- by Hankyung Ko(@HankyungKo) and Chanyang Ju(@wooju), Researcher at

Radius

. Thanks to

Tariz

and

AJ

for reviewing this post.

- Your feedback and opinions are highly valued. For the original content, please visit

our blog

.

1. Introduction

Radius is at the forefront of enhancing rollup composability through the development of ‘Shared Sequencer’. Enhancing composability between rollups allows users to access more opportunities, such as arbitrage, enhancing the blockchain ecosystem’s fluidity and potential for innovation. Service providers benefit as well, as they can manage multiple app-specific rollups and support seamless operations, scaling their services effectively. Our research indicates that the ‘Shared Sequencer’ is foundational to achieving the vision of inter-rollup composability. The ‘Shared Sequencer’ ensures atomic inclusion of transactions across multiple rollups or between Layer 1 (L1) and Layer 2 (L2) networks, representing a critical infrastructure component in our system.

The ‘Shared Sequencer’ must be a neutral entity, more so than any other within the ecosystem. As the entity responsible for constructing blocks across multiple rollups, it must operate without bias, adhering strictly to predefined rules. The essence of being the most neutral sequencer means that the sequencing of blocks must be free from any hidden agendas, creating a trustable system where blocks are produced based solely on these established rules. This neutrality is paramount to prevent any potential for manipulation or unfair advantage within the network.

Many of the security concerns in blockchain are addressed through cryptography or crypto-economic (optimistic) solutions, with the latter being effective when actions can be retrospectively verified for legitimacy. However, the role of the sequencer presents unique challenges; if a sequencer acts maliciously, such as censoring transactions or engaging in front-running attacks, these actions are not easily identifiable after the fact. Therefore, the development of a neutral shared sequencer necessitates a cryptographic approach. We see potential in an ‘encrypted mempool,’ where the encryption of transaction details hinders the ability of malicious actors to censor, reorder, or manipulate transactions for their benefit, fostering a more secure and trustworthy blockchain environment.

1.1. Delay Encryption within an Encrypted Mempool

We have concentrated on designing an encrypted mempool using ‘Delay encryption,’ a cryptographic algorithm that requires a predetermined amount of computation time to obtain the decryption key, utilizing timelock puzzles. This design ensures that the sequencer can commit to the transaction order before reaching the timelock parameter, thus enforcing an unbiased ordering process. The use of delay encryption introduces a mechanism where transactions are secured and ordered without the possibility of tampering or bias, establishing a fair and transparent sequencing process.

There are currently two main methods to create a censorship-resistant sequencer using delay encryption: Radius’s original Practical Verifiable Delay Encryption (PVDE) and Scroll’s Multiparty Delay Encryption (MDE). Both methods have their trade-offs. PVDE achieves complete trustlessness, allowing users to trust that their transactions are securely included in a block without relying on any external entities. However, this comes at the cost of higher computational expenses for both users and sequencers. On the other hand, MDE reduces computational costs for users and sequencers by introducing multiparty key generation, but this necessitates a 1-out-of-N trust assumption and, as our analysis shows, incurs significant gas fees for storing keys publicly with each block.

This article introduces Radius’s new encrypted mempool model, Single Key Delay Encryption (SKDE), which inherits the advantages of MDE while addressing its two main drawbacks: the trust assumption and high gas fees. We will briefly discuss how PVDE and MDE achieve their respective trade-offs and then delve into the innovative aspects of our SKDE model. The article will also present experimental results comparing the three models, showcasing the effectiveness and efficiency of SKDE in enhancing the security and trustworthiness of blockchain transaction sequencing.

2. Background

Delay encryption is a cryptographic tool that employs time-lock puzzles to enforce a predetermined delay on the availability of a decryption key. This method ensures that encrypted data cannot be decrypted until a specified amount of computational work, often measured in time, has been completed.

As its core, delay encryption involves two main phases: encryption and timed decryption. During the encryption phase, data is encrypted using a standard cryptographic algorithm, and a time-lock puzzle is generated. This puzzle is constructed in such a way that solving it requires a predictable amount of computational effort, effectively creating a “delay” before the information can be accessed.

