Research & Development

The goal of this research and development document is:

- to introduce the concepts of Multiparty Computation and Signatures (Single-Sig, Multi-Sig and Threshold-Sig)
- to describe the security and usability issues with current seed phrase mechanisms.
- to provide a high-level overview of the current state of MPC and BIP032 wallets.
- to design the architecture of the new system, specifying the roles and interactions of the main components.
- to determine the technical requirements and constraints of the project.
- to discuss potential next steps for further development and implementation.

Concepts

The control, security and accessibility of digital assets are defined by several technology concepts, which a user can choose from and needs to understand. Among them different types of custody, wallet and signature types. While custody describes who holds and controls the funds, wallets serve as the digital interface to access and manage these funds. Signatures are an integral part of wallets and are used to authorise transactions. Below these concepts are explained in more detail.

Custody types

Custodial / Third-party

With traditional money, a bank or other financial institution secures the money of customers. You don't have access to your money, but the bank allows you to access it. This is called a custodial wallet and is available in the crypto space as well. Big custodians are exchanges like Coinbase, Binance or Kraken. They are responsible for the security of your funds and you have to trust them to do so.

Custodians have several advantages, such as:

- Easy access to wallet by common authentication methods (e.g. username and password)
- Easy recovery of wallet (e.g. password reset)
- Easy to use (e.g. user-friendly interface)
- Easy to share access to wallet (e.g. shared access to account)

However, there are also disadvantages:

- The user does not own the funds (e.g. funds can be frozen or lost)
- The custodian is not decentralised (e.g. single point of failure)
- The custodian can be hacked, be mismanaged or go bankrupt
- Privacy concerns due to knowledge of transactions and user's assets

Self-Custodial / Non-custodial

The user has full control over assets and keys with self-custodial wallets. This is typically used with single signature wallets, but can be achieved with MPC wallets as well.

Shared Custodial

A shared custodial comes in form of a multi-sig wallet, where the keys are distributed over multiple parties, e.g. users, clients or third parties, so that no single party has full control over the wallet and multiple parties have to sign a transaction to send funds.

Signature types

When you want to send a transaction to someone, it needs to be signed so others can verify it's validity. This works as follows:

The sender generates a public/private-key pair and keeps the private key secret. The public key is shared with the world. The sender then signs the transaction with the private key and sends it to the receiver. Any receiver can then verify the signature with the public key. A digital signature, unless handwritten signatures, changes with the content that is signed, even with the same private key. This ensures integrity, meaning that the content has not been tampered with but also authenticity and non-repudiation, meaning that the holder of the private key signed the data and cannot deny it.

The security of signatures lies in the hardness of calculating the private key from a given public key. The security of a signature scheme depends on the size of the key: The larger the key, the harder it is to calculate the private key.

Wallets need to support signatures to sign transactions or decrypt backups. There are different types of signatures, which are explained below.

Single-Signature

Single-Signature is the most common and easiest to use type of wallet as it uses a single private key to sign transactions. The private key is stored on a single device.

However, it is the least secure because the funds are lost if the device is lost or stolen. Even with the strongest hardware security, access is guaranteed if the key is compromised. Since only one key is used, there is no option for backup or recovery builtin.

The most significant problem is, that this private key is derived from a seed phrase. This seed phrase is typically a mnemonic phrase of 12 or 24 words (making it more user-friendly than a long string of random numbers and letters). Although user-friendly but not very secure is the fact, that this seed phrase is the only thing needed to recover the private key. Most people store it on a sheet of paper or digitally. They can be lost, stolen, damaged, or hacked. More on the complications of seed phrases below.

Further, single signature wallets are not suited for groups, companies or organisations as a shared access to the wallet is not possible.

TLDR;

Pros	Cons
Easiest to use.	Least secure.
Widely supported and recognized.	Funds are (most likely) lost if seedphrase is lost or stolen.
Seed phrase makes recovery user-friendly.	No other way for recovery. The user has full responsibility.
More straightforward implementation.	Not suited for shared access (e.g. groups, companies).
No backup or recovery options.	

Multi-Signatures

Multi-signature wallets offer heightened security compared to single-signature wallets because they necessitate multiple parties for a transaction. This multi-signature mechanism reduces the single point of failure, as multiple devices would have to be compromised to gain unauthorized access. For instance, in a 2-of-3 configuration, even if one key is compromised or lost, the funds remain accessible. This enhances user-friendliness with regard to security significantly, but also involves more steps compared to single-signature wallets.

When combined with cold storage, multi-signature wallets further bolster security. Here, a hardware wallet signs transactions, ensuring the private key remains unexposed to the internet, even on a web-connected computer. Such a setup, referred to as an air-gapped device, also safeguards against potential physical threats or coercion.

However, each participant in a multi-signature scheme possesses a distinct public/private key pair (and seed phrase). During transaction sign-offs each party individually signs the transaction with their private key and then the individual signatures are combined. However, during the process these public keys become visible, potentially linking them to a specific wallet. This raises privacy concerns, as multiple transactions can be traced back to a single wallet. The performance also takes a hit since verification requires consolidating all the relevant keys. Furthermore, larger transaction and storage sizes elevate the associated fees.

Adjusting the security policy of a multi-signature wallet, such as modifying the threshold of signatories or introducing/removing participants, presents challenges. Typically, it necessitates the creation of a new wallet and fund transfer, leading to a new public address. Such alterations are unattractive for entities aiming for address consistency. Additionally, the fund transfer incurs another fee and leaves a transactional footprint on the blockchain, potentially revealing strategic information.

An aggregated signature offers a potential remedy by generating a consolidated signature from multiple signatures spanning diverse messages and signers. This method promises swift verification while reducing storage, throughput, and fees. However, not all wallets support this, it lacks standardization, and implementation can be intricate.

TLDR;

Pros	Cons
Enhanced security requiring multiple signatures.	Performance challenges due to longer key verification time.
Reduces single point of failure.	All public keys become visible in the transaction, raising privacy concerns.
Access, even when one key (or seed phrase) is lost.	Larger transaction sizes increase fees.
Combines effectively with cold storage and provides protection against physical threats.	Adjusting security policies can be complex and unattractive for maintaining address consistency.
	Fund transfers during policy changes incur fees and leave blockchain footprints.
	Still seed phrases used to generate private keys.

What are Threshold Signatures?

Threshold signatures, also called Threshold Signature Schemes (TSS), use multiple distributed private keys instead of one. They reside on multiple devices but can communicate to each other to sign transactions. When a transaction needs to be signed, a subset of involved parties sign the transaction with their individual private key and then the individual signatures are combined. This means that the private keys are never combined and therefore never exposed. This is a significant advantage over multi-signature wallets, where the private keys are combined and exposed during the signing process.

