JSC "Kazakh-British Technical University"
Faculty of Information Technology
Course: Introduction to Blockchain: Mathematical Foundations of Decentralized Systems

Report for Research and Project

Theme: Blockchain Technology Challenges: Scalability, Security, Privacy

Prepared by: Aiture Moldir
Kassymzhomart Yermakhan
Kenzhebay Askar

Checked by: Mukash Nazgul Kanyshkyzy

Almaty, 2025

CONTENT

INTRODUCTION.....3
MAIN BODY.....5
ANNEX.....18
CONCLUSION.....18
REFERENCES.....19

INTRODUCTION

Relevance

Modern public blockchains face three interrelated challenges that limit their adoption in real-world systems: **scalability, privacy, and security**. Layer-1 blockchains process transactions sequentially on-chain, which results in limited throughput and high transaction costs. At the same time, the transparent nature of public ledgers restricts the use of blockchain technology in applications that require confidentiality. Additionally, errors in smart contract logic have repeatedly led to severe financial losses, demonstrating that cryptographic guarantees alone do not ensure system security.

Project and Research Aim

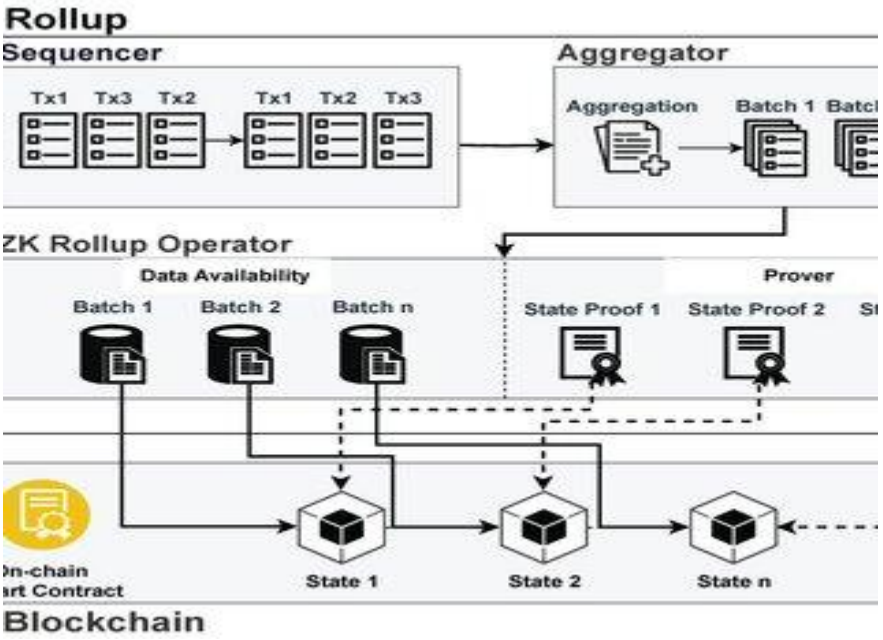
The primary aim of this project is to **design, implement, and analyze a research-oriented prototype of a Layer-2 blockchain system** that addresses the three fundamental challenges of modern public blockchains: **scalability, privacy, and security**.

The project investigates how a **Layer-2 ZK-Rollup architecture** can reduce on-chain computational load through off-chain transaction processing, preserve confidentiality using **zero-knowledge proofs**, and ensure system correctness and trustlessness via **on-chain cryptographic verification**.

By combining theoretical foundations with a practical prototype, the project aims to demonstrate how advanced cryptographic techniques can be applied to improve blockchain performance and security without compromising decentralization.

Scope

The scope of this project covers the **design, implementation, and analysis of a simplified Layer-2 ZK-Rollup prototype** with a focus on scalability, privacy, and security.



General ZK-Rollup architecture used in production systems (conceptual reference).

The diagram is shown for conceptual reference. The implemented prototype follows a simplified research-oriented ZK-Rollup model.

Project Objectives

To achieve the stated aim, the project pursues the following objectives:

1. **To analyze the limitations of Layer-1 blockchains** with respect to throughput, data transparency, and smart contract vulnerabilities, establishing the motivation for Layer-2 solutions.

