# Private Key Sharing for Threshold Signatures In Bitcoin

Juan Pablo Crespo Vargas

Departament of Information and Computer Science ICS
King Fahd University of Petroleum and Minerals KFUPM
g202393610@kfupm.edu.sa

***Abstract*— This report presents the design and evaluation of an ideal threshold ECDSA signature scheme for Bitcoin, combining Paillier cryptosystem-based secure computations with custom elliptic curve operations and Shamir's secret sharing. The scheme ensures efficient distributed signing and compatibility with Bitcoin's ecosystem while providing robust protection against adversaries. Testing confirmed secure key distribution and verifiable signature creation under asynchronous communication. The model's central coordinator simplified the implementation but can be replaced with fully distributed computations.**

***Keywords—Bitcoin, Paillier, Distribution Keys, Eliptic Curve Digital Signature Algorithm.***

## I. INTRODUCTION

Bitcoin has become a cornerstone of the global financial ecosystem, recognized both for its technological innovation and its ability to operate as a store of value in a digital environment. This digital asset, which redefined the concept of private property by making it absolute, was introduced to the world in 2008 through a white paper written by Satoshi Nakamoto [1]. It introduced a system for transferring value without relying on trusted intermediaries. Sustained by fundamental technologies such as SHA-256, PoW, and ECDSA, Bitcoin relies on several core principles and mechanisms to ensure its security, integrity, and functionality. One of Bitcoin's most important technological pillars is the SHA-256 hashing algorithm, which is used both to generate addresses and to ensure the integrity of the blocks. The Proof of Work (PoW) protocol ensures that blocks are validated through a competitive process of finding a nonce that meets the difficulty requirements set by the network. These mechanisms work together to ensure Bitcoin's resistance to censorship and manipulation.

Another essential component of Bitcoin is the use of digital signatures based on the Elliptic Curve Digital Signature Algorithm (ECDSA). To generate these signatures, a private key is first created, which is a secret random number granting the owner absolute control over the associated bitcoins. From this private key, a public key is derived using mathematical operations on elliptic curves; this public key is then converted into a payment address. The digital signature, generated with the private key, serves to authenticate transactions and demonstrate ownership of funds without revealing the private key, thereby ensuring the integrity and privacy of each operation. In the context of Bitcoin, possessing a private key equates to owning the bitcoins associated with it, as this key enables the generation of digital signatures to authorize transactions.

However, this reliance on the private key also constitutes a critical point of failure. Losing or compromising the private key can result in the irreparable loss of funds, underscoring the importance of effective protection. A commonly used solution to mitigate this risk is the use of multi-signature (multisig) transactions, where multiple keys are required to authorize a transaction[1].

Although multisig transactions significantly enhance security, they also present practical limitations. Bitcoin blocks have a maximum size of 1 MB and are generated approximately every 10 minutes, which means storage capacity is limited and costly. Multisig transactions occupy considerably more space within blocks compared to conventional transactions, increasing commission costs. The accompanying image illustrates how different transactions, including multisig ones, can vary in size within a block, highlighting the implications in terms of storage and costs. In this context, threshold signatures emerge as an innovative and more efficient solution. Based on Shamir's scheme for dividing the control of a private key, this approach allows a group of participants (t+1 out of n) to sign a transaction without any of them holding the complete private key. The key difference between a multisig scheme and a threshold scheme lies in how they manage keys: in a multisig scheme, each participant uses their own private key to generate a partial signature, whereas in a threshold scheme, several secrets are combined to form a single shared private key. This shared key enables participants to collaborate and produce a valid signature using the contributions of at least t+1 participants out of n, without any of them knowing the complete key.

This work proposes a distributed signing scheme using threshold signatures in an ideal environment, highlighting its potential to improve the security and efficiency of private key management.

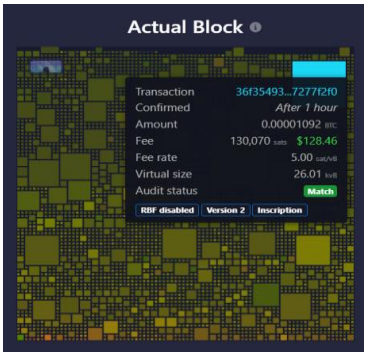


*Image showing the space ocupied by every transaction tx in one block*

In this work, we propose an efficient and practical threshold signature scheme specifically designed to secure Bitcoin keys. Unlike previous schemes, which required a larger number of participants to generate a signature or had significant inefficiencies, our approach minimizes the required servers to $n \geq t+1$ and ensures that signatures can be generated with only $t+1$ participants. Additionally, our protocol operates with a constant number of interaction rounds, constant computation time per participant, and minimal storage requirements. These improvements make our scheme both scalable and compatible with Bitcoin, providing a robust solution for distributing signing power without compromising security or usability.