TLDR;

Pros	Cons
Private keys are never combined, reducing risk of exposure.	Implementation complexity is typically higher than single-key solutions.
Provides redundancy; even if some keys are lost, the system can still function (up to the threshold).	Coordination between devices/parties is required, which can potentially delay transactions.
Provides a single public key and signature, making it indistinguishable from single-signature schemes on-chain.	If not properly implemented, weak points or vulnerabilities in the scheme could be exploited.

Pros	Cons
Can enhance privacy, as the number of participants/signers remains hidden.	Recovery mechanisms can be more complex compared to traditional wallets.
No single point of failure: compromising one key doesn't compromise the entire system.	Ensuring secure communication channels between devices/parties can be challenging.

Wallet types

Software / Hot wallets

These are offered by online platforms or exchanges that allow fast, easy and convenient access to a wallet by using the web browser, a desktop or mobile app. However, they are connected to the internet and therefore more susceptible to hacks, phishing and malware. They are always custodial wallets and come with the same advantages and disadvantages as described above.

Hardware / Cold wallets

Hardware wallets are self-custodial and store the private keys offline on physical devices. Popular choices are Ledger or Trezor. They are considered the most secure way to store crypto assets. However, they are not very user-friendly because a transaction can only be signed on the device itself, requiring a connection to a computer. They are also expensive and prone to loss, theft or damage.

Paper wallets

Paper wallets are self-custodial in a way that the public and private keys are printed out on a piece of paper and then removed from the digital wallet and the network. Similar to hardware wallets, they are offline and considered very secure but not very user-friendly. They are also prone to loss, theft or damage. Furthermore, they rely on the printer to work properly (no ink spots, no paper jams, etc.) and the printer or computer to not store any copies of the keys.

Smart contract wallets

Smart contract wallets allows the user to control the wallet with a smart contract and not with a private key. They give the user complete control over their funds as no other party holds access unless the user chooses to employ multi-sig options. It also offers recovery without a seed phrase, transfer limits to prevent drainage if the wallet is compromised and interoperability with dApps. They are also open-source, allowing for everyone to audit them. Smart contract wallets offer gasless, meta and batched transactions, allowing for others to pay for the transaction, bundling actions in one transaction or simultaneous execution of transactions. However, every new action, like adding/removing users or thresholds or recovery, require and on-chain transaction and therefore gas fees (Source).

However, smart contract wallets are prone to phishing attacks and bugs in the smart contract code. A smart contract wallet doesn't need a seed phrase because it uses trusted third parties named guardians to recover funds. Guardians can be friends, family or other devices the user owns. You need to trust these parties (again putting security in the hands of others) and they need to be available when needed. If a third party is used to set the wallet up, it could mean that the user does not have full control over the wallet.

Examples for Smart contract wallets are Gnosis, Argent and Unipass.

Multiparty Computation wallets

Multiparty Computation (MPC) is a cryptographic method to allow multiple parties to jointly compute a function without revealing the inputs to anyone. MPC protocols can generate keys and share them between (distrustful) parties, without them being in one place at any time. It can sign transactions over these distributed parties/keys without the need to combine the keys, diminishing the single point of failure that would otherwise exist.

MPC has been introduced to wallets because the other wallet (and signature) types have severe security problems. Users need full control over their keys and the usability of exchanges or custodians. Many users don't want to fully rely on themselves to keep something this important safe. With MPC it is possible to have the best of both worlds - security and usability.

As mentioned above, TSS employs MPC techniques but MPC wallets **must not** use TSS for transaction signing. They can also use other cryptographic protocols, such as Shamir's Secret Sharing Scheme (SSSS). However, with SSSS one party has to be trusted as the dealer, which generates, distributes and, at time of signing, recombines all shares. This is an undesired single point of failure.

MPC Threshold Signatures can be seen as a special case of MPC where the function to be computed by MPC is a cryptographic digital signature and the inputs are shares of a single private key. A threshold number of shares is enough to sign a transaction, thus the full private key is never reconstructed or exposed and the number and identity of signers are not revealed. This makes it indistinguishable from single-signature schemes on-chain and keeps privacy. Compared to Multi-Sig, the performance is better as the verification process is faster and the transaction size is smaller (Source).

TSS supports Distributed Key Generation (DKG) in the MPC scenario, where the protocol itself and not a single party is responsible for the key generation and knows the full private key. Also, any party can initiate the protocol.

Because MPC depends largely on cryptographic methods, it is difficult to implement, requires a lot of testing and susceptible to future vulnerabilities, which must be addressed.

The problems of seed phrases are mentioned in the next section. It needs to be said, that MPC wallets don't use seed phrases and therefore don't have these problems.

TLDR;

Pros	Cons
Only one signature, that leaves no trace on the number and identity of signers.	Key sharing must be secure.
Private key is never reconstructed or exposed.	Implementation is difficult.

Pros	Cons
Transaction size with TSS is smaller, meaning less blockchain space, fees and faster verification.	Depending on the number of involved parties, transactions might be slower than single-key transactions.
Any party can initiate the protocol in a TSS setup. No central dealer (like SSSS), where the full private key is restored.	Future vulnerabilities in cryptographic methods can have severe consequences.
Fault tolerance: Even if some devices or shares are lost, the remaining shares can still perform cryptographic operations as long as the threshold is reached.	
Resistance to attacks: An attacker would need to compromise a large number of shares (above the threshold) to gain control over the private key, making it significantly more challenging to breach the system.	
Scalability: The threshold can be adjusted according to the desired level of security and the number of devices or participants involved.	
MPC wallets that adhere to the ERC-4337 standard can work across all EVM-compatible blockchains.	
MPC allows granular access control with individual permissions. Especially helpful for organisations.	
MPC wallets cater to sophisticated transactional needs, such as time-locking, multi-step approval processes, and spending limits. This functionality is essential for organizations with complex financial operations or compliance requirements.	

Sources:

- Thirdweb Blog

Seed phrases

What is BIP32, BIP39 and BIP44?

BIP32 is the "Bitcoin Improval Proposal" which introduced the implementation of Hierarchical Deterministic (HD) wallets to Bitcoin. It allows to recover the entire wallet by using a single seed phrase. With BIP39, this seed phrase became a mnemonic phrase of 12 or 24 words, making it more user-friendly than a long string of random numbers and letters. The seed phrase is used to generate a master private key, which is then used to generate a tree of deterministic child keys. Deterministic means, that the same seed phrase will always generate the same tree of keys. This makes it possible to backup a wallet by simply writing down the seed phrase on paper.