2.1. Radius’ Original PVDE

[

pvde

1920×980 77.2 KB

](https://ethresear.ch/uploads/default/original/2X/5/5b65fa0efb6b614032462656e116a754e20a36b5.jpeg)

Radius’s original Practical Verifiable Delay Encryption (PVDE) is a sophisticated approach to securing transactions through the use of symmetric key encryption. This method places the onus of encryption key generation directly on the user, effectively eliminating the need for trust assumptions. This trustless model is pivotal in ensuring that users retain complete control over their transaction privacy without depending on external validators or third parties.

Nevertheless, the implementation of PVDE brings to the fore certain computational challenges, particularly attributed to its reliance on zero-knowledge proofs (ZKPs) and the requirement for sequencers to solve timelock puzzles for each transaction. The computational overhead is twofold:

1. Zero-Knowledge Proofs

: Users are required to conduct ZKP proving to validate their encryption keys and the integrity of the encryption process. While ZKPs offer a powerful means to achieve privacy and security by enabling the verification of information without revealing the underlying data, generating these proofs entails a significant computational effort on the part of the users.

1. Timelock Puzzle Solving by Sequencers

: Beyond the ZKP-related overhead, sequencers face an additional, substantial computational burden due to the necessity of solving a timelock puzzle for each transaction. Given a timelock parameter set to, for instance, 100 milliseconds, a sequencer operating on a single thread could decrypt fewer than 10 transactions per second. This limitation poses a significant bottleneck, severely impacting the system’s usability and scalability.

Furthermore, PVDE employs the zk-friendly Poseidon encryption algorithm for its encryption scheme. While Poseidon is lauded for its compatibility with zero-knowledge proof systems, it is inherently more computationally intensive than more conventional encryption algorithms. This additional complexity contributes to the decryption process’s overall computational demands on the sequencer.

2.2. Scroll’s Public Key Delay Encryption with Multiparty Key Generation

[

mde

1920×1021 121 KB

](https://ethresear.ch/uploads/default/original/2X/9/989b498b598377ee23d05d57a556dcb742d5af6f.jpeg)

Unlike PVDE, which relies on symmetric key encryption, Scroll’s MDE utilizes asymmetric key encryption. This shift brings about a new approach to generating encryption and extraction (enc, ext)

key pairs. Rather than being generated by the user, these key pairs are created by a committee, which operates under the assumption that at least one member is trustworthy. Once the (enc, ext)

key pair is made public, the decryption key can be extracted from the ext

key after a specified amount of computational effort, equivalent to a set period.

For sequencers, this model simplifies the process significantly since only one timelock puzzle needs to be solved

, making the timelock parameter more manageable. Furthermore, users are not required to generate ZKP proof for their transactions, and the system does not rely on zk-friendly encryption, leading to more efficient decryption. Consequently, this methodology allows for accommodating more transactions within a given time slot.

Advantages

- User's Computation Efficiency

: Scroll's MDE offers a more efficient computational process for users compared to Radius's PVDE, as it eliminates the need for generating zero-knowledge proofs. This enhancement is due to a shift in responsibility for creating Timelock Puzzle (TLP) parameters. In PVDE's architecture, users were tasked with generating TLPs, which necessitated proving the legitimacy of their puzzles to prevent potential attacks that could influence the sequencer's computation load. However, in MDE, the burden of TLP generation does not fall on individual users, substantially reducing the risk of such attacks. This change eliminates the need for users to prove the proper encryption of their transactions using their keys, streamlining the user experience without compromising security.

- Sequencer's Computation Efficiency

: The requirement to solve only one timelock puzzle significantly reduces the computational burden on sequencers. This efficiency boosts the capacity to process a higher volume of transactions in a predetermined time slot.

Limitations

- 1-out-of-N Trust Model

: The system necessitates a descent to a 1-out-of-N trust model, where trust is decentralized as much as possible to the KeyGen Committee. This approach introduces a reliance on the committee's integrity for key pair generation.

- Periodic Public Key Pair Storage

: Each period requires the public storage (e.g., on Ethereum) of committee members' partial key pairs. Assuming there are 20 sequencers, and without considering range proofs, this could result in a fee of approximately 0.768 ETH per block, illustrating a significant cost implication.

