Secure Asset Transfer Protocol
Internet-Draft
Intended status: Informational
Expires: 19 March 2026

V. Ramakrishna
V. Pandit
IBM Research
E. Abebe
Consensys
S. Nishad
D. Vinayagamurthy
IBM Research

15 September 2025

Protocol for Requesting and Sharing Views across Networks
draft-ramakrishna-satp-data-sharing-04

Abstract

With increasing use of DLT (distributed ledger technology) systems, including blockchain systems and networks, for virtual assets, there is a need for asset-related data and metadata to traverse system boundaries and link their respective business workflows. Systems and networks can define and project views, or asset states, outside of their boundaries, as well as guard them using access control policies, and external agents or other systems can address those views in a globally unique manner. Universal interoperability requires such systems and networks to request and supply views via gateway nodes using a request-response protocol. The endpoints of this protocol lie within the respective systems or in networks of peer nodes, but the cross-system protocol occurs through the systems' respective gateways. The inter-gateway protocol that allows an external party to request a view by an address and a DLT system to return a view in response must be DLT-neutral and mask the internal particularities and complexities of the DLT systems. The view generation and verification modules at the endpoints must obey the native consensus logic of their respective networks.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://ietf-satp.github.io/draft-ramakrishna-satp-data-sharing/draft-ramakrishna-satp-data-sharing.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ramakrishna-satp-data-sharing/.

Discussion of this document takes place on the Secure Asset Transfer Protocol Working Group mailing list (mailto:sat@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/sat/. Subscribe at https://www.ietf.org/mailman/listinfo/sat/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-satp/draft-ramakrishna-satp-data-sharing.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Blockchain systems, especially those of the permissioned variety, are a heterogeneous mix, fulfilling a diverse range of needs and built on several different DLT stacks that are not compatible with each other. Yet, in an interconnected world, the business processes running on each system cannot afford to remain isolated and the virtual assets they manage cannot afford to remain in siloes. These systems must be interoperable so that assets can move, and their properties can be reflected across network boundaries, and so that business processes can span multiple systems. Interoperability will, in effect, mimic a larger or scaled up, blockchain system composed of smaller blockchain systems without explicitly requiring those systems to coalesce.

At the core of any cross-blockchain system transaction is the ability of a system to project a view of assets and associated data recorded on its ledger to external parties, be they individual agents or other blockchain systems. On the reverse, a blockchain system must also have the ability to identify and address views of assets or associated data in another system, and further, to validate that view before its business process consumes it. View projection, addressing, and consumption, must eliminate or minimize the role of third parties to avoid loss of data privacy, control over business process, or diminishment of autonomy.

This document specifies an end-to-end protocol whereby a view request and the corresponding view response can be communicated using two or more gateway nodes connected to the end DLT systems. The gateways communicate with each other using a DLT-neutral protocol (like SATP [SATP], or variations of it) and therefore the view requests and responses must be encapsulated in DLT-neutral data formats. Beyond the gateways, and into the DLT systems, view generation and consumption rely upon native mechanisms exposed by those systems. The gateways must therefore be augmented to exercise these mechanisms to convey view requests and responses to their respective back-end systems.

The purpose of this document is to provide a technical framework to discuss the various aspects of a basic view request-response protocol via gateways acting on behalf of blockchain/DLT systems, including security and privacy considerations.

2. Terminology

The following are some terminology used in the current document:

* Blockchain system: 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 [NIST].

* Blockchain network: This is a blockchain system that is built on a network of nodes that maintain a common shared copy of the blockchain using a consensus protocol.

* Distributed ledger technology (DLT) system: Technology that enables the operation and use of distributed ledgers, where the ledger is shared (replicated) across a set of DLT nodes and synchronized between the DLT nodes using a consensus mechanism [ISO]. Every blockchain system is also a DLT system, and so we will mostly use the latter term in this draft when discussing protocols for cross-system interactions.

* Distributed ledger network: This is a DLT system that is built on a network of nodes that maintain a shared copy of the ledger or portions of it on different subsets of nodes using a consensus protocol.