2. **To study and formalize the cryptographic foundations** underlying zero-knowledge proofs and Merkle tree–based state commitments as mechanisms for privacy and data integrity.

3. **To design a simplified Layer-2 ZK-Rollup architecture** that clearly separates off-chain execution from on-chain verification, enabling scalable transaction processing.

4. **To implement a Mini ZK-Rollup prototype** that processes transactions off-chain, aggregates them into batches, and commits state updates using a Merkle root.

5. **To develop a zero-knowledge circuit and on-chain verifier** that proves the correctness of state transitions without revealing sensitive transaction data.

6. **To evaluate scalability improvements** by comparing gas consumption between individual Layer-1 transactions and batched Layer-2 submissions.
7. **To analyze smart contract security risks** by demonstrating a reentrancy vulnerability and its mitigation, highlighting the importance of secure contract design alongside cryptographic guarantees.

8. **To provide a clear and extensible research framework** that can be scaled to more complex rollup architectures, larger transaction batches, and advanced security mechanisms in future work.

Research Tasks

The project is structured around the following research and implementation tasks:

- Study existing academic and industrial approaches to blockchain scalability, privacy, and security.

- Formalize the state model and state transition rules used in a ZK-Rollup system.

- Implement off-chain transaction batching and state updates using a Merkle tree representation.
- Construct a zero-knowledge proof circuit that validates correct state transitions.
- Deploy and test on-chain smart contracts for proof verification and state finalization.
- Simulate a smart contract vulnerability and demonstrate a secure alternative.
- Analyze experimental results and discuss trade-offs, limitations, and future extensions.

Expected Contribution

The expected contribution of this project is a **coherent research-driven prototype** that demonstrates, in an accessible and verifiable manner, how **Layer-2 ZK-Rollups can simultaneously address scalability, privacy, and security challenges** in blockchain systems.

Rather than aiming for a production-ready implementation, the project focuses on **clarity of design, cryptographic correctness, and analytical depth**, making it suitable both as an academic study and as a foundation for further development.

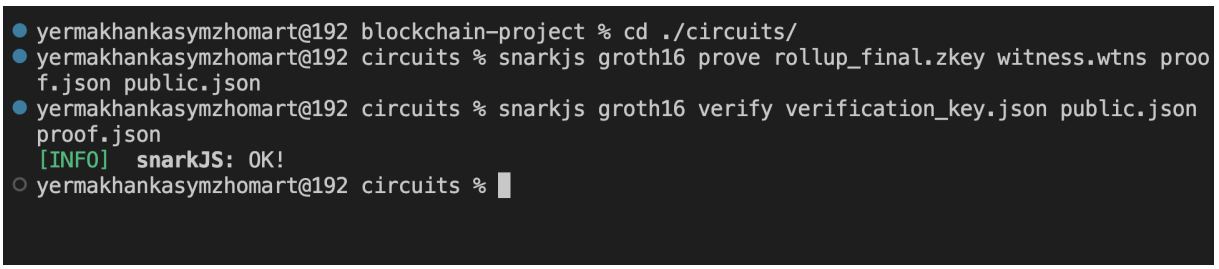
MAIN BODY

Methodology:

- Solidity, Ethereum Testnet
- Circom, SnarkJS
- Merkle Trees (Keccak256)
- Node.js / Python (sequencer)
- Slither / Mythril