In summary, Scroll's MDE marks a pivotal evolution in encrypted mempool technology, offering enhanced computational efficiencies for both users and sequencers. However, it also introduces specific trade-offs, including the need for a trust model reliant on a committee and potential cost associated with the system's operation. These factors present a complex yet promising landscape for MEV mitigation strategies within blockchain networks.

3. Radius' new SKDE

Our Single Key Delay Encryption (SKDE) design represents a step forward from Scroll's Multiparty Delay Encryption (MDE), inheriting its two primary advantages while addressing its limitations. Here's a breakdown of how the SKDE design enhances the framework:

Advantages Inherited from Scroll's MDE

- User's and Sequencer's Computational Efficiency

: Like Scroll's MDE, our SKDE design maintains high computational efficiency for both users and sequencers. It simplifies the decryption process and eliminates the need for users to generate zero-knowledge proofs (zkp) for their transactions.

Solutions to Limitations

1. Reduced Public Storage Cost

: Our design introduces an entity responsible for performing aggregate operations, significantly reducing the cost associated with storing partial key pairs in a public repository. We have reduced the gas fees from the previously estimated 0.768 ETH to merely 0.008 ETH for storing essential data on a platform like Ethereum. * The legitimacy of the aggregator's operations is proven through Zero-Knowledge Proofs (ZKP)

, where only compressed information (hash) of the partial keys is made public

. Each committee member signs the hash to validate that it corresponds to the key they generated. This method ensures privacy and integrity without revealing the actual keys, while also drastically cutting down on the blockchain storage costs associated with our system.

1. The legitimacy of the aggregator's operations is proven through Zero-Knowledge Proofs (ZKP)

, where only compressed information (hash) of the partial keys is made public

. Each committee member signs the hash to validate that it corresponds to the key they generated. This method ensures

privacy and integrity without revealing the actual keys, while also drastically cutting down on the blockchain storage costs associated with our system.

1. Minimized Trust Assumptions

: We have designed the system to allow individuals who do not trust the committee to participate in the key generation process directly. This inclusivity enhances the trustlessness of the system by decentralizing key generation further.

[

Trust Assumptions

1920×831 66.7 KB

](https://ethresear.ch/uploads/default/original/2X/5/5e16de604cc5e7a8700ac56c58f91015c516ab07.jpeg)

Design Implications

With the SKDE framework, the partial keys are no longer fully public, diverging from Scroll's MDE where all entities could directly verify partial keys and aggregate only the valid ones. As a result:

- Implemented Robust Crypto-Economic Model for Partial Key Verification

: We have designed and implemented a robust economic model for the verification of partial keys. This model is crafted to incentivize honest participation and deter malicious activities effectively. It strikes a balance between ensuring privacy and maintaining the system's integrity through trust, showcasing our proactive approach in fortifying the blockchain ecosystem against vulnerabilities.

In summary, our SKDE design not only adopts the computational efficiencies found in Scroll's MDE but also addresses its shortcomings by reducing public storage costs and minimizing trust assumptions. By incorporating an aggregation entity validated through ZKP and allowing direct user participation in key generation, we aim to foster a more secure, efficient, and trustless environment for MEV mitigation.

3.1. Main idea of our SKDE

[

skde

1920×1067 139 KB

](https://ethresear.ch/uploads/default/original/2X/9/98e2c52a286cee5920ca741ed7565fee5e272d35.jpeg)

The Single Key Delay Encryption (SKDE) process involves several entities: the Key Generation Committee, Key Aggregator, Sequencer, and Users. Each plays a crucial role in the secure and efficient handling of encrypted transactions.

Key Generation Committee

: Responsible for initiating the encryption process by generating partial keys.

[Phase 1]

1. Generate a partial key pair (k_{enc}, k_{ext})

.

1. Produce a hash h

to compress (k_{enc}, k_{ext})

.

1. Sign the compressed h

, creating a signature \sigma

.

1. Create a proof \pi

that validates the partial key's integrity.

1. Send the generated information (k_{enc}, k_{ext}, h, \sigma, \pi)

to the Key Aggregator and the Committee.

