*. Available at <https://github.com/EYBlockchain/nightfall>
- [68] ZoKrates (2020) *ZoKrates*. Available at <https://zokrates.github.io>
- [69] 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>
- [70] 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>
- [71] Aztec Protocol (2020) *Aztec Protocol - 1.0.0 specification*. Available at <https://github.com/AztecProtocol/specification>
- [72] Béres F, Seres I, Benczúr András, Quintyne-Collins M (2020) Blockchain is Watching You: Profiling and Deanonimizing Ethereum Users. *Eprint arXiv preprint arXiv:2005.14051*. Available at <https://arxiv.org/pdf/2005.14051.pdf>
- [73] BitcoinWiki (2020) *CoinJoin*. Available at <https://en.bitcoin.it/wiki/CoinJoin>
- [74] Chaum D (1984) Blind Signature System. *Advances in cryptology (Springer, Boston, USA)*. https://doi.org/10.1007/978-1-4684-4730-9_14
- [75] Liu Y, Wang Q (2017) An E-voting Protocol Based on Blockchain. *Cryptology ePrint Archive, Report 2017/1043*. <https://eprint.iacr.org/2017/1043.pdf>
- [76] Buterin V (2017) *Zk-SNARKs: Under the Hood*. (Medium). Available at <https://medium.com/@VitalikButerin/zk-snarks-under-the-hood-b33151a013f6>
- [77] 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. <https://people.csail.mit.edu/rivest/pubs/RST01.pdf>
- [78] 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
- [79] 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
- [80] 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-sharing-financial-services.pdf>
- [81] Enterprise Ethereum Alliance (2020) *Enterprise Ethereum Alliance Client Specification v6*. Available at <https://entethalliance.github.io/client-spec/spec.html>
- [82] Hyperledger Foundation (2020) *Hyperledger Besu*. Available at <https://www.hyperledger.org/projects/besu>
- [83] PegaSys (2020) *Orion Private Transaction Manager*. Available at <https://docs.orion.pegasys.tech/en/latest>
- [84] J.P. Morgan (2020) *Quorum*. Available at <https://github.com/jpmorganchase/quorum>
- [85] J.P. Morgan (2020) *Tessera*. Available at <https://github.com/jpmorganchase/tessera>
- [86] Enigma (2020) *Enigma*. Available at <https://enigma.co>
- [87] Secret Network (2020) *Secret Network*. Available at <https://scrt.network>
- [88] 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. <https://eprint.iacr.org/2015/675.pdf>
- [89] NuCypher (2020) *The NuCypher Network*. Available at <https://www.nucypher.com/network>
- [90] 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. <https://www.usenix.org/conference/usenixsecurity18/presentation/kalodner>
- [91] Hyperledger Foundation (2020) *Hyperledger Avalon*. Available at <https://www.hyperledger.org/projects/avalon>
- [92] Enterprise Ethereum Alliance (2020) *Enterprise Ethereum Alliance Off-Chain Trusted Compute Specification v1.1*. Available at <https://entethalliance.github.io/trusted-computing/spec.html>
- [93] Russinovich M, Ashton E, Avanesians 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, Vrousseau 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>
- [94] The Linux Foundation (2020) *Confidential Computing Consortium*. Available at <https://confidentialcomputing.io>
- [95] Coinbase (2020) *Rosetta - Documentation*. Available at <https://www.rosetta->