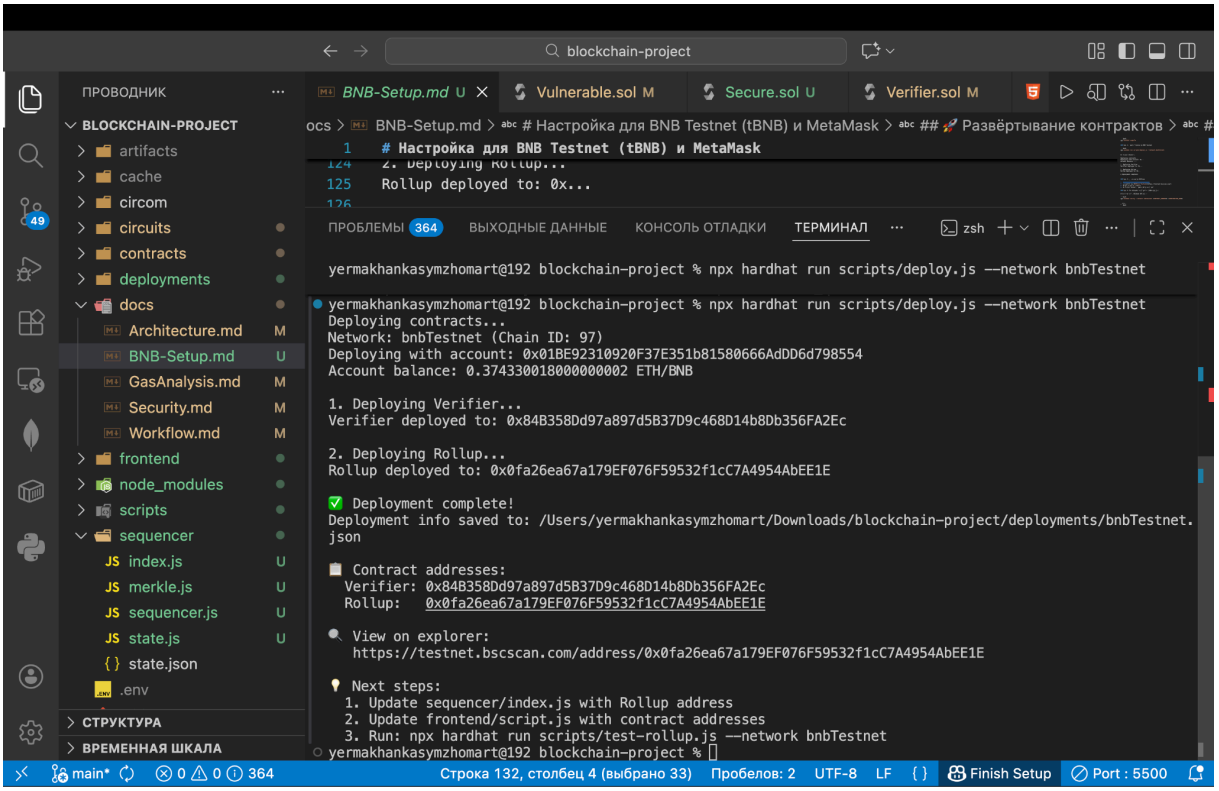
Implementation Details

A Layer-2 ZK-Rollup prototype was implemented, featuring off-chain transaction processing, a Merkle Tree–based state model, and on-chain verification of zero-knowledge proofs using SnarkJS. Smart contracts were developed in Solidity and deployed on an Ethereum test network, while transaction batching and state updates were handled by an off-chain sequencer.



Testing and Validation

The system was tested through local and testnet deployments, including ZK-proof generation and batch submission to the L1 contract. All tests confirmed successful proof verification and correct Merkle root updates.



