Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: 11 May 2023

M. Hargreaves
Quant Network
T. Hardjono
MIT
R. Belchior
Technico Lisboa

7 November 2022

Secure Asset Transfer Protocol
draft-hargreaves-sat-core-01

Abstract

This memo This memo describes the Secure Asset Transfer (SAT) Protocol for digital assets. SAT is a protocol operating between two gateways that conducts the transfer of a digital asset from one gateway to another. The protocol establishes a secure channel between the endpoints and implements a 2-phase commit to ensure the properties of transfer atomicity, consistency, isolation and durability.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This memo proposes a secure asset transfer (SAT) protocol that is intended to be deployed between two gateway endpoints, where the protocol itself is agnostic to the backend system or network behind the respective gateway.

The SAT protocol is defined for a unidirectional transfer, with the understanding that it can be deployed as a building block for a bidirectional transfer or exchange of assets.

There are several desirable properties of the protocol. The protocol must ensure that non-repudiability of transfers is achieved, and that the classic properties of atomicity, consistency, isolation, and durability (ACID) must be satisfied.

The requirement of consistency implies that the asset transfer protocol always leaves both networks in a consistent state (that the asset is located in one system/network only at any time).

Atomicity means that the protocol must guarantee that either the transfer commits (completes) or entirely fails, where failure is taken to mean there is no change to the state of the asset in the origin (sender) network.

The property of isolation means that while a transfer is occurring to a digital asset from an origin network, no other state changes can occur to the asset.

The property of durability means that once the transfer has been committed by both gateways, that this commitment must hold regardless of subsequent unavailability (e.g. crash) of the gateways implementing the SAT protocol.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in RFC 2119.

3. Terminology

The following are some terminology used in the current document:

Client application: This is the application employed by a user to interact with a gateway.

Gateway: The computer system functionally capable of acting as a gateway in an asset transfer.

Sender gateway: The gateway that initiates a unidirectional asset transfer.

Recipient gateway: The gateway that is the recipient side of a unidirectional asset transfer.

Claim: An assertion made by an Entity [JWT].

Claim Type: Syntax used for representing a Claim Value [JWT].

Gateway Claim: An assertion made by a Gateway regarding the status or condition of resources (e.g. assets, public keys, etc.) accessible to that gateway (e.g. within its network or system).

4. The Open Digital Asset Protocol

4.1. Overview

The Secure Asset Transfer Protocol (SAT) is a gateway-to-gateway protocol used by a sender gateway with a recipient gateway to perform a unidirectional transfer of a virtual asset.

The protocol defines a number of API endpoints, resources and identifier definitions, and message flows corresponding to the asset transfer between the two gateways.

```
                +----------+                +----------+
                |  Client  |                | Off-net  |
                | (Applic) |                | Resource |
                +----------+                +----------+
                     |                      |API Type-3|
                     |                      +----------+
                     |                           ^
                     V                           |
                +----------+                     |
                |API Type-1|                     |
     +------+   +----------+----+        +----+----------+   +------+
     |      |   |          |    |        |    |          |   |      |
     | Net. |   | Gateway  |API |        |API | Gateway  |   | Net. |
     |  L1  |---|    G1    |Type|<------>|Type|    G2    |---|  L2  |
     |      |   |          | 2  |        | 2  |          |   |      |
     +------+   +----------+----+        +----+----------+   +------+
```

Figure 1

4.2. SAT Model

The model for SAT is shown in Figure 1.

The Client (application) interacts with its local gateway (G1) over an interface (API Type-1) in order to provide instructions to the gateway with regards to actions to assets and related resources located in the local system (L1).

Gateways interact with each other over a gateway interface (API Type-2). A given gateway may be required to access resources that are not located in network L1 or network L2. Access to these types of resources are performed over an off-network interface (API Type-3).

4.3. Types of APIs

The following are the types of APIs in SAT:

* Gateway APIs for client (API Type-1): This the REST APIs that permit a Client (application) to interact with a local gateway, and issue instructions for actions pertaining to resources accessible to the gateway.