## II. METHODOLOGY

This section outlines the key components and mechanisms of the proposed threshold signature scheme. We begin by describing the communication model that ensures efficient and secure interactions between participants. Next, we categorize potential adversaries and analyze their capabilities within the system. Following this, we define the signature scheme, the threshold signature approach, and the measures implemented to ensure security and resilience against attacks, following the sugestion by Genaro et Al. [3]

### Communication Model

- Participants communicate through a peer-to-peer (P2P) network model, ensuring a direct connection between all nodes in the system. Figure 2 illustrates how this model operates.
- There is a free and public communication channel that all participants can access, used to share relevant information transparently and ensure synchronization during the protocol.

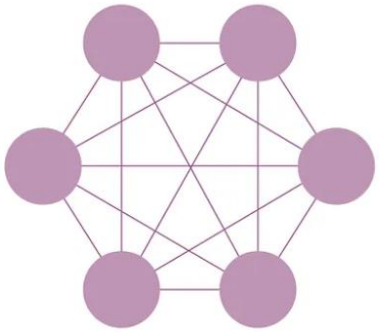

**Image 2. P2P model of communication for 6 participants**

### Adversaries

In our model, we consider two main types of adversaries:

1. **Honest but curious**: These adversaries strictly follow the protocol but attempt to gather additional information about other participants. Their goal is to exploit any data leakage within the established protocol limits.
2. **Malicious**: These adversaries do not follow the protocol and actively seek to compromise the scheme, either by altering the expected system behavior or attempting to generate invalid signatures.

The proposed model is partially synchronous, meaning there is a reasonable limit on communication delays, but perfect synchronization is not guaranteed. In this context, we will demonstrate how to mitigate and limit access to information in both an ideal model and a realistic environment.

### Signature Scheme

Our signature scheme is defined through a set of functions , where:

- **Key-Gen**: Generates a pair of public and private keys .
- **Sig**: Uses the private key to generate a digital signature from the hash of a message.
- **Ver**: Verifies the validity of a signature using the public key and the message.

A signature scheme is considered unforgeable if no adversary, even with access to the public key and signatures of previously chosen messages, can generate a valid signature for a new message with non-negligible probability depending on the security parameter . Here, represents the key length or cryptographic security level, directly influencing the likelihood of a successful attack. The larger , the lower the probability of compromise.

### Threshold Scheme

The threshold scheme divides the private key into parts, assigning one part to each participant, such that:

- Knowing up to of these parts provides no information about the original value nor enables reconstruction.
- Any set of t parts can be used to reconstruct efficiently.

The threshold signature scheme $S$ =(Key-Gen, Sig, Ver) in our proposal includes:

1. **Thresh-Key-Gen**: A distributed protocol that generates the key pair from n total contributions by the players, where the private key is divided into n parts following Shamir's threshold scheme. At the end of the protocol, each player obtains a private part, and all players know the public key.
2. **Thresh-Sig**: A distributed protocol where t+1 players at most collaborate using their parts and random numbers to generate a signature on a message, without revealing the complete private key or the total random number used.

The verification of threshold signatures uses the same verification algorithm as a traditional signature scheme, taking as input the hash of the signed message, the public key, the point to check, and the signature.

### Security in Threshold Schemes

A threshold scheme is considered secure if it meets the following properties:

1. **Unforgeability**: No malicious adversary corrupting up to players can generate a valid signature for a previously unsigned message with non-negligible probability. Here, is the security parameter determining the cryptographic strength of the scheme, with a lower probability of successful attack as increases.
2. **Secret Privacy**: The parts of corrupted players do not reveal meaningful information about the complete secret. Honest but curious participants attempting to gather another participant's secret would only obtain it in encrypted form, making it useless for reconstructing the secret.

This scheme ensures that, in both ideal and realistic scenarios, the security and privacy of private keys are protected against honest but curious and malicious adversaries.

### Description of the Ideal Environment

For the initial implementation of our threshold signature scheme, we operate within an idealized environment to evaluate its feasibility. This setup includes a trusted external participant who acts as a central coordinator. By definition, this coordinator is assumed to be honest and cannot behave maliciously, simplifying the scheme's deployment for testing purposes. In a real-world scenario, where a central coordinator cannot be trusted, the protocol would run entirely distributed across the participants' servers. For demonstration purposes, we utilize the Paillier cryptosystem, which supports homomorphic operations, to perform secure computations.

### Key Generation Process

The Paillier cryptosystem is based on modular arithmetic and supports additive homomorphic encryption. Its key generation process includes:

1. Selecting two large prime numbers and .
2. Calculating .
3. Defining the public key and the private key .

This cryptosystem ensures that encrypted values can be securely added without decryption, enabling collaborative computations.