- api.org/docs/principles_introduction.html
- [96] Graph Protocol (2020) *The Graph - A Query Protocol for Blockchains*. Available at <https://thegraph.com>
 - [97] Decentralized Identity Foundation (2020) *Universal Resolver*. Available at <https://github.com/decentralized-identity/universal-resolver>
 - [98] 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
 - [99] 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>
 - [100] Malhotra A, King D, Schwartz D, Zochowski M (2020) *PayID Protocol*. Available at <https://payid.org/whitepaper.pdf>
 - [101] 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>
 - [102] Interledger Project (2020) *Interledger*. Available at <https://interledger.org>
 - [103] Finlay D, Marks E (2019) *EIP 2255: Wallet Permissions Standard*. Available at <https://eips.ethereum.org/EIPS/eip-2255>
 - [104] Finlay D (2019) *Introducing Web3 Plugins*. (Medium - Metamask) Available at <https://medium.com/metamask/introducing-the-next-evolution-of-the-web3-wallet-4abdf801a4ee>
 - [105] 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>
 - [106] WalletLink (2020) *WalletLink*. Available at <https://www.walletlink.org>
 - [107] WalletConnect (2020) *WalletConnect*. Available at <https://www.walletconnect.org>
 - [108] Web3Modal (2020) *Web3Modal*. Available at <https://github.com/Web3Modal/web3modal>
 - [109] Decentralized Identity Foundation (2019) *DIF Identity Hubs*. Available at <https://github.com/decentralized-identity/identity-hub/blob/master/explainer.md>
 - [110] Filecoin (2020) *Filecoin*. Available at <https://filecoin.io>
 - [111] Protocol Labs (2020) *IPFS*. Available at <https://ipfs.io>
 - [112] Storj Labs (2018) *Storj*. Available at <https://storj.io>
 - [113] 3Box (2020) *3Box*. Available at <https://3box.io>
 - [114] Berry C (2019) *The Open Oracle System*. (Medium - Compound) Available at <https://medium.com/compound-finance/announcing-compound-open-oracle->

[development-cff36f06aad3](#)

- [115] 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.
<https://eprint.iacr.org/2016/168.pdf>
- [116] Ellis S, Juels A, Nazarov S (2017) *ChainLink - A Decentralized Oracle Network*. Available at <https://link.smartcontract.com/whitepaper>
- [117] Band Protocol (2020) *Band Protocol*. Available at <https://bandprotocol.com>
- [118] Buterin V (2019) *Control as Liability*. Available at https://vitalik.ca/general/2019/05/09/control_as_liability.html
- [119] 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>
- [120] 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.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8886793>
- [121] 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>
- [122] McConaghy T (2018) *Towards a Practice of Token Engineering*. (Medium – Ocean Protocol). Available at <https://blog.oceanprotocol.com/towards-a-practice-of-token-engineering-b02feeff7ca>
- [123] Coinbase (2018) *The Emergence of Cryptoeconomic Primitives*. (Medium – The Coinbase Blog). Available at <https://blog.coinbase.com/the-emergence-of-cryptoeconomic-primitives-14ef3300cc10>
- [124] 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>
- [125] Kyber Network, BitGo Inc, Republic Protocol (2019) *Wrapped Tokens*. Available at <https://www.wbtc.network/assets/wrapped-tokens-whitepaper.pdf>
- [126] tBTC (2020) *tBTC Security Model*. Available at <https://tbtc.network/developers/tbtc-security-model>
- [127] Cross-Chain Group (2020) *Introducing Cross-Chain Group*. Available at <https://crosschain.group>
- [128] J.P. Morgan (2020) *J.P. Morgan Creates Digital Coin for Payments*. Available at <https://www.jpmorgan.com/global/news/digital-coin-payments>

- [129] 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
- [130] Centre (2018) *Centre Whitepaper*. Available at <https://www.centre.io/pdfs/centre-whitepaper.pdf>
- [131] Maker Foundation (2020) *The Maker Protocol: MakerDAO's Multi-Collateral Dai (MCD) System*. Available at <https://makerdao.com/en/whitepaper>
- [132] 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
- [133] 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
- [134] The Digital Dollar Project (2020) *Digital Dollar Project*. Available at <https://www.digitaldollarproject.org>
- [135] 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
- [136] IBM (2020) *IBM Food Trust*. Available at <https://www.ibm.com/blockchain/solutions/food-trust>
- [137] Krma 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>
- [138] 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
- [139] Hyperledger Foundation (2020) *Supply Chain SIG*. Available at <https://wiki.hyperledger.org/display/SCSIG/Supply+Chain+SIG>
- [140] Hyperledger Foundation (2020) *Hyperledger Grid*. Available at <https://wiki.hyperledger.org/display/GRID/Hyperledger+Grid>
- [141] Centrifuge (2020) *Centrifuge*. Available at <https://centrifuge.io>
- [142] Wright A, Roon D, ConsenSys AG (2020) *OpenLaw*. Available at <https://www.openlaw.io>
- [143] Oasis Open Projects (2020) *Baseline Protocol*. Available at <https://docs.baseline->

