PrePrint - Report - Software Engineering and Human Computer Interaction aspects of Self custody with FROST

Pacu - ZWCD

March 2024

1 Scope

The scope of this paper is to introduce the reader to the problem of self-custody on hot wallets and propose a sovereign key custody pattern that allows users to increase safety of their daily transactions by splitting the spend authority to several devices within a threshold scheme that takes away individual spend authority to internet connected wallets making them less vulnerable to capture by adversaries without sacrificing versatility or requiring dedicated companion hardware.

2 background

Self-custody wallets that are connected to the Internet are the most widely used to interact with cryptocurrency protocols. According to a study on hot wallet popularity from 2023 by CoinGecko [21], the most popular browser self-custody wallet is Metamask [18] with more than 50 million users. While cryptocurrency use gains more popularity, so do malware targeting browsers extensions of compromised hosts [14]. Cryptocurrency users are an usual target for phishing and spoofing attacks on social media. While users of cryptocurrencies who have interacted with companies providing services to them can targeted through email phishing campaigns crafted with leaked or stolen email databases [19], users of self-custody wallets can be profiled and redirected to scam sites that either do phishing or promise substantial rewards from being early adopters of some project, appealing to the profit-seeking nature of inexperienced cryptocurrency day traders. Airdrops, meme coins and other kinds of crypto projects are attack vectors for identifying and targeting cryptocurrency users that are seeking to be “early” to novel projects to make a quick turnaround to an supposedly insignificant investment [13].

The key difference between malware attacks and scam projects is that the first one can be executed without direct action of the victim and the second one needs that the users performs a direct action that grants permission to the application to perform a transaction with the users' keys.

Malware, scammers and adversarial smart contracts are not the only foes users have to be protected from. There is one more threat users should actually fear the most: Themselves. Krombholtz et Al. [16] found that 43.2% of their subject who had gone through a loss-of-funds incident, was due to user error.

Cooling down the users' wallet means to keep the wallet connected to internet but reducing the spend capability of it. By limiting the spending power of the wallet, we can diminish the amount of exposure that the user has to attacks that rely on spend authority of the victim's wallet.

3 Shared Custody with Self

The paper "Design Patterns for Blockchain-based Self-Sovereign Identity" [17] describes a set of key management patterns present in blockchain clients and applications. The "Hot & Cold Wallet Storage" identifies two battling forces: Cyber Security and Usability. While a key may be compromised through the Internet connection the wallet constantly uses to communicate with the blockchain, a wallet that does not connect to the internet at all is less accessible to both adversaries and legitimate users. Liu identifies that one of the main usability drawbacks of the cold wallet pattern is that the cold wallet "might not be around" when the user needs it, implying that the cold wallet shall be stored and not be carried around.

Another pattern identified by Liu is "Key Shards", where "A key can be split up into several different pieces, and restored using enough key pieces". The pattern observes two forces: Loss of keys and Centralisation. Keys can be irreparably lost forever if there a single copy of them, while having multiple idempotent copies of the keys provides a backup mechanism, it also facilitates higher probabilities of finding copies of the keys by adversaries. Sharding a key into several pieces that do not provide neither view or spend access to the users' funds on their own but do work collectively as backups and spend authority is a way of leveling these forces present in the pattern.

A variant of the sharding key pattern was presented October 9th 2023 by the Zcash Foundation FROST team [12] under the name "2-of-3 user / shared custody service" (see figure 1). The pattern has three actors: the user's smartphone, a Custody Service provider and a cloud backup service. The user generates FROST scheme with three possible participants with a threshold of two. The user will have two "hot" keys that will be the ones used when interacting with the blockchain and signing transactions. One being the key piece on the user's smartphone wallet. The other a signature-capable "shared custody service" that can both custody the key share and sign a transaction at request of the user.

This use case features two third-party services the user needs to trust and depend on: the share custody and the cloud backup. Fröhlich et Al. [9] take Krombholtz's [16] work as reference and focus on the risk perceptions of users.

