Secure Asset Transfer Protocol
Internet-Draft
Intended status: Informational
Expires: 23 April 2026

H. Song
Seoul National Univ.
Y.-G. Hong
Daejeon Univ.
G. Jeong
TTA
T. Hardjono
MIT

20 October 2025

Secure Asset Transfer Protocol (SATP) Implementation Guide
draft-song-satp-implementation-guide-02

Abstract

This memo provides implementation guidelines for the Secure Asset Transfer Protocol (SATP). It is intended for developers of SATP gateway and digital asset networks they represent. This document also clarifies the core SATP processing workflow and offers recommendations to ensure interoperable implementations, particularly in environments where multiple gateways may represent the same network.

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://anawhj.github.io/draft-song-satp-implementation-guide/draft-song-satp-implementation-guide.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-song-satp-implementation-guide/.

Discussion of this document takes place on the Secure Asset Transfer Protocol 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/anawhj/draft-song-satp-implementation-guide.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 23 April 2026.

Copyright Notice

Table of Contents

1. Introduction

(Note: SATP core and architecture drafts are under progress of RFC publication as of October 2025, so major updates to this guide are expected following their RFC publication and new rechartering.)

The Secure Asset Transfer Protocol (SATP) provides a standardized mechanism for the secure and reliable transfer of digital assets across different asset networks. The protocol's primary scope is to define the messages and protocol flows for communication between peer gateways. In the SATP model, each gateway acts as a trusted representative for its respective network, managing the outbound (locking/burning) and inbound (unlocking/minting) transfer of assets.

A fundamental requirement for any cross-network value transfer is reliability. SATP is explicitly designed to provide transactional guarantees, ensuring that the ACID properties (Atomicity, Consistency, Isolation, and Durability) are maintained. The protocol flow, often leveraging a 2-Phase Commit (2PC) pattern, ensures that a transfer operation either completes successfully on both the origin and destination networks or is safely rolled back. This prevents critical failure states, such as the loss of an asset or its unintentional duplication.

This document, the "SATP Implementation Guide," serves as a non-normative companion to the core protocol specifications. Its purpose is to provide practical implementation guidance for SATP gateway developers and architects. It does not introduce any new normative requirements but instead clarifies the existing ones, outlines best practices, and provides concrete examples to aid in building secure, compliant, and interoperable SATP solutions.

All guidance provided herein is based on the normative definitions found in the "Secure Asset Transfer Protocol (SATP) Core" [SATcore] and the "Secure Asset Transfer (SAT) Interoperability Architecture" [SATarch] documents. Implementers are encouraged to consult those core specifications for all formal requirements, data models, and protocol state definitions. This guide SHOULD be used as a supplementary resource to understand how those normative requirements may be practically applied.

This guide elaborates on the abstract protocol by detailing common implementation scenarios. It provides concrete examples for core protocol flows, recommendations for payload structures, and best practices for critical areas such as security and privacy. Specific

attention is given to topics like key management, payload encryption (JWE) for sensitive compliance data, and mitigating protocol-level attacks.

The target audience for this document includes system architects designing gateway infrastructure, software engineers developing SATP client and server logic, and security professionals responsible for auditing and securing the implementation. This guide may also serve as a technical introduction for parties seeking to understand the practical workflow of the protocol beyond the abstract specification.

