Threshold Cryptography in Enhancing the Security and Scalability of DeFi

Alyssa Brittany Chen

Oct. 18, 2023

Contents

1 Introduction

1.1 Abstract

The world of Decentralized Finance (DeFi) has grown exponentially over the past few years, opening avenues for innovation but also bringing challenges in security and scalability. Threshold cryptography incorporates similar encoding processes utilized in classical cryptographic schemes such as RSA and ECC while simultaneously combining secret-sharing-esque decoding. This paper explores the integration of threshold cryptographic techniques in DeFi platforms as a primitive to key generation and signing to address new DeFi use cases as well as limitations of the technology.

1.2 Decentralization and Trust

The essence of DeFi lies in its decentralized nature, where traditional banking systems and intermediaries are replaced by smart contracts and protocols. Trust in DeFi is established not by institutions but by cryptographic proofs and consensus algorithms. While this decentralization offers increased transparency and control to users, it also creates vulnerabilities. If not properly secured, malicious actors can exploit these systems, leading to substantial financial losses.

1.3 Threshold Signature Schemes

1.3.1 Shamir's Secret Sharing and Lagrange

To intuitively go over threshold schemes, it is important to discuss the underlying ideas behind Shamir's scheme. Suppose you are a parent that doesn't want your kids to snack whenever they want, and wish to devise a scheme in which at least $n-1$ out of n kids who want snacks are needed to unlock the snack cabinet. Then, we can say the reconstruction threshold is $k = n$, and construct a degree $k-1$ polynomial $P(x)$, with $P(0) = code$ and distribute shares $P(i)$ to each child. When the required threshold of at least $n-1$ children come together, they can reconstruct the code for the lock and get their snacks! A.1

1.3.2 Overview of TSS

Threshold signature schemes (TSS) divide a private key among multiple parties, ensuring that no single party has access to the entire key. To produce a valid signature, a minimum number of these parties (the threshold) must collaborate. TSS provides an added layer of security as compromising a system would require breaching multiple parties, making attacks more difficult.

In distributed ledger technology (DLT) platforms, digital signatures play a critical role as they enable the identification and authentication of the parties involved in the network. A digital signature scheme consists of the private and public key, where the private key belongs to the owner and is used to sign data, and the public key is used by other parties in the network to validate the signatures of the private key holder.

Since the management of the private key is integral to the validity of the DLT platform, entrusting this responsibility to a single party creates a potential single point of failure that could compromise the system. However, there is a way to thwart this single point of failure by implementing threshold signature schemes, which is a branch of secure multiparty computation to reveal the private key when a subset of parties come together and combine their shares. This cryptographic technnique relies on the idea that multiple parties collaboratively compute a function over their inputs while keeping those inputs private from each other. For example, if Alice and Bob wanted to know who is richer without revealing their individual weaths, MPC would take in their salaries as the private inputs, and output the name of the person with more wealth. This is done so in a manner that keeps the input's of each part secret, allowing individuals who don't necessarily trust each other to perform operations without revealing sensitive information.

A secure protocol in a given threshold scheme should achieve three main properties:

1. Correctness: Each party receives the correct output as the result of the computation.
2. Privacy: Each party receives the correct output as the result of the computation.
3. Fairness: All parties should be guaranteed to receive the output.

Typically, the threshold scheme consists of the key generation stage followed by the signing and verification stages.

1.3.3 BLS Scheme

The Boneh Lynn Shacham (BLS) scheme is not inherently a threshold scheme, but is commonly used as an underlying infrastructure to contruct one. More details on the underlying mathematical implementation in A.2. This section covers a conceptual, high level understanding of the BLS scheme.

Aggregation: A special property the BLS scheme grants is the ability to condense multiple signatures into one, making it possible for increased efficiencies in storage, computational costs, and many more. For example, for m distinct signers and n signatures, the signatures can be aggregated by setting $\sum = \sum_i \sigma_i$. Alice will then broadcast σ_i and aggregate the partial signatures (after recieving t+1 shares of the partial signatures) to generate the threshold signature, $\sigma = \sum_{j \in S} \lambda_j \cdot \sigma_j$.