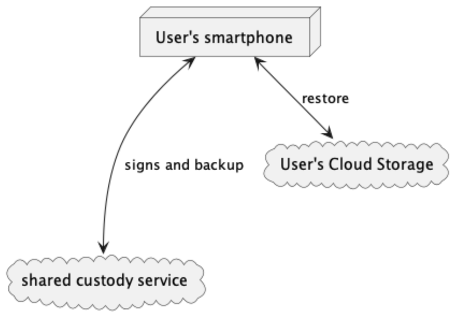

Figure 1: two out of three FROST scheme presented by ZF engineering team

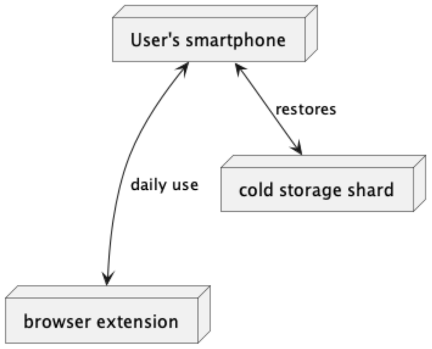

Figure 2: Shared custody with self.

According to them *"Users have to deal with the (1) Risk of Human Error, the (2) Risk of Betrayal and the (3) Risk of Malicious Attacks."*. The *Risk of Betrayal* refers to the risk of a trusted third party breaching their trust contract with their users. The shared custody and the cloud backup services would fall under this category of risks to mitigate when it comes to risks perceived by cryptocurrency users. It could also be argued that the hot wallet problem has been solved by "Hardware Wallets" which combine public keys on Hot Wallets with a dedicated disconnected device with a Trusted Execution Environment that contains the keys and signs the transactions without exposing the keys to the Internet connected wallet. Devices such as the ones manufactured by Ledger or Trezor, provide a high degree of security to their users, but ultimately they also fall under the category of *Risk of betrayal*, given that they can unilaterally ban cryptocurrencies from their software ecosystem forcing users move their funds out of their hardware wallets.

We propose a pattern called "shared custody with self". In which we combine FROST engineering team's approach with sovereign replacements for the third-party actors depicted in figure 1.

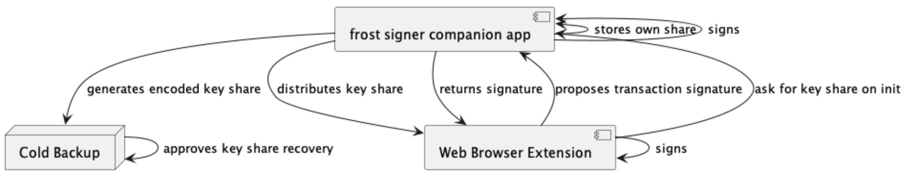
frost signer companion app
stores own share
signs
generates encoded key share
distributes key share
returns signature
proposes transaction signature
ask for key share on init
Cold Backup
approves key share recovery
Web Browser Extension
signs

Figure 3: Overview of Shared Custody with Self

Figure 2 shows a slightly different approach where the cloud backup is replaced with a cold storage of the third shard. Whereas the custody service is replaced by a hot wallet that the user will either use to transact regularly and a companion app that will sign the transactions. The user is sharing custody with itself by leveraging an additional device that it's used for signing as a hardware wallet would, but with a safety net, because none of the parts allows an adversary to spend the funds on its own. The browser extension wallet depicted in figure 2 even if compromised wouldn't be able to provide spendabilty to an adversary that could extract the keys such as some malwares attempt to. Same applies to the companion device that acts as co-signer of the 2-out-of-3 scheme. This lowers the security requirements for this device significantly, since it cannot spend funds on its own. Hardware wallets are dedicated hardware that are expensive, require specific knowledge to operate and if stolen could provide full spend capability to an attacker.

FROST allows threshold signatures schemes to be regenerated or refreshed when a key share is lost. The user can assess whether the loss is considered a security risk or not and decide whether it is safe to recover the lost share or refresh all shares otherwise.

4 Architecture

Figure 3 shows how a web browser extension wallet and a signer companion app can interact to generate the 2-of-3 scheme of the shared custody with self pattern and how their role in when signing transactions.

The companion app is both a FROST coordinator that can create the scheme from a given seed phrase and a second participant for a user that transacts using a "view-only + key share" Zcash wallet on a browser extension or mobile wallet. A browser extension is used in this example because in this scenario, the user is not require to have two devices in its possession. It is rather preferable that the mobile device of the user is the second participant instead of requiring a secondary companion device. Although there is no official announcement of a production-ready Zcash wallet, there are teams working on its creation [2].

