SQUARELINK

Security and privacy

Patented, Non-custodial Solution

(USPTO app: 15/974802)

Our patented, non-custodial key management solution, developed out of Harvard University research labs, enables users to interact with Squarelink via username and password while ensuring complete and exclusive ownership of keys.

> **"At Squarelink, we understand the significance of user security and don't take it for granted that we're providing a platform for users to hold potentially millons of dollars in cryptocurrency."**
>
> \- Alex Patin, Founder and CTO of Squarelink

To generate a user's Master Key, their password is first hashed to a 256-bit output length using PBKDF2 with n iterations where n is a random number between 20000 and 200000. A 256-bit, high-entropy, cryptographic salt is then generated on the client-side of the application.

Together, the user's ID, password hash, and salt are appended to one another and hashed again using PBKDF2 with n iterations, resulting in the user's 512-bit Master Key. To make use of asymmetric encryption later, we use Elliptic Curve Cryptography (ECC) to derive a public key from the user's Master Key. Along with their salt and n, this public key is then sent to Squarelink servers and stored relationally with the user's email. With this information, Squarelink can leverage Zero-knowledge Proofs to authenticate users and encrypt wallets for safe-storage without ever touching the user's password.

Supercharged Security

In addition to our novel non-custodial approach, the Squarelink protocol lends itself to enhanced, centralized security additions to further protect against brute force attacks and unauthorized access to a user's salt. Users can add MFA protection to their salt or stick with our default ReCaptcha-coupled authentication.

Account Recovery

Squarelink leverages PGP technology and novel ZKP solutions to allow non-custodial account recovery via email, phone, and security questions. We've left the 24-word mnemonic in the dust! Additionally, we've supercharged the process with centralized mechanisms – we always require email and MFA verification before a user can begin to recover their account.

Read about the full protocol at squarelink.com/Squarelink.pdf

The Squarelink Difference

Many popular, iframe-based solutions like Portis and Fortmatic must often compromise on basic security requirements you'd expect from standard connected applications.

Phishing

Using iframes creates vulnerabilities allowing malicious dapps to easily phish users credentials. With wallets that offer clean & simple UI/UX, mimicry by hackers is trivial. Any interaction with Squarelink is done via window popup to mitigate phishing risks.

CSRF/XSS

OWASP-compliance requires the usage of browser-side CSRF tokens in addition to cookie-based JWTs to offer both XSS and CSRF protection together. However, modern browsers block 3rd party cookies in iframes. This forces iframe solutions to forgo these critical authentication protections and thus open their wallets to trivial CSRF attacks.

Client-to-client Communication

Many dapps open themselves to attacks by malicious users themselves when they trust information coming directly from a 3rd party web-client (i.e. "reputation" on Portis). Squarelink is **the only OAuth 2.0-compliant** web3 wallet, so all information you get from a user is retrieved from our API using an OAuth access token rather than directly from a potentially malicious user. What good is a "reputation" if you can't trust it? No need to worry about OAuth implementation details though, our client-side OAuth interaction is all wrapped nicely in our Web3 Provider SDK.

"Trusted" DApps Vulnerabilities

Several solutions offer the ability for a "trusted" 3rd party to sign transactions on behalf of a user. First, this requires a mechanism for storage of private keys in persistent browser storage. This is incredibly dangerous in solutions that have the aforementioned XSS risks. Furthermore, because communication with these dapps is unauthenticated (no client secret is used unlike Squarelink's OAuth solution), a malicious dapp could copy the client/dapp ID of a user's "trusted" dapp to send transactions unrestricted on their behalf.

Read more at squarelink.com/Squarelink.pdf

Why Trust Us?

OWASP-Compliant
The Squarelink team does comprehensive reviews on each build to ensure utmost protection against all common web2 attacks.

OAuth 2.0/OpenID Connect
Our web3 SDK doesn't compromise either. No sensitive information is ever accessed without protection for the user and the dapp (see iframe vulnerabilities).

Rapid7 Certified
All of our builds are thoroughly pen-tested by world-class hackers using the best tools.

Data at Rest
While we never store sensitive data on our servers, we still go the extra mile and encrypt all data at rest.

HSMs Aren't That Great

Sure, HSMs are great for things like testnet faucets. But at the end of the day, HSMs are only as secure as the web service sitting in front of it. With solutions like Fortmatic, if their centralized service is compromised, so is their HSM. As you can see to the left, it's unlikely these HSMs are as protected as they should be. Having access to a private key itself isn't necessary if you trigger transactions to empty wallets, unrestricted, through a compromised web service.