Why is this important? For example, as Ethereum made the shift from a Proof-of-Work to Proof-of-Stake model, the task of verifying signatures efficiently became an issue. When a block is proposed on the Ethereum blockchain, many other validators must also sign that block in order to attest it– Ethereum currently has over 800k validators as of September 2023, therefore verification and scalability could pose a significant problem. However, because of the BLS scheme's aggregtion property, we can aggregate multiple different signatures for the same message into one and acheieves faster verification, as we still only need two pairings are required for verification. [maybe rewrite this to be more concise]

1.3.4 FROST Protocol

In the current landscape of DeFi, many threshold schemes are rapidly developing and evolving. One such protocol is the FROST (Flexible Round-Optimized Schnorr Threshold signature scheme) protocol, the threshold counterpart of the Schnorr signature scheme.

Signatures produced by the FROST method and Schnorr signatures are indistinguishable and similarly verifiable per the standard verification process.

On a high level, FROST consists of either (1) a two round signing protocol or (2) a single round protocol with preprocessing [prob explain this more], and unlike similar schemes, signing operations achieve more security from concurrently performed actions. FROST can be implemented with a Distributed Key Generation (DKG) Protocol, an n-wise Shamir Secret Sharing protocol in which each participant takes the role of a dealer. The protocol also favors efficiency by allowing honest participants to rerun the protocol after identifying and excluding the misbehaving participant.

The authors of the original FROST whitepaper instantiates the protocol using an optionally implemented signature aggregator role ($\mathcal{SA}$), delegated to report dishonest parties and publish the signature– allowing for optimized communication overhead and increased practicality. Additionally, even if up to $t-1$ participants, including the $\mathcal{SA}$ are misbehaving, the infrastructure of FROST guarantees that without the approval of at least one honest party, the signature cannot be published and verified. For details on the more specific implementation of the protocol, refer to section A.3.

1.4 Use Cases, Scalability & Security Improvements

In a multisignature scheme, two or more participants who each hold their own public key are usually required to sign off on a transaction which can be a lengthy process. Using a threshold implementation, m number of parties hold shares of the public key, but only up to $m-1$ partcipants are required to sign off on the transaction. Because threshold signature schemes are able to be computed and operate off chain, they are more cost effective than the multisig approach, which relays each transaction separately to the blockchain as compared to the threshold scheme, which delivers each contribution as a single transaction. At the time of writing, many companies such as Coinbase, Binance, and Bitgo are working on incorporating MPC protocols into their wallets infrastructure.

Schnorr signatures (which are used in the FROST protocol) allow for an anonymity characteristic, making it more difficult for an attacker to distinguish between single and multisig spends when observing on-chain activity and for observers to determine which participants signed a transaction and which did not. Similarly to the BLS scheme, Schnorr signatures also have an aggregation property, and compared to the currently used Elliptic Curve Digital Signature Algorithm (ECDSA), Schnorr signatures are more suitible for cryptocurrency and DeFi infrastructure as they require fewer computations and allow for faster efficiency with smaller signature sizes and faster verification processes.

In addition to the active security measures present in the underlying structure of threshold schemes, there are additional mechanisms that can be implemented in order to achieve

proactive security. A secret is distributed amongst n members, and $t+1$ shares are needed to uncover the secret, so a successful attacker must breach at least $t+1$ shares. If we implement a key refreshing mechanism and refresh secret shares after a certain period, assuming an attacker successfully breaches a percentage of shares in one period, and breaches the remaining in the next, they are unable to uncover the secret. This has an especially interesting use case in multisignature (multisig) wallets.

If we update the shares of a multisig wallet periodically, this introduces a moving target for attackers. Even if they compromise a fraction of the shares within one time frame, the refresh process would invalidate these efforts in the next period. This method not only increases the complexity of a successful attack but also provides a window of opportunity for detecting and responding to breaches before the secret/access to the wallet can be reconstructed. In practice, this is more suitable for maintaining continuous security and ongoing trust among participants in the system.

This security approach can be particularly effective in environments where the threat landscape is dynamic and the value of the protected asset (like cryptocurrency in a multisig wallet) is high, and the appliations of this approach is infinitely dynamic– the multisig wallet is just one of many use cases.

The tBTC is a project that integrates with Base, a Layer 2 blockchain incubated by Coinbase aimed to lower costs and improve scalability in DeFi using threshold cryptographic techniques. Read more here: Navigating the Future of DeFi: tBTC Launches on Base. Another use case utilizes threshold cryptography to aid in social recovery for Web3 wallets by breaking users' private keys into multiple shards, and allowing for recovery with a certain threshold of those shards.