The cooled-down-less-heated wallet in this case would be a browser extension that no longer has complete spend authority. Instead, it stores a FROST key share in its local encrypted storage. The encryption is done locally through a passphrase that the user has to input in order to use the browser wallet and authorize transactions. This is how browser wallets like Metamask currently operate. In this case if an adversary gains full control of the browser extension, even though it could access the viewing key that powers the wallet watching capabilities, it will not be able to spend the funds. Privacy will be leaked and a proper security protocol would have to be enforced to move the funds to a new seed phrase with the adequate measures that protect the user from timing attacks that allow the attacker that might be already able to watch the attacked wallet's incoming and outgoing transactions to learn the whereabouts of the new wallet the funds are being transferred to. By using a FROST 2-of-3 scheme the user was able to avoid losing it funds to an adversary.

Figure 4 shows the main components of a FROST-capable browser extension. The browser depicts a Zcash Metamask Snap [6] that allows users to use ZEC with the Metamask browser extension wallet. The *zcash snap ui* component is a Typescript / Javascript module that uses the Metamask Snaps API to show results of the *zcash snap logic* to the user through the provided hooks and callbacks in the browser extension user interface. The *zcash snap logic* is the wallet's "business logic" that knows how to knit the *zcash javascript sdk*, *frost core wasm* API and the Snap's encrypted store to provide the user the ability to receive funds, see its transactions and spend the funds with the aid of a second FROST participant.

The "zcash js sdk" component refers to a software development kit that enables Javascript / Typescript developers interact with the Zcash light client protocol [22] server implementation, lightwalletd [7]. For this to happen, researchers at Chainsafe [5] have determined that the GRPC implementation of the Light client protocol has to be proxied in order for REST/HTTP clients to connect to it [2]. Figure 4 shows how this would look like in a deployment scenario. The Zcash javascript SDK would connect to a lightwalletd instance through a GRPC Web Proxy

Another participant of this 2-of-3 scheme is a companion application. The user would carry it on its mobile phone. Figure 3 shows that this FROST companion application has many responsibilities at specific moments of the scheme's life-cycle. At creation the companion app will be the one knowing the specific requirement to act both as Coordinator and participant to generate the three FROST key shares that the user will distribute between the (not so) hot wallet, the cold wallet backup and the companion app itself. If needed, the companion app should be able to start a "recovery workflow" (see 5) with or without its own key share. We consider recovery and creation unusual events. The most common use case of the companion application should be acting as a participant/signer coordinator at the users' request. The cooled-down wallet will initiate a transaction plan. The user will have to initiate the two-round FROST signing protocol in order to sign the transaction and submit it to the network. This workflow resembles to the one detailed on the demo chapter of the FROST Book [8] where the user has to start up a coordinator and a number of signers to meet (or exceed) the threshold number. The FROST protocol [15] does not specify neither recommend how to the determine which one of the involved parties is the most suitable to act as a coordinator. This architecture intends