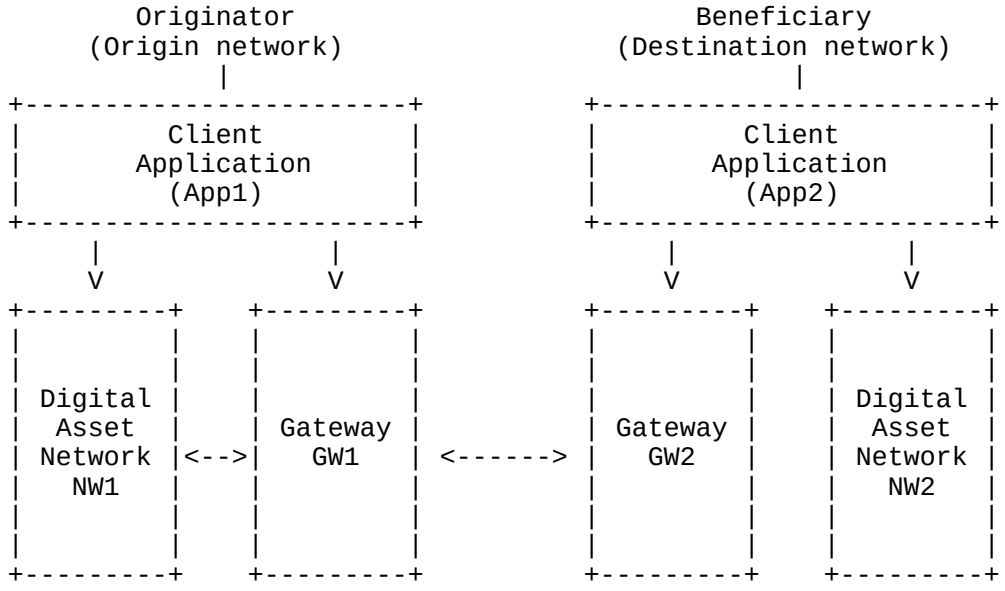

Figure 1. Scope of the SATP implementation

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

* SATP (Secure Asset Transfer Protocol): A protocol that defines the communication flow and messages between peer gateways to execute the secure transfer of a digital asset from one network to another.

* Gateway: A software component that acts as a representative for a Digital Asset Network. It implements the SATP and manages the locking, unlocking, burning, or minting of assets on its represented network.

* Originator Gateway: The gateway that initiates an asset transfer request on behalf of the asset sender.

* Beneficiary Gateway: The gateway that receives the asset transfer request on behalf of the asset recipient.

* Intermediary Gateway: An optional gateway that participates in a multi-hop transfer, acting as both a Beneficiary (to the preceding gateway) and an Originator (to the subsequent gateway).

* Digital Asset: A digital representation of value that is being transferred via SATP (e.g., a cryptocurrency, token, stablecoin, or NFT).

* Digital Asset Network (or Asset Network): The underlying system, such as a Distributed Ledger (DLT) or a centralized database, on which a digital asset is recorded and maintained.

* VASP (Virtual Asset Service Provider): The real-world operational entity (e.g., a financial institution, a crypto exchange) that operates a Gateway and is responsible for regulatory compliance.

* 2PC (Two-Phase Commit): A protocol pattern used by SATP to coordinate a transaction between two or more distributed systems (Gateways) to ensure Atomicity. It consists of a "Prepare" (or "Vote") phase and a "Commit" (or "Decision") phase.

* ACID (Atomicity, Consistency, Isolation, Durability): A set of properties that guarantee the reliability of transactions. SATP aims to provide these guarantees for the cross-network transfer itself.

* Payload: The application-level data within an SATP message, typically formatted as JSON [RFC8259] and secured with JWS [RFC7515], containing details of the transfer, assets, and compliance information.

* PII (Personally Identifiable Information): Sensitive data about the originator or beneficiary of a transfer (e.g., name, address) that is often required for regulatory compliance, such as the FATF Travel Rule.

* Secure Channel: An encrypted communication path between two gateways, typically established using Transport Layer Security (TLS) [RFC8446], which SATP mandates for all communication.

4. Core Protocol Flows

This section provides non-normative examples of the message flows between peer gateways for common SATP use cases. These flows illustrate the practical application of the 2-Phase Commit (2PC) model defined in the SATP Core [SATcore] specification, which is the foundation for providing ACID guarantees.

The protocol is divided into two primary phases:

Phase 1 (Prepare): The Originator Gateway (OG) proposes a transfer (TransferRequest), and the Beneficiary Gateway (BG) validates it and "votes" by reserving the resources and responding (TransferAcknowledgement or TransferError).

Phase 2 (Commit/Abort): The OG receives the vote. If the vote was "YES" (Ack), the OG commits the transfer on its side and instructs the BG to do the same (TransferCommit). If the vote was "NO" (Error), the OG aborts the transfer and rolls back any changes.

4.1. Use Case 1: Simple Asset Transfer (Successful)

This is the "happy path" scenario for a straightforward asset transfer where compliance checks are minimal or integrated seamlessly.

Message Flow: The flow begins when the Originator Gateway (OG) initiates the Prepare phase by sending a TransferRequest message. This message contains the asset details, amount, and destination. The Beneficiary Gateway (BG) receives this request, validates the address, and confirms it can receive the asset. The BG then sends a TransferAcknowledgement message, signaling its "Vote YES" to proceed. Upon receiving this acknowledgment, the OG enters the Commit phase, sending a TransferCommit message to the BG. The BG receives this final instruction, finalizes the transfer, and replies with a TransferFinalization (or a final simple Ack) to complete the protocol.