* Resource Domain: The collection of resources and entities participating within a blockchain or DLT system. The domain denotes a boundary for permissible or authorized actions on resources.

* Interior Resources: The various interior protocols, data structures and cryptographic constructs that are a core part of a blockchain or DLT system. Examples of interior resources include the ledger (blocks of confirmed transaction data), public keys on the ledger, consensus protocol, incentive mechanisms, transaction propagation networks, etc.

* Exterior Resources: The various resources that are outside a blockchain or DLT system, and are not part of the operations of the system. Examples include data located at third parties such as asset registries, ledgers of other blockchain/DLT systems, PKI infrastructures, etc.

* Nodes: The nodes of the blockchain or DLT system which form the peer-to-peer network, which collectively maintain the shared ledger in the system by following a consensus algorithm.

* Gateway nodes: The nodes of the blockchain or DLT system that are functionally capable of acting as a gateway in an asset transfer. A gateway node conforms to the Secure Asset Transfer Architecture [SATA] and implements the Secure Asset Transfer Protocol (SATP) [SATP]. Being a node on the blockchain/DLT system, a gateway has at least read access to the interior resources (e.g., ledger) of the blockchain. Optionally, it may have write access to the ledger and also the ability to participate in the consensus mechanism deployed for the system. Depending on the blockchain/ DLT system implementation, some or all of the nodes may be gateway-capable.

* Gateway device identity: The identity of the device implementing the gateway functions. The term is used in the sense of IDevID (IEEE 802.1AR) or EK/AIK (in TPM1.2 and TPM2.0) [IDevID].

* Gateway owner: The VASP who legally owns and operates a gateway node within a blockchain system.

* Clients: Entities are permitted to invoke smart contracts to read or update ledger state in a blockchain or DLT system. They possess unique identities in the form of public keys. In a permissioned system, identity certificates are issued by the system’s internal providers (or certificate authorities).

* Wallets: Collection of client identities represented by public-private key pairs, and optionally certificate issued by a blockchain or DLT system’s identity providers (or certificate authorities).

* Virtual Asset: A virtual asset is a digital representation of value that can be digitally traded, or transferred as defined by the FATF [FATF].

* Virtual Asset Service Provider (VASP): Legal entity handling virtual assets as defined by the FATF [FATF].

* Ledger state: A snapshot of the information held in a distributed shared ledger, typically (though not necessarily) as a set of blockchain or DLT system. Examples of interior resources include key-and-value pairs. This information includes records of virtual assets and the state of business processes that are meaningful to the DLT system’s participants and clients. State elements and subsets may be scoped under the namespaces of given smart contracts, thereby being accessible only through invocations on those contracts.

* Smart contracts: Business workflows written in programming languages that manage the states of assets and business processes on a shared ledger in a DLT system. These contracts constrain the ability of system clients to modify ledger state and embed guards around state elements. Contracts can be invoked to read from, or write, to, a ledger. They can be deployed on several system nodes, who must agree on a given ledger state update via a consensus protocol.

* Consensus protocol/mechanism: Process by which nodes agree on a ledger state update, typically (though not mandatorily) through a smart contract invocation.

* Local transaction: A transaction triggered by a client to update the ledger state in a blockchain or DLT system, typically (though not mandatorily) through a smart contract invocation.

* View: A projection of a blockchain or DLT system’s ledger state for external consumption, i.e., for parties outside that system. This can be a single element, a subset of the state, or a function over a subset.

* View Address: A unique identifier or locator for a view into a blockchain or DLT system’s ledger. This is analogous to an HTTP URL.

* Source system: Blockchain or DLT system governing the ledger from which a view is produced.

* Destination system: System in which a view is consumed. This can be a blockchain or DLT system.

* Remote system: Counterparty system in a view request-response protocol instance. From the vantage point of the source system, this refers to the destination system, and vice versa.

* View requestor: Person or organization triggering a view request from a source network.

* Proof: A data structure containing evidence linking a view to its source system’s blockchain, or more generally, ledger. The evidence may be probabilistic and in some cases cryptographically verifiable.