protocol.org

- [144] uPort (2020) *uPort*. Available at <https://www.uport.me>
- [145] Blockcerts (2020) *Blockcerts*. Available at <https://www.blockcerts.org>
- [146] Hyperledger Foundation (2020) *Hyperledger Indy*. Available at <https://www.hyperledger.org/use/hyperledger-indy>
- [147] Durant E, Trachy A (2017) *Digital Diploma debuts at MIT*. (MIT News). Available at <http://news.mit.edu/2017/mit-debuts-secure-digital-diploma-using-bitcoin-blockchain-technology-1017>
- [148] Sansone K (2019) *Malta is first country to put education certificates on blockchain*. (MaltaToday). Available at https://www.maltatoday.com.mt/news/national/93148/malta_is_first_country_to_put_education_certificates_on_blockchain#.XrMK1C-z3UJ
- [149] 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>
- [150] VON (2019) Verifiable Organizations Network. Available at <https://vonx.io>
- [151] Dapper Labs (2020) *CryptoKitties*. Available at <https://www.cryptokitties.co>
- [152] Maecenas Fine Art (2020) *Maecenas*. Available at <https://www.maecenas.co/whats-maecenas>
- [153] Decentraland (2020) *Decentraland*. Available at <https://decentraland.org>
- [154] Unlock (2020) *Unlock Protocol*. Available at <https://unlock-protocol.com>
- [155] Shrestha A, Vassileva J (2019) User Data Sharing Frameworks: A Blockchain- Based Incentive Solution. *IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8936137>
- [156] Ocean Protocol Foundation (2020) *Ocean Protocol*. Available at <https://docs.oceanprotocol.com/concepts/introduction>
- [157] Born C (2019) *Tokenized Reputation*. (Medium - TheCapital). Available at <https://medium.com/the-capital/tokenized-reputation-dee463fbc631>
- [158] Jones C, Tonetti C (2020) Nonrivalry and the Economics of Data. *National Bureau of Economic Research*. https://christophertonetti.com/files/papers/JonesTonetti_DataNonrivalry.pdf
- [159] Joint Development Foundation Projects (2020) *Trust Over IP Foundation*. Available at <https://trustoverip.org>
- [160] Nakamoto S (2008) *Bitcoin: A Peer-to-Peer Electronic Cash System*. Available at <https://bitcoin.org/bitcoin.pdf>
- [161] Proof of Stake Alliance (2020) *PoS Alliance*. Available at

<https://www.proofofstakealliance.org>

- [162] Buterin V (2015) *On Public and Private Blockchains*. Available at <https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains>
- [163] WebAssembly Community Group (2020) *WebAssembly*. Available at <https://webassembly.org>
- [164] 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>
- [165] Decentralized Identity Foundation (2020) *SideTree Protocol*. Available at <https://identity.foundation/sidetree/spec>
- [166] 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. <https://arxiv.org/pdf/1902.00878.pdf>

2065

2066

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 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 that 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, based on their respective stakes, may be allowed to propose blocks and then vote for the winning block. 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 [160]. Both of these types of consensus models enable transactions between untrusted participants, in permissionless networks, and are generally seen as providing a high level of immutability 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 [161].

