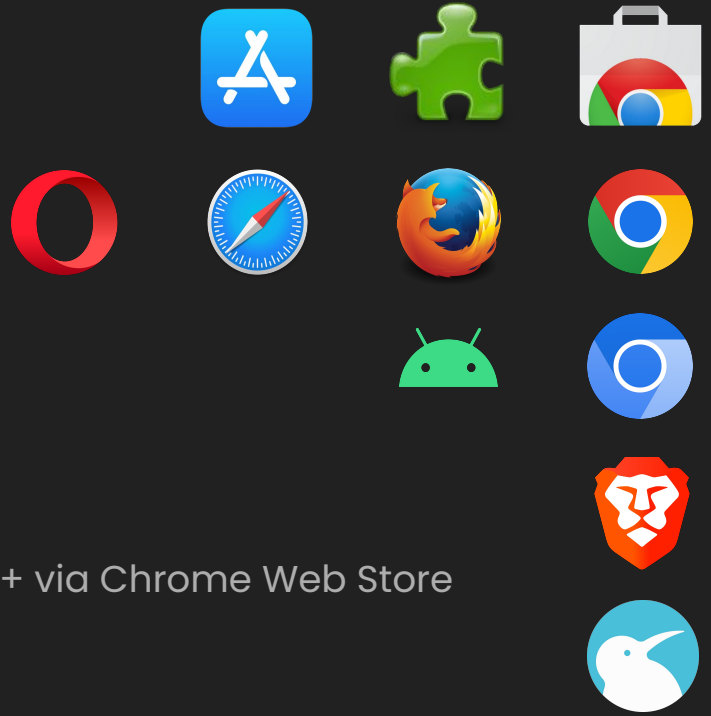
StarShell

Technology Profile - Aug 2022

Created by Blake Regalia

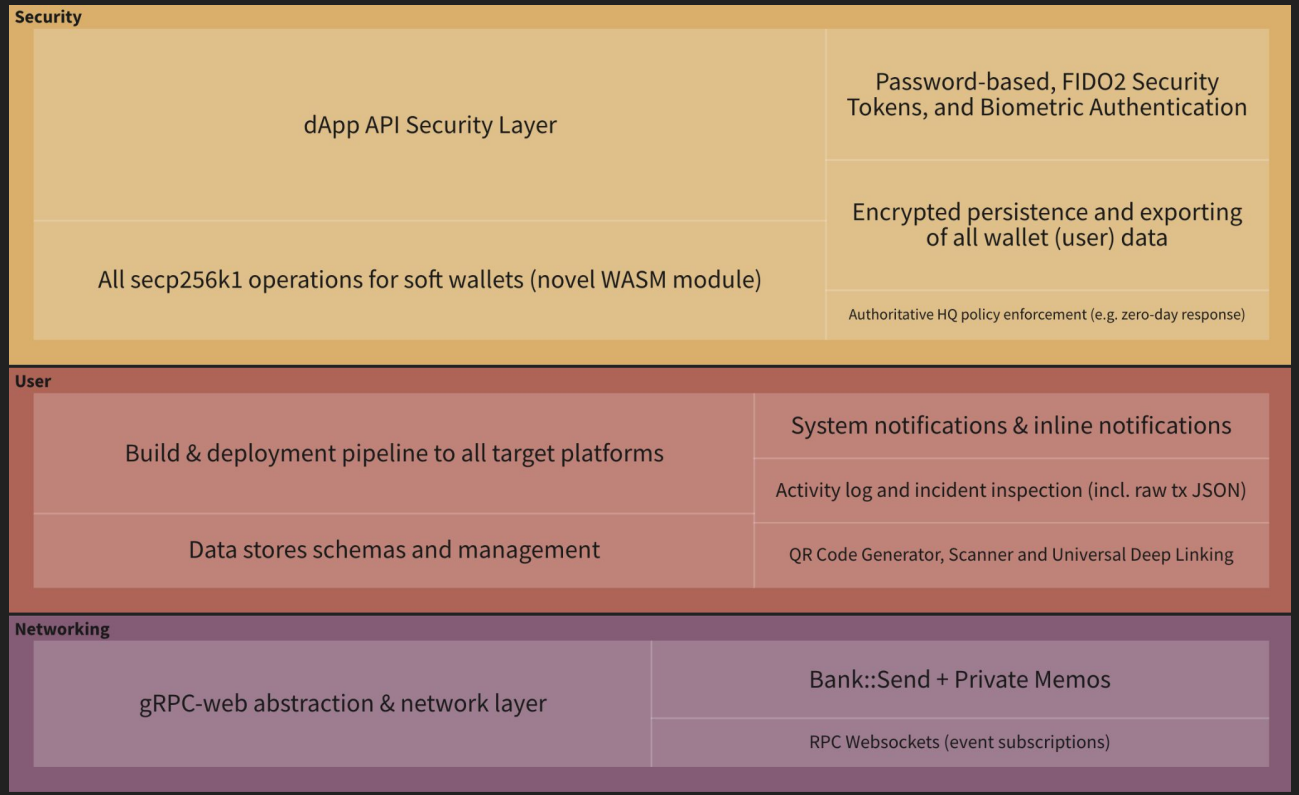
Platform/Vendor Compatibility



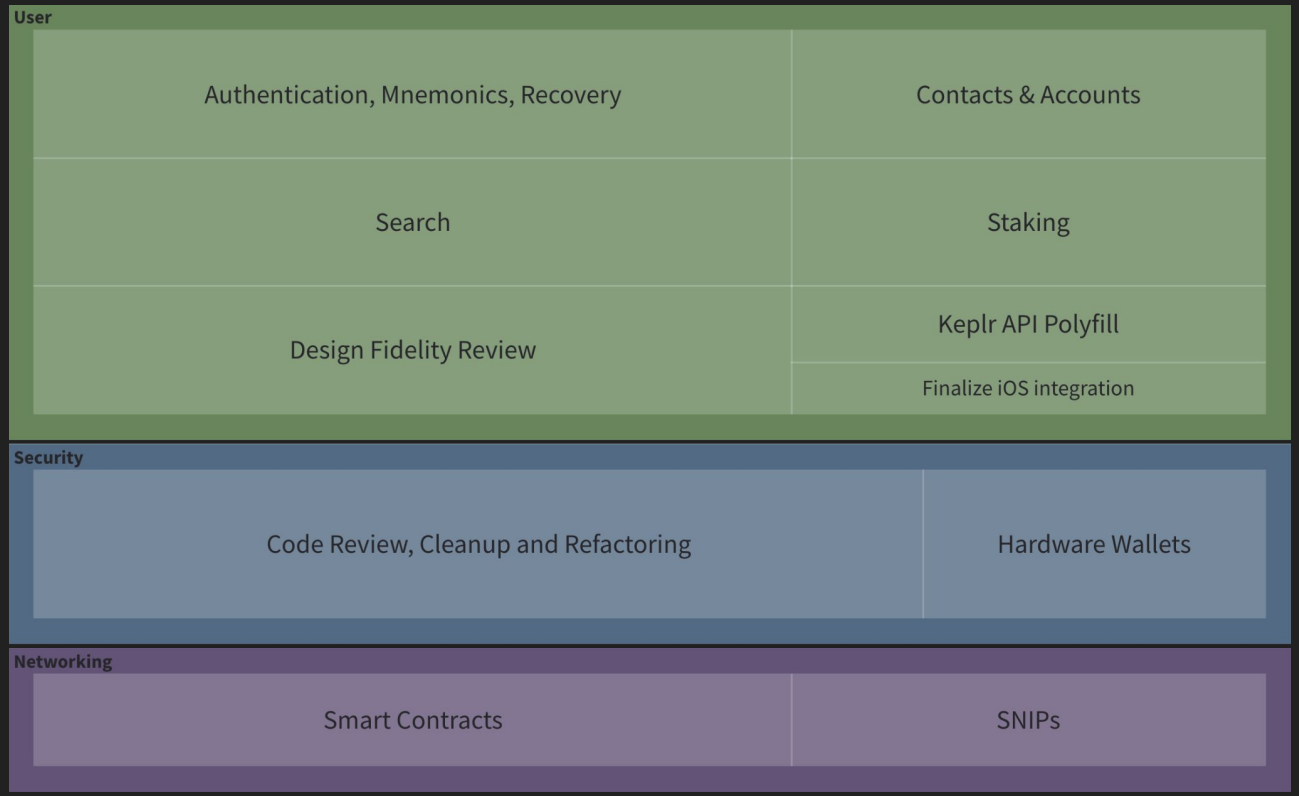
- iOS, iPadOS, macOs
  - Safari app extension via App Store
- Android
  - Firefox for Android & Kiwi Browser
- Desktop
  - Chrome, Firefox, Edge, Opera & all Chromium+ via Chrome Web Store