* Gateway APIs for peer gateways (API Type-2): This is the REST APIs employed by two (2) peer gateways in performing unidirectional asset transfers.

* APIs for validation of off-network resources (API Type-3): This is the REST APIs made available by a resource server (resource owner) at which a gateway can access resources.

The use of these APIs is dependent on the mode of access and the type of flow in question.

4.4. Types of Flows

The SAT protocol defines the following three (3) flows:

* Transfer Initiation flow: This flow deals with commencing a transfer from one gateway to another. Several tasks are involved, including (but not limited to): (i) gateway identification and mutual authentication; (ii) exchange of asset type (definition) information; (iii) verification of the asset definition, and others.

* Lock-Evidence flow: This flow deals with the conveyance of evidence regarding the lock status (e.g. escrow) of an asset by one gateway, and the verification of the evidence by the other gateway.

* Commitment Establishment flow: This flow deals with the asset transfer and commitment establishment between two gateways.

These flow will be discussed below.

4.5. Resources and Identifiers

* (a) Resource addressing for systems or networks, using the URL syntax.

* (b) Client identification based on the URN format. These are for identifying clients (developers and applications) who access these resources, and which in some use-cases require access authorization.

* (c) Protocol message family for negotiating authentication, authorisation, and parameters for confidential channel establishment.

- (d) Resource discovery mechanism for developers and applications to discover resources hosted at a gateway. The gateway response is subject to the level of access granted to that developer or application.

5. SAT Message Format, identifiers and Descriptors

5.1. Overview

This section describes (i) the phases of the SAT protocol; (ii) the format of SAT messages; (iii) the format for resource descriptors; (iv) a method for gateways to implement access controls; (iv) protocol for negotiating security capabilities; (v) discovery and accessing resources and provisions for backward compatibility with existing systems.

5.2. SAT Message Format

SAT messages are exchanged between applications (clients) and gateways (servers). They consist of protocol negotiation and functional messages.

Messages are in JSON format, with protocol specific mandatory fields, support for several authentication and authorization schemes and support for a free format field for plaintext or encrypted payloads directed at the gateway.

JSON format message, mandatory fields are shown below:

- Version: SAT protocol Version (major, minor).

- Session ID: unique identifier (UUIDv2) representing a session

- Sequence Number: monotonically increasing counter that uniquely represents a message from a session.

- SAT Phase: The current SAT phase.

- Resource URL: Location of Resource to be accessed.

- Developer URN: Assertion of developer / application identity.

- Action/Response: GET/POST and arguments (or Response Code)

- Credential Profile: Specify type of auth (e.g. SAML, OAuth, X.509)

- Credential Block: Credential token, certificate, string

* Payload Profile: Asset profile and capabilities

* Application Profile: Vendor or Application specific profile

* Payload: Payload for POST, responses, and local networks. The payload is specific to the current SAT phase.

* Payload Hash: hash of the current message payload.

* Message signature: Gateway EDCSA signature over the message

Other relevant attributes may exists that need to be captured for logging purposes.

5.3. Digital Asset Resource Descriptors

Resources are identified by URL [RFC 1738] as described below:

* The type is new: application/satres

* The access protocol is SAT.

Data included in the URL includes the folowing:

5.3.1. Organization Identifier

This MAY be a Legal Entity Identifier (LEI) or other identifier linking resource ownership to a real world entity. Any scheme for identifying gateway owners may be implemented (e.g. LEI directory, closed user group membership, SWIFT BIC, etc.).

The developer or application MAY validate the identity with the issuing authority. The identifier is not a trusted identity, but MAY be relied on where trust has been established between the two parties (e.g. in a closed user group).

The mechanisms to determine organizations identifiers is out of scope for the current specification.

5.3.2. Gateway / Endpoint ID

FQDN of the SAT compliant gateway. Required to establish IP connectivity. This MUST resolve to a valid IP address.

5.3.3. Network or system Identifier

Specific to the gateway behind which the target network operates. This field is local to the gateway and is used to direct SAT interactions to the correct underlying network.

For example: "tradelens-network", "EU-supply-chain".