In our configuration:

1. The coordinator generates the public and private key pair for the Paillier cryptosystem.
2. The coordinator shares in the public channel, along with the agreed values of (number of participants) and (threshold).
3. Each participant generates their own random value, encrypts it using the shared , and sends the ciphertext to the coordinator via the public channel.

In a real-world scenario, the Paillier key generation would be distributed among the participants. This ensures that neither honest nor malicious adversaries can gain any useful information about the shared values.

### Threshold Secret Sharing

Once the coordinator receives encrypted random values from all participants, the following steps are performed:

1. The coordinator uses the additive homomorphic property of Paillier to securely sum the encrypted values, creating an aggregated ciphertext. This step could also be replicated in a distributed manner by the participants in a real-world implementation.
2. The coordinator decrypts the aggregated ciphertext using to obtain a private key .
3. The corresponding public key is derived from using the secp256k1 standard from Bitcoin.
4. Using Shamir's secret sharing scheme, the coordinator divides into parts with a threshold , ensuring that any parts can reconstruct , while or fewer parts reveal no information.
5. Each participant receives their secret share via the P2P channel.
6. The coordinator publishes the public key in the public channel.

In a real-world scenario, participants can encrypt their secret shares with a unique public key before receiving them if the P2P channels are considered vulnerable.

### Signature Generation

With the secret shares distributed, the signature generation process proceeds as follows:

1. Participants compute the hash of the message to be signed.
2. Each participant generates a random value and sends their secret share and their (both encrypted with Paillier) to the coordinator.
3. The coordinator reconstructs the private key and combines the encrypted random values from all participants homomorphically to compute a new aggregated random value.
4. After decrypting the aggregated random value, the coordinator uses it and to compute the signature based on the threshold signature scheme.
5. The final signature is publicly shared via the public channel.

**Proposed Scheme**

**Description of the Ideal Environment**

For the initial implementation of our threshold signature scheme, we operate within an idealized environment to evaluate its feasibility. This setup includes a trusted external participant who acts as a central coordinator. By definition, this coordinator is assumed to be honest and cannot behave maliciously, simplifying the scheme's deployment for testing purposes. In a real-world scenario, where a central coordinator cannot be trusted, the protocol would run entirely distributed across the participants' servers. For demonstration purposes, we utilize the Paillier cryptosystem has demostrated good performance[2], which supports homomorphic operations, to perform secure computations.

**Key Generation Process**

The Paillier cryptosystem is based on modular arithmetic and supports additive homomorphic encryption. Its key generation process includes:

1. Selecting two large prime numbers p and q.
2. Calculating n=p*q.
3. Defining the public key and the private key.

This cryptosystem ensures that encrypted values can be securely added without decryption, enabling collaborative computations.
In our configuration:

1. The coordinator generates the public and private key pair for the Paillier cryptosystem.
2. The coordinator shares in the public channel, along with the agreed values of n (number of participants) and t (threshold).
3. Each participant generates their own random value, encrypts it using the shared public key pk, and sends the ciphertext to the coordinator via the public channel.

In a real-world scenario, the Paillier key generation would be generated and distributed among the participants. This ensures that neither honest nor malicious adversaries can gain any useful information about the shared values.

**Threshold Secret Sharing**

Once the coordinator receives encrypted random values from all participants, the following steps are performed:

1. The coordinator uses the additive homomorphic property of Paillier to securely sum the encrypted values, creating an aggregated ciphertext. This step could also be replicated in a distributed manner by the participants in a real-world implementation.
2. The coordinator decrypts the aggregated ciphertext using to obtain a private key.
3. The corresponding public key is derived from using the secp256k1 standard from Bitcoin.
4. Using Shamir's secret sharing scheme, the coordinator divides the private key into parts with a t threshold, ensuring that any parts less than t+1 can reconstruct while or fewer parts reveal no information.
5. Each participant receives their secret share via the P2P channel.
6. The coordinator publishes the public key in the public channel.

In a real-world scenario, participants can encrypt their secret shares with a unique public key before receiving them if the P2P channels are considered vulnerable.

**Signature Generation**

With the secret shares distributed, the signature generation process proceeds as follows:

1. Participants compute the hash of the message to be signed.
1. Each participant generates a random value and sends their secret share and their (both encrypted with Paillier) to the coordinator.
2. The coordinator reconstructs the private key and combines the encrypted random values from all participants homomorphically to compute a new aggregated random value.
3. After decrypting the aggregated random value, the coordinator uses it and to compute the signature based on the threshold signature scheme.
4. The final signature is publicly shared via the public channel.