✅ **Beta has been tested and is working in all the above environments**

Notable Engineering Accomplishments in Q3



Tasks Remaining to Reach Production

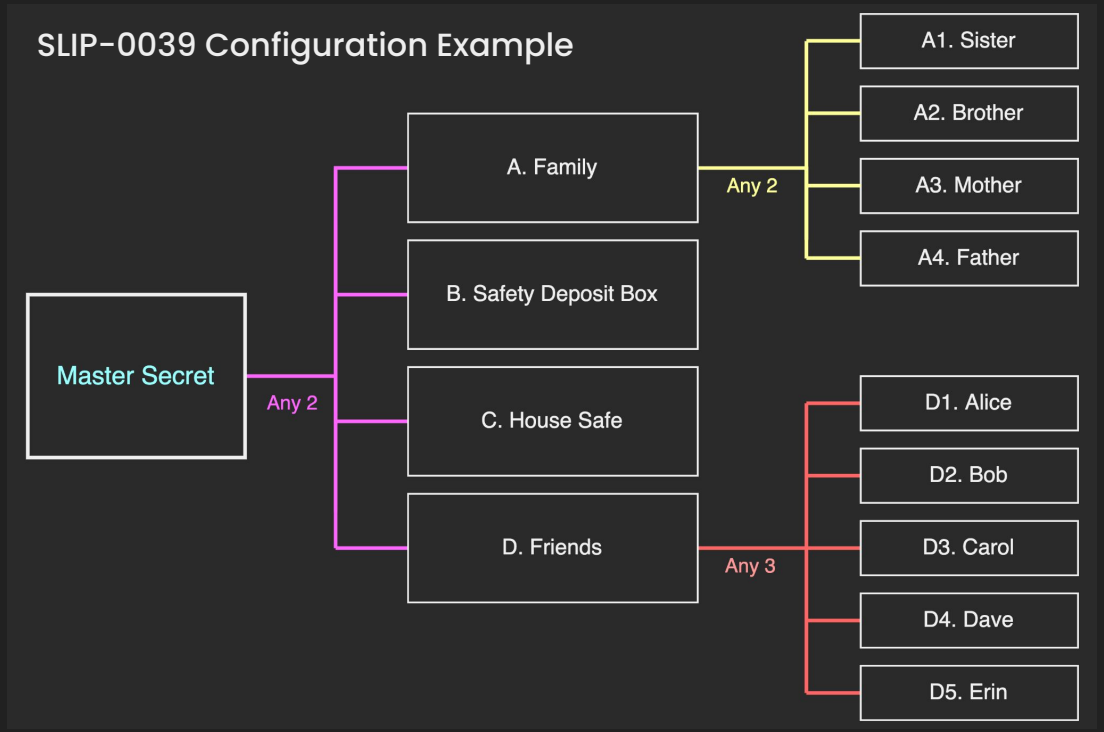


Part 1. Leveraging State of the Art

Reliability and Redundancy

Mnemonic Backup Redundancy

StarShell implements SLIP-0039, allowing users to **divide their seed phrase** mnemonic into a two-level threshold scheme consisting of “groups” and “shares”.



https://github.com/satoshilabs/slips/blob/master/slip-0039.md

Leveraging native cryptography APIs

Cosmos has recently added support for signing and verifying transactions (ECDSA) over the **secp256r1** curve.

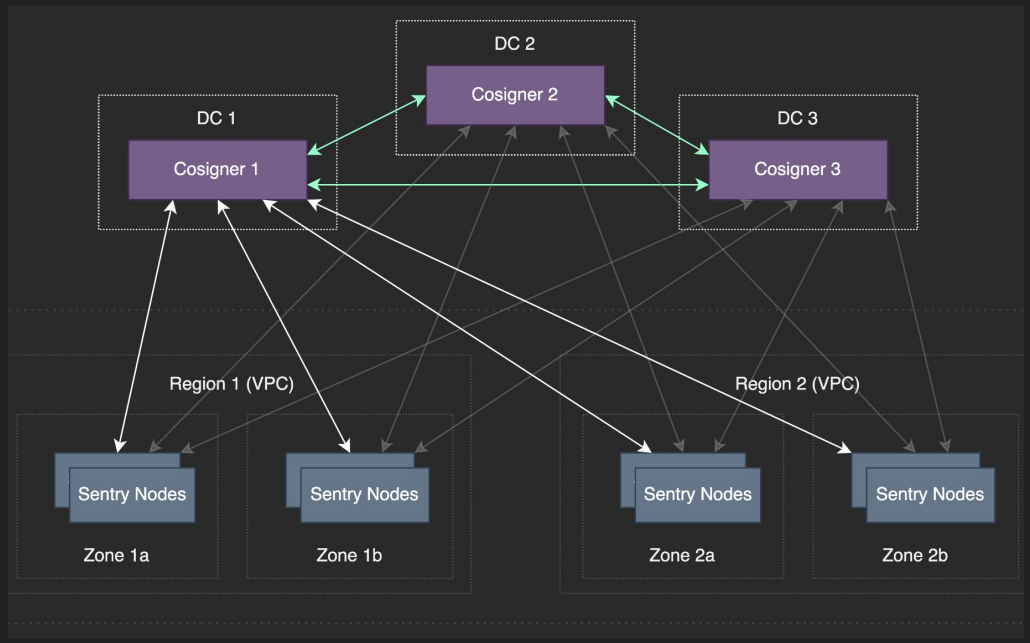
This allows new soft wallet accounts to take advantage of **hardware-level ECDSA** native to their device, for example on their laptop or mobile phone.

