WebAuthn on SUI

zkLogin PoC

zktx.io

OpenID

Summary

- OpenID is an internet user authentication standard developed in 2005, allowing the use of a single digital ID across multiple websites.

Key Features

- Single Sign-On (SSO): Users can log in to multiple sites with one ID.
- Simplified Login Process: Reduces the need to remember multiple passwords.

WebAuthn

Summary

- WebAuthn is a standard for passwordless authentication developed by the FIDO Alliance and W3C.

Key Features

- Passwordless Login: Uses asymmetric keys stored on user devices for authentication.
- Security: Private keys are securely stored on user devices.
- Hardware Security Authenticators: Utilizes built-in secure device biomatrics or external devices.

zkLogin

Summary

- zkLogin allows users to access blockchain without managing private keys, leveraging OpenID for initial user authentication.

Key Benefits

- Ease of Use: No need for passwords or mnemonics.
- Speed: Fast authentication process.
- Freedom from Mnemonic: Users do not need to remember or record mnemonics.

Relationship between WebAuthn and OpenID

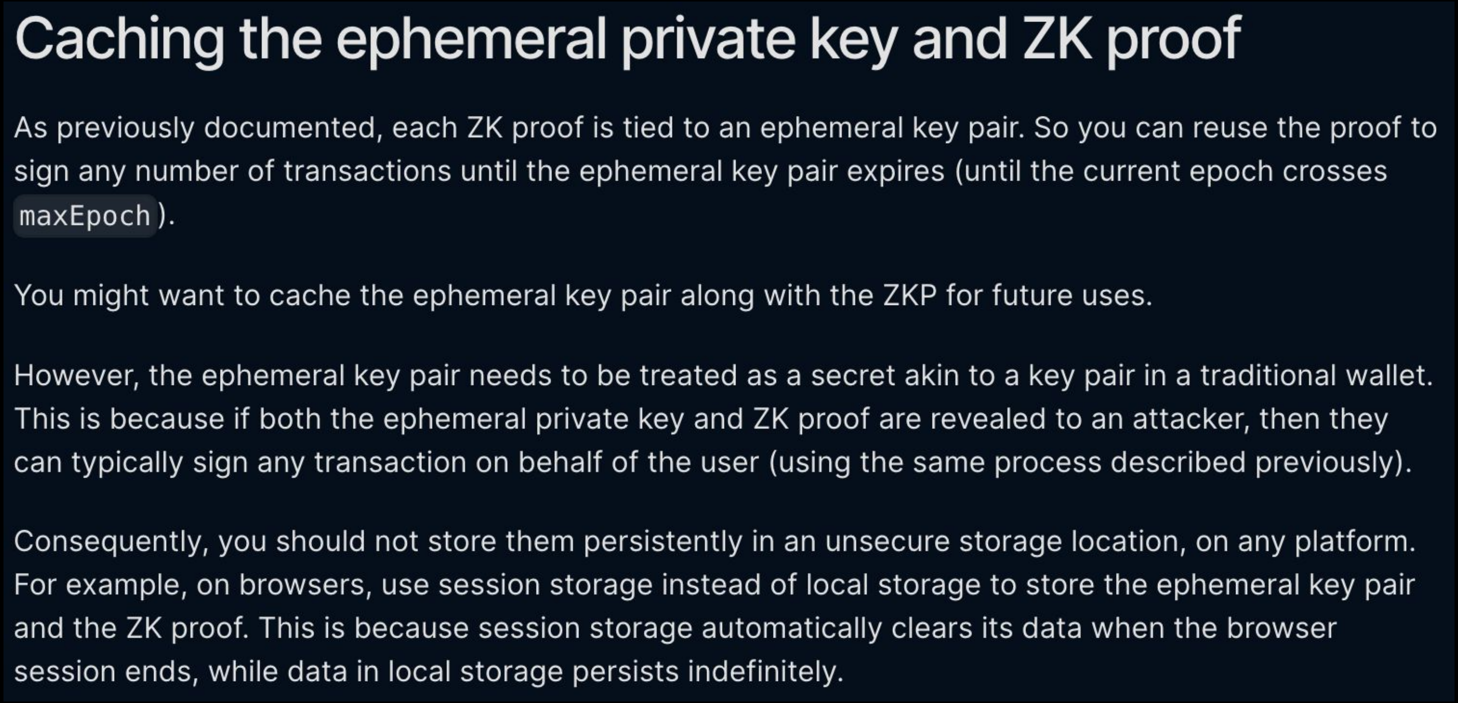
Common Goals:

- Simplify user authentication and enhance security.