Network Roles: On the Originator Network, the OG places a "lock" on the user’s assets before sending the TransferRequest or immediately after, but before the Commit phase. This lock ensures the assets cannot be double-spent. Upon receiving the TransferAcknowledgement and sending the TransferCommit, the OG executes the final action (e.g., "burning" the asset or transferring it to a gateway-controlled

vault). On the Beneficiary Network, the BG's TransferAcknowledgement signifies it is ready to credit the user. Upon receiving the TransferCommit, the BG executes its final action (e.g., "minting" the new asset or releasing funds to the recipient's account).

4.2. Use Case 2: Transfer with PII Compliance (Successful)

This flow is common for transfers between VASPs that must adhere to regulations like the FATF Travel Rule, requiring the exchange of PII.

Message Flow: The message flow is identical to the simple transfer (Request -> Ack -> Commit -> Finalization). However, the content of the TransferRequest message is expanded. It now also contains the required originator and beneficiary PII, which is encrypted using JWE [RFC7516] as described in the Privacy Considerations section.

Validation and Network Roles: The primary difference lies in the BG's validation logic. Before the BG can send the TransferAcknowledgement ("Vote YES"), it decrypts the PII payload. It then validates this PII against its internal compliance engine (e.g., checking for sanctions lists, AML policies, or data formatting). If the PII is valid and passes all checks, the BG proceeds with the TransferAcknowledgement. The Originator Network's role is unchanged, but the OG is responsible for correctly collecting and encrypting this PII. The Beneficiary Network's action (minting/releasing funds) is now contingent not only on a valid TransferCommit but also on the prior success of this compliance check.

This flow ensures that no asset is moved or committed on the Originator Network until the Beneficiary Gateway has confirmed it can legally accept the transfer and the associated PII.

4.3. Use Case 3: Transfer Rejection (Rollback)

This "unhappy path" scenario demonstrates the 2PC "Abort" phase, which is critical for maintaining atomicity when a transfer cannot be completed.

Message Flow: The OG initiates the transfer by sending a TransferRequest. The BG receives this request and performs its validation (as in Use Case 1 or 2). During this validation, the BG discovers a critical error. This could be a failed compliance check (e.g., a sanctioned recipient), an invalid asset type, a non-existent destination address, or any other business rule violation.

Because it cannot proceed, the BG "Votes NO" by sending a TransferError message instead of an acknowledgment. This TransferError message contain a specific error code and a human-

readable description of the problem. Upon receiving the TransferError, the OG immediately enters the Abort phase. The protocol stops here; no TransferCommit message is ever sent.

Network Roles: The Beneficiary Network does nothing; it received a request but rejected it before taking any action on its ledger. The atomicity guarantee is fulfilled by the Originator Network. When the OG receives the TransferError, it executes a rollback. This involves removing the "lock" placed on the user's assets, returning them to the sender's control. The OG then reports the failure (and the reason provided by the BG) to the originating user.

4.4. Use Case 4: Multi-Hop (Intermediary) Transfer

This more complex scenario involves routing a transfer through one or more Intermediary Gateways (IG). This flow (e.g., A -> B -> C) essentially "chains" two separate SATP 2PC flows together.

Message Flow: The flow is initiated by the OG (Gateway A), which sends a TransferRequest to the IG (Gateway B). This request identifies Gateway C as the final beneficiary. Gateway B receives this request and acts as a "Beneficiary" to Gateway A, but also as an "Originator" to Gateway C.

Gateway B (IG) immediately initiates its own TransferRequest (Phase 1) to Gateway C (BG), forwarding the relevant asset and PII data. Gateway B doesn't send its TransferAcknowledgement ("Vote YES") back to Gateway A until it has first received a TransferAcknowledgement from Gateway C.

Network Roles and Atomicity: This nested 2PC flow creates an end-to-end atomic guarantee. The "Vote YES" (TransferAcknowledgement) messages propagate backward from the final destination (C -> B, then B -> A). The TransferCommit messages propagate forward from the origin (A -> B, then B -> C).

If the final BG (Gateway C) sends a TransferError ("Vote NO"), the IG (Gateway B) relays this failure by sending a TransferError back to the OG (Gateway A). This triggers a full, cascading rollback across all participating networks. The Intermediary Network (Network B) never takes custody of the asset; it merely acts as a routing conduit, and the protocol ensures the asset is only ever committed at the final destination.