This provides even better security for soft wallets, but some users may prefer to stick with secp256r1 out of concern for NSA backdoors in NIST P-256.

Node Service Infrastructure and Validator Redundancy

StarShell is building a **high-availability** node service infrastructure (Iac) in Azure.

In addition to managed service provided features like DDoS mitigation, our infra also performs **load-balancing, self-healing** and **auto-scaling** with intelligent backup/restore and key management, spanning multiple regions.



WebAuthn and gRPC-Web

StarShell will implement and expose a WebAuthn Authenticator interface to dApps so that **users can register new accounts** that are ultimately **derived from their seed key**, effectively turning their soft/hardware wallet into a web authenticator.

gRPC-Web ensures that calls to the nodes are accurately generated directly from the protobuf definitions defined in Cosmos-SDK or in a chain’s extensions.

Deep Conditional Typing

StarShell's approach to implementing the client software starts with **metaprogramming** conditional type definitions in TypeScript that generate deeply typed inferences used for comprehensive type-checking at compile-time.

This practice streamlines development and testing for existing and new contributors alike. It also provides rich and reusable type definitions for downstream apps.

```
/**
 * === _**@starshell/meta**_ ===
 *
 * ```ts
 * Vocab.New<
 *    source: {
 *       [key: string]: {
 *          value?: JsonValue;
 *          response?: JsonValue;
 *       },
 *    },
 *    config?: {
 *       each?: JsonObject,
 *    },
 * > ==> Vocab
 * ```
 * Creates a new Vocab
 */
export type New<
   h_source extends Source,
   gc_vocab extends Config={},
> = Compute<{
   // use mapped type to transform each entry; destructure each entry's value into `g_source`
   [si_key in keyof h_source]: h_source[si_key] extends infer g_source
      ? g_source extends h_source[si_key]
         ? Merge<
            {
               // create the message struct; starting with the message `type` key
               message: MergeAll<{
                  type: si_key;
               }, [
                  // merge the `value` type if one was provided
                  g_source extends {value:JsonValue}
                     ? {value:g_source['value']}
                     : {},

                  // merge the 'each' type provided by config
                  gc_vocab['each'] extends JsonObject
                     ? gc_vocab['each']
                     : {},
               ]>;
            },
            // append the response struct if one was provided
            g_source extends {response:JsonValue}
               ? {response:g_source['response'];}
               : {}
            >
         : {}
      : {};
}>;
```

Part 2. Innovations

Privacy, Security, and IBC

Privacy while Browsing the Web

StarShell has created the **Covert Discovery** model to help protect user privacy.

With StarShell, Web dApps must first **request to see the wallet**, rather than automatically having access to pre-injected globals such as **`window.keplr`**

The problem with pre-injected globals is that they **are present on every page,** (not just dApps!) exposing users to being profiled and enhancing fingerprinting.

https://medium.com/@starshellwallet/web3-wallets-have-serious-privacy-and-security-flaws-5023f8f872b1

Privacy from Snooping

StarShell allows users to **protect their on-chain address** from prying websites by using derived “shadow accounts” when interacting with dApps.

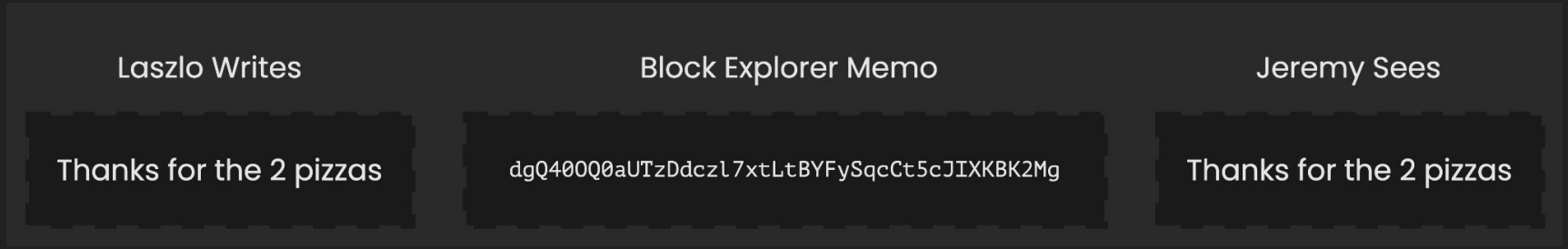
When enabled, **web apps see a false public key** while StarShell automatically transforms the outgoing and incoming messages to the correct public key.

Shadow accounts are recoverable, ensuring that should a user later discover the website recorded public keys for airdrop rewards, bridge transfer, or some other external action, user can still access those funds.