* Access/exposure policy: Set of rules governing the release of a view to an external party (i.e., outside the source system), held in consensus by nodes on the source system’s ledger.

* Verification policy: Rules for validation of proofs associated with views maintained in a destination system. If the destination system is a blockchain or DLT system, these rules are held in consensus by nodes on that system’s ledger.

* View Request: A request for a made by an external party to a source blockchain or DLT system. The external party may be a client in a destination blockchain or DLT system. The request consists of a view address and various metadata, including optionally a verification policy.

* Asset locking or escrow: The conditional mechanism used within a blockchain or DLT system to make an asset temporarily unavailable for use by its owner. The condition of the asset release can be based on a duration of time (e.g., hash time locks) or other parameters.

* Gateway crash recovery: The local process by which a crashed gateway (i.e., device or system fault) is returned to a consistent and operational state, ready to resume the asset transfer protocol with the peer gateway prior to the crash event.

Further terminology definitions can be found in [NIST] and [ISO]. The term ’blockchain’ and ’distributed ledger technology’ (DLT) are used interchangeably in this document.

3. Assumptions and Principles

The following assumptions and principles underlie the design of the current interoperability architecture and protocol enabling the requesting and sharing of data from across DLT systems using gateways, and correspond to the design principles of the Internet architecture.

3.1. Design Principles

The protocol must not involve any centralized intermediaries or trusted third parties, including settlement blockchains, other than gateways. The protocol must not decrease the security of endpoint blockchain systems nor impose a higher bar on their internal consensus. Further, the protocol must not reveal any private information from a system beyond what the nodes of that network have agreed to through consensus. The protocol must enable view requests and response from systems built on any arbitrary (present or future) blockchain or distributed ledger technology. Finally, the endpoint systems must retain full autonomy over internal governance and framing of policies governing how views can be exposed and how proofs can be validated.

3.2. Operational Assumptions

The following conditions are assumed to have occurred prior to the construction of a view request:

* Syncing remote system structure, memberships, and identities: see Section 5 for details.

* Recording view access control policies in the source ledger: see Section 5 for details.

* Recording view verification policies in the destination ledger: see Section 5 for details.

4. Interoperability and Data Sharing Among Distributed Ledger Systems

Distributed ledger systems, typically maintained through consensus on a network of peer nodes, run business workflows (often as "smart contracts" [Ethe22]) and maintain transaction logs. The ledgers and their transaction histories are distinct, but the business workflows they run are often interlinked in the real world. This necessitates interoperability among the systems/networks as well as the ledgers maintained by them. Certain modes, or patterns, of interoperability have been identified as typical and useful for the enterprises and consortia that run these systems. These are listed in the Secure Asset Transfer architecture draft [SATA], namely as asset transfer, asset exchange, and data sharing, respective. Motivating use cases for these different modes have been presented in the Secure Asset Transfer use cases draft [SATU].

In this document, we will address the data sharing mode of interoperability, whereby DLT systems can export views of their ledger state with associated proofs of authenticity, control access to these views, and also address views exported by a remote DLT system. The formats of the views, addresses, requests, and access control policies are beyond the scope of this draft, and are specified in the Views and View Addresses draft [SATV]. This draft specified a request-response protocol whereby a view can be requested from one DLT system to another via those systems' respective gateways, and obtained and consumed via the same gateways. In effect, data maintained on the ledger in one system is shared with another in a way that requires no third-party system or agent acting as a mediator.

This draft will only specify a bilateral protocol, i.e., whereby a view is requested by a DLT system to exactly one other DLT system and whereby a view is supplied by a DLT system with exactly one other DLT system. Extrapolating or augmenting this protocol to involve more than two DLT systems is beyond the scope of the present draft.

5. Data Sharing Protocol

This section describes a protocol using which an external entity can request and process a view from a distributed ledger.

5.1. Goal and Overview of Protocol

Using this protocol, any entity, be it a single application or agent or a distributed ledger system itself, can supply a view address and a verification policy to a distributed ledger system and obtain a view in response, which it can then validate before processing as per need. This protocol makes no assumption about the nature of such processing, which can be executed by some module (like a smart contract) but rather prescribes the steps to be followed until the point where the view is dispatched to the module.

