Secure Asset Transfer Protocol (SATP)
Implementation in the Hyperledger Cacti Interoperability Framework

(draft-ietf-satp-core-02) &
(draft-belchior-satp-gateway-recovery-00)

Zakwan Jaroucheh (Edinburgh Napier University)

Mentors: V. Ramakrishna (IBM), Rafael Belchior(Técnico Lisboa), Sandeep Nishad (IBM)

IETF 118: Secure Asset Transfer Working Group
Prague, Czech Republic
November 9, 2023
Reference Implementation of SATP in Hyperledger Cacti

- Hyperledger Mentorship 2023 project 2023
  - Augment Cacti “relay” according to SATP draft specs
    - SATP-standard endpoints and SATP message parsing capabilities
    - Error handling and crash recovery support
  - ETA: end of November 2023

- Relay is a configurable module running gRPC services built on Rust
  - Not built for any specific DLT; compatible with any
  - Fits the specification for an SATP gateway

[Diagram of Hyperledger Cacti]

IETF 118: SATP Working Group
Cacti Relay Architecture

Formally verifiable message based protocol conversation with remote relay

Best of breed SDK for interacting with DLT network

Protocol adapter maps protocol semantics to DLT protocol

Driver for interacting with network, enclosed in a container

Rust-based reference implementation of interoperability protocol standard enclosed in a container


Code: https://github.com/hyperledger/cacti/tree/main/weaver/core/relay

IETF 118: SATP Working Group
SATP Between Cacti-Augmented Networks

IETF 118: SATP Working Group
Augmenting Cacti Components for SATP

- **Cacti Relay**
  - Rust-based and runs gRPC services and clients
  - Added a SATP Service
    - Added the new service `satp_service.rs`
    - Added a new client `satp_client.rs`
    - Added/Changed the relevant library helper files

- **SATP protobuf (service interface)**
  - In `github.com/hyperledger/cacti/weaver/common/protos`, added the `satp.proto`

- **Hyperledger Fabric App**
  - In `github.com/hyperledger/cacti/weaver/samples/fabric`, added `satpsimpleasset` chaincode (`satpsimpleasset.go`), i.e., smart contract for Fabric
  - Augmented interoperation helper SDK for Fabric clients
    - `github.com/hyperledger/cacti/weaver/core/fabric-driver/server/server.ts` to include the following functions: `performLock`, `createAsset`, `extinguish`, and `assignAsset`

- Reference: [https://github.com/hyperledger/cacti/pull/2748](https://github.com/hyperledger/cacti/pull/2748)
Relay Support for SATP Operations

Reference: draft-ietf-satp-core-02
SATP Protobuf (Service Interface for Relay)
SATP Service Sample Function

```rust
async fn transfer_proposalClaims(
    request: Request<TransferProposalClaimsRequest>,
) -> Result<Response<Ack>, Status> {
    let transfer_proposal_claims_request: TransferProposalClaimsRequest = request.into_inner().clone();
    let request_id: String = get_request_id_from_transfer_proposal_claims(transfer_proposal_claims_request.clone());
    let conf: RwLockReadGuard<_, Config> = self.config.lock_read().await;

    let log_entry: LogEntry = LogEntry(-);
    log::debug!("{}", log_entry);

    match process_transfer_proposal_claims_request(
        transfer_proposal_claims_request.clone(),
        conf.clone(),
    ) {
        Ok(ack: Ack) => {
            let reply: Result<Response<Ack>, Status> = Ok(Response::new(ack));
            let log_entry: LogEntry = LogEntry(-);
            log::debug!("{}", log_entry);
            reply
        }
        Err(e: Error) => {
            let error_message: String = "Transfer proposal claims failed.".to_string();
            let reply: Result<Response<Ack>, Status> = create_ack_error_message(request_id.clone(), error_message.clone(), e);
            let log_entry: LogEntry = LogEntry(-);
            log::error!("{}", log_entry);
            reply
        }
    }
}
```
SATP Gateway Calls Fabric Driver