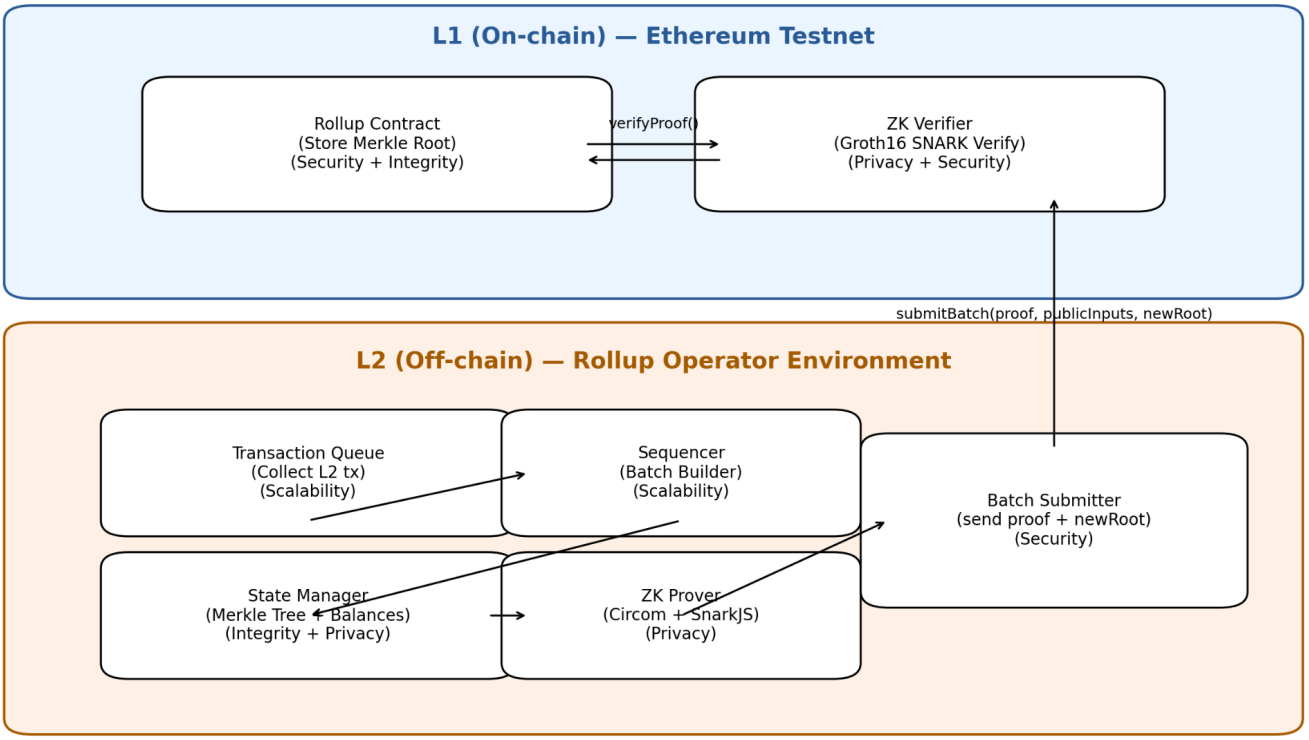
Results

The results demonstrate reduced gas costs due to transaction batching and the correct functioning of zero-knowledge verification. Visual outputs include transaction data, blockchain state information, and gas consumption analysis.