The master key can be represented in its extended form, often referred to as the "xpriv", where an extended public key ("xpub") can be derived from. The extended keys allow to derive child public keys, which can be used to generate addresses and receive funds without exposing the master keys. When new addresses are generated for every transaction, transactions cannot be linked to a user.

BIP44 was implemented to allow HD wallets to import extended keys and find the bitcoin stored in a wallet by defining the standard derivation path for keys and addresses, so that the extended keys start with the prefix "xpub" and "xprv". If the "xpub" is exposed, no security but privacy can be compromised as it enables to derive all public keys and see past and future transactions. Extended keys offered the possibility to store private keys offline in a cold wallet and use the xpub online to generate new addresses and receive funds.

What are the security and usability issues with current seed phrase mechanisms?

One single seed phrase allows to backup and recover an entire wallet which makes it very powerful and more user-friendly than a long random string of numbers and letters. The most used wallets today, i.e. Metamask, Ledger, are still using seed phrases, but they are prone to many errors.

The user is solely responsible for keeping the seed phrase safe, but often fails to do so. They often ...

- don't understand the importance of the seed phrase and expose it to others.
- store it on a paper, which can be lost, stolen or damaged.
- misremember words or the order of words and make typos.
- store it digitally, which can be exploited. If stored on an USB-stick, it can be lost, stolen or damaged and also corrupted, making it unreadable.

The seed phrase acts as a single point of failure with no further backup or recovery option possible, and when compromised, funds are lost. With almost $1B in digital assets stolen until August 2023, it's evident that security is a major concern for the industry (Source). A recent event in September 2023 showed how easy and fast a wallet can be emptied. A brasilian streamer unluckily showed his private key on a live stream and lost close to $50k in MATIC (Source). Furthermore, phishing attacks are common, where users are tricked into entering their seed phrase on a fake website.

Lastly, there is the problem of missing standardisation across wallets in terms of the number of words or the used dictionary. This decreases compatibility and makes it difficult to recover a wallet with a seed phrase from a different wallet.

In conclusion, recoverability is crucial for mass adaption and the hurdle of seed phrases is too big for new users. You can give this responsibility to custodians or exchanges, but the common saying "not your keys, not your crypto" shows, that you don't own your funds when others have access to your keys. But this is what stands digital money apart from traditional money, because it is mathematically and cryptographically possible to ensure you that only you have access to your funds. MPC solves these problems.

Current state of MPC wallets

Important aspects when comparing MPC wallets are the supported schemes, signatures and type of custody. There is also a difference between (open-source) wallets, wallet infrastructure and Wallet-as-a-Service (WaaS).

Wallet infrastructure providers like Web3Auth, Magic Wallet and Sepior (acquired by Blockdaemon) help to build wallets and Wallet-as-a-Service like Coinbase WaaS and Cobo offer a ready but customizable wallet. Wallet infrastructure refers to the underlying systems and technologies that support cryptocurrency wallet operations, necessitating that companies set up, manage, and maintain it themselves. This often allows for a greater degree of privacy customization based on the architecture chosen. Conversely, Wallet as a Service (WaaS) is a service model where wallet functionalities are offered by a third party. Though the service provider typically manages maintenance, updates, and security in WaaS, the inherent centralized nature might lead to potential privacy considerations or compromises. Both benefit from a robust and established system, but there is full relience on the tools, services, and technologies that they offer. Switching to a different infrastructure down the line can be time-consuming and potentially costly.

On the other hand, using open-source libraries to build a wallet from scratch provides more flexibility and autonomy. This approach allows to avoid vendor lock-in and potentially create a unique solution tailored to exact needs. However, it typically requires more initial development work, and proactive maintenance about security and updates.

Wallet Infrastructure and Wallet as a Service (WaaS) providers

A short list of the above mentioned providers and their properties shall be given for comprehensiveness.

Sepior (acquired by Blockdaemon)

- Sepior offers a combination of advanced Cryptography & Wallet services.
- It's backed by world-renowned cryptographers with over 20 years of experience in MPC.
- Supports threshold signatures ECDSA and EdDSA.
- Capacity to manage millions of keys and handle up to 10,000 TPS
- Support any device: VMs, Container, Desktop, Mobile
- Any protocol, ranging from Bitcoin and Ethereum to newer entrants like Tether and Uniswap.
- Integration through webhooks, REST APIs, or SDKs
- Plugins that cater to the growing NFT and DEFI markets, along with KYC processes.
- Any operational model: Hot, warm, cold, trading
- Any hosting model: On-prem, public cloud, private cloud

Web3Auth

- Web3Auth is a pioneer in combining passwordless authentication with MPC, crafting a unique WaaS infrastructure specifically tailored for dApps and wallets.
- Support various signature schemes, including GG19, GG20, DKLS19, and FROST.

- Using GG19/GG20, a typical 2-party transaction might need 4-7 seconds for pre-compute on a browser. In contrast, DKLS19 considerably speeds things up, clocking in at approximately 1.5-2.5 seconds, allowing for a seamless user experience.
- Strong partnerships with Universal, Ubisoft, Trust Wallet, and Binance.
- A notable milestone for Web3Auth was their beta launch in April 2023, backed by a significant $13M Series A funding round from Sequoia Capital.

Magic

- Out-of-the-box (but customizable) wallet UI for login, NFT Checkout & Gallery, fiat on-ramp.
- Over 20 authentication methods and MFA.
- Self-custodial with support for key export.
- Magic Wallet offers an array of solutions, prominently employing signature schemes like DKLS19 and FROST.
- They provide support for popular signature types like ECDSA and EDDSA, catering to a diverse set of blockchain applications.
- In terms of transaction processing, Magic Wallet boasts a performance rate of 1.5-2 seconds, making it a competitive choice for users seeking swift transactions.
- An added advantage is their regionally distributed infrastructure, ensuring optimal performance across different geographies. Regional distribution can drastically reduce latency issues*
- A unique feature of Magic Wallet is its ability to support gasless transactions, which can be built on top of their existing infrastructure. Gasless transactions can significantly reduce costs and attract a wider user base.

Cobo Wallet

- Offers a range of custody solutions, including Full Custody, Self-Custody, and WaaS.
- Primarily designed with institutional needs in mind, prioritizing security and operational efficiency.
- Cobo MPC Lite: a self-custody solution targeting all-size Web3 companies (from $99/month with 2 wallets, 3 users and until $200k).
- Over 500 institutional clients and over 1 million registered individual users across 4 continents.
- 100B+ transactions in 6 years continuous service
- Provides compatibility with a vast number of tokens and chains, totaling over 1,800 tokens and 70+ chains.
- Equips developers with a rich set of APIs and SDKs for integration purposes.
- Holds both SOC 2 Type 1 and Type 2 certifications.