5. Data Model and Payload

This section provides non-normative examples of the data models used in SATP messages. The core SATP specification [SATcore] defines the normative data structures and formats. All SATP messages are required to be encapsulated within a secure envelope, such as a JWS (JSON Web Signature) [RFC7515] or COSE (CBOR Object Signing and Encryption) [RFC9052], to ensure data integrity and sender authentication.

The payload of this secure envelope is the SATP message itself (e.g., TransferRequest, TransferAcknowledgement, TransferError). Implementers need to consider to agree on the serialization format, with JSON [RFC8259] and CBOR [RFC8949] being the recommended formats. The JSON payload is required to use UTF-8 encoding.

5.1. Gateway Identifiers

Gateways is identified using unique, routable identifiers. While the core specification may allow various formats, this guide RECOMMENDEDs using either a DID (Decentralized Identifier) [W3C-DID-CORE] or a secure HTTPS-based URI. The use of DIDs is particularly beneficial for automated discovery of public keys and service endpoints.

Specific examples of identifiers that can be supported:

DID (vLEI): did:vlei:GLEIF.SF4G.393O.5O9B (Uses a vLEI for VASP identification)

DID (web): did:web:vasp-a.com

URI: httpsa://satp.vasp-b.com/gateway

These identifiers are used in the originatorGateway and beneficiaryGateway fields of SATP messages.

5.2. Example Payload: TransferRequest (JSON)

The following example shows the decoded payload of a JWS token for a TransferRequest message, corresponding to Use Case 2 (Transfer with PII Compliance). This entire JSON object is what is signed and placed within the payload field of the JWS envelope.

JSON

{ "messageType": "TransferRequest", "jti": "a7d9f8b1-c2e3-4d5f-b6a7-e8f9b0c1d2e3", "iat": 1761000000, "originatorGateway": "did:vlei:GLEIF.SF4G.393O.5O9B", "beneficiaryGateway":

"did:vlei:GLEIF.9876.WXYZ.5432", "assetTransfer": { "assetType": "urn:iso:std:iso:4217:USD", "amount": "1500.00", "destinationAddress": "0xABC123...[beneficiary_address]" }, "compliancePayload": "eyJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiRUNESC1FUytB..." }

In this example, the jti (JWT ID) and iat (Issued At) claims are used for anti-replay protection, as described in the Security Considerations. The compliancePayload field contains a JWE (JSON Web Encryption) [RFC7516] token. This JWE encapsulates the sensitive PII (e.g., originator's name, address) and is encrypted for the Beneficiary Gateway, in line with the Privacy Considerations.

5.3. CBOR Representation

For implementations prioritizing minimal data size, especially in constrained environments, CBOR [RFC8949] is the recommended binary format. The SATP message can then be encapsulated using COSE [RFC9052] (e.g., a COSE_Sign1 structure) instead of JWS.

The content of the COSE payload would be the CBOR-encoded representation of the JSON object shown above. The core SATP specification [SATcore] is required to define integer mappings for common JSON string keys (e.g., "messageType" might be mapped to the integer 1, "jti" to 2, etc.) to achieve significant payload reduction. Implementers are encouraged to adhere to these official mappings to ensure cross-format interoperability.

6. Interoperability Profiles

The SATP core specification [SATcore] provides a robust and flexible framework for secure asset transfers. This flexibility, however, can lead to interoperability challenges if not properly managed. For example, one gateway may implement the protocol using JSON/JWS, while another uses CBOR/COSE. Similarly, one gateway may require mandatory PII compliance payloads for all transfers, while another only supports simple, non-compliance transfers.

To mitigate this and to provide a clear path for interoperability, this guide defines a set of Interoperability Profiles. A profile is a named set of normative features, algorithms, message formats, and protocol flows that an implementation is required to support in full to claim conformance to that profile. This allows gateways to easily discover and verify that they share a compatible set of capabilities before attempting a transfer.

Gateways are required to publicly declare which profiles they support. This can be achieved through a service discovery mechanism, such as a well-known URI (as defined in [RFC8615]) hosted at the gateway's domain, which returns a machine-readable JSON object describing the gateway's capabilities and supported profiles.

This section introduces a baseline profile, SATP-Core-v1.0. All gateways claiming to be SATP-compliant are required to implement this profile at a minimum, as it ensures baseline functionality for simple transfers. Additional profiles, such as a SATP-FATF-Compliance-v1.0 profile, may be defined in this document or in separate companion documents to address specific use cases like regulatory compliance.