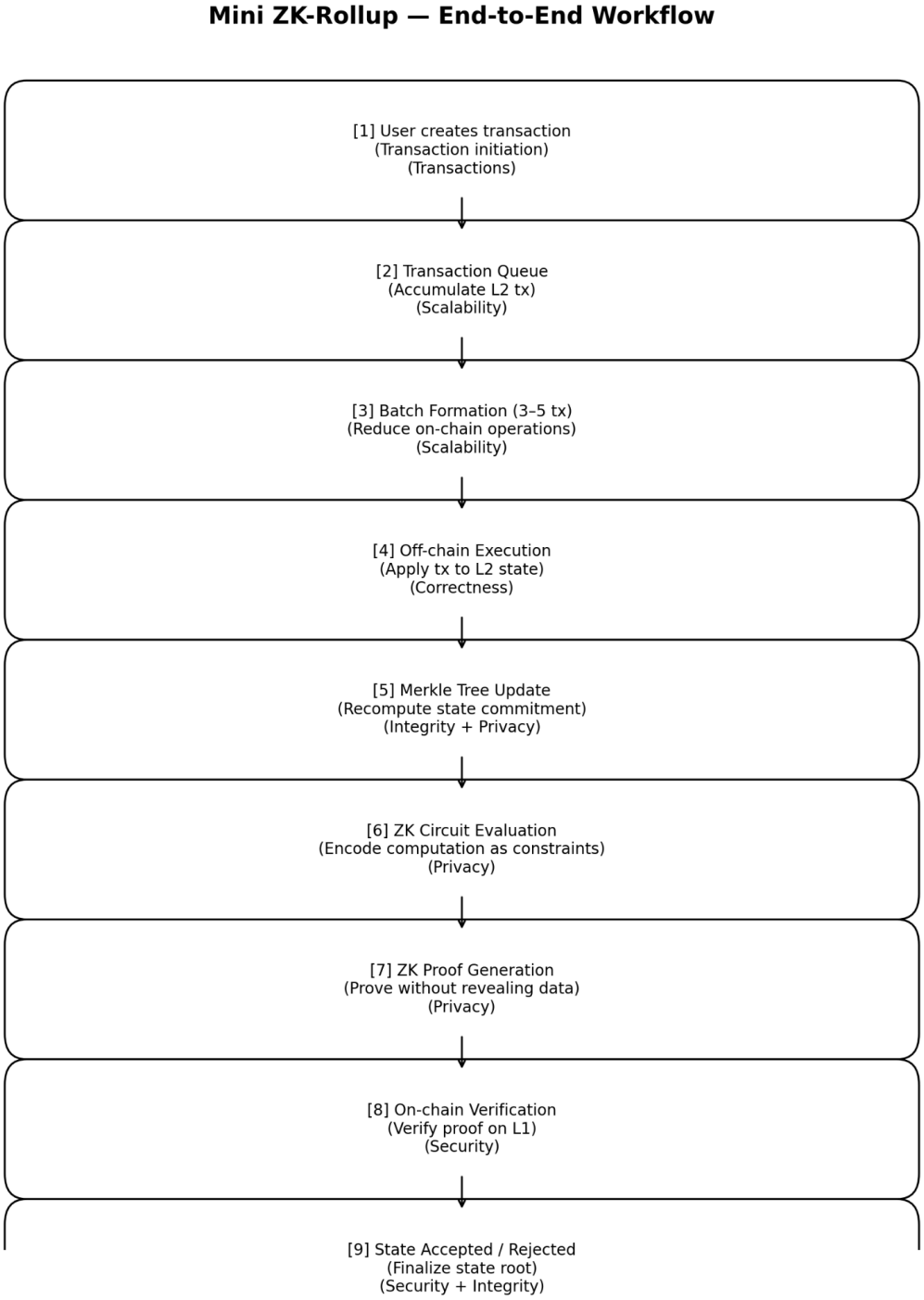
Project architecture:

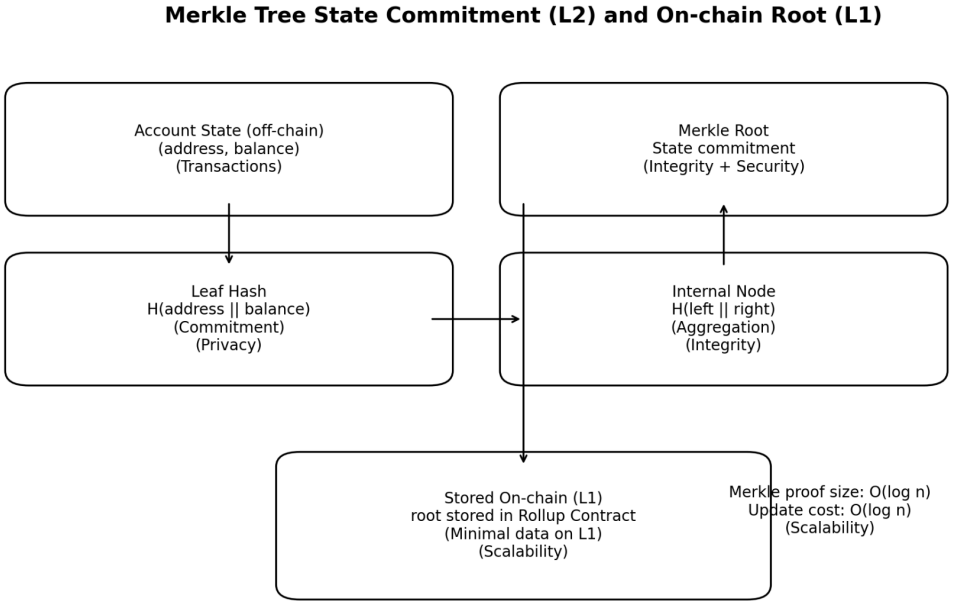
L2 performs computation and batching (Scalability), commits state using a Merkle root (Integrity), produces a ZK proof of correct state transition without revealing internal data (Privacy), and L1 verifies and finalizes the new commitment (Security).