5.3.4. Resource

Specifies a resource held on the underlying network. This field must be meaningful to the network in question but is otherwise an arbitrary string. The underlying object it points to may be a network address, data block, transaction ID, alias, etc. or a future object type not yet defined.

5.3.5. Examples

satres://quant/api.gateway1.com/swift

5.4. Digital Asset Resource Client Descriptors

Resources are identified by URN as described below:

* The type is new: application/satclient

The URN format does not imply availability or access protocol.

Data included in the URN includes the folowing:

5.4.1. Organization Identifier

Legal Entity Identifier (LEI) or other identifier linking resource ownership to a real-world entity. Any scheme for identifying Gateway owners may be implemented (e.g. LEI directory, closed user group membership, BIC, etc.).

The Gateway MAY validate the identity with the issuing authority. The identifier is not a trusted identity, but MAY be relied on where trust has been established between the two parties (e.g. in a closed user group).

5.4.2. Gateway / Endpoint ID

Applications which interact with multiple networks can operate in a mode whereby the application connects to its local gateway, which then forwards application traffic to local networks and to remote networks via other SAT gateways.

Where this is the case, this field identifies the "home" gateway for this application. This may be required to carry out gateway to gateway handshaking and protocol negotiation, or for the server to look up use case specific data relating to the client.

5.4.3. Organizational Unit

The organization unit within the organization that the client (application or developer) belongs to. This assertion should be backed up with authentication via the negotiated protocol.

The purpose of this field is to allow gateways to maintain access control mapping between applications and resources that are independent of the authentication and authorization schemes used, supporting future changes and supporting counterparties that operate different schemes.

5.4.4. Name

A locally unique (within the OU) identifier, which can identify the application, project or individual developer responsible for this client connection. This is the most granular unit of access control, and gateways should ensure appropriate identifiers are used for the needs of the application or use case.

5.4.5. Examples

satclient:quant/api.overledger.quant.com/research/luke.riley

5.5. Gateway Level Access Control

Gateways can enforce access rules based on standard naming conventions using novel or existing mechanisms such as AuthZ protocols using the resource identifiers above, for example:

satclient://hsbc/api.overledger.hsbc.com/lending/eric.devloper

can READ/WRITE

satres://quant/api.gateway1.com/tradelens

AND

satres://quant/api.gateway1.com/ripple

These rules would allow a client so identified to access resources directly, for example:

satres://quant/api.gateway1.com/tradelens/xxxxxADDRESSxxxxx

This method allows resource owners to easily grant access to individuals, groups and organizations. Individual gateway implementations may implement access controls, including subsetting and supersetting or applications or resources according to their own requirements.

5.6. Negotiation of Security Protocols and Parameters

5.6.1. TLS Established

TLS 1.2 or higher MUST be implemented to protect gateway communications. TLS 1.3 or higher SHOULD be implemented where both gateways support TLS 1.3 or higher.

5.6.2. Client offers supported credential schemes

Capability negotiation prior to data exchange, follows a scheme similar to the Session Description Protocol [RFC 5939]. Initially the client (application) sends a JSON block containing acceptable credential schemes, e.g. OAuth2.0, SAML in the "Credential Scheme" field of the SAT message.

5.6.3. Server selects supported credential scheme

The server (recipient Gateway) selects one acceptable credential scheme from the offered schemes, returning the selection in the "Credential Scheme" field of the SAT message.

If no acceptable credential scheme was offered, an HTPP 511 "Network Authentication Required" error is returned in the Action/Response field of the SAT message.

5.6.4. Client asserts or proves identity

The details of the assertion / verification step are specific to the chosen credential scheme and are out of scope of this document.

5.6.5. Sequence numbers initialized

Sequence numbers are used to allow the server to correctly order operations from the client, some of which may be asynchronous, synchronous, idempotent with duplicate requests handled in different ways according to the use case.

The initial sequence number is proposed by the client (sender gateway) after the finalization of credential verification. The server (recipient gateway) MUST respond with the same sequence number to indicate acceptance.

The client (sender gateway) increments the sequence number with each new request. Sequence numbers can be reused for retries in the event of a gateway timeout.

