**Satslink SDD**

Author: Sama Labs

**1. Overview**

Satslink is a privacy-focused communication application that provides end-to-end encryption for user data and includes all the traditional features of a VPN. The application leverages the full-stack data encryption communication technology of the Sama network, ensuring that all data generated or accessed by users is transmitted in an encrypted manner. This effectively prevents hijacking, monitoring, and analysis by centralized entities. Additionally, the Sama network enhances security and privacy by separating the consensus layer from the business shards.

Currently, Satslink operates as an Android application that provides VPN functionality. It is developed based on the Sama SDK, which separates upper-layer and lower-layer functions and provides a series of interfaces for Satslink to call. These interfaces cover the core functionalities of Satslink, including creating application public and private key pairs, generating ETH accounts, setting VPN node information, starting and stopping VPN ss services, exiting ss services, and obtaining proxy ports.

To further enhance the user experience and simplify the payment process, this upgrade will support MSQ user authorization and ICRC-1 payment functionality. Specifically, users with ETH accounts can use Metamask snap to authorize access to the MSQ platform, where they can obtain an ICP account. Using this ICP account, users can make payments and enjoy the VPN services provided by Satslink.

This upgrade ensures that Satslink continues to offer robust VPN functionality while also expanding payment options by integrating ICP accounts, making the application more adaptable to the needs of various users, especially those already using ETH wallets.

**1.2 Objectives**

The goal of Satslink is to replace centralized VPNs, reducing the risks of data leakage and auditing, and making it an essential tool for every Web3 user.

**Main Objectives**

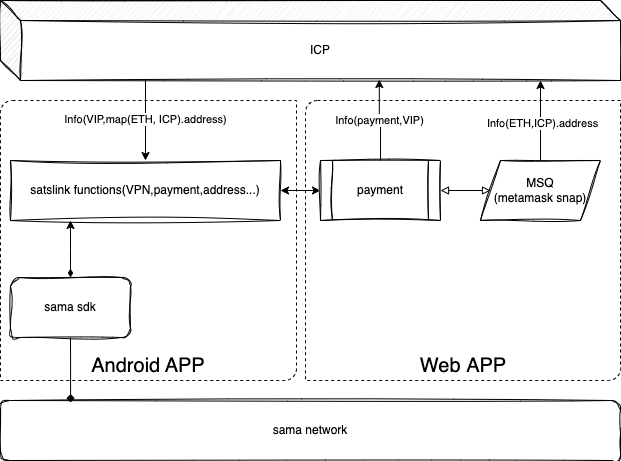
* User Integration: Through MSQ user authorization, ensure that each user receives a unique private identity when interacting with different domains, enhancing the security of user authentication.
* Payment System: Based on MSQ, support direct purchase of ICP and ICRC-1 tokens and payment for VPN services within the IC ecosystem, simplifying the payment process and providing a convenient payment experience.
* Pricing Packages: Design based on the IC Exchange Rate Canister, ensuring that Satslink service prices are dynamically adjusted and fair, offering users the best value.

Achieving these objectives will significantly enhance Satslink’s competitiveness in user authentication, payment convenience, and service pricing, making it a more secure and user-friendly Web3 privacy communication tool.

**2. System Architecture**

**2.1 Architecture Diagram**

This section will present the current system architecture and the upgraded system architecture. The diagrams will clearly illustrate how the current Satslink system achieves VPN functionality through the Sama SDK and how the upgraded system integrates ICP account authentication and payment modules.



The architecture diagram is primarily divided into three parts includes **Sama SDK and Android App、Payment Web App**.Satslink, as a distributed decentralized VPN application, utilizes the SDK and Android suite to implement basic VPN functions, enabling the application to offer the required functionalities as a distributed application.

* **Sama SDK**

Sama SDK provides core interfaces, including starting and stopping the VPN service, setting node information, and ETH account authentication. This SDK encapsulates all underlying functions and offers them to the upper-layer Android application for use.