System Architecture (L1/L2 interaction)



Mini ZK-Rollup End-to-End Workflow (step-by-step)





Role of Participant 1: Research Architect & Project Manager

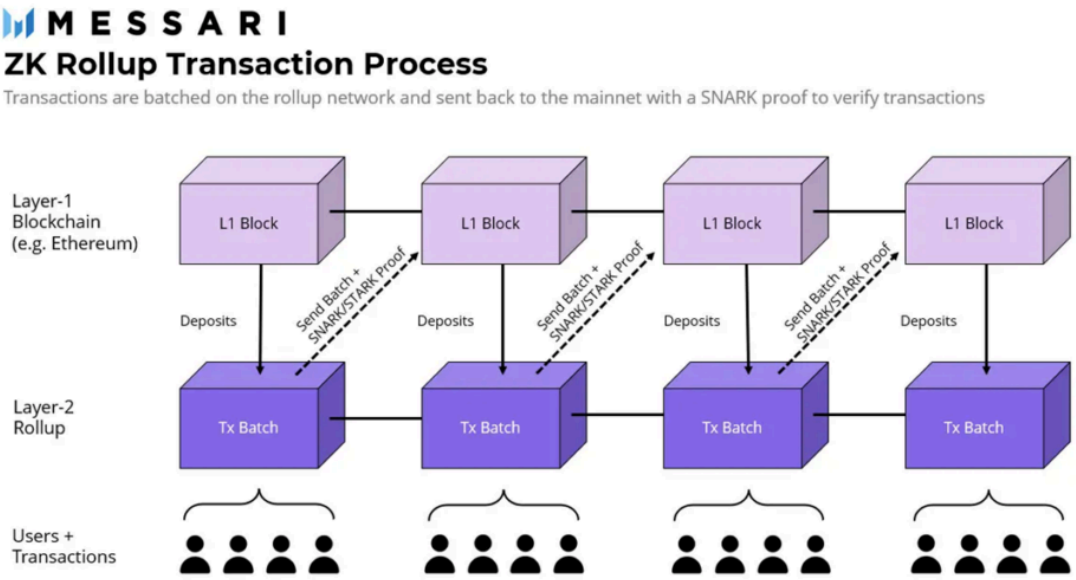
Scope of Responsibility

Participant 1 was responsible for the **overall research architecture and analytical coherence of the project**. This role focused on ensuring that all technical components formed a unified system addressing the core blockchain challenges of **scalability, privacy, and security**.

The responsibilities included:

- Designing the **conceptual Layer-2 ZK-Rollup architecture**, including the separation between off-chain execution and on-chain verification.
- Formalizing the **system interaction model**, defining how transaction batching, state commitments, zero-knowledge proofs, and on-chain verification interact within a single workflow.
- Structuring and authoring the **theoretical and analytical sections** of the research, including the literature review, methodology, and discussion.
- Integrating contributions from all participants into a **consistent research narrative**, ensuring logical continuity between theory, implementation, and results.

- Interpreting experimental outcomes, including scalability and gas cost observations, and deriving **general conclusions and limitations** of the proposed approach.



Conclusions from Responsibilities

Through architectural design and analytical synthesis, Participant 1 established a coherent research framework demonstrating how Layer-2 ZK-Rollup systems can simultaneously address scalability, privacy, and security challenges. The role ensured that the project remained research-oriented, conceptually sound, and extensible for future development, while accurately reflecting the implemented prototype.