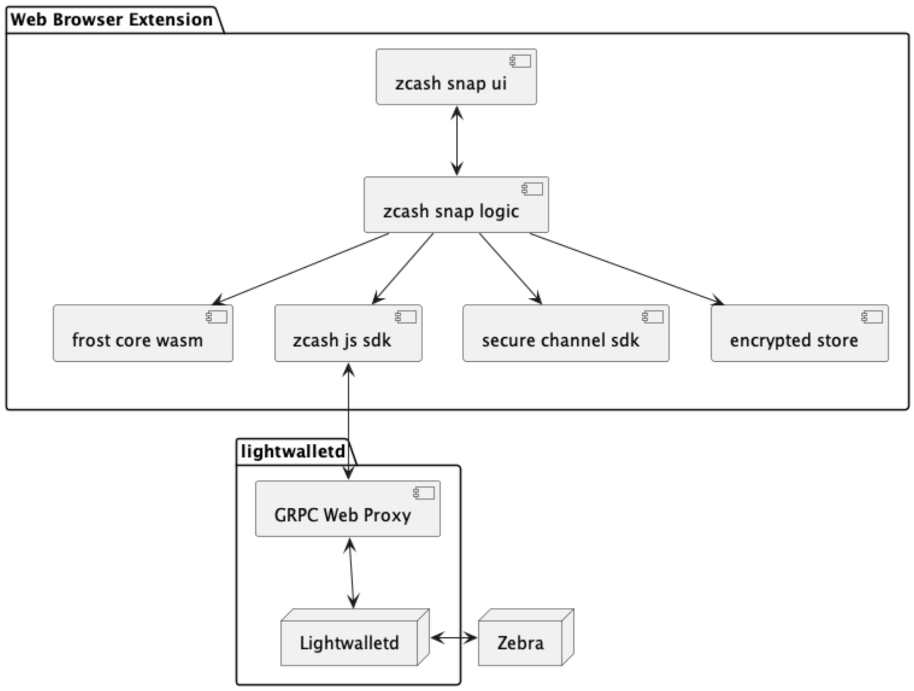

Figure 4: Component diagram of a FROST enabled browser extension

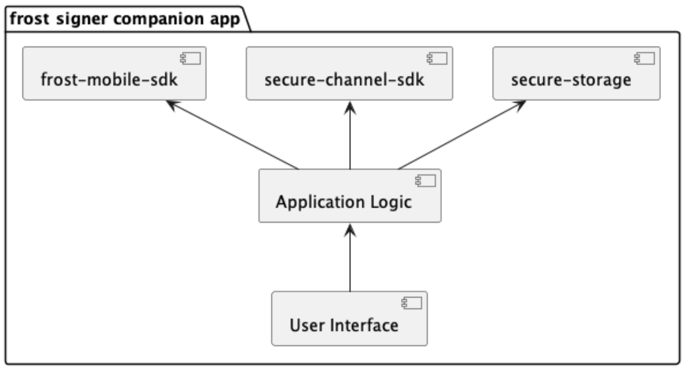

Figure 5: Diagram showing the components of the FROST signer companion application.

to keep the cooled-down wallet as lightweight as possible by delegating FROST protocol domain knowledge to the companion application, but it has to be noted that this is a design choice and not a protocol specification. The companion application will act as both coordinator and signer. It will track the number of signatures received so far and inform the users whether the threshold as been met or not. Also the coordinator is responsible of detecting bad actors and tear down the signature attempt.

Figure 5 shows the different modules composing the companion application. The User Interface module refers to the all the code needed to display the GUI and the user experience elements. The UI module calls the application logic module which contains the use cases and requirements needed for the key distribution, coordinator, participant and recovery use cases. As this use cases will use sensitive information the application relies on a form of secure storage that can be provided by the device's Trusted Execution Environment / Secure Enclave or any other form of encrypted storage.

The Application Logic contains a wide range of use cases and requirements both derived from the UI and UX elements of the companion application itself and from the FROST protocol. Although the core functionality of the latter will come from a module we called "frost mobile sdk" depicted on figure 6.

The FROST mobile software development kit is designed in a similar fashion as the Zcash Mobile SDKs for Swift [4] and Kotlin [3]. With the exception that it would be a state-less SDK sharing a single API for both platforms. Inside the *frost-mobile-sdk* component there is a *frost-rust* sub-component comprised of RustLang crates for *frost-core* and a *frost-redpallas* implementation of the *frost-core* traits to work with orchard and sapling transactions. This module would be the one exposed through a *frost-uniffi* interface. Mozilla UniFFI is a tool to generate Rust bindings to other languages such as Swift, Kotlin, Python, GoLang and C# allowing developers to code the core of the business logic in Rust and reusing it in different platforms. For this case, the *frost-uniffi* logic

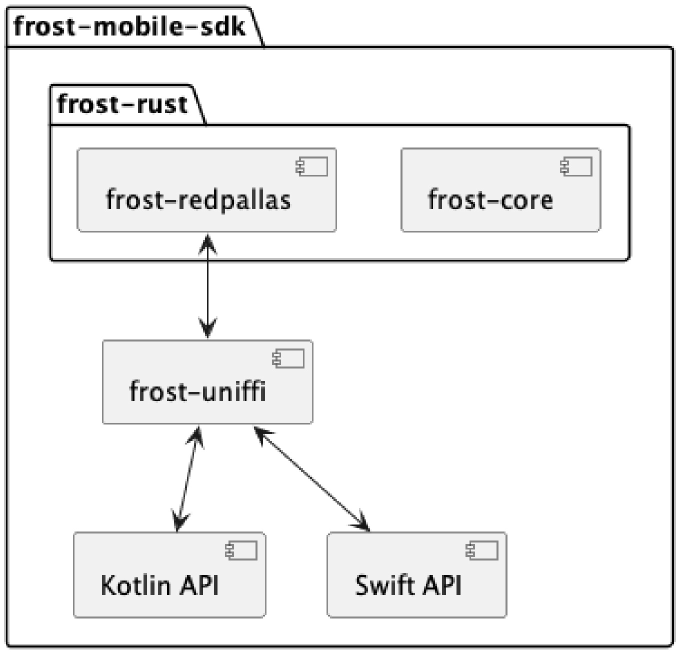

Figure 6: Proposed structure of a FROST mobile SDK

would expose an API that the Kotlin and Swift companion applications can use natively without having to implement an FFI bridge logic themselves (as it is currently the case for the Zcash Mobile SDKs).