Complementary Relationship

- OpenID: Provides single sign-on (SSO) for multiple sites using one ID.
- WebAuthn: Offers passwordless, secure authentication.
- Integration Potential:
  - WebAuthn adds security to OpenID.
  - zkLogin can also benefit from WebAuthn for secure key management, creating a robust and user-friendly process.

zkLogin Integration Guide



Caching the ephemeral private key and ZK proof

As previously documented, each ZK proof is tied to an ephemeral key pair. So you can reuse the proof to sign any number of transactions until the ephemeral key pair expires (until the current epoch crosses `maxEpoch`).

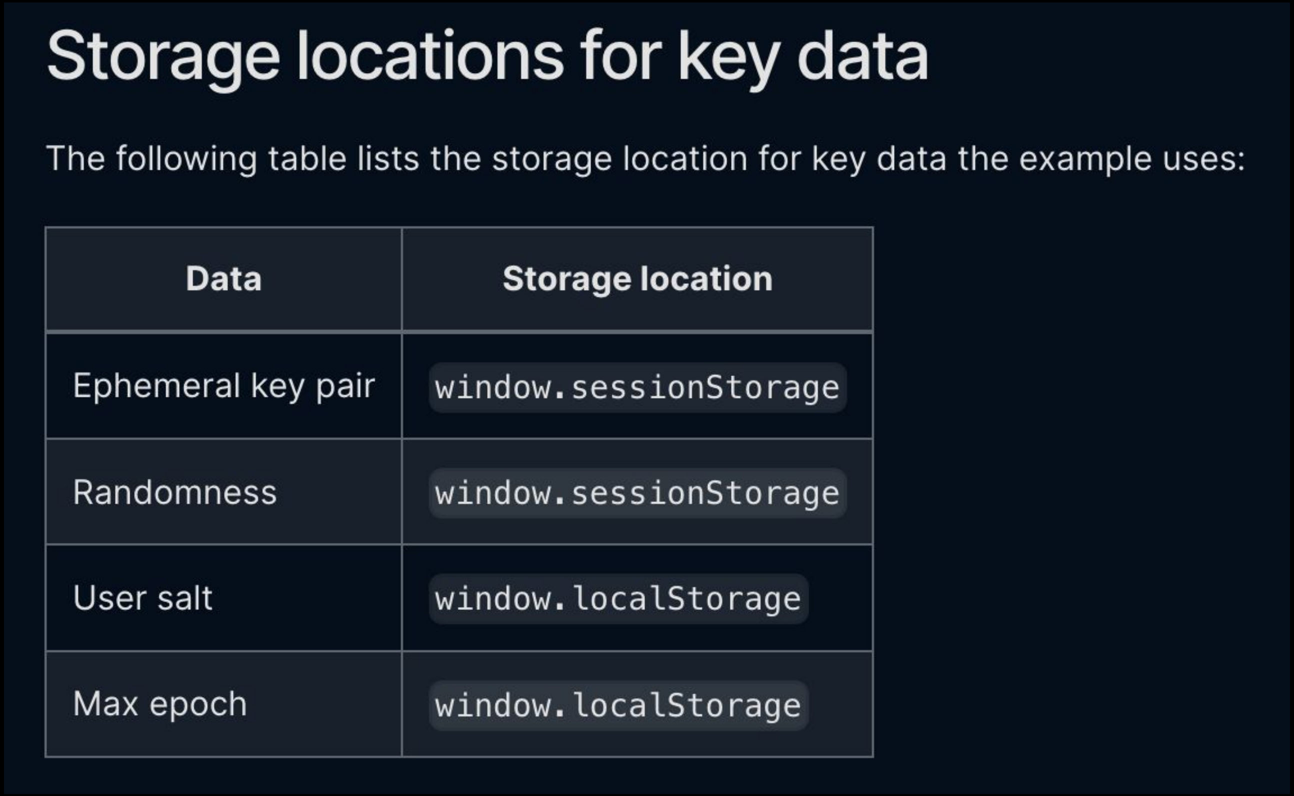
You might want to cache the ephemeral key pair along with the ZKP for future uses.