Participant 2: ZK & Cryptography Engineer

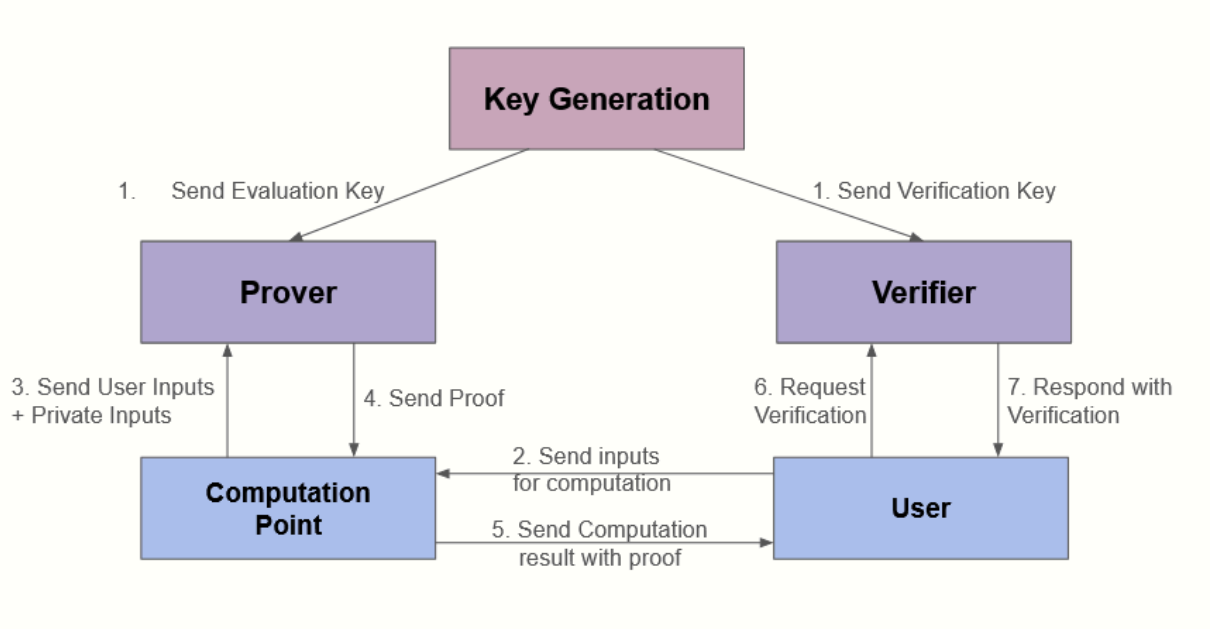
Scope of Responsibility

Participant 2 was responsible for the **cryptographic core of the project**, focusing on the design, implementation, and validation of zero-knowledge proof mechanisms used in the ZK-Rollup prototype.

The responsibilities included:

- Designing and implementing the **zero-knowledge circuit** using Circom to formalize correct state transition logic.
- Generating zero-knowledge proofs using **SnarkJS**, including the preparation of private witnesses and public inputs.
- Developing and integrating the **on-chain verifier smart contract** (`verifier.sol`) for proof verification on the Ethereum test network.

- Contributing to the **theoretical analysis of zero-knowledge proofs**, including the explanation of SNARK properties, privacy guarantees, and verification correctness within the research text.



Conclusions from Responsibilities

Participant 2 successfully demonstrated the practical application of zero-knowledge cryptography within a Layer-2 context. The implemented ZK circuit and proof-generation pipeline validated that state transitions could be proven correct without revealing underlying transaction data. This contribution provided the cryptographic foundation for the project's privacy guarantees and enabled secure on-chain verification, directly supporting the research objectives related to privacy and security.

Role of Participant 3: Rollup & Security Engineering

1. Scope of Responsibility

Within the project, Participant 3 was responsible for the implementation of rollup state management, off-chain batch processing, and on-chain security analysis. The main focus of this contribution was the practical realization of rollup mechanics and the evaluation of smart contract security through a real-world attack scenario.

The responsibilities included:

- off-chain transaction batching,
- Merkle Tree–based state representation,
- rollup smart contract logic,

- demonstration of a reentrancy vulnerability,
- development of a secure contract version.

2. Off-chain Batch Processing

To simulate rollup behavior, transactions were processed off-chain in batches. Each batch updates the global system state before being committed on-chain. This approach reflects how rollups reduce on-chain computation and gas costs.

The off-chain batch logic applies transactions sequentially, validates balances, and produces an updated state. This logic was implemented in Python and serves as a simplified model of rollup execution.