Privacy in Micropayments

Using existing standards and technologies within Cosmos-SDK, StarShell enables **end-to-end encrypted memos**, allowing everyday users to add context to their micropayments when sending to peers.

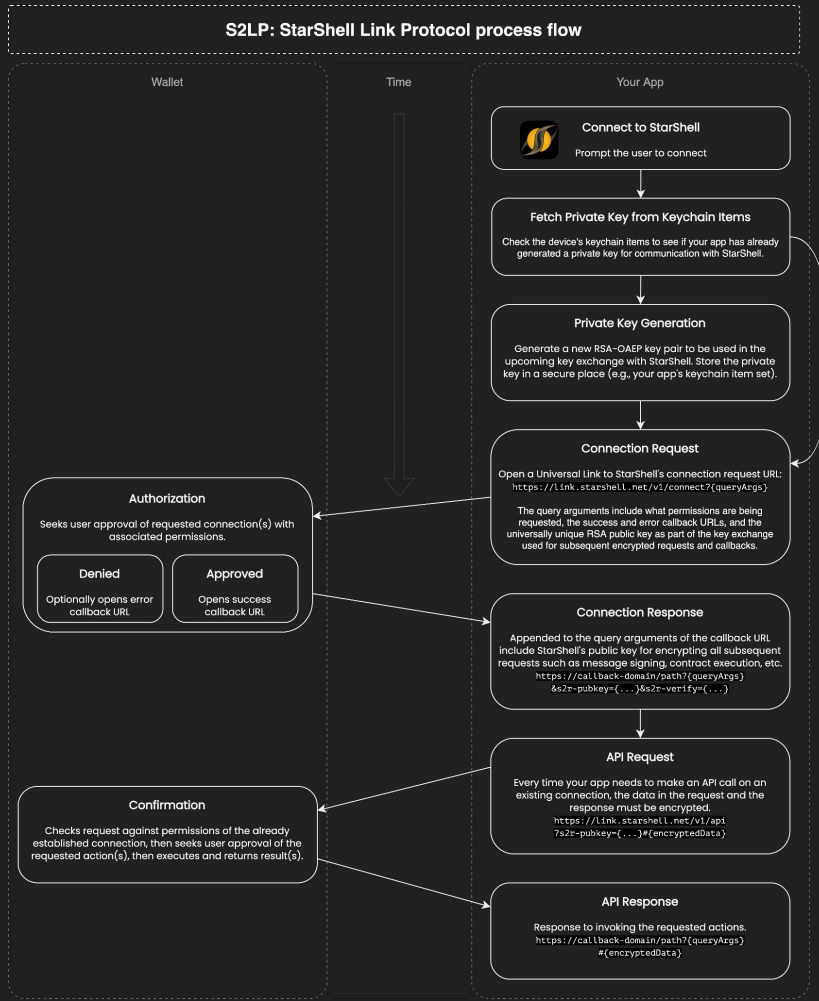
With our implementation, this feature is compatible with **all Cosmos chains**.



*example truncated for brevity; actual memos padded to constant length

Privacy and Security on Mobile

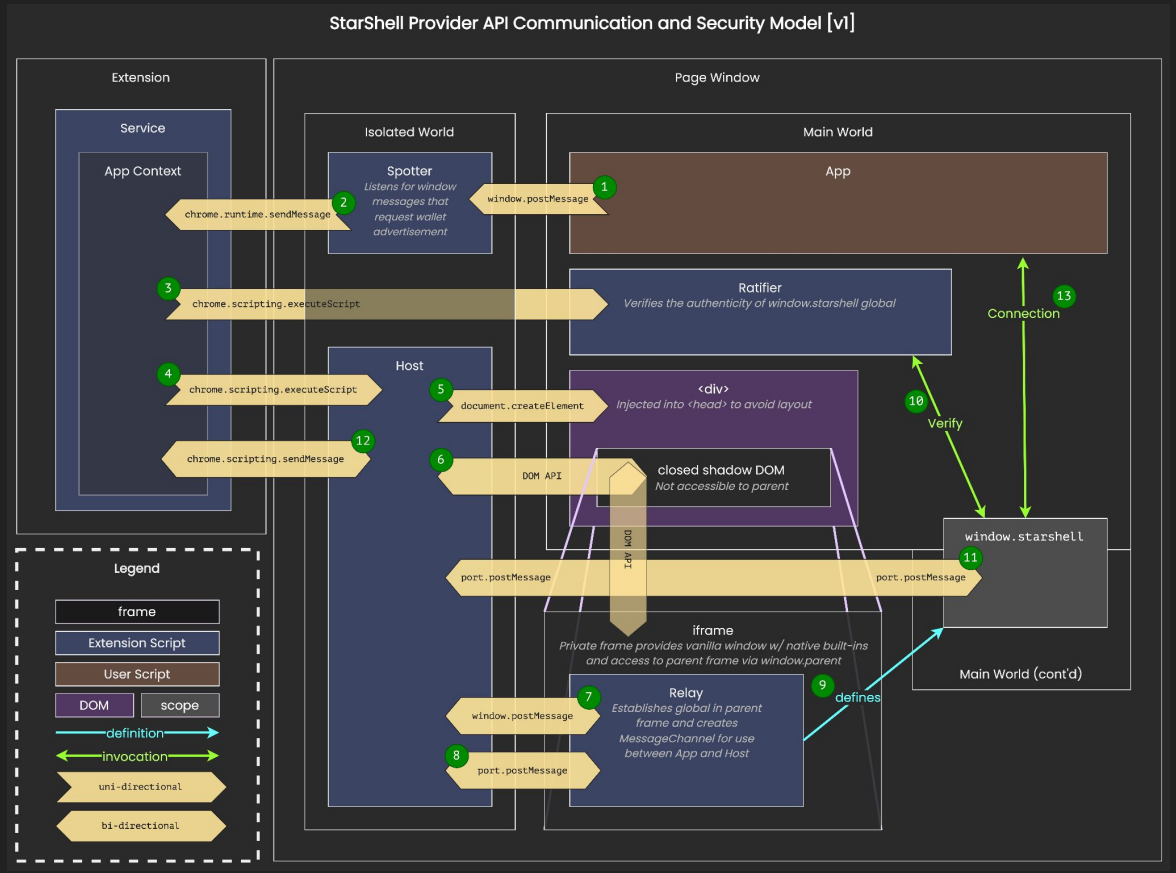
We drafted the StarShell Link Protocol (S2LP), which is designed to reduce data leakage for inter-app communication on mobile devices, and allows for the same multi-chain, multi-account connection features offered thru the browser extension APIs.