However, the ephemeral key pair needs to be treated as a secret akin to a key pair in a traditional wallet. This is because if both the ephemeral private key and ZK proof are revealed to an attacker, then they can typically sign any transaction on behalf of the user (using the same process described previously).

Consequently, you should not store them persistently in an unsecure storage location, on any platform. For example, on browsers, use session storage instead of local storage to store the ephemeral key pair and the ZK proof. This is because session storage automatically clears its data when the browser session ends, while data in local storage persists indefinitely.

https://docs.sui.io/guides/developer/cryptography/zklogin-integration#caching-the-ephemeral-private-key-and-zk-proof

zkLogin Integration Guide

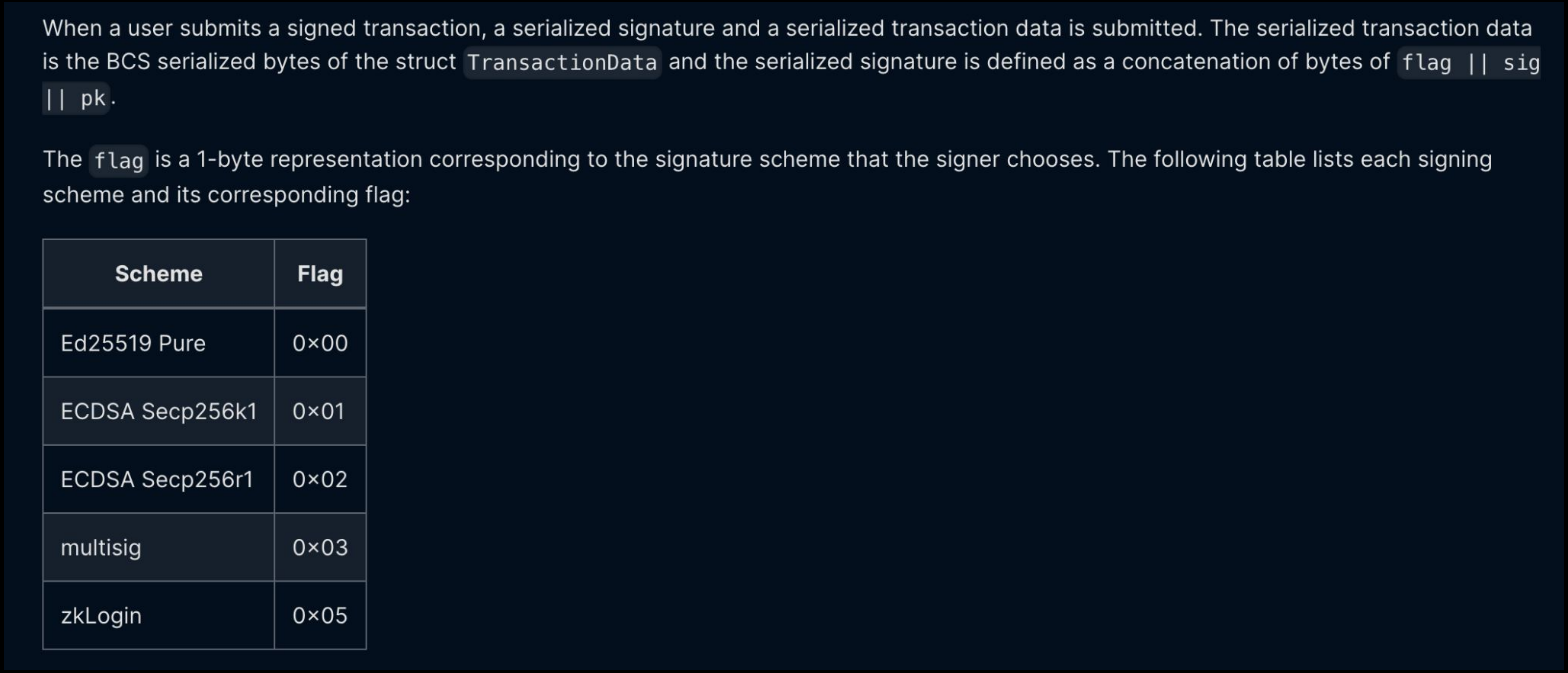


Storage locations for key data

The following table lists the storage location for key data the example uses:

Data	Storage location
Ephemeral key pair	`window.sessionStorage`
Randomness	`window.sessionStorage`
User salt	`window.localStorage`
Max epoch	`window.localStorage`