Krayon

- Provides Wallet Infrastructure along with WaaS API.
- Offers an MPC Wallet tailored to small and mid-sized businesses.
- Capable of processing payments for up to 100 customers simultaneously.
- Underwent security measures including Sayfer AUDIT and penetration testing.
- Operational since 2022 and trusted by over 50 organizations with records of 10k+ wallets created, 110+ users onboarded, and 15,868 transactions processed.
- Marketed as a more cost-effective alternative to platforms like Fireblocks or Gnosis Safe.
- Supports cryptocurrencies like Bitcoin, Ethereum, and Polygon.

- Received backing from notable entities such as Emurgo, Blockchain Founders Fund, GSR, Saison Capital, among others.
- Holds SOC 2 Type 1 certification.

Particle Network

- A simple self-custodial auth infra for Web3 apps and wallets, with focus on dApps.
- Also offer WaaS with the offer to integrate an in-ap wallet in under 30 min. on iOS, Android, Web and Unity.
- Integration with any wallet, platform, chain or social login.
- Codebase audit by Certik and Salus.

Silence Laboratories

- Deep-tech, decentralized, authentication-infrastructure provider.
- Offers developer-focused SDKs and libraries, for protocols, wallets, and projects that can be used to embed/build distributed, zero-trust authentication and authorization products.
- For ease of integration, we are application, protocol, custodian, and stack agnostic, collaboration-friendly, and future-forward.

Coinbase WaaS

- Introduced to the public in March 2023.
- No mnemonic seed phrase
- Two key shares created at wallet creation and both needed for signing transactions (2-of-2):
  - One at Coinbase, safeguarded by public encryption and verified without direct access to private keys.
  - Another user-oriented one, emphasizing speed and security. It supports cloud, local, and hardware backups.
- Enforces key management principles ensuring that Coinbase cannot unilaterally act and that users can restore or export wallet without Coinbase's involvement.
- Uses the "Threshold verifiable random function (VRF)" for key derivation instead of the more common HMAC-SHA512.
- Aims to incorporate a DApp browser into the wallet in the near future.
- HD Wallet functionality supported.
- Does not currently combine MPC with multi-sig and employs an M-of-M scheme.

Fireblocks WaaS

- Embed custodial or self-custodial wallets into any application or platform.
- Hot, warm or cold wallets.
- Support for every blockchain and leading token standards for Web3 and NFTs.
- Private key shares are created and encrypted in isolated Secure Hardware Enclaves across multiple cloud centers. To sign transactions, the key shares are used to perform multiple rounds of computation all without ever being brought into the same environment.
- CCSS Level 3 certification. We are SOC 2 Type II certified and complete regular pen testing from ComSec and NCC Group. Fireblocks is also the first crypto tech company to be certified by the

International Organization for Standards in security (ISO 27001), cloud (ISO 27017) and privacy (ISO 27018).
- Offers dedicated Specie insurance through our insurance broker partners.
- $4T+ transactions secured and 130M+ wallets created.

Fireblocks also has a comprehensive treasury management platform with the same properties as their WaaS solution, where they have over $45B in digital assets under custody. However, they are a high-cost solution for big corporations and institutions with 100M in digital assets.

Fireblocks has their own open-sourced and peer-reviewed MPC protocol called MPC-CMP, which does not suffer the same vulnerabilities as GG18, GG20 and CGGMP21 (see below). It is explained in more detail in the section about open-source libraries below.

Self custodial wallets

Bitizen

- Offers a keyless and seedless system:
  - Bypasses seed phrases, focusing on ease of backup and recovery.
  - Avoids the creation, sharing, or storing of private keys via TSS and MPC, employing the Distributed Key Generation (DKG) protocol.
  - Signing transactions is facilitated through the phone combined with the Bitizen server or an alternative secondary device.
- Incorporates biometric authentication for enhanced security.
- Provides one encrypted backup share on Google Drive.
- Compatibility spans a broad range of chains, including but not limited to Bitcoin, Ethereum, BSC, and Polygon.
- Extends support to NFTs on Ethereum and EVM-compatible chains, adhering to ERC-721 and ERC-1155 standards.
- App available for both Android and Apple platforms.
- Security audit conducted by slowmist.

Qredo

- Offers a self-custodial wallet with a focus on security and privacy.
- All funds and transactions are recorded on chain on their Layer 2 Blockchain, Qredo network.
- Are a self described **distributed** MPC protocol, which is not open-source.
  - Qredo secures keys and signatures using a distributed MPC (Multi-Party Computation) network running on TEEs (Trusted Execution Environments) and governs its operations via QredoChain.
  - QredoChain is a purpose-built Tendermint application that uses network consensus to secure state transitions and MPC signatures.
  - This design allows users or developers to represent ownership and define rules (Policies) to govern keys and assets without relying on a central party.
  - Qredo's architecture circumvents typical issues in asset custody, such as centralizing access control in databases, relying on rigid smart contracts, or requiring users to run critical software. Simultaneously, it provides comparable security properties and assurances.
- API to plug into other platforms.

- Qredo Web3 Wallets enable access dApps through both MetaMask Institutional and WalletConnect.
- ISO 22301 and ISO/IEC 27001 certified.

Rainmaker.nyc

- Uses account abstraction to allow smart contracts ownership & control over blockchain accounts. This mechanism aids in facilitating gas-less transactions and simplifying DeFi operations.
- Likely operational since January 2023.
- Offers optional on-device biometric encryption for encrypted shares.
- Documentation is currently a work in progress.
- Supports networks such as Ethereum, Polygon, Avalanche, and Arbitrum.
- Implements a complex share management system:
  - The primary share is stored on an unspecified blockchain and is encrypted with a key accessible via email/password and 2FA through Firebase.
  - A secondary share exists on the user's device and is encrypted on the blockchain. It can be backed up on platforms like iCloud or Google Drive.
  - An accompanying third share and keypair mechanism encrypts a private key on the blockchain, subsequently storing it there.
  - This system might be convoluted and may require additional clarification.
  - Encrypted private shares are pulled from the blockchain using verifier contracts and are decrypted on the device, with access restricted to authorized users.
- Has dedicated apps for Apple and Android devices, with a web version anticipated soon.