https://github.com/SolarRepublic/prerelease-docs/blob/main/StarShell-Wallet-API-Primer.md#starshell-link-protocol-s2lp

Security from Malware and XSS

StarShell has engineered the first page ⇔ extension connection protocol **designed to resist MITM attacks** from cross-site scripts, malicious co-installed extensions and system malware.



Are these types of attacks even realistic? **YES**

https://www.cisecurity.org/advisory/multiple-vulnerabilities-in-google-chrome-could-allow-for-arbitrary-code-execution_2022-073

Security from Theft: Hardening Soft Wallet Security

All soft wallets that run in the browser perform elliptic cryptography in JavaScript.

Problems with doing secp256k1 in JavaScript:

- it is **completely vulnerable to side-channel attacks** during key generation, signing, etc.
- none of the current JS libraries zero-out **sensitive data** (infeasible w/ string and bigint)
- JS impls tend to be very opinionated, leading to possible divergent implementations
- the most popular and widely-used JS libraries are susceptible to **supply-chain attacks** w/ over hundreds of dependencies

StarShell is bringing the **libsecp256k1** C library from bitcoin-core to WebAssembly:

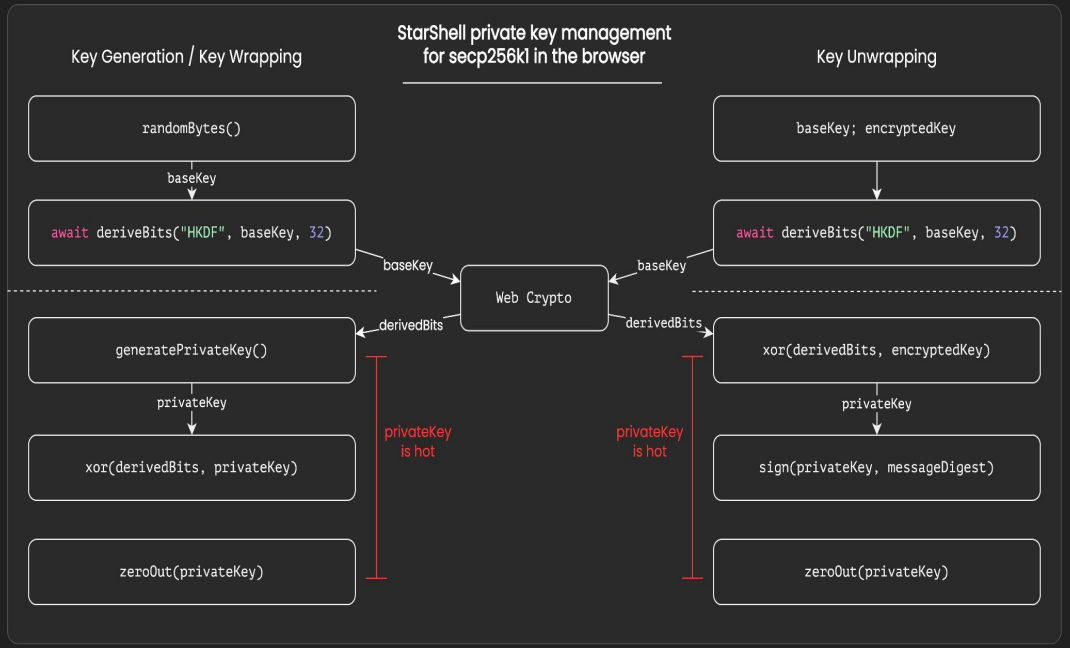
- constant-time and constant-memory **greatly reduces exposure to side-channel attacks**
- all key material is **immediately & synchronously zeroed out** after use
- upstream is very reliable, highly optimized, and thoroughly audited