Step 1.1 → Step 2.1A
SATP Gateway Calls Fabric Driver

Stage 2
Step 2.1A

```rust
data_transfer_service.rs
event_publish_service.rs
event_subscribe_service.rs
helpers.rs
logger.rs
mod.rs
network_service.rs
satp_helper.rs
satp_service.rs
types.rs
client_tls.rs
client.rs
dbo.rs
derrors.rs
dmain.rs
relay_proto.rs
satp_client.rs
target
tests
dockerignore
env
.env
.env.template
```

```rust
pub fn spawn_send_perform_lock_request(
    driver_info: Driver,
    perform_lock_request: PerformLockRequest,
) {
    tokio::spawn(async move {
        let request_id: String = perform_lock_request.session_id.to_string();
        println!(
            "Locking the asset of the lock assertion request id: {:?},",
            request_id
        );
// TODO: pass the required info to lock the relevant asset
// Call the driver to lock the asset
        let result: Result<(), Error> = call_perform_lock(driver_info, perform_lock_request).await;
        match result {
            Ok(()) => {
                println!("Perform lock request sent to driver\n"");
            }
            Err(e) => {
                println!("Error sending perform lock request to driver: {:?}\n", e);
// TODO: what to do in this case?
            }
        }
    });
```
SATP Gateway Calls Fabric Driver

Stage 2
Step 2.1A
Fabric Driver Invokes `satpsimpleasset` Chaincode

A sample/reference implementation for networks built on Hyperledger Fabric that use SATP gateways

Function to process asset received from another network

**Step 3.6A**
Support for Other Kinds of Networks

• Relay is DLT-agnostic, so the SATP augmentation will work for any DLT (not just Fabric)

• Hyperledger Cacti supports connectivity to various kinds of DLTs: Hyperledger Fabric, Hyperledger Besu (permissioned Ethereum), Quorum, R3 Corda, Hyperledger Sawtooth, and others; more in the pipeline
  • Offers a connector/driver for each
    • Need to augment these to talk to the relay, just as we did for the Fabric driver in this project
  • Offers a client library/SDK for each
    • Need to add functions just like we did for the Fabric SDK in this project
  • Offers sample apps (smart contracts or DApps) for each
    • Need to add asset transfer endpoint functions (to lock, burn, mint, redeem, assets)
Logging (Crash Recovery)