5.6.6. Messages can now be exchanged

Handshaking is complete at this point, and the client can send SAT messages to perform actions on resources, which MAY reference the SAT Payload field.

5.7. Asset Profile Identification

The client and server must mutually agree as to the asset type or profile that is the subject to the current transfer from the client and server. The client must provide the server with the asset-identification number, or the server may provide the client with the asset-identification numbers for the digital asset supported by the server.

Formal specification of asset identification is out of scope of this document. Global numbering of digital asset types or profiles is expected to be performed by a legally recognized entity.

5.8. Application Profile Negotiation

Where an application relies on specific extensions for operation, these can be represented in an Application Profile.

For example, a payments application tracks payments through the use of a cloud based API and will only interact with gateways that log messages to that API, a resource profile can be established:

Application Name: TRACKER

X-Tracker_URL: https://api.tracker.com/updates

X-Tracking-Policy: Always

As gateways implement this functionality, they support the TRACKER application profile, and the application is able to expand its reach by periodically polling for the availability of the profile.

This is an intentionally generalized extension mechanism for application or vendor specific functionality.

5.9. Discovery of Digital Asset Resources

Applications located outside a network or system SHOULD be able to discover which resources they are authorized to access in a network or system.

Resource discovery is handled by the gateway in front of the network. For instance using a GET request against the gateway URL with no resource identifier could return a list of URLs available to the requester. This list is subject to the access controls above.

Gateways MAY allow applications to discover resources they do not have access to. This should be indicated in the free text field, and gateways SHOULD implement a process for applications to request access.

Formal specification of supported resource discovery methods is out of scope of this document.

6. Identity and Asset Verfication Flow (Phase 0)

Prior to commencing the asset transfer from the sender gateway (client) to the recipient gateway (server), both gateways must perform a number of verifications steps. The types of information required by both the sender and recipient are use-case dependent and asset-type dependent.

The verifications include, but not limited to, the following:

* Gateway identity mutual verification: This is the identity of the gateway at the protocol and network layer. This may include the public-keys of the gateways.

* Gateway owner verification: This is the verification of the identity (e.g. LEI) of the owners of the gateways.

* Gateway device and state validation: This is the device attestation evidence [RATS] that a gateway must collect and convey to each other, where a verifier is assumed to be available to decode, parse and appraise the evidence.

* Originator and beneficiary identity verification: This is the identity and public-key of the entity (originator) in the origin network seeking to transfer the asset to another entity (beneficiary) in the destination network.

These are considered out of scope in the current specifications, and are assumed to have been successfully completed prior to the commencement of the transfer initiation flow.

7. Transfer Initiation Flow (Phase 1)

This section describes the SAT initialization phase, where a sender gateway interacts with a recipient gateway, proposing a session.

For this, several artifacts need to be validated: asset profile, asset ownership evidence, identities, and logging-related operations (log profile, access control profile).

In this phase, gateways implement the Transfer Initiation Flows endpoint.

In the following, the sender gateway takes instructions from an application, while the recipient gateway may act on behalf of its applications.

The flow follows a request-response model. The sender gateway makes a request (POST) to the Transfer Initiation endpoint at the recipient gateway.

Gateways MUST support the use of the HTTP GET and POST methods defined in RFC 2616 [RFC2616] for the endpoint.

Clients (sender gateway) MAY use the HTTP GET or POST methods to send messages in this phase to the server (recipient gateway). If using the HTTP GET method, the request parameters may be serialized using URI Query String Serialization.

The client and server may be required to sign certain messages in order to provide standalone proof (for non-repudiation) independent of the secure channel between the client and server. This proof may be required for audit verifications (e.g. post-event).

(NOTE: Flows occur over TLS. Nonces are not shown).

7.1. Initialization Request Message

This message is sent from the sender gateway (client) to the recipient gateway (server).

The purpose of this message is for the client to initiate an asset transfer. Depending on the proposal, multiple rounds of communication between the client and the server may happen.

The parameters of this message consists of the following:

- Version: SAT protocol Version (major, minor).