Security from Theft: Improving Key Management

StarShell has implemented a mechanism allowing it to leverage **platform-specific key management** from the browser (which in turn uses system-available **hardware security** such as keychain enclaves) to encrypt and persist secp256k1 private keys.

Keys are subsequently restored for use, using **one-time pad**, and only exist in memory for **very short periods of time** (on the order of milliseconds).

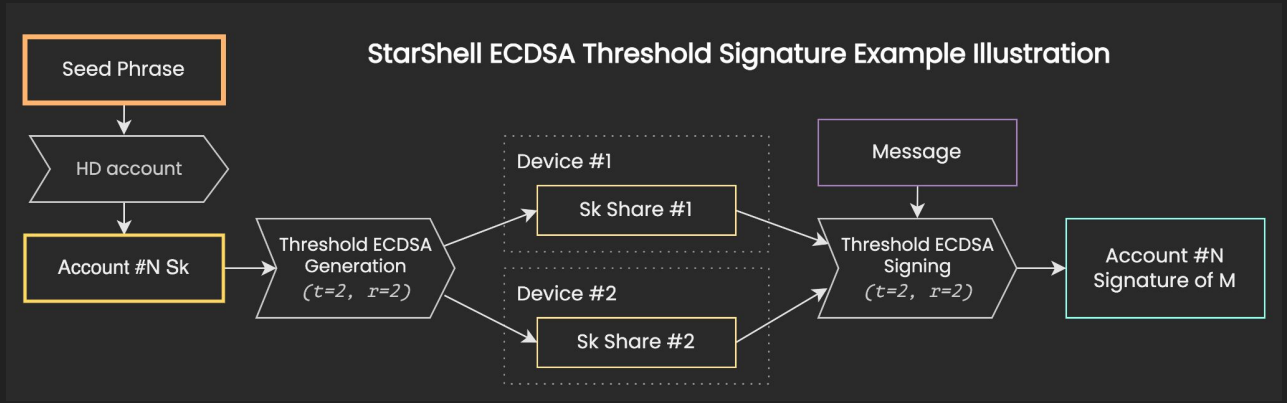
This further protects key material from **cold-boot and key-finding attacks** on browser RAM both while the data is at-rest and in-use.



Security from Theft: Even on Compromised Device

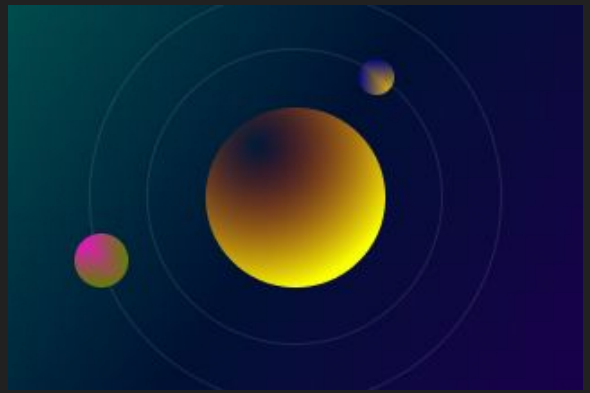
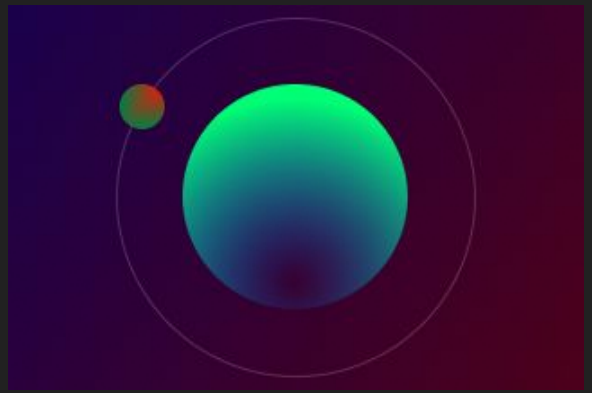
StarShell is currently researching a multi-party threshold ECDSA scheme that would **allow users to require signatures from multiple devices**.

This system would provide a more secure alternative to people who do not have access to hardware wallets by allowing them to effectively **split their master key** between multiple devices, such as their laptop and their phone.