The diagram below illustrates the protocol whereby a distributed ledger system requests and consumes a view from another distributed ledger system. The protocol units are indicated, with sequence numbers associated with procedures or messages.

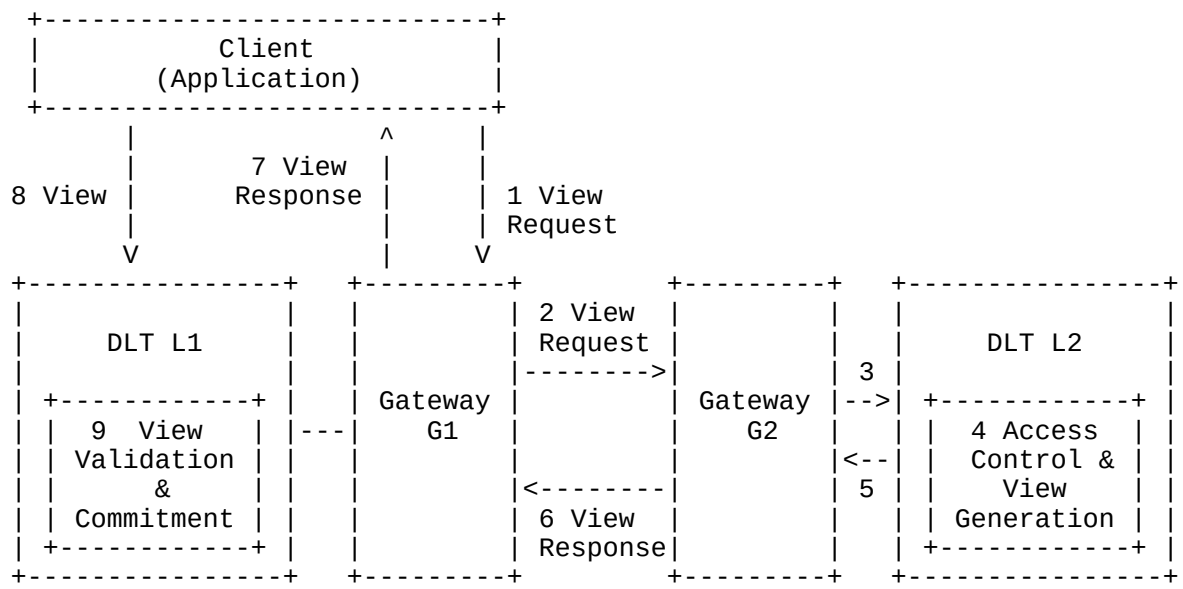

Figure 1

In a shorter form of the protocol, the requesting system is a unitary entity rather than a network of peers maintaining a distributed ledger and a client application above it. The diagram below illustrates this protocol and its units, with sequence numbers associated with procedures or messages.

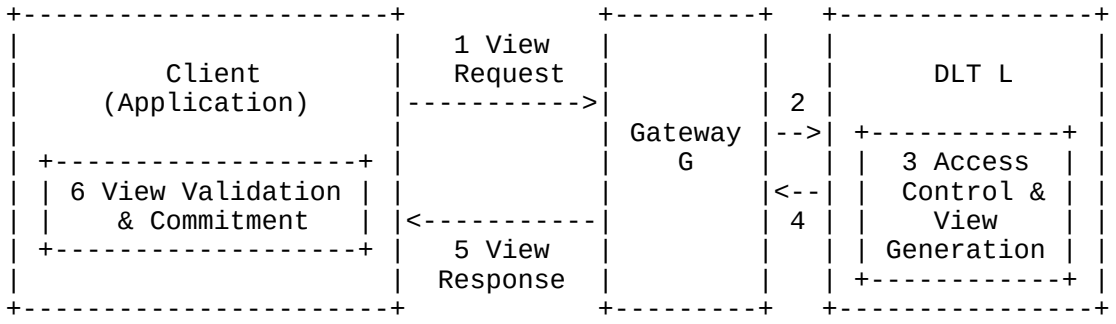

Figure 2

The distributed ledger system on the right is referred to as the source system, as it is the source of the information being requested for through a view. The system on the left is referred to as the destination system, as the desired information ends up there either in raw or in processed form. The destination system can be a traditional centrally governed system or a distributed ledger system maintained by a decentralized network.