One of the unresolved challenges of this architecture lies on the "secure channel sdk". The communication between coordinator and participants must be done through a secured end-to-end communication channel. The architecture does not specify which one should be since the user experience would vary depending on the operating systems and platforms involved. For example, a within the Apple ecosystem could leverage "continuity" features to share information between a browser running in its computer and the mobile device, while this would not be possible if any of those devices was "out" of such ecosystem (by running Android, Linux, F-Droid, Windows, etc). We a desirable solution would be one that allows a seamless experience for different platforms. One implementation detail that could be abstracted as a requirement for the secure transfer development kit would be a tool that allows to perform a secure and brief one-time exchange of information between two parties that trust each other but may distrust the channel, such as the Magic Wormhole protocol [24] which allows peers to exchange information in an "ephemeral letter box" manner. Regardless of how the secure channel question is answered. The network traffic should be directed through a mixing network that allows such information to be far from any preying eyes and observers [23].

id	As a	I would like to	so that
1	Coodinator	generate a new FROST protocol from a UFVK	I share custody a key
2	User	Start a threshold signature scheme from a new seed phrase	I can start a new shared custody with self wallet
3	User	receive a key share	I can act as a participant of a threshold signature
4	Coordinator	share a key share to a participant through secure channel	I can complete trusted key share scheme
5	User	be helped to create a backup of the seed phrase	I make sure I have done it correctly
6	User	be asked to remove the seed phrase from the application	I can be sure no can recover from seed other than myself
7	User	protect my key share with a passcode	can be sure only I access it
8	User	protect my key share with a biometric ID	can be sure only I access it
9	User	join a signing protocol as a participant	so that I can decide whether a transaciton should be signed
10	Participant	see the transaction details before signing	so that I can decide whether a transaciton should be signed
11	Participant	agree to sign a transaction	so that the coordinator adds me to the ongoign FROST protocol
12	Participant	refuse to sign a transaction	so that the coordinator knows that I don't agree with what I've been proposed to sign
13	Participant	communicate my signature decision to the coordinator through a secure channel	so that the coordinator may act accordingly
14	Coordinator	receive a tx plan and a commitment from the cooled down wallet	I can start the sgining protocol
15	Coordinator	create a signing commitment and a nonce	I can be a participant of the signing protocol
16	Coordinator	create a new signing package with the message and the signing commitments	I can distribute those to all participants through a secure channel
17	Coordinator	send the signing package to the participants through a secure channel	they can perform round two of the signing protocol
18	Participant	create a signing commitment and a nonce	I can be a participant of the signing protocol
19	Participant	receive a Signing package from the Coordinator	I can participate in round 2 of the signing protocol
20	Participant	create a Signature Share with the Signing package, nonces and Key Package	I can perform round two signature share
21	Participant	send round two signature share to the coordinator through a secure channel	the coordinator can then aggregate my signature
22	Coordinator	inform the user that one of more signatures are found to be invalid	the user can be notified that the protocol has been aborted and the reasons
23	Coordinator	aggregate the public key package, Signing Package and Singnature Shares	I can perform the FROST signature
24	Coordinator	inform the user that the signature was done correctly	the user can acknowledge the progress
25	Coordinator	send the signed transaction to the cooled down wallet	the user can review and submit the transaction
26	User	be able to review the signed transaction	I can be sure that I agree to what's been collectively signed
27	User	submit a FROST-signed transaction to the network	I can fulfill the payment I originally intended to