* **Android App**

Satslink, as a front-end application, interacts with the Sama SDK to manage VPN services, authenticate ETH accounts, and select nodes. Users can choose nodes and operate VPN services within the app.

* **Payment Web App**

The web app, as an extension of the application, is built on MSQ for processing payments and ICP account authentication. This expansion is a critical step for Satslink's continued development and commercialization.

**2.1.1 Current System Architecture**

The current system is based on the Sama SDK, responsible for separating upper-layer and lower-layer functions, implementing ETH account authentication, VPN node management, starting and stopping VPN services, and retrieving proxy ports. The main modules include:

- VPN Core Function Module

- ETH Account Authentication Module

- UI Module

**2.1.2 Upgraded System Architecture**

The upgraded architecture will integrate ICP account creation and authentication modules and payment modules into the existing system. The new ICP account creation module will generate an ICP account from an ETH account through MSQ snap and bind it to the existing VPN service.

The payment module will allow users to pay for VPN services using ICP or ICRC-1 tokens through MSQ. It will also incorporate the MSQ payment function to accept any assets from the ICP ecosystem. Since the VPN application charges based on a gold standard model, it will use the Exchange rate canister to perform real-time detection of current ICRC-1 token prices for real-time pricing and billing.

**2.2 Application Modules**

As part of the Android app, the application provides a user interface that supports VPN node selection, account generation, authentication, and VPN operations such as starting and stopping the service. The web app extends the application by enabling payment and ICP account authentication through MSQ, which is a key step toward the continuous development and commercialization of the application. The system's functional modules are divided as follows:

* **VPN functions Module**

**Function:** Provides the core functionality of the VPN, such as node selection, VPN service activation and deactivation, and proxy port retrieval.

**Responsibility:** Interacts with the underlying VPN service through the Sama SDK, ensuring the proper operation of the VPN service and the encrypted transmission of user data.

* **Account Authentication Module**

**Function:**On the first login, the system generates an ETH account and private key unique to Satslink through the SDK interface. This module handles the authentication of the user's ETH wallet address, ensuring the uniqueness and security of the user identity. Once the user triggers the payment module, they will be redirected to the web interface where MSQ will generate an ICP account from the ETH account and perform authentication. The Metamask snap will then generate an ICP payment account, allowing the user to make payments and correctly bind the ICP account with the ETH account, followed by proper permission management.

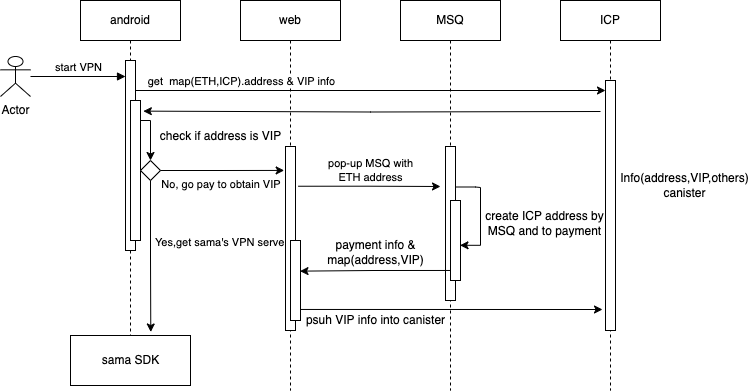
**Responsibility:** Uses MSQ snap for ETH account verification and generates ICP accounts for users as necessary.

* **Payment Module**

**Function:** Primarily implemented on the web end, the module ensures compatibility with Metamask and Metamask snap, facilitating payments with mainstream cryptocurrencies. As a snap-compatible MSQ system, it supports payments using ICP and ICRC-1 tokens, allowing users to pay for VPN services with these tokens.

**Responsibility:** Handles payment requests, interacts with payment interfaces within the IC ecosystem, ensuring secure and convenient payment processes.