Sui Signature Scheme

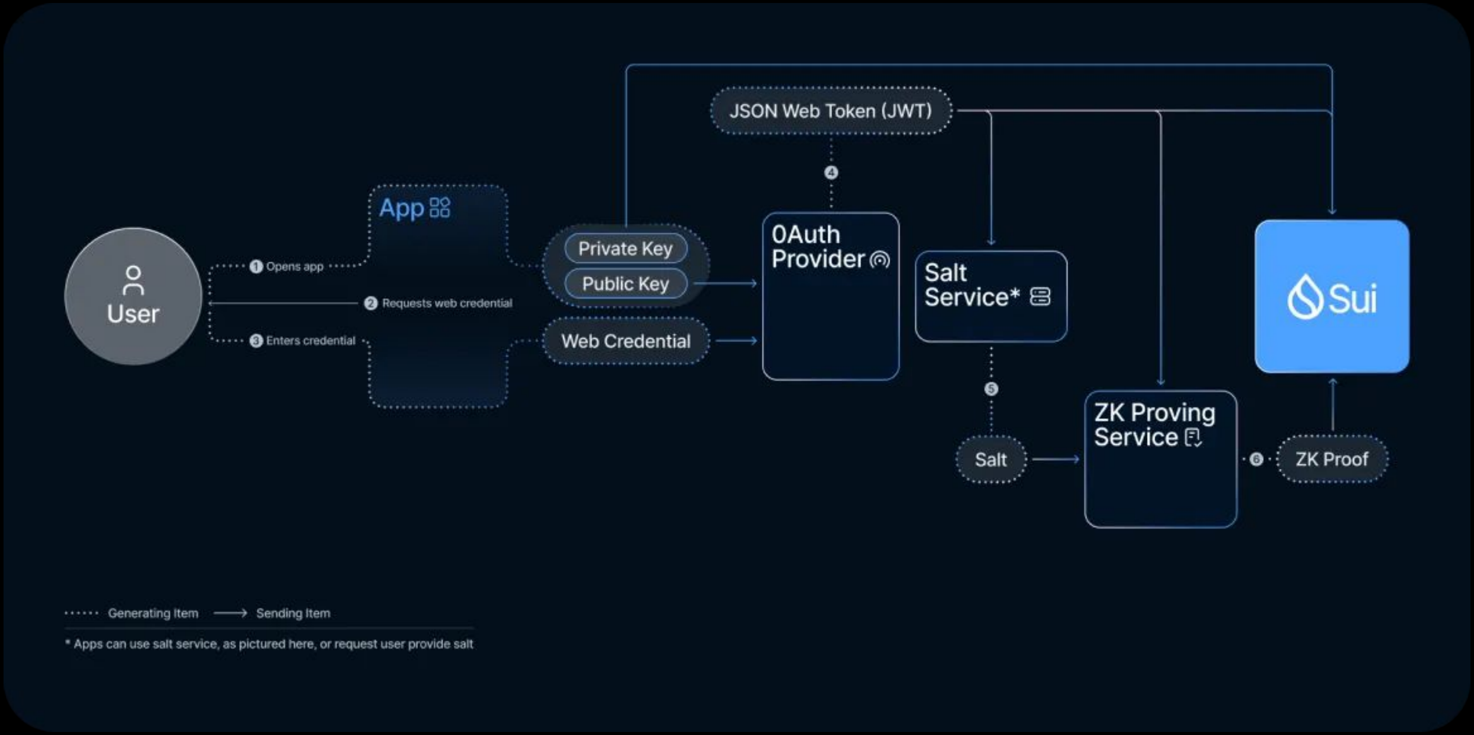


When a user submits a signed transaction, a serialized signature and a serialized transaction data is submitted. The serialized transaction data is the BCS serialized bytes of the struct `TransactionData` and the serialized signature is defined as a concatenation of bytes of `flag || sig || pk`.