## III. Experimental Settings

**Implementation Details**

The implementation of the proposed threshold signature scheme and protocol was developed entirely in Python and is available at:
https://github.com/jpcrespo/thresholdsign_ECDSA

For this project, we utilized the Python library **PHE** to quickly integrate the Paillier cryptosystem, enabling homomorphic encryption. However, the code for elliptic curve operations, specifically the secp256k1 standard, and Shamir's secret sharing was implemented from scratch to ensure full control and understanding of the underlying algorithms.

**Simulation Environment**

The system simulates communication between participants via peer-to-peer (P2P) channels and a public channel. These communications are represented as text output in the terminal for simplicity and debugging purposes. The implementation includes:

1. **Coordinator and Participant Classes**:
    - A class for the central coordinator, responsible for managing the key

generation, secret distribution, and signature creation.
    - A class for each participant, handling local computations and interactions with the coordinator.
2. **Configurable Threshold Scheme**:
    - The protocol supports an configuration, where represents the total number of participants, and defines the threshold for collaborative signing. These parameters are defined prior to the protocol's execution.
3. **Simulated Channels**:
    - The public channel is used to share non-sensitive information, such as public keys and aggregated results.
    - P2P channels simulate secure private communication between the coordinator and each participant, ensuring isolation of secret shares.

This setup provides a practical environment to test and validate the functionality of the threshold signature protocol, while also demonstrating how communication and computation are structured in a distributed system. Extensive tests were conducted with different sets of participants and varying thresholds, resulting in:

1. Successfully generating valid signatures with distributed participants collaborating.
2. Ensuring that previous signatures of a message, along with the public key and up to distinct secret shares, do not allow the reconstruction of the signature or reveal additional information about the private key.
3. Guaranteeing that the public key in this scheme cannot be inferred unless participants collaborate to reconstruct the private key.

## IV. Conclusions

### Summary of Findings

This work demonstrated the theoretical feasibility of implementing an ideal ECDSA threshold signature scheme for Bitcoin. Through the proposed implementation, several key objectives were achieved:

1. The scheme successfully generates valid signatures with distributed participants collaborating, ensuring the threshold requirements are met.
2. It maintains temporal and spatial complexity bounded to asynchronous communication among participants, allowing efficient execution even in distributed environments.
3. Security against various types of adversaries was validated, provided that Paillier keys and secret shares remain secure. Neither the public key nor the private key can be inferred by adversaries, even with access to prior signatures and up to secret shares.
4. While this work relied on a central coordinator to simplify the simulation and testing environment, it has been conceptually demonstrated that the protocol can operate in a fully distributed manner, eliminating the need for a trusted coordinator in real-world applications.

The compatibility of the scheme with Bitcoin's existing infrastructure suggests its potential for adoption in real-world scenarios to enhance wallet security without disrupting the ecosystem.

### Perspectives and Future Work

Using insights from the referenced paper by Boneh et al. [4], several enhancements to this scheme can be envisioned through the incorporation of Level-1 Fully Homomorphic Encryption (FHE). This approach can reduce the round complexity of the protocol, significantly improving its efficiency in practical scenarios where network latency is a concern.

- **Reduction in Round Complexity**: By leveraging Level-1 FHE, the signing process could be streamlined to fewer rounds compared to current implementations. This would address latency issues in distributed environments, as highlighted in the referenced work.
- **Improved Security and Flexibility**: Level-1 FHE enables computations over encrypted data with enhanced security while maintaining efficiency. This could strengthen the security guarantees of the proposed scheme, especially under scenarios involving higher adversarial threats.
- **Parallelization Opportunities**: The modularity and reduced interaction requirements introduced by FHE facilitate better parallelization of operations, thereby reducing the overall computation time in large-scale deployments.

The adoption of these enhancements could further bridge the gap between theoretical feasibility and practical utility, positioning the threshold signature scheme as a robust solution for Bitcoin wallet security and beyond.

## References


[1] Hamidi, A., & Ghodosi, H. (2022, December). Efficient Distributed Keys Generation of Threshold Paillier Cryptosystem. In *International Conference on Information Technology and Communications Security* (pp. 117-132). Cham: Springer Nature Switzerland.

[2] Jia, L., Chen, X., Liu, L., Wang, X., Xiao, K., & Xu, G. (2023). Blockchain data secure sharing protocol based on threshold Paillier algorithm. *High-Confidence Computing*, *3*(4), 100132.

[3] Gennaro, R., Goldfeder, S., & Narayanan, A. (2016). Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In *Applied Cryptography and Network Security: 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings 14* (pp. 156-174). Springer International Publishing.

Boneh, D., Gennaro, R., & Goldfeder, S. (2017, September). Using level-1 homomorphic encryption to improve threshold DSA signatures for bitcoin wallet security. In *International Conference on Cryptology and Information Security in Latin America* (pp. 352-377). Cham: Springer International Publishing.