**2.3 Data Flow**

****

* **User Account Authentication and Binding**

Users obtain their ETH wallet address and private key through the app. After jumping to the web interface for payment, the web system verifies the ETH account via MSQ snap and retrieves the user's identity information. Once verification is successful, the system checks whether an ICP account exists; if not, it creates one. After successful creation, the web system stores the ICP account information and payment details, binding them with the user's ETH account.

Interaction Process:ETH Account Authentication Module -> MSQ snap -> ICP Account Creation Module -> Data Storage Module -> User Interface (UI) Module

* **Payment Process Data Flow**

Users initiate payment from the Android mobile app, selecting the service package they wish to pay for. The system displays the payment amount (which is locked in based on gold standard detection via the Exchange rate canister) and the available payment methods. The user confirms the payment method and submits the payment request. The payment module processes the payment through MSQ’s receiving interface, and upon confirmation of successful payment, the corresponding VPN service for the bound account is activated.

Interaction Process: Android Payment Channel Module -> Web Payment Module -> MSQ Payment Interface -> ICP Canister Data Storage Module

* **VPN Service Activation and Encrypted Data Flow**

When the user selects a VPN node and activates the VPN service, the system retrieves the VIP information stored in the ICP canister to verify if the user is a VIP. If confirmed as a VIP, the system starts the ss service through the Sama SDK and retrieves the proxy port. User data is transmitted through the VPN tunnel in an encrypted manner, ensuring privacy and security.

Interaction Process: User Interface (UI) Module -> VIP Detection Module -> VPN Module -> Sama SDK -> Data Transmission Module

Through the coordinated work of the above modules, the system can manage the entire process from user authentication and account creation to payment and VPN service, ensuring the stability and user experience of Satslink.

**3. New Features and Interfaces**

* **ETH Account Authentication and ICP Account Creation**

**Feature Description:** When the user jumps from the Android mobile app to the web interface for payment, the ETH account is first authenticated using MSQ snap. Within the extended web system integrated with MSQ, the ETH account will be used to create the corresponding ICP account on the first connection to MSQ.

**Input/Output:** The user's ETH wallet address as input, with the authentication result as output, along with the creation of an ICP account and any ICRC-1 asset accounts supported by MSQ.

**Interface Description:** Describes the relevant interfaces and data formats for interacting with MSQ.

**Pseudocode:**

```javascript

const result = await MsqClient.createAndLogin();

const { msq\_identity } = result.Ok; // Obtain the identity object

const pseudonym: string = await identity.getPseudonym();```

The ETH address can be used as an alias, mapping the ICP account and related VIP payment data. The data is encrypted using the ETH public and private keys before being pushed to the ICP canister for storage.

* **Payment**

**Feature Description:** Supports payment functionality based on the ICP account, including subscription packages, currency selection, and payment processing.

**Input/Output:** The user selects the payment method and amount, with the payment status as output.

**Interface Description:** Lists the interfaces and data formats for interacting with the payment system.

**Pseudocode:**

```javascript

const blockId: bigint | null = await msq.requestICRC1Payment({

tokenCanisterId,

to: {ownersubaccount},

amount,

memo

});

```

This code block outlines the necessary data for the receiving currency, receiving account, and amount. After payment is complete, the blockId is used to verify whether the transaction has been confirmed on the blockchain.

* **Other Relevant Interfaces**

1. **Query Available Nodes**

- \*\*URL\*\*: `/api/v1/nodes`

- \*\*Purpose\*\*: Node Query

- \*\*Method\*\*: POST

- \*\*Response\*\*: List of node information

- \*\*Request Parameters\*\*:

```json

{

"headers": {

"X-Token": "API user X-Token",

"Content-Type": "application/json"

},

"payload": {

"user": "API user ID",

"hostip": "" // Optional parameter

}

}

```

1. **Meal Package List**

- \*\*URL\*\*: `/api/v1/meallist`

- \*\*Purpose\*\*: Query available meal packages

- \*\*Method\*\*: POST

- \*\*Response\*\*: List of available meal packages

- \*\*Request Parameters\*\*:

```json

{

"headers": {

"X-Token": "API user X-Token",

"Content-Type": "application/json"