Table 1: User Stories for the Companion App component of the Shared Custody With Self pattern

id	As a	I would like to	so that
1	User	initialize my wallet with a Unified Full Viewing Key and a key Share	I can use my cooled down wallet in my broswer extension
2	User	my UFVK and key share to be stored in a password encrypted storage	accessing that information requires a password that I know of
3	Participant	Create a transaction plan and signing commitments and nonces	I can tell a Coordinator to run this ceremony for me
4	Participant	Send a coordination my transaction plan and commitments through a secure channel	I can tell a Coordinator to run this ceremony for me
5	Participant	receive a Signing package from the coordinator through a secure channel	have the information needed to perform round two of the FROST signing protocol
6	Participant	create a signature share with the signing package, nonces and key share	I can then share the signature share with the coordinator
7	Participant	send the signature share to the coordinator through a secure channel	the coordinator can aggregate the signature shares and perform the signature
8	User	receive the signed transaction from the coordinator	I can later verify the signature and review it
9	User	review the signed transaction	I can agree or reject it
10	User	agree with what has been signed through the FROST signing protocol	I can tell the application to submit it to the network
11	User	reject the signed transaction	I can tell the application to roll back any state and delete the transaction attempt.

Table 2: User Stories of the self-custody with self enabled browser extension

5 Requirements

We will focus on the specific requirements that are needed to implement the proposed elements of the architecture. In terms of Privacy Coin wallets, the are different pieces of literature that already cover those in more depth [11] [10]. The subsections below detail a preliminary list of user stories comprising the requirements for the companion application (table 1) that can act as participant, coordinator and recovery actor within the roles established in the FROST protocol. Table 2 lists a preliminary collection of user stories condensing requirements of the Web Browser Extension wallet component porposed by the Shared Custody with Self pattern that are related to an initiating participant of FROST protocol for signing a given transaction.

5.1 FROST Signer Companion Application

5.2 Web Browser Extension Wallet

6 Conclusions

In this work we introduced the readers to the problems of self-custody of cryptocurrencies and we presented studies that showed how probable were those risks withing a group of 990 Bitcoin users, identifying user error as the most prominent source of loss-of-funds. We presented an custody pattern we called "Shared Custody with self" that uses FROST threshold signatures to mitigate the presented risks and also add extra recovery options provided by the Key Share recovery capabilities of the FROST protocols. The presented custody pattern introduces the notion of a Companion Application that contains the logic of the FROST protocol and avoids that such logic has to be present in the (cooled down) hot wallet. Additionally the pattern allows ZEC holders to have a reliable, robust and fail-proof self-custody with tools that are funded and developed completely within the Zcash Ecosystem and do not rely on third parties.

7 Feasibility and Future Work

The present paper introduces components that are yet to be developed. The Browser Extension cooled-down wallet is not currently available but at the moment to writing this report, there are efforts being done in such direction such as the Zcash Javascript SDK [2] and the integration of Zcash to the Brave Browser wallet [1]. The FROST Mobile SDK is one of the areas the Zcash developer ecosystem should work on in order to bring the FROST protocol to mobile devices for self custody and organizational custody schemes. The secure channel required by the FROST protocol could be eventually be routed through a Nym Mixnet. Such capability is being developed by Nym Technology's team with a ZCG Grant [23].

8 Acknowledgements

The present report is part of Deliverable 2.4 of the ZWCD grant by Zcash Community Grants Committee [20]. Thanks to Conrado Gouvea for proof reading and brainstorming ideas for this report.

References

[1] coinmarketcap.com. Brave partners with electric coin and filecoin to advance web3 privacy. `https://coinmarketcap.com/community/articles/650dd01b65f8d9726cecbecb/`, 2023. [Accessed 20-03-2024].

[2] Zcash Community Grants Committee. ZCash SDK Feasibility Study [JS/TS] — zcashgrants.org. `https://zcashgrants.org/gallery/25215916-53ea-4041-a3b2-6d00c487917d/44833322/`. [Accessed 17-03-2024].

[3] Electric Coin Company and Zcash Developers. GitHub - Electric-Coin-Company/zcash-android-wallet-sdk: Native Android SDK for Zcash — github.com. `https://github.com/Electric-Coin-Company/zcash-android-wallet-sdk`, 2019. [Accessed 18-03-2024].

[4] Electric Coin Company and Zcash Developers. GitHub - Electric-Coin-Company/zcash-swift-wallet-sdk: iOS light client Framework proof-of-concept — github.com. `https://github.com/Electric-Coin-Company/zcash-swift-wallet-sdk`, 2019. [Accessed 18-03-2024].

[5] Chainsafe Developers. ChainSafe Systems - Blockchain Research and Development — chainsafe.io. `https://chainsafe.io/`, 2024. [Accessed 18-03-2024].