[Phase 2]

(After “Key aggregation” process)

1. Validate the integrity of all partial keys, as the initial steps performed by the Key Aggregator.
2. If any discrepancies are found or if the aggregation process does not meet specified conditions—such as \pi_{agg}

verification, inclusion of invalid key pairs, or omission of valid key pairs—claims are raised to the Key Aggregator.

Key Aggregator

: Serves as the central figure in consolidating partial keys and making them known to Sequencers.

1. Compute h

using the partial keys and verify the signatures \sigma

.

1. Verify the validity of all partial keys using \pi

.

1. Exclude any invalid partial keys and aggregate the rest.
2. Generate a proof of aggregation \pi_{agg}

.

1. Announce the Aggregated key (k_{enc}^{agg}, k_{ext}^{agg})

to Sequencers.

Sequencer

: Responsible for finalizing the transaction sequence and decryption process.

1. Verify \pi_{agg}

.

1. Upon User request, provide the Aggregated encryption key k_{enc}^{agg}

.

1. Upon receiving an Encrypted Transaction, commit its order and using the Aggregated extraction key k_{ext}^{agg}

to compute the decryption key k_{dec}^{agg}

.

1. Decrypting the transactions.

User

: Engages with the Sequencer to encrypt transactions using the Aggregated encryption key k_{enc}^{agg}

.

1. Query the Sequencer for the Encryption key.
2. Encrypt their transaction using the received Encryption key k_{enc}^{agg}

and send CT

to the Sequencer.

3.2. Addressed security challenges

Our Single Key Delay Encryption (SKDE) design effectively navigates through several security challenges inherent in encrypted mempool systems. By addressing the following attack scenarios, SKDE enhances the integrity and confidentiality of transactions within the blockchain. Here's how the SKDE framework addresses these challenges:

1. Key Contamination Attack

: A potential attack where an invalid partial key pair (k_{enc}, k_{ext})

contaminates the aggregated key, rendering all encrypted transactions undecryptable and compromising the Sequencing Layer.

1. Partial Key Verifiability

: The SKDE design ensures that each partial key pair's validity can be verified, preventing the Key Aggregator from generating a contaminated aggregated key. We achieve this through: * Creating proofs \pi

for key validity, utilizing Sigma protocols for verifying the relationship of partial key pairs and range proofs to ensure aggregated keys are the unique combination of partial keys.

1. Creating proofs \pi

for key validity, utilizing Sigma protocols for verifying the relationship of partial key pairs and range proofs to ensure aggregated keys are the unique combination of partial keys.

1. Key Generation Non-repudiation

: Ensures that Key Setup Committee members cannot deny having generated an invalid partial key. This is done by signing the hash of the partial key, allowing the Aggregator to verify this commitment and depend on crypto-economics to encourage claims against discrepancies.

1. Aggregator Responsibility

: The Aggregator is held accountable for excluding invalid partial keys, backed by a penalty protocol that enforces economic consequences for contaminating the block with invalid keys.

1. Decryption Key Leakage Attack

: A scenario where the Aggregator prematurely discloses an aggregated key (k_{enc}^{agg}, k_{ext}^{agg})

, potentially exposing transaction privacy and enabling MEV attacks.

1. Aggregate Computation Verifiability

: By generating zero-knowledge proofs that validate the legitimacy of aggregate operations, SKDE ensures that the public can verify the correctness of the published aggregated key.

1. Commitment Consistency Verifiability

: The framework relies on cryptographic economics to verify that the commitments made public by the Aggregator match the partial keys used in the aggregation process.

1. Timelock Privacy

: SKDE's design guarantees that, given a minimum of one honest Sequencer, no collusion among the others can compromise the privacy of a user's transaction until the designated time has passed.

- Additionally, SKDE allows users to partake in the key generation process, offering a fundamental property where, regardless of any collusion among committee members, the confidentiality of the decryption key is preserved. This inclusion ensures that no single group can undermine the system's security, significantly reinforcing the trustworthiness of the key generation and aggregation phases.

- Additionally, SKDE allows users to partake in the key generation process, offering a fundamental property where, regardless of any collusion among committee members, the confidentiality of the decryption key is preserved. This inclusion ensures that no single group can undermine the system's security, significantly reinforcing the trustworthiness of the key generation and aggregation phases.