The following is an example of a machine-readable JSON definition for the baseline SATP-Core-v1.0 profile. This object could be used in a service discovery document.

JSON

{ "profileName": "SATP-Core-v1.0", "version": "1.0", "description": "Baseline interoperability for simple asset transfers without PII compliance payloads.", "basedOn": ["SATcore", "SATarch"], "supportedFlows": ["SimpleAssetTransfer", "TransferRejection"], "payload": { "format": "JSON", "encoding": "UTF-8" }, "security": { "envelope": "JWS", "signatureAlgorithms": ["ES256", "ES256K"], "transport": "mTLS", "tlsVersion": ["TLSv1.3"] }, "compliance": { "supported": false }, "messageTypes": ["TransferRequest", "TransferAcknowledgement", "TransferError", "TransferCommit", "TransferFinalization"] }

7. Open Sources and Test Tooling

A robust and diverse ecosystem of implementations is essential for the success and adoption of SATP. This ecosystem includes both public, open-source projects, which serve as reference implementations and testbeds, and non-public (closed-source) commercial implementations, which validate the standard against real-world business requirements. The availability of both types of implementations allows the community to build a common understanding and refine the protocol.

On the open-source front, a key reference implementation is being developed within the Linux Foundation Decentralized Trust (lfdecentralizedtrust) community. The Hyperledger Cacti project, an interoperability framework, provides a modular architecture for connecting heterogeneous DLTs. This project hosts the SATP-Hermes plugin, which implements a fully functional SATP Gateway. This SATP-Hermes module acts as a vital, public reference implementation that other developers can test against and contribute to.

In the non-public and commercial domain, Quant Network’s Overledger platform is a prominent example. As a major contributor to the SATP standardization process, Quant utilizes the SATP gateway model as a core component of its Overledger architecture. Overledger is a commercial platform designed to provide interoperability between diverse blockchains and legacy enterprise systems, and its implementation validates the SATP framework’s applicability to institutional and enterprise-grade use cases.

The existence of multiple, independent implementations necessitates common test tooling to ensure true interoperability. It is recommended that the SATP community develops a shared Test Suite and a set of comprehensive Test Vectors (examples of which may be found in the Appendix of this guide). This test suite is required to validate not only the "happy path" flows but also all defined error states, security requirements, and protocol edge cases.

Implementers of both open and closed-source solutions are encouraged to participate in community-driven interoperability testing events, such as IETF Hackathons or "Interop Fests". These events provide a crucial venue for testing different implementations against one another, identifying ambiguities in the specification, and collectively strengthening the interoperability of the entire SATP ecosystem.

8. Security Considerations

This section expands on the normative security requirements defined in the SATP core specification [SATcore], offering practical guidance to help implementers avoid common pitfalls and securely configure their gateways.

Transport Security Configuration: The SATP core specification mandates the use of a secure channel. Implementers are encouraged to ensure this channel is properly configured to prevent eavesdropping, man-in-the-middle (MITM) attacks, and replay of the transport session. It is good practice to support TLS 1.3 [RFC8446] or a subsequent version. Implementations are encouraged to avoid negotiating or supporting any version of TLS prior to 1.2, and it is

recommended to only support TLS 1.3+. Implementations are encouraged to also avoid cipher suites known to be weak (e.g., those using RC4, 3DES, or anonymous Diffie-Hellman). It is recommended to configure servers to only support cipher suites that provide Perfect Forward Secrecy (PFS). To ensure that only authorized gateways can communicate, implementations are encouraged to use Mutual TLS (mTLS). In an mTLS exchange, the Beneficiary Gateway (server) authenticates the Originator Gateway (client) by verifying its client-side X.509 certificate, in addition to the standard server-side certificate validation performed by the client.

Gateway Key Management Lifecycle: SATP messages rely on digital signatures (JWS/COSE) for integrity and non-repudiation. The private keys used for this purpose are a high-value target and require careful protection. Private keys are required to not be stored in plaintext in configuration files or source code. It is strongly recommended that private keys be generated and stored within a Hardware Security Module (HSM) or an equivalent isolated secure environment (e.g., TEE, Secure Enclave) that prevents key exfiltration. Implementers are encouraged to have a defined policy for periodic key rotation (e.g., annually or upon suspected compromise). Implementations are encouraged to also be designed with "crypto-agility" in mind. Hard-coding specific algorithms or curves (e.g., ES256K) is discouraged, and the system is required to be capable of upgrading to new signing algorithms if vulnerabilities are discovered in the currently used ones.