Coinbase Retail

- A "semi-custodial" wallet since May 2022.
- No HD wallet functionality.
- Using FROST and DKLS18 schemes and ECDSA and EdDSA signatures.
- User does not have backup and key exports are not supported.
- Accessible via iOS, Android, and web platforms.
- Features gasless transactions
- User is responsible for their key, Coinbase holds the other. Coinbase.com accounts are another custodial solution, where Coinbase holds all keys.
  - Coinbase app offers a convenient and highly recoverable option where the user does not need to hold an encryption key. By verifying their government-issued ID, users can recover their wallet with Coinbase's help. This streamlined process eliminates the complexities of key management and resembles the experience of reclaiming traditional financial accounts with banks. It sets us apart from other wallets, providing top-level cryptographic security and ensured recovery. (Source)

ZenGo

- Launched in 2019, ZenGo is the first self-custodial wallet emphasizing its advanced MPC wallet security architecture.
- No seed phrase vulnerability, boasting over 800,000 users without a single hack or account takeover since inception.
- Aims to enable universal, secure participation in a borderless economy.
- Differentiates itself with aggressive investments in security tools and novel approaches.

- Using FROST schemes and ECDSA and EDDSA signatures with a performance time of 2 seconds.
- Operates with a regionally distributed infrastructure for optimal performance.
- Does not support gasless transactions.
- Incorporates multifactor authentication for enhanced security.
- Trust is paramount as ZenGo retains a share of keys.
- Ensures free recovery without KYC requirements, even guaranteeing access if ZenGo stops its operations (Blog article).
- The core cryptography is open-source, but the application and server codes remain proprietary (Zengo Open Source policy)

Further sources:

- A comprehensive list of crypto wallets based on MPC-TSS (06.2023)
- Comparison of wallet infrastructure by Web3Auth (Note: A bit outdated on some details.)

Comparison of above MPC wallets

There are many MPC wallets available.

Qredo has written a good guideline on what to look for when comparing MPC wallets. The main questions are:

- **Is it database or consensus driven?**

  Database driven means that the wallet is built on a database and not on a blockchain. This means that the wallet is not decentralized and the database can be hacked. Consensus driven means validators must confirm any change of ownership or invoked transaction and thus create an immutable entry on the blockchain. This is the more secure option.

Qredo, Bitizen, Rainmaker use some form of record on the blockchain, whereas ZenGo, Coinbase and Fireblocks keep shares in their database.

- **Who holds sensitive key material and where is it stored?**

  Basically: Is it custodial, semi-custodial or self-custodial? What happens if a share is lost?

Bitizen offers to store a backup on GDrive, Rainmaker stores one share on the blockchain and another in a users cloud. ZenGo and Coinbase only store one share in their database. However, ZenGo has made precautions so that users can recover their funds even if ZenGo stops its operations. Fireblocks stores shares in Secure Hardware Enclaves across multiple cloud centers, which have been hacked before. Every time when the user is responsible to keep a share secure and private, there is higher risk of loosing access to their wallet and therefore loosing funds.

Qredo moved completely decentralised because the sensitive key material will be held by independent blockchain validators. This will create the first trustless and consensus-driven implementation of MPC.

- **Is the protocol peer-reviewed?**

Because real-world practical implementations of MPC are still relatively new, it is important to look at the protocol and the implementation. The protocol should be peer-reviewed, audited and adapt to new findings, vulnerabilities and features (like key rotation). As can be seen below, a few are not adapting quick enough to new findings and vulnerabilities.

Many of them have not been around long (Rainmaker and Web3Auth) and most are not open-source. The most popular and used ones are Coinbase, ZenGo and Fireblocks, where ZenGo and Fireblocks offer insights into their code repositories and provide loads of valuable documentation on their blog. However, most wallets were vulnerable in an attack in March 2023 (see below) and only a few have fixed the vulnerability.

Vulnerabilities in MPC technology

MPC and TSS are not new technology but since they depend closely on cryptography and mathematics, they have to be secure and tested thoroughly.

In a recent example from March 2023, Verichains, a provider of security solutions, has found a key recovery vulnerability of MPC based threshold signature ECDSA based on the GG18, GG20 and CGGMP21 protocols. They named it TSSHOCK and stated that nearly all TSS implementations based on this protocol, including many Golang and Rust open-source libraries are vulnerable.

As the article from Cointelegraph states, "Verichains expects at least $8 billion of value in total assets to be at risk" and recommends all vendors to update their software to the latest version.

16 vendors are affected, including implementations from big companies like Coinbase WaaS, Binance, BitGo and Zengo.

When looking at open-source solutions that are interesting for this work, Zengo must not be used, as they will not address the vulnerability because they deprecated the library in it's entirety.

The libraries TSSlib from BnB Chain and Taurus-TS-ECDSA have fixed the vulnerability.

The open-source library CMP-MPC from Fireblocks is also not infected as it incorporates a different protocol.

Architecture and roles in MPC wallet

For the architecture it is important to understand the roles of the different parties involved in the MPC process.

The architecture consists of these parts:

- A client, which is the user's device with wallet interface, where the user can initiate the DKG and signing process as a background task.
- One or multiple server nodes where the DKG and signing happens. They can be on the cloud or other user devices.

- Encrypted communication.
-

Needs to be validated:

- Another node that watches processes.
- Storage: A database is easy but a blockchain allows for a trustless and immutable record of operations (See here for more details)

Blockdaemon says to have as much parties as the security model needs and no more. The more parties, the more complex it gets. For most scenarios and top-notch security a setting of 2 of 2 or 2 of 3 parties is sufficient, as [Blockdaemon](Blockdaemon-Intro to Threshold Cryptography) is stating. Anything above 20 parties is too inefficient and not recommended.

It is not desired to require all nodes to be available, inter-connected and actively communicating all the time. If the user device has a share on their mobile phone, it is prone to connectivity issues and the user might not be able to sign a transaction. Therefore, it is important to have a system where the user can sign a transaction asynchronously and the nodes can communicate with each other asynchronously.

The user must always sign a transaction, but the other required nodes can be chosen depending on availability.

Technical requirements

There are three main components of a threshold signature process in an MPC wallet:

- Distributed Key generation (DKG)
- Distributed signing
- Key rotation, backup & recovery

1.DKG

A DKG protocol is run on the users mobile device and on the server, which generates two individual encrypted shares. One share is stored on the server and the other is on the users mobile device, preferably in a secure enclave (SE) or Trusted Execution Environment (TEE). SE notably used by Apple are very secure and tested. TEE is found in various devices and with more implementations than just encryption and biometrics like SE, including mobile payments and content protection. However, even the OS cannot access the SE, whereas TEE can be used by many applications.