Characteristically, permissioned consensus models do not rely on blockchain-native tokens. Scalability is generally increased, though this comes at the expense of reduced expectations of immutability [162]. It is possible that 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

2108 nodes but do not involve exclusive governance by a central authority. They are referred to as
2109 partially decentralized since governance rights are shared among consortium members running
2110 nodes in the network. Unlike permissionless blockchains, the rules to become a node in the
2111 network are not deterministically and independently enforced by the blockchain protocol. There
2112 must generally be an explicitly defined external governance framework that organizes a
2113 community of known participants. It can have on-chain features, such as access control or
2114 tokenized identity artifacts. Depending on business requirements and design characteristics,
2115 decentralizing governance is not always relevant. A central authority may be deemed trustworthy,
2116 a transition period preferable, or the benefits not necessarily worth the costs. Blockchains with
2117 centralized governance (i.e., a central authority selects the nodes or delegates that power), often
2118 referred to as *private blockchains*, have been used to build append-only distributed ledgers with
2119 fault tolerance and cryptographically verifiable transaction logs that are open for others to use
2120 without relinquishing control of the system.

2121 Per EEA's Client Specification [81], *transaction finality* "occurs when a transaction is definitely
2122 part of the blockchain and cannot be removed. A transaction reaches finality after some event
2123 defined for the relevant blockchain occurs. For example, an elapsed amount of time or a specific
2124 number of blocks added." In a consortium blockchain, where nodes are partially trusted to behave
2125 appropriately, transaction finality can usually be considered *deterministic* (or *absolute*). When
2126 deterministic, a transaction is deemed final as soon as it provably satisfies an explicit condition,
2127 such as being added to a block. In a permissionless blockchain, transaction finality is often
2128 *probabilistic*. The more blocks are added after a transaction is posted, the more final the
2129 transaction. This has fundamental ramifications for participants' expectations of data integrity and
2130 risks.

2131 Additionally, blockchain protocols enable virtual machines that offer limited instruction sets (e.g.,
2132 Bitcoin Script) or provide general-purpose programming environments (e.g., the WebAssembly
2133 open standard, or Wasm) [163], allowing *smart contract* execution for second layer protocols.
2134 Note that some blockchain platforms provide highly modular and configurable protocols (e.g.,
2135 pluggable consensus models, programming environments).

2136 **Appendix B—Acronyms**

2137 Selected acronyms and abbreviations used in this paper are defined below.

2138	API	Application Programming Interface
2139	CBDC	Central Bank Digital Currency
2140	DAD	Decentralized Autonomic Data
2141	DEX	Decentralized Exchange
2142	DID	Decentralized Identifier
2143	DLT	Distributed Ledger Technology
2144	D2D	Device-To-Device
2145	EEA	Enterprise Ethereum Alliance
2146	ENS	Ethereum Name Service
2147	ERC	Ethereum Request for Comment
2148	HD	Hierarchical Deterministic
2149	HSM	Hardware Security Module
2150	HTLC	Hashed Timelock Contract
2151	ISO	International Organization for Standardization
2152	ITL	Information Technology Laboratory
2153	JSON	JavaScript Object Notation
2154	JWT	JSON Web Token
2155	LTE	Long-Term Evolution
2156	MPC	Multi-Party Computation
2157	NIST	National Institute of Standards and Technology
2158	NIST-IR	National Institute of Standards and Technology Internal Report
2159	NFC	Near-Field Communication
2160	NFT	Nonfungible Token
2161	TTE	Trusted Execution Environment
2162	TTI	Token Taxonomy Initiative
2163	UTXO	Unspent Transaction Output
2164	WLAN	Wireless Local Area Network
2165	W3C	World Wide Web Consortium

2166 **Appendix C—Glossary**

2167	Account	An entity in a blockchain identified with an address and usually managed in a wallet.
2168		
2169	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.
2170		
2171		
2172	Airdrop [15]	A distribution of digital tokens to a list of blockchain addresses.
2173	Atomic Swap	An exchange of tokens that does not involve the intervention of any trusted intermediaries and automatically reverts if all of the provisions are not met.
2174		
2175		
2176	Authentication [164]	Verifying the identity of a user, process, or device, often as a prerequisite for allowing access to resources in an information system.
2177		
2178		
2179	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.
2180		
2181		
2182		
2183		
2184		
2185		
2186		
2187	Blockchain Explorer	A software for visualizing blocks, transactions, and blockchain network metrics (e.g., average transaction fees, hashrates, block size, block difficulty).
2188		
2189		
2190	Blockchain Subnetwork	A blockchain network that is interconnected with one or more other blockchain networks, as found in sharding and sidechains.
2191		
2192	Consensus Model [1]	A process to achieve agreement within a distributed system on the valid state.
2193		
2194	Consortium	A group of organizations or individuals with the objective of mutualizing resources for achieving a common goal (e.g., operating a consortium blockchain).
2195		
2196		
2197	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.
2198		
2199		
2200		
2201		
2202		
2203	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
2204		