5.2. Protocol Phases

The protocol can be divided into the following phases:

* Phase 1: View request creation in destination network.

* Phase 2: Cross-gateway discovery and request.

* Phase 3: View creation in source network.

* Phase 4: Cross-gateway response.

* Phase 5: View validation and commitment in destination system.

5.3. Phase 1: View request creation in destination network

All operations in this phase are conducted by a client, typically acting through an application, in the destination system.

* The client constructs a view address corresponding to the desired view.

* The client looks up the verification policy corresponding to this view address from the destination system's database. If the destination system is a distributed ledger system, this lookup should involve a query to one or more shared ledgers within the system.

* The client sends a view request comprising of the view address and the verification policy to the destination system's gateway.

5.4. Phase 2: Cross-gateway discovery and request

All operations in this phase are conducted by a gateway belonging to, or acting on behalf of, the destination system.

* The destination system's gateway looks up or tried to discover the address of one or more gateways belong to, or working on behalf of, the source system. If no address can be found, the protocol terminates.

* The destination system's gateway selects an address and sends the view request to the source system's gateway at that address.

5.5. Phase 3: View creation in source network

Operations in this phase are orchestrated by a source system gateway via the system's smart contract transaction and network consensus mechanisms.

* The source system's gateway converts the view request into a distributed ledger query or a smart contract invocation (if the source system supports smart contracts). The following steps show how this happens in a permissioned distributed ledger system with smart contracts.

  - The source system's gateway parses the view address within the view request to formulate a view data query and determine the ledger from which a view is desired.

  - The source system's gateway parses the verification policy in the view request to determine a subset of peers within its network that maintain this ledger and to which this query can be sent.

  - The source system's gateway invokes a smart contract to process the view data query, in effect by sending the query to the selected network peers.

  - Each peer that receives the query:

- Validates the query against the source system's access control policy.
- Processes the query and produces an output if the validation is successful.
- Packages and signs the output, which can be a query response or a failure report, as it would do with any regular smart contract transaction result. The signature constitutes part of the proof for satisfaction of the verification policy.
- Sends the signed package to the source system's gateway.

* The source system's gateway creates a view from the smart contract invocation result. If this invocation follows the process described in the preceding steps, this is done by collecting and enveloping the received packages into a view structure.

5.6. Phase 4: Cross-system response

* The source system's gateway sends the view to the destination system's gateway. It may do this as a response in a long-running session that began with the latter sending a view request to the former. Alternatively, this could be done by the former posting an event to the latter. Sending of the view could be a best effort process or guaranteed through suitable fault tolerance mechanisms built into gateways. Such mechanisms are beyond the scope of this draft.

* The destination system's gateway sends the view to the client that created the original view request. The same mechanisms and considerations apply as in the cross-gateway response in the preceding step.

5.7. Phase 5: View validation and commitment in destination system

All operations in this phase are conducted by the client in the destination system that created the original view request.

* The client parses the view to determine if the request was successful. If not, the protocol terminates.

* If the destination system is a centrally governed system with a database, the client submits a transaction, using an available API, with the view in the arguments. If the destination system is a distributed ledger system, the client submits a smart contract transaction, with the view in the arguments, to the destination system's network peers.

* The backend system (if the destination system is centrally governed) or every peer in the network (if the destination system is a distributed ledger system) that receives the transaction validates the proof within the view against the verification policy recorded in the database or the shared ledger.

* If the proof is determined to be invalid, the protocol terminates. Otherwise, the transaction is executed and then committed to storage, either through a unitary procedure (centrally governed system) or through distributed consensus (distributed ledger system).