[6] Metamask developers. Customize your wallet with MetaMask Snaps — metamask.io. `https://metamask.io/snaps/`, 2024. [Accessed 18-03-2024].

[7] Zcash developers. GitHub - zcash/lightwalletd: Lightwalletd is a backend service that provides a bandwidth-efficient interface to the Zcash blockchain — github.com. `https://github.com/zcash/lightwalletd`, 2018. [Accessed 18-03-2024].

[8] Zcash Foundation. Ywallet Demo - The ZF FROST Book — frost.zfnd.org. `https://frost.zfnd.org/zcash/ywallet-demo.html`, 2023. [Accessed 18-03-2024].

[9] Michael Fröhlich, Felix Gutjahr, and Florian Alt. Don't lose your coin! investigating security practices of cryptocurrency users. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference*, DIS '20, page 1751–1763, New York, NY, USA, 2020. Association for Computing Machinery.

[10] Francisco Gindre. *Arquitectura de software en wallet de código abierto para privacy coin en dispositivos móviles.* PhD thesis, Universidad Nacional de La Plata, 2021.

[11] Francisco Gindre, Matias Urbieta, and Gustavo Rossi. Patterns for anonymity enhancing cryptocurrencies non-custodian mobile wallets. In *Proceedings of the 29th Conference on Pattern Languages of Programs*, PLoP '22, USA, 2023. The Hillside Group.

[12] Conrado Guvea. Zeal Call: FROST for Zcash + Q&A — youtube.com. `https://www.youtube.com/watch?v=XG-5txt4Cko`, 2024. [Accessed 14-03-2024].

[13] Andrea Horch, Christian H Schunck, and Christopher Ruff. Adversary tactics and techniques specific to cryptocurrency scams. 2022.

[14] Erhan Kahraman. Hodlers beware! New malware targets MetaMask and 40 other crypto wallets — cointelegraph.com. `https://cointelegraph.com/news/hodlers-beware-new-malware-targets-metamask-and-40-other-crypto-wallets`, 2022. [Accessed 14-03-2024].

[15] Chelsea Komlo and Ian Goldberg. Frost: Flexible round-optimized schnorr threshold signatures. Cryptology ePrint Archive, Paper 2020/852, 2020. https://eprint.iacr.org/2020/852.

[16] Katharina Krombholz, Aljosha Judmayer, Matthias Gusenbauer, and Edgar Weippl. The other side of the coin: User experiences with bitcoin security and privacy. In Jens Grossklags and Bart Preneel, editors, *Financial Cryptography and Data Security*, pages 555–580, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg.

[17] Yue Liu, Qinghua Lu, Hye-Young Paik, and Xiwei Xu. Design patterns for blockchain-based self-sovereign identity, 2020.

[18] Metamask. The Ultimate Crypto Wallet for DeFi, Web3 Apps, and NFTs — MetaMask — metamask.io. https://metamask.io/, 2024. [Accessed 14-03-2024].

[19] Kirsty Moreland. Message by LEDGER's CEO - Update on the July data breach. Despite the leak, your crypto assets are safe. — Ledger — ledger.com. https://www.ledger.com/message-ledgers-ceo-data-leak, 2020. [Accessed 14-03-2024].

[20] Pacu. Zcash Wallet Community Developer 2024 — zcashgrants.org. https://zcashgrants.org/gallery/25215916-53ea-4041-a3b2-6d00c487917d/45148172/, 2024. [Accessed 20-03-2024].

[21] Lim Yu Qian. Most Popular Crypto Hot Wallets for Self-Custody. https://www.coingecko.com/research/publications/most-popular-crypto-hot-wallets, 2023. [Accessed 14-03-2024].

[22] George Tankersley and Matthew Green. ZIP 307: Light Client Protocol for Payment Detection — zips.z.cash. https://zips.z.cash/zip-0307, 2018. [Accessed 18-03-2024].

[23] Nym Technologies. The Nym mixnet for Network Privacy for Zcash — zcashgrants.org. https://zcashgrants.org/gallery/25215916-53ea-4041-a3b2-6d00c487917d/44416053/, 2023. [Accessed 20-03-2024].

[24] Brian Warner. Magic-Wormhole: Get Things From One Computer To Another, Safely; Magic-Wormhole — magic-wormhole.readthedocs.io. https://magic-wormhole.readthedocs.io/en/latest/, 2017. [Accessed 18-03-2024].