- Developer URN: Assertion of developer / application identity.

- Credential Profile: Specify type of auth (e.g. SAML, OAuth, X.509)

- Payload Profile: Asset Profile provenance and capabilities

- Application Profile: Vendor or Application specific profile

- logging_profile REQUIRED: contains the profile regarding the logging procedure. Default is local store.

- Access_control_profile REQUIRED: the profile regarding the confidentiality of the log entries being stored. Default is only the gateway that created the logs can access them.

- Initialization Request Message signature REQUIRED: Gateway EDCSA signature over the message

- Sender_gateway_pubkey REQUIRED: the public key of the gateway initiating a transfer

- Sender_gateway_network_id REQUIRED: the ID of the source network

- Recipient_gateway_pubkey REQUIRED: the public key of the gateway involved in a transfer

- Recipient_gateway_network_id REQUIRED: the ID of the recipient network

- Escrow type: faucet, timelock, hashlock, hashtimelock, multi-claim PC, destroy/burn (escrowed cross-claim).

- Expiry time: when will the escrow or lock expire

- Multiple claims allowed: true/false

- Multiple cancels allowed: true/false

- Permissions: list of identities (public-keys or X.509 certificates) that can perform operations on the escrow or lock on the asset in the origin network.

- Origin: along with the sender gateway identification, allows identifying from where are the asset is escrowed/provided

- Destination: along with the recipient gateway identification, allows identifying to where are the escrowed asset is going

- Subsequent calls: details possible escrow actions

- History (optional): provides an history of the escrow, in case it has previously been initialized.

The sender gateway makes the following HTTP request using TLS.

Example: TBD.

7.2. Initialization Request Message Response (ACK)

After receiving an Initialization Request Message, the recipient gateway needs to validate the profiles.

This validation could be performed automatically (using a defined set of rules), or by requiring approval by an application.

If one of the profiles is rejected, the recipient gateway constructs a Initialization Denied Message, stating what was rejected, and proposing an alternative (if applicable).

Otherwise, if approved, the recipient gateway constructs an Initialization Request Message Response.

The purpose of this message is for the server (recipient gateway) to indicate agreement to proceed with the proposed operations, under the proposed profiles.

This message is sent from the recipient gateway to the sender gateway (client) in response to a Initialization Request from the sender gateway.

The message must be signed by the server.

The parameters of this message consists of the following:

- Session ID: unique identifier (UUIDv2) representing a session.

- Sequence Number: monotonically increasing counter that uniquely represents a message from a session.

- SAT Phase: The current SAT phase.

* Hash of the Initialization Request Message REQUIRED: the hash of the proposal.

* Destination: if the recipient gateway calculates the destination address dynamically.

* Timestamp REQUIRED: timestamp referring to when the Initialization Request Message was received.

* Processed Timestamp REQUIRED: timestamp referring to when the Initialization Response Message was constructed.

Example: TBD.

8. Lock-Evidence Verification Flow (Phase 2)

This section describes the conveyance of claims regarding the status of the asset (or resource) from a sender gateway to a recipient gateway.

In this phase, the gateways implement the Lock-Evidence Agreement endpoint.

In the following, the sender gateway takes the role of the client while the recipient gateway takes the role of the server.

The flow follows a request-response model. The client makes a request (POST) to the Lock-Evidence Agreement endpoint at the server.

Gateways MUST support the use of the HTTP GET and POST methods defined in RFC 2616 [RFC2616] for the endpoint.

Clients MAY use the HTTP GET or POST methods to send messages in this phase to the server. If using the HTTP GET method, the request parameters may be serialized using URI Query String Serialization.

The client and server may be required to sign certain messages in order to provide standalone proof (for non-repudiation) independent of the secure channel between the client and server. This proof may be required for audit verifications post-event.

(NOTE: Flows occur over TLS. Nonces are not shown).

8.1. Transfer Commence Message

This message is sent from the client (sender gateway) to the Transfer Request Endpoint at the server (recipient gateway). It signals to the server that the client is ready to start the transfer of the digital asset.