When a user wishes to access the wallet or execute a transaction, a request is sent to all participants or devices holding key shares. A threshold number of key shares must be gathered to allow access. Various additional authentication measures can be employed for participants. This can include biometrics, PINs, hardware tokens, or multi-factor authentication, ensuring that even if one device is compromised, the attacker cannot access the wallet or make unauthorized transactions.

Like Silence laboratories are doing during signature, "the plugin initiates the signature generation with the phone wallet through a challenge-response protocol embedded in TSS. As such user just needs to approve a push notification on the phone to kickstart the signature generation. The final signature is generated on the plugin side and pushed to the chain."

When DKG happens, address creation and key backup process should be initialised (See more below).

Choose the right threshold MPC protocol

There are various MPC protocols, each with its own advantages and disadvantages. The choice of protocol depends on the desired level of security, performance, and functionality. Each threshold MPC protocol implements different signature schemes, which are detailed in the next section about signing.

Lindell's Two-Party ECDSA:

The protocol allows two parties to jointly produce an ECDSA signature with minimized communication overhead, thereby reducing potential vulnerabilities. It seeks a balance between straightforwardness and robust security. Even if one party deviates from protocol specifications, the overall system remains secure. Requires 8 rounds to sign a transaction.

Optimal for 2-party scenario

DKLs (Doerner, Kondi, Lee and shelat, 2018):

Two parties can jointly sign a message with ECDSA in just over two milliseconds by exchanging just 3 messages, and all the while transmitting about 120 total kilobytes of data. Requires 6 rounds to sign a transaction.

GG18 and GG20 (Gennaro & Goldfeder):

Both protocols are designed to work with ECDSA signatures, where GG20 is the successor of GG18 and fixes some issues. Instead of 2 rounds, it's a three-round protocol that's even more efficient and works better for larger groups. It comes with additional optimizations, making it more efficient in terms of communication overhead.

They are the most used protocols in MPC wallets but as the section about vulnerabilities showed, they were vulnerable in a recent attack. The protocol also has a large communication latency as it requires the user to wait for up to 9 signatures for a transaction.

Fireblocks MPC-CMP:

Fireblocks improved the GG18 protocol to allow the fastest available signature process with only **1 round to sign a transaction**. Furthermore, they added a key refresh every minute, so that adversaries don't have enough time to collect enough shares to compromise the system and allow hot and cold signing with at least one key stored offline. It is also open-source and will never be patented, says Fireblocks. Lastly, it is universally composable (UC), which allows it to be used as a building block for larger systems without compromising security, allowing for modular design of systems. Their Github repository shows options for EdDSA signatures as well, but is written in C++.

CGGMP (2021):

Building on the above protocols, this is a threshold ECDSA protocol for any number of signatories and any threshold, low latency, compatibility with cold-wallet architectures, proactive security, identifiable abort and composable security. Only one round requires the message to be signed and the other rounds happen in the preprocessing stage. It can withstand adaptive corruption of signatories.

Ideal for threshold wallets for ECDSA-based cryptocurrencies.

The Taurus-TS-ECDSA library is based on this protocol. It features ECDSA and FROST signatures and BIP32 key derivation.

Conclusion:

> For 2P use either Lindell or DKLs, depending on what frameworks have integrated it. For nP use CGGMP or Fireblocks MPC-CMP. CGGMP uses Go, whereas MPC-CMP uses C++.

Sources to look into:

- Silence laboratories DKG

2. Signing

When a signature is needed (for transactions, smart contract or dApp) the user signs it with his key fragment and request a partial signature from the other node. The two signatures are merged by the MPC protocol and broadcasted to the blockchain.

Choosing the right signature scheme

Digital signature schemes secure digital transactions by ensuring the integrity, authenticity, and non-repudiation of a message. They are based on elliptic curve cryptography and before implementing any, it should be checked if it is still secure: SafeCurves.

There are various signature schemes:

ECDSA (Elliptic Curve Digital Signature Algorithm)

- The most widely supported curve; prominently used in Bitcoin and Ethereum.
- High security with short key lengths (256 bits ECDSA has same security level as 3072-bit RSA key (NIST recommendation for RSA: 2048-bit)).
- Uses elliptic curves for efficiency and security.
- Threshold signing with ECDSA is much harder than Schnorr and BLS.
- Requires a genuinely random nonce for security, otherwise it can be exploited.
- Was not designed for cryptographic advantage but to circumvent the Schnorr patent, making the signature somewhat mallable.
    - a valid ECDSA signature on the same message with the same key, a property which has induced some occasional trouble (namely, transaction replay attacks in Bitcoin).

BLS (Boneh-Lynn-Shacham)

- Operates on more intricate mathematical structures; might be less efficient than Schnorr.
- More straightforward than ECDSA signing.
- Supports MultiSig.
- Provides the most compact signatures.
- Can aggregate numerous signatures into a single one, optimizing space and computational costs.
- Signature Aggregation: Multiple signatures generated under multiple public keys for multiple messages can be aggregated into a single signature.
- BLS are not quantum-secure and smart contract wallets using BLS aggregation were vulnerable to rogue-attacks, where a malicious actor could create a valid signature for a message without knowing the private key. This was fixed in the BLS12-381.

Schnorr Signatures

Schnorr is a digital signature algorithm that's widely appreciated for its simplicity, efficiency, and flexibility. At its core, Schnorr is essentially a zero-knowledge proof of knowledge of the discrete log of the public key. It natively supports Multi-Sig and is the underlying basis for many cryptographic protocols such as EdDSA and FROST

EdDSA (Edwards-curve Digital Signature Algorithm)

- Compared to ECDSA: Faster with equivalent security and more compatible with TSS.
- Ed25519, its popular variant, employs SHA-512 and Curve25519, utilized in devices like Apple Watch and iPhone, as well as in numerous software libraries.
- Immune to side-channel attacks, where indirect information about the system, such as timing information, power consumption, electromagnetic leaks, or even sound are exploited, rather than directly targeting software or protocol weaknesses.
- Operates without depending on an external randomness source.
- Ristretto permits systems using Ed25519 to smoothly incorporate zero-knowledge protocols.

FROST (Flexible Round-Optimized Schnorr Threshold Signatures)

- FROST brings multi-party and threshold signatures to Schnorr.
- Optimized for round-efficiency, enhancing signing processes in blockchain or distributed systems.
- Simplifies the process compared to ECDSA signing.
- Designed to fend off rogue-key attacks, ensuring added protection in decentralized frameworks.
- Allows a flexible threshold of participants to collaborate and produce a valid signature.
- Supports non-malleability, thus it is not possible to take that signature and generate a new valid signature for another message
- To allow unlimited concurrent operations, it sacrifices robustness, i.e. the ability to work in the presence of malicious adversaries. A single dishonest participant is enough to sabotage the signing process, forcing the process to start over (after kicking out the dishonest participant). This loss of robustness is a theoretical necessity to provide security against t malicious signers, given that robust designs only tolerate up to n/2 attackers.