5.8. Desirable Properties of Data Sharing Protocols

The desirable properties of a data sharing protocol include, but are not limited to, the following:

* Decoupling gateways from systems: the protocol should not rely on a specific implementation of a gateway or a cross-gateway communication protocol, nor should the protocol be restricted to a particular pair of view generation-validation processes in the respective systems. Different components and even systems can be replaced without requiring modifications to the high-level protocol semantics.

* Technology-neutral cross-gateway communication: the cross-gateway communication protocol, e.g., SATP [SATP], should be oblivious to the nature and implementation of the endpoint systems. As a corollary, the protocol units internal to the system should also be oblivious to the cross-gateway discovery and communication mechanisms.

* Trust through native consensus: end-to-end trust should be realized by implementing the view generation and validation procedures the same way as any distributed consensus-driven operation in the respective systems. This respects the native decision-making processes in those systems and enables multi-party groups backing those systems to interact with each other as units.

* Minimize trust in gateways: the integrity of the protocol should not be dependent on the reliability of either gateway. The next subsection elaborates on the desired security properties.

* Preserving system autonomy and self-sovereignty: the endpoint systems should have complete freedom in determining their respective access controls and verification policies, and in choosing when to initiate a view request or whether to respond to a view request. The internal activities of each system should be oblivious to the other and should not affect data sharing protocol instance.

* Accommodating heterogeneity: the end-to-end protocol should work the same regardless of the distributed ledger technology implementation backing either endpoint system.

* Avoid synchronization: the protocol should not rely on any externally guided synchronization mechanism, such as a global clock, and instead rely purely on asynchronous message transfers.

5.9. Security Considerations

The security considerations of the protocol include, but are not limited to, the following: - Distributed consensus: rely on the consensus mechanism of the counterparty system both to authenticate view requests and to validate views. This can mitigate (or avoid) Byzantine failure of malicious nodes within the endpoint systems. It can also prevent a malicious client from supplying a fake view in the destination system. - Integrity: rely on digital signatures over the view requests and responses (made by the client in the destination system and peers in the source system) so that the gateways cannot tamper with the messages undetected. - Confidentiality: rely on end-to-end encryption of the view response so that the gateways cannot exfiltrate the view contents and/or the associated proofs. Exfiltration will violate the access control policies of the source system. - Availability: rely on redundancy and failover to mitigate against denial-of-service attacks mounted by either gateway. Multiple gateways should be configured and kept on standby in practice.

5.10. Privacy Considerations

Access control policies allow source systems to have privacy by default, and only reveal the minimum ledger information necessary to external entities. Any data shared across two systems is private between them only if the gateways are maintained by stakeholders within the respective systems. This should be kept in mind when selecting or discovering gateways for data sharing instances.

5.11. Fault Tolerance and Crash Recovery

The usual considerations of fault tolerance, crashes, and session maintenance, apply to gateways involved in data sharing. Various techniques for failover and session state and log backups can be used to guard against, or recover from, the possibility of either gateway failing during a data sharing instance. Details are beyond the scope of this draft, which makes no recommendations on this topic either.

6. Pre-request setup and configuration

Either endpoint system in a data sharing protocol instance must have the following configured in their respective data stores, which, if the system is a distributed ledger system, should be a shared ledger.

* View request access control policies: as described in Section 8 in the Views and View Addresses draft [SATV], one or more access control policy rules according to the given specification should be recorded before a data sharing protocol instance. If a view request is received by a system and there is no appropriate policy rule in the system's record, the principle of least privilege should be applied, and the view request rejected.

* View verification policies: as described in Section 6 in the Views and View Addresses draft [SATV], one or more verification policies according to the given specification should be recorded before a data sharing protocol instance. If a view is received by a system and there is no appropriate policy rule in the system's record, the view should be deemed invalid. The client itself ought to avoid sending a view request if an appropriate verification policy cannot be retrieved, but even if a malicious client proceeds with a spurious view request, the destination system's back end should declare the proof accompanying the view response to be invalid.