},

"payload": {

"user": "API user ID",

"meal\_id": "", // Optional parameter, meal package ID (numeric type)

"meal\_name": "", // Optional parameter, meal package name (string type)

"meal\_type": "" // Optional parameter, package type (string type), supports "month" or "quarter", where "month" is calculated as 30 days; this parameter will adjust with business needs

}

}

```

1. **Verify Blockchain Transaction**

- \*\*URL\*\*: `/api/v1/payinfo`

- \*\*Purpose\*\*: Verify if the transaction was successful

- \*\*Method\*\*: POST

- \*\*Response\*\*: Blockchain transaction status, with possible outcomes: `conn-failure` (interface call failure, requires re-submission for confirmation), `success` (verification successful), or `failure` (verification failed)

- \*\*Request Parameters\*\*:

```json

{

"headers": {

"X-Token": "API user X-Token",

"Content-Type": "application/json"

},

"payload": {

"user": "API user ID",

"user\_addr": "", // Required parameter, user's address (string type)

"trans\_hash": "", // Required parameter, transaction hash to be verified (string type)

"meal\_id": "", // Required parameter, meal package ID from the `/meallist` API response

"pay\_address": "", // Required parameter, user's payment address

"currency": "" // Required parameter, currency from the `/currency` API data

}

}

```

1. **Supported Currency Query**

- \*\*URL\*\*: `/api/v1/currency`

- \*\*Purpose\*\*: Query supported transaction currencies

- \*\*Method\*\*: POST

- \*\*Response\*\*: List of supported transaction currencies

- \*\*Request Parameters\*\*:

```json

{

"headers": {

"X-Token": "API user X-Token",

"Content-Type": "application/json"

},

"payload": {

"user": "API user ID"

}

}

```

1. **Query User’s Meal Package Data**

- \*\*URL\*\*: `/api/v1/usermeal`

- \*\*Purpose\*\*: Query the valid time period of the user's meal package

- \*\*Method\*\*: POST

- \*\*Response\*\*: Expiration timestamp

- \*\*Request Parameters\*\*:

```json

{

"headers": {

"X-Token": "API user X-Token",

"Content-Type": "application/json"

},

"payload": {

"user": "API user ID",

"user\_addr": "Paid user's address" // Required parameter

}

}

```

**4. Security**

In Satslink, the security of data transmission, user authentication, and payment processes are critical. We use encryption protocols to ensure that user data is not intercepted or tampered with during transmission. Additionally, we employ multi-factor authentication to enhance the security of user accounts, especially during the payment process, where sensitive information is handled.

The integration of MSQ and ICP for account management and payments brings added layers of security through blockchain technology, which inherently provides transparency, immutability, and decentralization. These characteristics help in safeguarding against unauthorized access and ensuring that user data and transactions are secure.

By combining these security measures with our robust system architecture, Satslink aims to provide a highly secure environment for users to communicate privately and safely within the Web3 ecosystem.

**4.1 Data Privacy Storage**

User payment-related VIP data and information associated with accounts need to be stored privately in the ICP Canister. Android users can only retrieve and decrypt the encrypted data from the Canister after passing the authorization verification. Since the data in ICP Canisters is publicly accessible by default, any user with appropriate permissions can read this data by invoking the Canister's interface. Therefore, we need to carefully design and program to achieve data privacy. We will enhance data privacy through access control and data encryption.

**4.2 Authentication**

To ensure the security of user identity, including ETH account authentication and the secure creation of ICP accounts, we will incorporate permission control mechanisms into the Canister's code. This will ensure that only users who have paid for a VPN package and whose permissions are still valid can access the data. This authentication mechanism will restrict access to specific interfaces to authorized users only, thereby enhancing security.

**4.3 Data Encryption**

We will store encrypted data in the Canister, and only users with the decryption key will be able to read and decrypt this data, ensuring private data access. Data encryption and decryption will use the user's asymmetric public and private keys. Data will be encrypted before being transmitted to the Canister, ensuring its security even while stored in the Canister.