> Polkadot uses Schnorrkel/Ristretto x25519 ("sr25519") as its key derivation and signing algorithm. Sr25519 is based on the same underlying Curve25519 as its EdDSA counterpart, Ed25519. However, it uses Schnorr signatures instead of the EdDSA scheme (Source).

Comparison:

EdDSA and Schnorr-based schemes are recognized for their performance and versatility, whereas ECDSA, despite its widespread use, has an archaic slow design that encourages security-destroying implementation errors, while the EdDSA family of signature schemes is a modern design that avoids those errors. Any particular instance of ECDSA, such as ECDSA over the curve secp256k1 with SHA-256 (as Bitcoin uses), is incompatible with any other instance of it, such as ECDSA over the curve nistp521 with SHA-512. (Source)

Threshold signing is easier with Schnorr and EdDSA than with ECDSA. Schnorr beat BLS in terms of signature size, public key size, and verification time.

Conclusion:

> Use EdDSA or FROST.

3. Key rotation, backup & recovery

Key rotation

Key rotation is the process of replacing an old key with a new one. It is a security best practice that ensures that a compromised key does not compromise the entire system. It is also a way to ensure that the system is not vulnerable to future attacks. After the transaction or access is complete, the temporary session data is wiped, ensuring that the full private key or any critical information is not left vulnerable.

Backup & Recovery

Choosing how to backup and recover the wallet key is a tradeoff between security and convenience.

Cloud, local or hardware backups lets the user in full control to move / recover their wallet without using the service provider. It is convenient and fast but still prone to loss or theft.

The system must have the capability to add/remove parties for recovery.

Idea: Offline Party for backup and asynchronous authorisation

- **Enhanced Security**: Incorporating offline parties adds another hurdle for potential attackers. They might compromise online signing parties but still need approvals from offline parties for certain actions.
- **Asynchronous Authorization**: Offline parties don't need to be online simultaneously for the protocol execution. They grant approvals asynchronously, catering to scenarios where all participants might not be available simultaneously.
- **Backup and Recovery**: Holding shares of the key, offline parties ensure continuity of operations and recoverability of the wallet, even if an online entity loses its share or is compromised.

- **Risk Diversification**: By dividing roles between online and offline participants, trust is distributed. It ensures that no single entity or group holds undue power or control.
- **Operational Flexibility**: Diverse time zones or connectivity issues can hinder synchronous operations. Offline parties offer a solution by allowing approvals at different times.
- **Compliance and Governance**: In regulated setups, asynchronous authorization by offline parties can meet regulatory or internal audit needs, ensuring operations are vetted by different entities.

Sources:

- Coinbase: Building userfocused web3 wallets blog post

Conclusion: Our solution must have:

Hot / Cold Wallet Support

- Algorithm compatibility with cold storage.
- Consideration of air-gapped offline shares.

Zero-knowledge Proof

- Enforce that parties use the correct input and output for key derivation.
- Guard against malicious value modifications in key generation.
  - *Source: Coinbase MPC Whitepaper*

Performance

- **Transaction Latency**
  - Focus on reducing cycle times and use of pre-processing.
- **Throughput**
  - Determine required Transactions per Second (TPS).
  - Factors: compute resources, system latency, computational efficiency.
  - Optimization of containers, VMs, servers.
  - MPC algorithm and code optimization.
- **Scale**
  - A system must be expressly designed for large-scale operations.
  - Performance trade-offs: More parties can reduce key refresh frequency.
- **Offline Availability**
  - Some MPCs allow asynchronous approvals with offline parties.
  - Not all parties need to be online all the time.
  - *Note on Byzantine Fault Tolerance: A party is byzantine if offline, malfunctioning, or actively sabotaging.*

Audit

- Ensure security within set conditions.
- Security attestation benchmarks include FIPS 140-3.

- Look for third-party assessments like those from NCC Group.
  - *Source: Blockdaemon Blog*

Backup and Recovery

- Secure backup and disaster recovery.
- Support key refresh and standard key derivation.
- Capability to add/remove parties for recovery.

Flexibility

- Balance between security and usability.
- Support for multiple coins/systems and signing algorithms (e.g., Bip32, Bip44).

Authentication

- Mandatory for backup and user recovery.
- Device and wallet security measures: facial recognition needed for transactions.
  - *Reference: Zengo Wallet*

BIP32 compliant wallet

TODO: Can stay here for references but needs to be excluded in the final report.

Functionalities (from Unbound) Source

- **Generate generic secret:** Used to generate an initial secret that is used for BIP derivation.

  - Use DKG so no dealer knows the secret key

- **BIP derivation:** Derive keys in MPC, using the initial secret, according to the BIP32 standard. The result of this step is a key that can be used in ECDSA. Thus, both parties receive the derived public key and hold random shares of the associated private key. Since MPC is used, neither party knows the full private key.

- **ECDSA/EdDSA key generation:** Uses MPC for two parties to directly generate an ECDSA or EdDSA key. The result is a public key known to both parties, and a random sharing of the associate private key. Since MPC is used, neither party knows the full private key.

  - Not just generating and splitting the secret but using MPC to generate the keys themselves without them being ever exposed on a single device.
  - Every participant generates a fragment of the private key.
  - Using a secure distributed protocol, participants combine their fragments to create a public key, without revealing their private fragments to each other.
  - Multiparty Paillier generation is impractical, even 40s for 2P
    - UnboundSecurity used Additively-homomorphic ElGamal in exponent

- **Backup:** The backup method receives an RSA backup public key and generates a publicly-verifiable backup of the parties' shares that is guaranteed to be correct. The procedure uses a zero-knowledge proof, guaranteeing that nothing is revealed about the private key shares, even though it is possible to validate correctness. The backup verification function verifies the zero-knowledge proof and can be used by any entity to ensure that the backup is valid (this function receives the public ECDSA/EdDSA key and so validates that the backup is of the private key of the given public key). Finally, the backup restore method takes the backup information and the RSA backup private key and outputs the ECDSA/EdDSA private key.