**Types:** Init, Exec, Done

**Reference:** draft-belchior-satp-gateway-recovery-00
Logging (Crash Recovery)

Storage Implementation: SQLite database
Sample Snapshot of Log Entries

<table>
<thead>
<tr>
<th>log entry</th>
<th>log_id</th>
<th>event_time</th>
<th>event_type</th>
<th>event_id</th>
<th>event_id_str</th>
<th>event_id_str</th>
<th>event_id_str</th>
<th>event_id_str</th>
<th>event_id_str</th>
</tr>
</thead>
<tbody>
<tr>
<td>session_1</td>
<td>128</td>
<td>2023-10-31</td>
<td>DEBUG</td>
<td>00:00</td>
<td>session_1</td>
<td>(message_type=&quot;message_type1&quot;,&quot;asset_id&quot;:&quot;asset_id1&quot;,&quot;session_id&quot;:&quot;session_id1&quot;)</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>session_2</td>
<td>130</td>
<td>2023-10-31</td>
<td>DEBUG</td>
<td>00:00</td>
<td>session_2</td>
<td>(message_type=&quot;message_type2&quot;,&quot;asset_id&quot;:&quot;asset_id2&quot;,&quot;session_id&quot;:null)</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>session_3</td>
<td>133</td>
<td>2023-10-31</td>
<td>DEBUG</td>
<td>00:00</td>
<td>session_3</td>
<td>(message_type=&quot;message_type3&quot;,&quot;asset_id&quot;:&quot;asset_id3&quot;,&quot;session_id&quot;:null)</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>session_4</td>
<td>134</td>
<td>2023-10-31</td>
<td>DEBUG</td>
<td>00:00</td>
<td>session_4</td>
<td>(message_type=&quot;message_type4&quot;,&quot;asset_id&quot;:null,&quot;session_id&quot;:null)</td>
<td>4.0</td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td>session_5</td>
<td>136</td>
<td>2023-10-31</td>
<td>DEBUG</td>
<td>00:00</td>
<td>session_5</td>
<td>(message_type=&quot;message_type5&quot;,&quot;asset_id&quot;:null,&quot;session_id&quot;:null)</td>
<td>5.0</td>
<td>5.0</td>
<td>5.0</td>
</tr>
<tr>
<td>session_6</td>
<td>137</td>
<td>2023-10-31</td>
<td>DEBUG</td>
<td>00:00</td>
<td>session_6</td>
<td>(message_type=&quot;message_type6&quot;,&quot;asset_id&quot;:null,&quot;session_id&quot;:null)</td>
<td>6.0</td>
<td>6.0</td>
<td>6.0</td>
</tr>
<tr>
<td>session_7</td>
<td>138</td>
<td>2023-10-31</td>
<td>DEBUG</td>
<td>00:00</td>
<td>session_7</td>
<td>(message_type=&quot;message_type7&quot;,&quot;asset_id&quot;:null,&quot;session_id&quot;:null)</td>
<td>7.0</td>
<td>7.0</td>
<td>7.0</td>
</tr>
<tr>
<td>session_8</td>
<td>139</td>
<td>2023-10-31</td>
<td>DEBUG</td>
<td>00:00</td>
<td>session_8</td>
<td>(message_type=&quot;message_type8&quot;,&quot;asset_id&quot;:null,&quot;session_id&quot;:null)</td>
<td>8.0</td>
<td>8.0</td>
<td>8.0</td>
</tr>
<tr>
<td>session_9</td>
<td>140</td>
<td>2023-10-31</td>
<td>DEBUG</td>
<td>00:00</td>
<td>session_9</td>
<td>(message_type=&quot;message_type9&quot;,&quot;asset_id&quot;:null,&quot;session_id&quot;:null)</td>
<td>9.0</td>
<td>9.0</td>
<td>9.0</td>
</tr>
<tr>
<td>session_10</td>
<td>141</td>
<td>2023-10-31</td>
<td>DEBUG</td>
<td>00:00</td>
<td>session_10</td>
<td>(message_type=&quot;message_type10&quot;,&quot;asset_id&quot;:null,&quot;session_id&quot;:null)</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
</tr>
</tbody>
</table>

Storage Implementation: SQLite database
Project Status

• SATP Service
  • All endpoints for Stage 1 to Stage 3 have been implemented
  • An endpoint has been added to allow the driver to update the status of an asset for the steps: 2.1B, 3.2B, 3.4B and 3.6B.
  • Placeholders have been added to validate each incoming requests
  • Placeholders have been added to derive the corresponding request object (that needs to be sent to the other gateway) based on the incoming requests
  • Demo how an asset can be transferred from one Fabric network to another Farbic network using SATP protocol implementation

• TODO
  • All endpoints related to Stage 0
  • Fill the above placeholders according to the SATP logic
  • Remove some hardcoded values used for demoing creating an asset, assign it and destroy it
  • Add a looping mechanism that enables the gateway to repeatedly inquire about the asset's status from the driver.
Feedback from Implementer

• It would be beneficial to have the format of these messages documented in a Git repository to serve as the definitive source of truth.
• Interface between gateway and driver (executor) is not clear (*Note: out of scope*)
• Unclear how to get a unique ID for each request
• How do gateways negotiate compatible signature algorithms? (*Note: Stage 0*)
• Ambiguity about asset state inference in the face of failure
  • If a gateway recovers after a crash and wishes to resume SATP but discovers that an asset is locked, how does it know that the asset was locked by the in-progress SATP instance and not by an unrelated process?
  • *Note: do we need a generic interface for networks to expose the states of digital assets to third parties?*
• Managing contention and avoid overhead: read asset state first before attempting to lock (atomic operation)
Thank you and Q&A

Zakwan Jaroucheh: zakwanj@gmail.com
Rafael Belchior: rafael.belchior@tecnico.ulisboa.pt
V. Ramakrishna: vramakr2@in.ibm.com
Sandeep Nishad: sandeep.nishad1@ibm.com

Zakwan’s demo with voiceover: https://we.tl/t-dWH2vFeNt4