Security from Spoofing and Proof of Authenticity

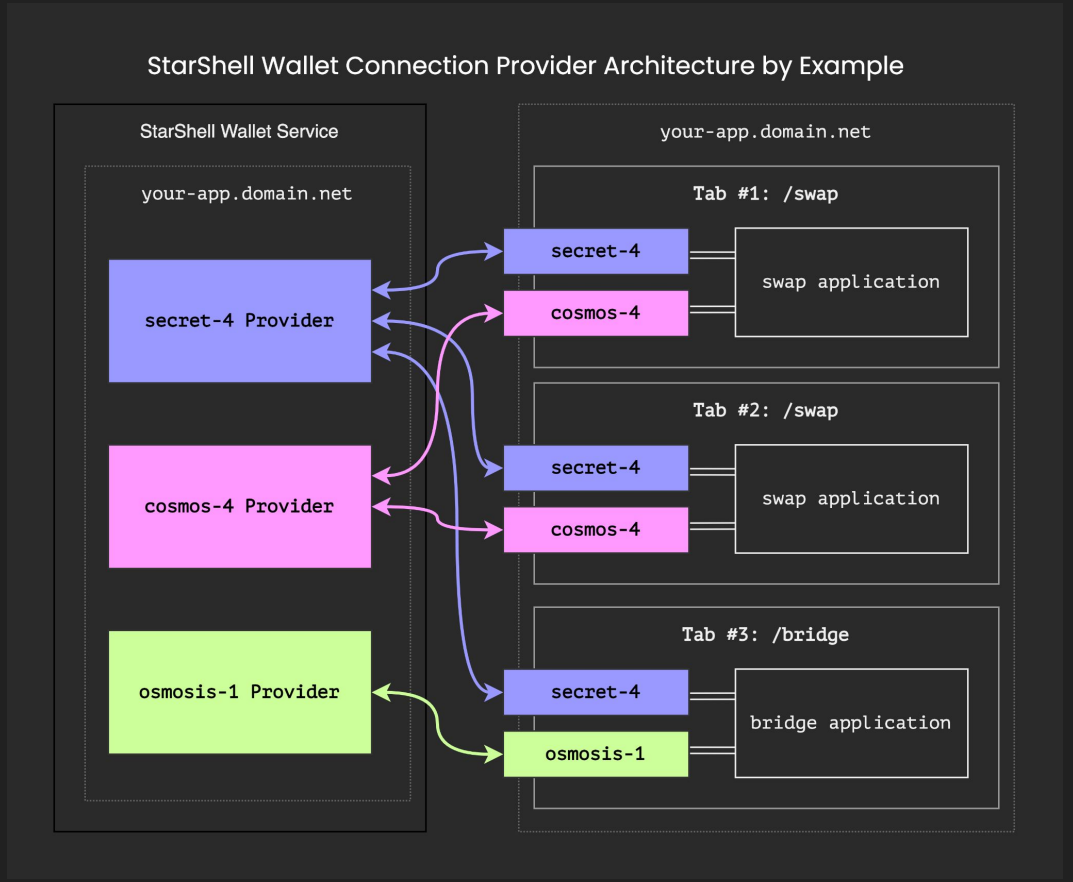
The wallet derives profile pictures from a **deterministic, multi-party signature** between user's account and StarShell's web services, ensuring that no other extensions, websites, etc., can spoof the wallet or trick users. Also provides guarantee of sync-ness across devices and across account restores.


Multi-Channel Networking

Allows dApps to establish **multiple, simultaneous** connections to **different chain networks**, greatly improving *developer experience* for writing IBC and cross-chain applications.

Also seamlessly handles cases where same application is connecting from multiple tabs.



Part 3. Vertical Integration

Validator, Node Services, and HW

StarShell Validator

We will run a Secret Network validator in a **cosigning cluster configuration**

- High-availability thru (3, 5) threshold signature means 2 nodes can be offline
- Improved opsec: no individual cosigner holds the master validator key
- Cosigners divided among regions, colocated instances run different OS
- Considering purchasing bare-metal machines with HSM for signing
- https://github.com/strangelove-ventures/horcrux

Node Services

StarShell will support clients with its **own infrastructure** designed to be elastic.

Additionally, StarShell will pursue a B2B model by continuing cloud development in order to provide **fully managed node services** for applications that want dedicated resources for their apps, including private testnets.

This plan would be a phased approach, starting with hosted VPS resources and working our way towards reducing costs by purchasing bare metal hardware to install in leased rack space at colocations (data centers).

Wallet Features

Additional new provider API features:

- Encrypted key/value stores API (v1)
- Alert configuration management (v1)
- Query plugins (v2)

Complementary services:

- Beacon (v1)
- Utility bots (v1.5)
- Graffiti (v2)

starshell.net