The `flag` is a 1-byte representation corresponding to the signature scheme that the signer chooses. The following table lists each signing scheme and its corresponding flag:

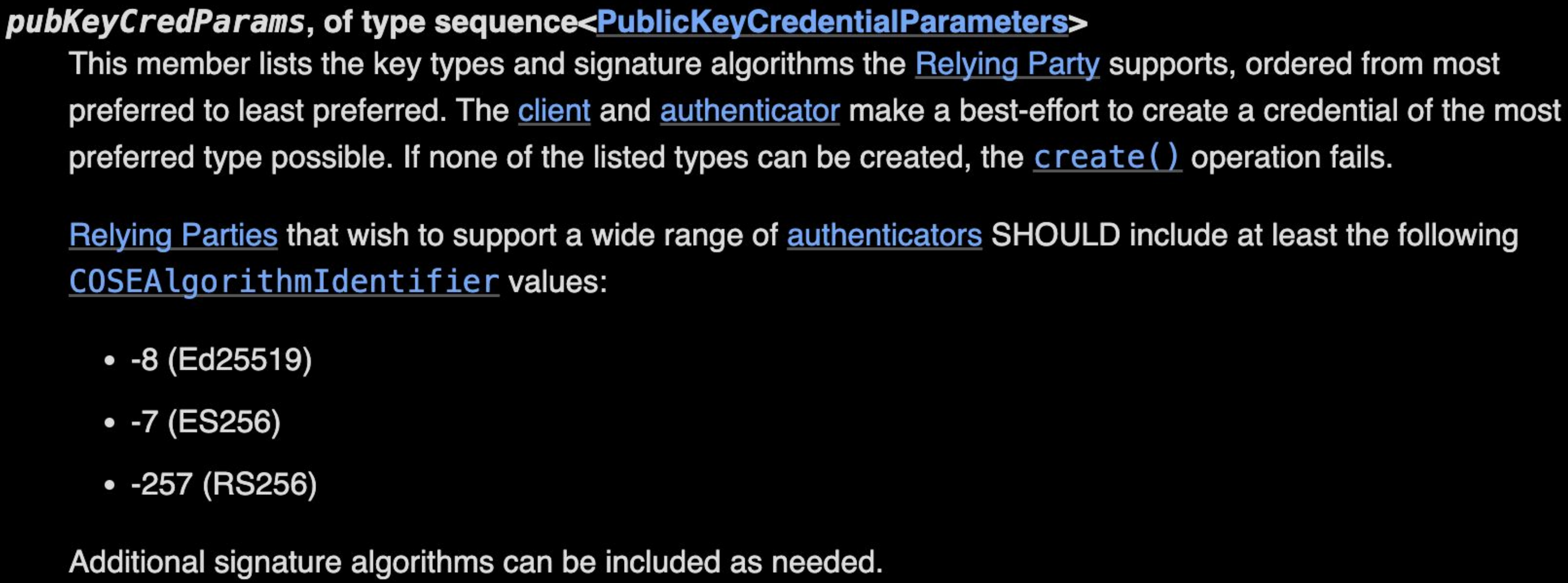
Scheme	Flag
Ed25519 Pure	0×00
ECDSA Secp256k1	0×01
ECDSA Secp256r1	0×02
multisig	0×03
zkLogin	0×05