The message must contain claims related to the information from the previous flow (Phase 1). It must be signed by the client.

The parameters of this message consists of the following:

* message_type REQUIRED. MUST be the value urn:ietf:sat:msgtype:transfer-commence-msg

* originator_pubkey REQUIRED. This is the public key of the asset owner (originator) in the origin network or system.

* beneficiary_pubkey REQUIRED. This is the public key of the beneficiary in the destination network.

* sender_gateway_network_id REQUIRED. This is the identifier of the origin network or system behind the client.

* recipient_gateway_network_id REQUIRED. This is the identifier of the destination network or system behind the server.

* client_identity_pubkey REQUIRED. The public key of client who sent this message.

* server_identity_pubkey REQUIRED. The public key of server for whom this message is intended.

* hash_asset_profile REQUIRED. This is the hash of the asset profile previously agreed upon in Phase 1.

* asset_unit OPTIONAL. If applicable this is the unit amount of the asset being transferred, previously agreed upon.

* hash_prev_message REQUIRED. The hash of the last message in Phase 1.

* client_transfer_number OPTIONAL. This is the transfer identification number chosen by the client. This number is meaningful only the client.

* client_signature REQUIRED. The digital signature of the client.

For example, the client makes the following HTTP request using TLS (with extra line breaks for display purposes only):

```
POST /token HTTP/1.1
  Host: server.example.com
  Authorization: Basic awHCaGRSa3F0MzpnWDFmQmF0M2ZG
  Content-Type: application/x-www-form-urlencoded

      {
      "message_type": "urn:ietf:sat:msgtype:transfer-commence-msg",
      "originator_pubkey":"zGy89097hkbfgkjvVbNH",
      "beneficiary_pubkey": "mBGHJjjuijh67yghb",
      "sender_net_system": "originNETsystem",
      "recipient_net_system":"recipientNETsystem",
      "client_identity_pubkey":"fgH654tgeryuryuy",
      "server_identity_pubkey":"dFgdfgdfgt43tetr535teyrfge4t54334",
      "hash_asset_profile":"nbvcwertyhgfdsertyhgf2h3v4bd3v21",
      "asset_unit": "ghytredcfvbhfr",
      "hash_prev_message":"DRvfrb654vgreDerverv654nhRbvder4",
      "client_transfer_number":"ji9876543ewdfgh",
      "client_signature":"fdw34567uyhgfer45"
      }
```

Figure 2

8.2. Transfer Commence Response Message (Ack)

The purpose of this message is for the server to indicate agreement to proceed with the asset transfer.

This message is sent from the server (recipient gateway) to client (sender gateway) in response to a Transfer Commence Request from the client.

The message must be signed by the server.

The parameters of this message consists of the following:

* message_type REQUIRED urn:ietf:sat:msgtype:transfer-commenceack-msg

* client_identity_pubkey REQUIRED. The client for whom this message is intended.

- server_identity_pubkey REQUIRED. The server who sent this message.

- hash_commence_request REQUIRED. The hash of previous message.

- server_transfer_number OPTIONAL. This is the transfer identification number chosen by the server. This number is meaningful only to the server.

- server_signature REQUIRED. The digital signature of the server.

An example of a success response could be as follows:

```
     HTTP/1.1 200 OK
     Content-Type: application/json;charset=UTF-8
     Cache-Control: no-store
     Pragma: no-cache

     {
       "message_type":"urn:ietf:sat:msgtype:transfer-commenceack-msg",
       "client_identity_pubkey":"fgH654tgeryuryuy",
       "server_identity_pubkey":"dFgdfgdfgt43tetr535teyrfge4t54334",
       "hash_commence_request":"DRvfrb654vgreDerverv654nhRbvder4",
       "server_transfer_number":"ji9876543ewdfgh",
       "server_signature":"aaw34567uyhgfer66"
     }
```

Figure 3

8.3. Lock Evidence Message

The purpose of this message is for the client (sender gateway) to deliver the relevant asset locking (or escrow) evidence to the server (recipient gateway).

The format of the evidence is dependent on the network or system fronted by the client and is outside the scope of this specification.