* External identities and certifications: participating systems can be abstracted into one or more security domains that represent the stakeholders of that system. For effective policy enforcement, i.e., to authenticate a view request or validate a view, knowledge of a remote system's security domains—identity providers, certification authorities, and group structure if the system is a distributed ledger—is necessary. This information is typically, though not limited to, a set of certificate hierarchies, which should be determined and recorded to the system's data store before a data sharing protocol instance. In the absence of such recorded information about the counterparty system, the protocol should terminate in the view request access control check step or in the view validation step.

7. Related Open Issues

This draft provides a specification for views and how to addresses them. It further describes a protocol whereby one system can request a view from another through gateways. But there are several aspects of the end to-end process, which are extraneous to the data sharing protocol yet crucial to its successful completion. Though detailed specifications of these are beyond the scope of this draft, we list them in this section for readers' considerations.

7.1. Global identification of blockchain systems and public keys

To construct a view address as well as determine a suitable gateway to send a view request to, we need identifiers and naming systems for distributed ledgers and the networks that maintain them. In addition, we need ways to associate public keys with these names for authentication purposes. Further, we need mechanisms to store and retrieve these identifiers and keys in a distributed, and ideally decentralized, manner. The concepts of Decentralized Identifiers [DID] and Self-Sovereign Identity [SSI] are promising technologies to begin devising such specifications.

7.2. Discovery of gateways nodes within a blockchain system

To initiate a data sharing protocol, a client in a distributed ledger system needs to identify a suitable gateway node. This can be done in several different ways, one of which involves registering the set of gateways on a shared ledger within the system. Because, ideally, the gateway does not need to be highly trusted, there are few security implications but potentially larger efficiency implications in the choice of mechanism for registration and discovery of local gateway nodes.

7.3. Remote gateway discovery

How gateways can discover other gateways acting on behalf of remote distributed ledger systems is an open question. The problem can be related to the routing problem (i.e., discovery of routing paths) in packet networking [RFC791].

7.4. Decentralized identity management across distributed ledger systems

Mechanisms are required to continuously sync external identities and certifications as described in Section 5 as a basis for data sharing. To keep these mechanisms are decentralized as possible and leverage existing identity registries, we can leverage the concepts of decentralized identifiers [DID] and verifiable credentials [VC22]. Protocols for this purpose, which use DID registries and trust anchors, have been proposed [WRFCDID], and we can use them as starting points for this specification.

8. References

8.1. Normative References

[DID] Sporny, M., Longley, D., Sabadello, M., Reed, D., Steele, O., and C. Allen, "Decentralized Identifiers (DIDs) v1.1, Core architecture, data model, and representations (W3C Candidate Recommendation Snapshot)", September 2025, <https://w3c.github.io/did/>.

[FATF] FATF, "International Standards on Combating Money Laundering and the Financing of Terrorism & Proliferation - The FATF Recommendations", October 2018, <http://www.fatf-gafi.org/publications/fatfrecommendations/documents/fatf-recommendations.html>.

[ISO] ISO, "Blockchain and distributed ledger technologies- Vocabulary (ISO:22739:2024)", January 2024, <https://www.iso.org/standard/82208.html>.

[NIST] Yaga, D., Mell, P., Roby, N., and K. Scarfone, "NIST Blockchain Technology Overview (NISTR-8202)", October 2018, <https://doi.org/10.6028/NIST.IR.8202>.

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/rfc/rfc791>.

[SATA] Hardjono, T., Hargreaves, M., Smith, N., and V. Ramakrishna, "Secure Asset Transfer (SAT) Interoperability Architecture, IETF, draft-ietf-satp-architecture-08", July 2025, <https://datatracker.ietf.org/doc/draft-ietf-satp-architecture/>.

[SATP] Hargreaves, M., Hardjono, T., Belchior, R., Ramakrishna, V., and A. Chiriac, "Secure Asset Transfer Protocol (SATP) Core, IETF, draft-ietf-satp-core-11", August 2025, <https://datatracker.ietf.org/doc/draft-ietf-satp-core/>.

[SATU] Ramakrishna, V., Hardjono, T., and C. Liu, "Secure Asset Transfer (SAT) Use Cases, IETF, draft-ietf-satp-usecases-06", July 2025, <https://datatracker.ietf.org/doc/draft-ietf-satp-usecases/>.