- Mismatched Key Replay Attack

: A scenario where the encryption key delivered to the user does not match the actual aggregated key, allowing Sequencers or intermediaries to decrypt and re-encrypt transactions under a guise of normalcy.

1. Mitigation

: Users can verify the aggregated encryption key published in the block against the key received. If discrepancies are found, users have the ability to claim against the Sequencer they communicated with, enhancing transaction privacy and deterring potential MEV attacks.

- Improving proving performance is an ongoing research area that could enable users to verify the validity of keys upon receipt instantaneously, further mitigating the risk of MITM attacks. Our commitment to advancing SKDE's proving performance promises to strengthen the security framework against sophisticated attack vectors, ensuring a more secure and trustless environment for all network participants.

- Improving proving performance is an ongoing research area that could enable users to verify the validity of keys upon receipt instantaneously, further mitigating the risk of MITM attacks. Our commitment to advancing SKDE's proving performance promises to strengthen the security framework against sophisticated attack vectors, ensuring a more secure and trustless environment for all network participants.

3.3. The lifecycle of a key

[

key_lifecycle

2630×454 114 KB

](https://ethresear.ch/uploads/default/original/2X/2/273d8fd884dc7350b31e4c7bcebee14fe2013df1.jpeg)

In the lifecycle of a key within systems like Single Key Delay Encryption (SKDE), we can delineate three distinct phases.

Preparation Phase

: During this initial stage, the key is generated and undergoes several crucial steps to ensure its readiness for encryption purposes. Key activities include:

- Generation of the partial keys, marking its inception.
- To share the partial keys among Sequencers.
- Aggregation of partial keys to form a robust, unified encryption key.
- Dissemination of the aggregated key for upcoming encryption tasks.

Accessibility Phase

: This phase is characterized by the system's readiness to accept and process transactions using the previously prepared key. It involves:

- Collection of encrypted transactions from users.
- To Solve the key, ensuring readiness for decryption.
- Execution of fair sequencing and transmission of order commitments.
- Production of proofs for the aggregation process, affirming the validity of the encryption key.

Lockdown Phase

: The final stage locks the usage of aggregated key and focuses on transitioning from encryption to decryption. This phase includes:

- Decryption of the amassed transactions, rendering them into their original, intelligible state.

[

key_cycle_period

1142×474 49.9 KB

](https://ethresear.ch/uploads/default/original/2X/4/47c4c39b11428643f12782db9a05646489fe40ec.png)

So far, we have discussed the birth and eventual retirement of a single key within our system. Now, we turn our attention to the cyclical nature of key generation and how it perpetuates throughout the system's operation. Given that all entities utilize a single key, there is a need for a defined periodic cycle to manage its use. We have partitioned the key's lifecycle into three distinct phases to delineate when users can access and utilize the key. Also, the length of the Lockdown phase directly influences the volume of transactions that can be decrypted and ordered within the given cycle. Therefore, we have

established our key usage cycle around the Lockdown phase, setting a rhythm for the generation, active use, and phasing out of keys.

This systematic cycle ensures that key generation and retirement are synchronized with the network's operational needs, facilitating a seamless flow of transaction processing and maintaining the security and efficiency of our encrypted mempool system.

4. Experimental results

In our journey to refine encrypted mempool technology and bolster MEV mitigation efforts, Radius has achieved substantial progress by rolling out the Curie Testnet and Portico Testnet, both powered by the original Practical Verifiable Delay Encryption (PVDE). Additionally, we have implemented Scroll's Multiparty Delay Encryption (MDE) and our Single Key Delay Encryption (SKDE). In this chapter, we present the experimental analysis that offers a detailed comparison of the computational costs entailed by the PVDE, MDE, and SKDE frameworks for each participating entity.

[

performance_table

1468×542 93.5 KB

](https://ethresear.ch/uploads/default/original/2X/6/65724d17b398784454f5c8d1a8b0dbe0e74e80af.png)

The experiments are conducted on an Apple M3 Pro with 18GB memory. The Zero-Knowledge Proofs (ZKP) utilized in PVDE and SKDE are both constructed using the Halo2 proving system. In this context, n

represents the number of users and c

denotes the number of Key Generation Committee members. T_s

denotes a short-cycle timelock parameter for PVDE, and T_l

is a long-cycle timelock parameter for MDE and SKDE.