This message is sent from the client to the Evidence Validation Endpoint at the server.

The server must validate the lock evidence claims (payload) in this message prior to the next step.

The message must be signed by the client (sender gateway).

The parameters of this message consists of the following:

* message_type REQUIRED urn:ietf:sat:msgtype:lock-evidence-req-msg

* client_identity_pubkey REQUIRED. The client who sent this message.

* server_identity_pubkey REQUIRED. The server for whom this message is intended.

* lock_evidence_claim REQUIRED. The lock or escrow evidence (on the network behind the client).

* lock_claim_format OPTIONAL. The format of the evidence.

* lock_evidence_expiration REQUIRED. The duration of time of lock on the asset (after which the lock is released).

* hash_commence_ack_request REQUIRED. The hash of the previous message.

* client_transfer_number OPTIONAL. This is the transfer identification number chosen by the client. This number is meaningful only to the client.

* client_signature REQUIRED. The digital signature of the client.

8.4. Lock Evidence Response Message (Ack)

The purpose of this message is for the transfer server (recipient gateway) to indicate acceptance of the asset-escrow (or lock) evidence delivered by the client (sender gateway) in the previous message.

The message must be signed by the server.

The parameters of this message consists of the following:

* message_type REQUIRED urn:ietf:sat:msgtype:lock-evidence-ack-msg

* client_identity_pubkey REQUIRED. The client for whom this message is intended.

* server_identity_pubkey REQUIRED. The server who sent this message.

* hash_lockevidence_request REQUIRED. The hash of previous message.

* server_transfer_number OPTIONAL. This is the transfer identification number chosen by the server. This number is meaningful only to the server.

* server_signature REQUIRED. The digital signature of the server.

9. Commitment Establishment Flow (Phase 3)

This section describes the transfer commitment agreement between the sender gateway to a recipient gateway.

This phase must be completed within the asset-lock duration time specificied in the previous lock_evidence_expiration parameter.

In this phase gateways implement the Transfer Commitment endpoint.

In the following, the sender gateway takes the role of the client while the recipient gateway takes the role of the server.

The flow follows a request-response model. The client makes a request (POST) to the Transfer Commitment endpoint at the server.

Gateways MUST support the use of the HTTP GET and POST methods defined in RFC 2616 [RFC2616] for the endpoint.

Clients MAY use the HTTP GET or POST methods to send messages in this phase to the server. If using the HTTP GET method, the request parameters maybe serialized using URI Query String Serialization.

The client and server may be required to sign certain messages in order to provide standalone proof (for non-repudiation) independent of the secure channel between the client and server. This proof maybe required for audit verifications post-event.

(NOTE: Flows occur over TLS. Nonces are not shown).

9.1. Commit Preparation Message

The purpose of this message is for the client to indicate its readiness to begin the commitment of the transfer.

The message must be signed by the client.

The parameters of this message consists of the following:

* message_type REQUIRED. It MUST be the value urn:ietf:sat:msgtype:commit-prepare-msg

* client_identity_pubkey REQUIRED. The client who sent this message.

* server_identity_pubkey REQUIRED. The server for whom this message is intended.

* hash_lockevidence_ack REQUIRED. The hash of previous message.

* client_transfer_number OPTIONAL. This is the transfer identification number chosen by the client. This number is meaningful only the client.

* client_signature REQUIRED. The digital signature of the client.

9.2. Commit Preparation Response

The purpose of this message is for the server to indicate to the client its readiness to proceed with the commitment finalization step.

The message must be signed by the server.

The parameters of this message consists of the following:

* message_type REQUIRED. It MUST be the value urn:ietf:sat:msgtype:commit-prepare-ack-msg

* client_identity_pubkey REQUIRED. The client for whom this message is intended.

* server_identity_pubkey REQUIRED. The server who sent this message.

* hash_commitprep REQUIRED. The hash of previous commit preparation message.

* server_transfer_number OPTIONAL. This is the transfer identification number chosen by the server. This number is meaningful only the server.

* server_signature REQUIRED. The digital signature of the server.