2205		custodian may act as an exchange and provide additional services,
2206		such as staking, lending, account recovery, or security features.
2207	Fungible	Refers to something that is replaceable or interchangeable (i.e., not
2208		uniquely identifiable).
2209	Hash [15]	The output of a hash function (e.g., hash(data) = digest). Also known
2210		as a message digest, digest, hash digest, or hash value.
2211	JSON Web Token	A data exchange format made of a header, payload, and signature
2212	[14, Adapted]	where the header and the payload take the form of JSON objects.
2213		They are encoded and concatenated with the aggregate being signed
2214		to generate a signature.
2215	Merkle Tree [1]	A data structure where the data is hashed and combined until there
2216		is a singular root hash that represents the entire structure.
2217	Mint	A protocol-level operation that creates and distributes new tokens to
2218		blockchain addresses, either individually or in batch.
2219	Multi-Signature	A cryptographic signature scheme where the process of signing
2220		information (e.g., a transaction) is distributed among multiple
2221		private keys.
2222	Non-Custodial	Refers to an application or process that does not require users to
2223		relinquish any control over their data or private keys.
2224	Nonfungible [15]	Refers to something that is uniquely identifiable (i.e., not
2225		replaceable or interchangeable).
2226	Off-Chain [15]	Refers to data that is stored or a process that is implemented and
2227		executed outside of any blockchain system.
2228	On-Chain [15]	Refers to data that is stored or a process that is implemented and
2229		executed within a blockchain system.
2230	Oracle [15]	A source of data from outside a blockchain that serves as input for a
2231		smart contract.
2232	Permissioned [1]	A system where every node, and every user must be granted
2233		permissions to utilize the system (generally assigned by an
2234		administrator or consortium).
2235	Permissionless [1]	A system where all users' permissions are equal and not set by any
2236		administrator or consortium.
2237	Permissions [1]	Allowable user actions (e.g., read, write, execute).
2238	Resolver [15]	Software that retrieves data associated with some identifier.
2239	Separation of Concerns	A design principle for breaking down an application into modules,
2240		layers, and encapsulations, the roles of which are independent of one
2241		another.

2242	Sidechain	A blockchain with its own consensus mechanism and set of nodes that is connected to another blockchain through a two-way bridge.
2243		
2244	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 second layer.
2245		
2246		
2247	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.
2248		
2249		
2250	State Update	An on-chain transaction used to anchor the current state of an external ledger onto the underlying blockchain.
2251		
2252	Stateful	Refers to a data representation or a process that is dependent on an external data store.
2253		
2254	Stateless	Refers to a data representation or a process that is self-contained and does not depend on any external data store.
2255		
2256	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.
2257		
2258		
2259		
2260		
2261		
2262	Sybil Attack	A cybersecurity attack wherein an attacker creates multiple accounts and pretends to be many persons at once.
2263		
2264	Token	A representation of a particular asset that typically relies on a blockchain or other types of distributed ledgers.
2265		
2266	Token Factory Contract	A smart contract that defines and issues a token.
2267	Transaction [15]	A recording of an event, such as the transfer of tokens between parties, or the creation of new assets.
2268		
2269	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.
2270		
2271		
2272	Wallet [20]	An application used to generate, manage, store or use private and public keys. A wallet can be implemented as a software or hardware module.
2273		
2274		
2275	Zero-Knowledge Proof [15]	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).
2276		
2277		
2278		