zkLogin Flow



* Apps can use salt service, as pictured here, or request user provide salt

WebAuthn



***`pubKeyCredParams`*, of type sequence<PublicKeyCredentialParameters>**

This member lists the key types and signature algorithms the Relying Party supports, ordered from most preferred to least preferred. The client and authenticator make a best-effort to create a credential of the most preferred type possible. If none of the listed types can be created, the `create()` operation fails.

Relying Parties that wish to support a wide range of authenticators SHOULD include at least the following `COSEAlgorithmIdentifier` values:

- -8 (Ed25519)
- -7 (ES256)
- -257 (RS256)

Additional signature algorithms can be included as needed.

Problem & Solution

Problems with Existing Methods

- Security risks associated with key caching.
- Session storage is not secure and provides an uncomfortable experience.

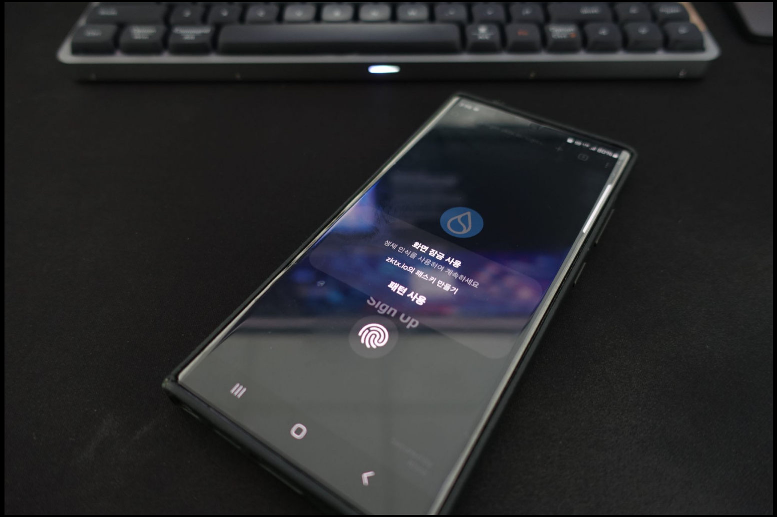
Solution

- The most reliable method is to manage ephemeral keys within hardware security zones.