1.5 Risks, Challenges & Limitations

Just as any given cryptographic protocol, it is essential to provide sufficient mathematical proofs to demonstrate the MPC protocol to operate as desired. However, given the complex nature of the technology, this requires a high level of mathematical maturity and understanding. Additionally, the protocol must not only be designed correctly, but also have an efficient technological implementation. [can explain this a bit more] Finally, to address the funcitonal perspective of an MPC protocol, especially in terms of security factors, several considerations must be taken into account. For example:

1. Scalability- How does the protocol scales with the number of participants? Is there a trade-off between the number of participants and the protocol's efficiency or security?
2. Recovery mechanism- What recovery mechanisms are in place if a computation is disrupted? (this includes data redundancy and protocol restart capabilities).
3. Threshold- Up to how many malicious parties can the protocol tolerate? The minimum number of parties needed to collaborate to perform computations or to reconstruct the secret?
4. Privacy- Can the protocol ensure that no information other than the computation

outcome is leaked, even if some participants are compromised?

References

[A] G. Tillem, O. Burundukov. Threshold Signatures using Secure Multiparty Computation

[B] Chelsea Komlo, Ian Goldberg. FROST: Flexible Round-Optimized Schnorr Threshold Signatures

[C] Ahmet Ramazan Agirtas, Jorge Arce-Garro, Yevgeny Zaytman, Jelilat Anofiu. Threshold Signature Schemes

[D] Omer Shlomovits, ZenGo. Threshold Signatures Explained

[E] Cassandra Heart, Arash Afshar. Threshold Digital Signatures

[F] Chainlink. What Is a Schnorr Signature?

[G] Panther. Threshold Cryptography, MPC, and MultiSigs: A Complete Overview

[H] Adi Shamir. How to Share a Secret

A Appendix

A.1 Shamir

Shamir's Secret Sharing is based off polynomial interpolation over a Galois (or finite) field $GF(q)$, where $q > n$ shares being generated. Since a polynomial is uniquely determined by degree $k+1$ points, the secret a_0 is able to be reconstructed from k or more shares of $\mathcal{S}$ using the formula for Lagrange interpolation $a_0 = f(0) = \sum_{i=0}^{k} y_i \prod_{i=j} \frac{x-x_j}{x_i-x_j}$. For further reading: Shamir's Secret Sharing: Explanation and Visualization is a quick read that offers great visuals.

A.2 BLS Definitions

1. e is a bilinear pairing function which maps $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_3$ are groups of prime order p and satisfies two properties:
    a. bilinearity– multiplication in one group corresponds to exponentiation in the target group.
    b. non-degeneracy– there exist elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ such that their pairing is not the identity element in $\mathbb{G}_3$.
2. g_1, g_2 are generators of $\mathbb{G}_1, \mathbb{G}_2$, respectively. A generator is an element such that repeated application of the group operation to it generates every element of the group.

3. h is an output of hash function $\mathcal{H}$, which maps arbitrary length input message $\mathcal{M}$ onto group $\mathbb{G}_1$. The hashed value of the message is then used in the signature generation and verification process.
4. pk, the public key is computed as $pk = sk \cdot g_2$, where sk represents the secret key and g_2 is a generator of $\mathbb{G}_2$.
5. σ, the signature is computed as $\sigma = sk \cdot h$.

Key Generation:

- Partial signature computed by $\sigma_i \leftarrow s_i$
- Secret key uniformly at random
- Computing public key

Signature Generation:

- Hashing the message
- Computing the signature

Verification Phase:

- Given a message $\mathcal{M}$ and signature σ, verify and accept the signature if $e(h, pk) = e(\sigma, g_2)$

A.3 Schnorr Algorithm

- Given a message m, secret key $s \in \mathbb{Z}_q$, and public key $Y = g^s \in \mathbb{G}$:
  1. Signer $(x, Y) \leftarrow$ KeyGen()
  2. Verifier asks to sign message in respect to their public key (m, Y)
  3. Generate nonce $k \leftarrow \mathbb{Z}_q$, [also add in the money symbol] public commitment $R = g^k \in \mathbb{G}$, challenge $c = H(R, Y, m)$ where H is a hash function, and response $z = k + c \cdot x$.
- Signature tuple $(m, \sigma = (R, z))$
- To verify signature:
  1. Verifier rederives challenge from computing $c = H(R, Y, m)$.
  2. Using the public key, response, and challenge compute $R' = g^z \cdot Y^- c$ [Fix this]
  3. Output $R =^? R'$