9.3. Commit Final Message

The purpose of this message is for the client to indicate to the server that the client (sender gateway) has completed local extinguishment of the asset on its network or system (L1), and that now on its part the server (recipient gateway) must re-generate the asset on its network or system (L2).

The message must contain standalone claims related to the extinguishment of the asset by the client. The claim must be signed by the client.

The parameters of this message consists of the following:

* message_type REQUIRED. It MUST be the value urn:ietf:sat:msgtype:commit-final-msg

* client_identity_pubkey REQUIRED. The client who sent this message.

* server_identity_pubkey REQUIRED. The server for whom this message is intended.

* commit_final_claim REQUIRED. This is one or more claims signed by the client that the asset in question has been extinguished by the client in its local network or system.

* commit_final_claim_format OPTIONAL. This is the format of the claim provided by the client in this message.

* hash_commitprepare_ack REQUIRED. The hash of previous message.

* client_transfer_number OPTIONAL. This is the transfer identification number chosen by the client. This number is meaningful only the client.

* client_signature REQUIRED. The digital signature of the client.

9.4. Commit Final Response Message

The purpose of this message is for the server to indicate to the client that the server has completed the asset re-generation at its network or system (L2).

The message must contain standalone claims related to the re-generated of the asset by the server. It must be signed by the server.

The parameters of this message consists of the following:

* message_type REQUIRED. It MUST be the value urn:ietf:sat:msgtype:commit-final-ack-msg

* client_identity_pubkey REQUIRED. The client for whom this message is intended..

* server_identity_pubkey REQUIRED. The server who sent this message.

* commit_acknowledgement_claim REQUIRED. This is one or more claims signed by the server that the asset in question has been regenerated by the server in its local network or system.

* commit_acknowledgement_claim_format OPTIONAL. This is the format of the claim provided by the server in this message.

* hash_commitfinal REQUIRED. The hash of previous commit final message.

* server_transfer_number OPTIONAL. This is the transfer identification number chosen by the server. This number is meaningful only the server.

* server_signature REQUIRED. The digital signature of the server.

9.5. Transfer Complete Message

The purpose of this message is for the client to indicate to the server that the asset transer has been completed and that no further messages are to be expected from the client in regards to this transfer instance.

The message logically closes the first message of Phase 2 (Transfer Commence Message). It must be signed by the client.

The parameters of this message consists of the following:

* message_type REQUIRED. It MUST be the value urn:ietf:sat:msgtype:commit-transfer-complete-msg

* client_identity_pubkey REQUIRED. The client who sent this message.

* server_identity_pubkey REQUIRED. The server for whom this message is intended.

- hash_commit_final_ack REQUIRED. The hash of previous message.

- hash_transfer_commence REQUIRED. The hash of the Transfer Commence message at the start of Phase 2.

- client_transfer_number OPTIONAL. This is the transfer identification number chosen by the client. This number is meaningful only the client.

- client_signature REQUIRED. The digital signature of the client.

10. Security Consideration

Gateways are of particular interest to attackers because they are a kind of end-to-end pipeline that enable the transferral of digital assets to external networks or systems. Thus, attacking a gateway may be attractive to attackers instead of the network behind a gateway.

As such, hardware hardening technologies and tamper-resistant crypto-processors (e.g. TPM, Secure Enclaves, SGX) should be considered for implementations of gateways.

11. IANA Consideration

(TBD)

12. References

12.1. 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/info/rfc2119>.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, DOI 10.17487/RFC2234, November 1997, <https://www.rfc-editor.org/info/rfc2234>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

12.2. Informative References

[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>.

[RFC5939] Andreasen, F., "Session Description Protocol (SDP) Capability Negotiation", RFC 5939, DOI 10.17487/RFC5939, September 2010, <https://www.rfc-editor.org/info/rfc5939>.

Authors' Addresses

Martin Hargreaves
Quant Network
Email: martin.hargreaves@quant.network

Thomas Hardjono
MIT
Email: hardjono@mit.edu

Rafael Belchior
Technico Lisboa
Email: rafael.belchior@tecnico.ulisboa.pt