[SATV] Ramakrishna, V., Pandit, V., Abebe, E., Nishad, S., and K. Narayanam, "Views and View Addresses for Secure Asset Transfer, IETF, draft-ramakrishna-satp-views-addresses-06", September 2025, <https://datatracker.ietf.org/doc/draft-ramakrishna-satp-views-addresses/>.

[SSI] Tobin, A. and D. Reed, "The Inevitable Rise of Self-Sovereign Identity, A white paper from the Sovrin Foundation", March 2017, <https://sovrin.org/wp-content/uploads/2018/03/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf>.

[VC22] Sporny, M., Longley, D., Chadwick, D., and I. Herman, "Verifiable Credentials Data Model v2.1 (W3C Editor's Draft)", May 2025, <https://w3c.github.io/vc-data-model/>.

8.2. Informative References

[ABCH20] Ankenbrand, T., Bieri, D., Cortivo, R., Hoehener, J., and T. Hardjono, "Proposal for a Comprehensive Crypto Asset Taxonomy", May 2020, <https://arxiv.org/abs/2007.11877>.

[BVGC20] Belchior, R., Vasconcelos, A., Guerreiro, S., and M. Correia, "A Survey on Blockchain Interoperability: Past, Present, and Future Trends", May 2020, <https://arxiv.org/abs/2005.14282v2>.

[Clar88] Clark, D., "The Design Philosophy of the DARPA Internet Protocols, ACM Computer Communication Review, Proc SIGCOMM 88, vol. 18, no. 4, pp. 106-114", August 1988.

[Ethe22] "Ethereum Whitepaper", September 2022, <https://ethereum.org/en/whitepaper/>.

[Gray81] Gray, J., "The Transaction Concept: Virtues and Limitations, in VLDB Proceedings of the 7th International Conference, Cannes, France, September 1981, pp. 144-154", September 1981.

[Herl19] Herlihy, M., "Blockchains from a Distributed Computing Perspective, Communications of the ACM, vol. 62, no. 2, pp. 78-85", February 2019, <https://doi.org/10.1145/3209623>.

[HLP19] Hardjono, T., Lipton, A., and A. Pentland, "Towards an Interoperability Architecture for Blockchain Autonomous Systems, IEEE Transactions on Engineering Management", June 2019, <https://ieeexplore.ieee.org/document/8743548>.

[HS2019] Hardjono, T. and N. Smith, "Decentralized Trusted Computing Base for Blockchain Infrastructure Security, Frontiers Journal, Special Issue on Blockchain Technology, Vol. 2, No. 24", December 2019, <https://doi.org/10.3389/fbloc.2019.00024>.

[IDevID] Richardson, M., "A Taxonomy of operational security considerations for manufacturer installed keys and Trust Anchors. IETF draft-irtf-t2trg-taxonomy-manufacturer-anchors-11", August 2025, <https://datatracker.ietf.org/doc/draft-irtf-t2trg-taxonomy-manufacturer-anchors/>.

[SRC84] Saltzer, J., Reed, D., and D. Clark, "End-to-End Arguments in System Design, ACM Transactions on Computer Systems, vol. 2, no. 4, pp. 277-288", November 1984.

[WRFCDID] Ramakrishna, V., Narayanam, K., Chandra Ghosh, B., and E. Abebe, "Decentralized Network-Identity Discovery and Management for Interoperation, Hyperledger Cacti - Weaver, RFC 01-011, LFDT", August 2022, <https://github.com/hyperledger-cacti/cacti/blob/main/weaver/rfcs/models/identity/network-identity-management.md>.

Authors' Addresses

Venkatraman Ramakrishna
IBM Research
Email: vramakr2@in.ibm.com

Vinayaka Pandit
IBM Research
Email: pvinayak@in.ibm.com

Ermyas Abebe
Consensys
Email: ermyas.abebe@consensys.net

Sandeep Nishad
IBM Research
Email: sandeep.nishad1@ibm.com

Dhinakaran Vinayagamurthy
IBM Research
Email: dvinaya1@in.ibm.com