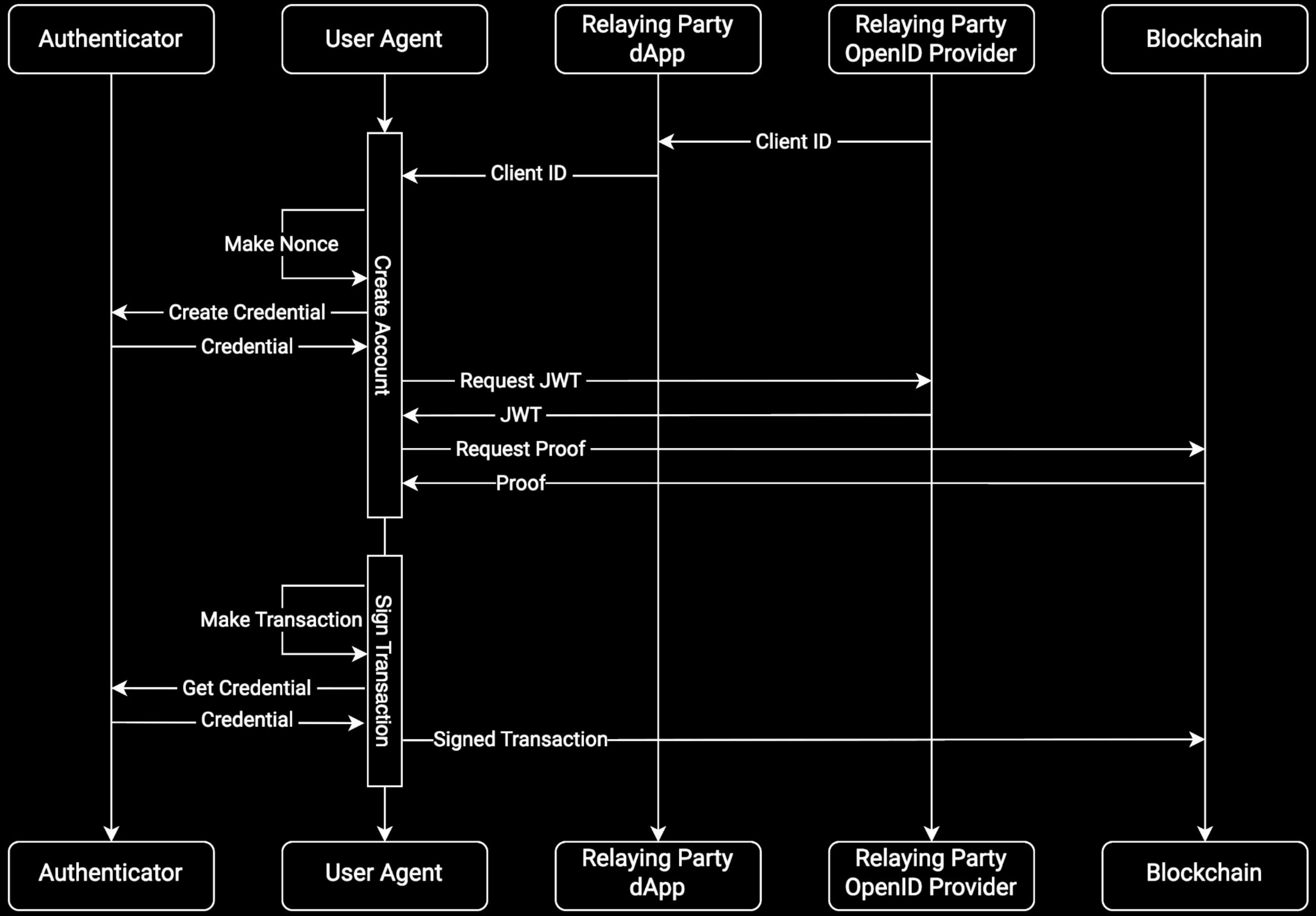
Integration

Proof of Concept

- Cryptographic agility is core to Sui.
- The primary cryptography used in WebAuthn is secp256r1, which Sui also supports.
- Defining new signature scheme to use WebAuthn is beneficial, but integrating WebAuthn with zkLogin will significantly enhance the user experience and have a substantial impact.



Flow


Authenticator
User Agent
Relaying Party dApp
Relaying Party OpenID Provider
Blockchain
Client ID
Client ID
Make Nonce
Create Account
Create Credential
Credential
Request JWT
JWT
Request Proof
Proof
Make Transaction
Sign Transaction
Get Credential
Credential
Signed Transaction
Authenticator
User Agent
Relaying Party dApp
Relaying Party OpenID Provider
Blockchain

Expected Benefits

Enhanced Security

- Prevent unauthorized access: Secure key management through hardware security zone.
- Ensure confidentiality of ephemeral keys: Keys are securely stored and managed.

Improved User Experience

- Increased convenience: Easier access for users without the need for additional UX to security.
- Longer Usage: By extending maxEpoch, WebAuthn can be used for a longer period.
- Superior Experience on Mobile: Provides the best user experience, especially on mobile devices.

Conclusion

Importance of Integration

- Emphasize how integrating zkLogin with WebAuthn enhances user experience and security.

Summary of Expected Benefits

- Improvement in both security and convenience.
- Contribution to the mainstream adoption of blockchain technology.

Links

https://docs.zktx.io

https://zklogin.zktx.io/ (Poc)

https://github.com/zktx-io/zklogin-webauthn-poc

https://github.com/sui-foundation/sips/pull/30

zktx.io