Message Signature and Verification Best Practices: Improper validation of JWS [RFC7515] or COSE [RFC9052] signatures can lead to severe vulnerabilities. When verifying a signature, the alg (algorithm) header parameter is required to be checked against an explicit allow-list of acceptable algorithms maintained by the verifier. Any message received with an alg value not on this list is rejected. In particular, it is critical that the alg:none value is rejected. This guide recommends that the signature be applied to the canonicalized form of the entire message payload to prevent ambiguity. The implementation guide provides clear test vectors showing exactly which bytes are included in the signature computation. Signers are encouraged to include a Key ID (e.g., kid in JWS) in the protected header to aid the verifier in selecting the correct public key for validation, especially during and after key rotation.

Protocol-Level Attack Mitigation: The stateful nature of the 2-Phase Commit (2PC) protocol can be exploited for denial-of-service (DoS) or replay attacks. All state-changing messages (especially TransferRequest) are required to include a unique identifier (e.g., a jti claim) and a timestamp (e.g., an iat claim). A Beneficiary

Gateway maintains a record of recently processed jti values to detect and reject replayed messages. Implementations enforces strict timeouts for 2PC transactions. A Beneficiary Gateway that receives a Phase 1 (Prepare) request releases any locks or reserved resources if the transaction is not finalized (Committed or Aborted) within a predefined, reasonable timeframe (e.g., 5 minutes). This prevents resource exhaustion DoS attacks. Gateway endpoints are encouraged to implement rate limiting based on source IP, client certificate, or other identifiable metrics to mitigate volumetric DoS and brute-force attacks.

9. Privacy Considerations

Privacy is a fundamental consideration for SATP gateways, especially since the protocol often carries sensitive Personally Identifiable Information (PII) for regulatory needs (like the FATF Travel Rule). Implementers are encouraged to adopt a "Privacy by Design" approach. This involves protecting data not just from external attackers, but also managing how legitimate participants handle data to prevent misuse or leakage.

Data Minimization: It’s good practice for a gateway to avoid sending the complete set of PII it holds about an originator in every request. Instead, the TransferRequest ideally only contains the minimum set of PII fields that are strictly required by the Beneficiary Gateway’s jurisdiction or its stated compliance policy. A policy-driven approach is recommended to dynamically determine the appropriate data payload for each transfer, thereby limiting the privacy risk and exposure in every transaction.

Privacy-preserving Logging: Implementations are required to aim to ensure privacy-preserving logging. Sensitive PII (such as customer names, residential addresses, or national identification numbers) is not written in plaintext to any general-purpose application, debug, or INFO level logs. Traceability and debugging can be achieved using non-sensitive correlation identifiers, such as the jti of the TransferRequest. Furthermore, any PII received is subject to purpose binding (used only for the compliance check it was intended for) and storage limitation (securely deleted after the mandated legal retention period has expired).

Transaction Linkability and Metadata Protection: Even when the PII payload is encrypted (e.g., using JWE), the metadata of the SATP flow— such as the identities of the participating gateways, the frequency of transfers, and the timing of the messages— remains visible to the gateways and potentially to network observers. This metadata can be used to infer sensitive patterns about users and institutions. While the mandated use of TLS mitigates passive

network observation, implementers are encouraged to avoid placing any unnecessary unique identifiers in unencrypted headers to reduce the risk of linkability.

10. IANA Considerations

There are no IANA considerations related to this document.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

11.2. Informative References

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://www.rfc-editor.org/rfc/rfc7516>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/rfc/rfc8615>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

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

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

[W3C-DID-CORE] Sporny, M., Guy, A., Sabadello, M., and D. Reed, "Decentralized Identifiers (DIDs) v1.0", July 2022, <https://www.w3.org/TR/did-1.0/>.

Acknowledgments

TBA

Authors' Addresses

Hyojin Song
Seoul National Univ.
South Korea
Email: fun@snu.ac.kr

Yong-Geun Hong
Daejeon Univ.
South Korea
Email: yonggeun.hong@gmail.com

GukSik Jeong
TTA
South Korea
Email: jgsigi@tta.or.kr

Thomas Hardjono
MIT
United States of America

Email: hardjono@mit.edu