T_s

in PVDE : This parameter is designed to accommodate the generation of a time lock puzzle for every individual transaction within PVDE. It is set to ensure that there is sufficient time to send an order commit to the user before the conclusion of the time lock parameter. The shorter cycle is necessitated by the per-transaction time lock puzzle approach, requiring a quicker turnaround to maintain transaction flow and commit timing.

T_l

in MDE and SKDE : Unlike PVDE, MDE and SKDE operate under a model where only one time lock puzzle needs to be solved within a defined cycle. This allows for a longer time lock parameter T_l

, providing the flexibility to accumulate transactions over a longer period. The extended time lock cycle facilitates not only a First-Come-First-Served (FCFS) sequencing but also enables the aggregation of transactions to apply predetermined rules (such as fee auctions) for ordering and committing transactions. This longer duration ensures there is ample time to order transactions according to these rules, enhancing the system's capacity to handle transactions efficiently and securely.

Through this analysis, we have determined that SKDE stands out in terms of computational efficiency for both users and sequencers. Moreover, it has reduced the gas fee significantly, which is required for storing partial keys each block period. It is also reasonable to project that if blob storage were to be factored into the equation, both MDE and SKDE would likely see a corresponding decrease in fees.

The introduction of an aggregator entity makes an advancement in our framework, adding a layer of complexity and efficiency. The security of this aggregator is ensured through the application of Zero-Knowledge Proofs (ZKP) and crypto-economic principles, forming a robust design that balances privacy, efficiency, and trust.

Regarding the ZKP

entry in the table, it indicates that the aggregator's computational effort is denoted as ZKP

- (78c)ms

. The ZKP proving computation is linear to the committee size c

. Current ongoing research efforts are focused on optimizing the ZKP

computation of the aggregator, aiming to minimize its operational overhead while maintaining the integrity and efficacy of the

system.

5. Conclusion

In summarizing the key points from our exploration of PVDE, MDE, and SKDE, we draw conclusions across several critical dimensions: security, user computation, sequencer computation, aggregator computation, and public storage gas fee.

Security

: Our SKDE design approaches ‘trustlessness’, being architected to minimize trust assumptions wherever possible. We have identified potential attack vectors, defined security requirements formally, and proved mathematically to validate these security claims. This rigorous approach ensures a high level of confidence in the robustness of our system against MEV vulnerabilities.

User Computation

: Unlike previous PVDE, SKDE does not require users to generate individual timelock puzzles, eliminating the risk of DDoS attacks stemming from malformed puzzles. This change effectively removes the need for user-generated proofs, resulting in a revolutionary decrease in user-side computational requirements.

Sequencer Computation

: By only needing to solve a single timelock puzzle for a defined cycle, SKDE significantly lightens the computational load for sequencers, compared to the multiple puzzle resolutions required in PVDE.

Aggregator Computation

: The aggregator is a novel entity introduced within the SKDE framework. It was developed to address the significant gas fees associated with MDE’s public key storage. The aggregator performs computational tasks including the creation of ZKPs, contributing to the overall system efficiency while ensuring the integrity of the aggregation process.

Public Storage Gas Fee

: SKDE achieves a remarkable reduction in the gas fees for public storage, mainly due to the reduced frequency and size of key information that needs to be stored on-chain.

Acknowledging PVDE’s strength in offering a ‘fully trustless’ environment, Radius plans to provide both PVDE and SKDE as viable options, catering to diverse needs within the blockchain ecosystem. While SKDE offers reduced computational demands and gas fees, PVDE remains a powerful choice for those prioritizing maximal trustlessness. Both systems signify Radius’ commitment to offering tailored solutions that enhance security, efficiency, and composability in the rollup landscape.

References

[KAJ+23] Khajehpour, Amirhossein, et al. “Mitigating MEV via Multiparty Delay Encryption.” Cryptology ePrint Archive

(2023).

[PVDE] MEV-resistant ZK-Rollups with Practical VDE (PVDE)

[Curie] Curie Testnet | Radius

[Portico] Portico Testnet | Radius