  - Signing protocol is not enough, backup is needed to achieve user-friendly experience without sacrificing self-custody but it requires many cryptographic tools (HD derivation in MPC, input and output enforcement, publicly-verifiable backup, and more). Source
  - **Zero Knowledge Proofs:**
    - A prover proves to a verifier that a statement is true without revealing any information.
      - Simplified example: If you want to find a dog in a picture full of cats, I can prove to you that it is in the picture by laying a sheet of paper with a hole on top of it, where you can see the dog through the hole but does not leak any other information, i.e. the cats.
    - Must fulfill three properties:
      - Soundness: If the statement is false, the prover cannot convince the verifier that it is true.
      - Completeness: if the statement is true, an honest verifier is convinced by the prover.
      - Zero-knowledge: no information shall be learned from the proof apart from that the statement is true.
    - Can be used to verify the backup is valid before transferring any funds to an address.
      - Other use cases at unbound security [Note:](1 Proof that a committed value and an encrypted value are the same. 2 Proof that a committed value is in some range of values. 3 Proof of knowledge that a player knows the discrete log of an elliptic-curve point (ZK-DL). 4 Proof that a Paillier public-key was generated correctly (ZK-PPK). 5 Proof that the decryption of a given Paillier ciphertext is the discrete log of a given elliptic curve point (ZK-PDL). 6 Proof that an RSA ciphertext ci encrypts a value xi such that Qi = xi ⋅ G, where Qi is known. This is used for cold backup.)
      - Each party encrypts its share of the key with RSA and proves in zero knowledge that it encrypted the correct share. The flow is as follows: 1. Generate a key (using distributed MPC protocol) 2. Generate backups of shares with zkp 3. Verify proofs and store encryptions 4. Only after all the above: transfer funds to address.
        - Use Fiat-Shamir for public verifiability.

- **ECDSA/EdDSA signing:** This method uses a previously generated key to sign in MPC on a message m that is approved by both. Since MPC is used, neither party can generate such a signature alone, or trick the other into signing on a different message than the one it approves.

  - MPC generates a single signature
  - When a signature is needed, participants collaborate using a secure MPC protocol.
    - Each participant generates a signature fragment.
      - The fragments are combined to produce the final signature, which can be verified with the public key.
  - Ephemeral key k = random()

        - S = sig(msg, k, x,l), where x = private key and l = ECDSA constant
            - With key shares not x and k but x1, x2 and k1, k2

- **Key sharing refresh:** This is used by the parties to jointly generate new random shares of an existing shared key so that the old shares become useless. If an attacker steals a share from one device before a refresh and from the other device after a refresh, it learns nothing about the private key. Thus, it must breach and be resident on both parties before any refresh takes place. This security property is called “proactive” in the academic MPC literature.

    - Key shares refresh at every signature process
        - Example with Schnorr / EdDSA signing
            - The key itself does not change but the shares are changed by running a MPC coin-toss protocol to generate a random string r that is added to one share x1 and subtract r from the other share x2, where x1 + x2 = key x
    - Protocols should choose random and (randomly shifting) key shares and large ranges of potential values to make it impractical to randomly guess By Blackdaemon

Sources:

- Intro to Threshold ECDSA concepts
- List of loads of signature related pages

Libraries to test

MPC library:

Silence Labs

- MPC as a library

    - React native SDK
    - Cloud node SDK

For node deployment:

Multichain

- Node installment
- fastMPC mainnet goes live

Apache Milagro decentralised security infrastructure

Torus node

- A cluster of Torus nodes form the Torus Network, which hold private key shares for Torus users. These private keys are generated via distributed key generation (DKG) and retrieved via OAuth logins (eg. Google, Reddit)

Anyswap

Go

BnB Chain Tsslib

- ECDSA and EdDSA, (BLS builtin)
- Many other networks or projects, Threshold Network(tBTC), Swingy, Multichain,
- Used by IBM - https://github.com/IBM/TSS
- permissively MIT license
- Affected by TSSHOCK but fixed

Taurus TS ECDSA

- BIP32 key derivation
- Parallel processing
- ECDSA and FROST signatures
- parallel processing
- communication rounds: 4 online and 7 in preprocessing phase
- actively maintained (last commit: August 2023)
- Affected by TSSHOCK but fixed

OKX threshold lib

- Improved 2-2 to {2,n} to prevent unrecoverability - See more here
- 2-party ECDSA signature, using Feldman's VSS generate key shares and Lindell 17 protocol for 2-party signature.
- 2-party Ed25519 signature.
- Bip32 key derivation, support key share unhardened derivation, chaincode is generated by n parties.
- Key share refresh, when one party key share is lost or a new participant comes in, support refresh

Rust

Zengo

- **Zengo-X ECDSA** - Multiparty-ecdsa
  - ECDSA
  - Includes GG20
  - Signal messenger (p2p network) instead of broadcast channel
  - **NodeJS wrapper** - https://github.com/builderssquad/node-mpc-ecdsa
    - Under heavy development
    - Last commit Jan 2023
  - Affected by TSSHOCK but will not be fixed
    - No longer maintained
- **Two party signatures JS SDK** - https://github.com/ZenGo-X/thresh-sig-js
- **Bitcoin HD wallet using 2P ECDSA** - https://github.com/ZenGo-X/gotham-city/tree/master
  - It is not clear how to use this

Axelar(tofn) ECDSA Rust implementation of GG20

- Affected by TSSHOCK, no fix yet.

OpenTSS by LatticeX-Foundation - https://github.com/LatticeX-Foundation/opentss

- Only ECDSA
- Offline/Online signing
- Key generation
- Last commit: Sep 2022

C/ C++

Fireblocks MPC-lib

- ECDSA (offline and online) with 1 round to sign a transaction
- EdDSA (offline and online)

Uboundsecurity blockchain-crypto-mpc

- No longer supported
- Good for experimenting

Safeheron threshold ECDSA with GG18, GG20 and CMP protocols implemented

- How to mitigate the attack: add zkps for proving the well-formedness of Alices N In GG18 and GG20.
  - Source
  - Attack

Skalenetwork libBLS

- Actively maintained and used by skale labs
- BLS threshold signatures, DKG and Threshold encryption
- Works with ethereum cryptography
- Signatures can be verified in solidity

Signatures

- Testing ECDSA and EdDSA in plain JS
- Schnorrkel: with js, python

Conclusion:

> Start with BnB Chain Tsslib, Taurus TS ECDSA and OKX threshold lib. Fireblock has the best algorithm but their code is not that straightforward.

Proof of Concept

1. Build a 2 of 2 system
2. Add a third node (MPC-TSS scenario)
3. Add offline node

Glossary

Non-malleability: A property of a cryptographic scheme to prevent adversaries from changing the content of messages/signatures. An example for signatures: A non-malleable signature ensures that an adversary cannot take another signature and create a new valid signature for another message.