3. Merkle Tree State Representation

To represent the rollup state compactly and securely, a Merkle Tree structure was used. Each account balance is stored as a leaf node, and the Merkle root serves as a cryptographic commitment to the entire state.

```
from dataclasses import dataclass

from typing import Dict, List

from merkle_tree import hash_leaf, merkle_root

@dataclass

class Tx:

    sender: str

    receiver: str

    amount: int

def apply_batch(state: Dict[str, int], txs: List[Tx]) -> Dict[str,
int]:

    new_state = dict(state)

    for tx in txs:

        s = tx.sender.lower()

        r = tx.receiver.lower()

        a = tx.amount

        if new_state.get(s, 0) < a:

            raise ValueError("Not enough balance")
```

```
        new_state[s] -= a
        new_state[r] = new_state.get(r, 0) + a
    return new_state
def compute_root(state: Dict[str, int]) -> str:
    leaves = []
    for addr in sorted(state.keys()):
        leaves.append(hash_leaf(addr, state[addr]))
    root = merkle_root(leaves)
    return "0x" + root.hex()
```

Transactions are provided to the batch executor as a list of operations. Additionally, a JSON-based representation of transactions (transactions.json) is included in the project to illustrate the expected input format for off-chain batches. This file serves as an example data structure and is not required for the core execution logic.

This approach allows efficient verification of state correctness without storing all data on-chain, which is a core principle of rollup systems.

```
import hashlib
from typing import List
def sha256(data: bytes) -> bytes:
    return hashlib.sha256(data).digest()
def hash_leaf(address: str, balance: int) -> bytes:
    text = f"{address.lower()}:{balance}".encode()
    return sha256(text)
def hash_pair(left: bytes, right: bytes) -> bytes:
    return sha256(left + right)
def merkle_root(leaves: List[bytes]) -> bytes:
    if not leaves:
```

```
        return sha256(b"empty")
    level = leaves[:]
    while len(level) > 1:
        next_level = []
        for i in range(0, len(level), 2):
            left = level[i]
            right = level[i + 1] if i + 1 < len(level) else level[i]
            next_level.append(hash_pair(left, right))
        level = next_level
    return level[0]
```

Initial state: {'0xaaa': 100, '0xbbb': 50, '0xccc'

Old Merkle Root: 0x501a986d9e4b08682d8b59b9143705f

New state: {'0xaaa': 90, '0xbbb': 45, '0xccc': 15}

New Merkle Root: 0xf149d52bee254fb94bc0a52d0220da9

4. Vulnerable Rollup Contract

An initial rollup contract was implemented with an intentional security flaw to demonstrate a reentrancy vulnerability. In this version, Ether transfer occurs before the user balance is updated, allowing a malicious contract to repeatedly re-enter the withdrawal function.

This vulnerable design reflects a common mistake found in early smart contracts and serves as a basis for security analysis.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract RollupVulnerable {
    mapping(address => uint256) public balances;
    function deposit() external payable {
        balances[msg.sender] += msg.value;
    }
    function withdraw(uint256 amount) external {
```

```
        require(balances[msg.sender] >= amount, "Not enough");

        // ❌ УЯЗВИМОСТЬ: сначала перевод, потом обновление

        (bool ok,) = msg.sender.call{value: amount}("");

        require(ok, "Transfer failed");

        balances[msg.sender] -= amount;

    }

}
```

5. Reentrancy Attack Demonstration

To validate the vulnerability, a dedicated attacker contract was implemented. The attacker exploits the improper order of operations in the vulnerable contract by recursively calling the withdrawal function during Ether transfer.

The successful execution of this attack demonstrates that the vulnerability is exploitable in practice, not only theoretical.

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

import "./rollup_vulnerable.sol";

contract Attacker {

    RollupVulnerable public target;

    uint256 public rounds;

    uint256 public maxRounds = 3; // сколько раз повторно войдём

    constructor(address _target) {

        target = RollupVulnerable(_target);

    }

    receive() external payable {

        // ограничиваем количество повторных входов,

        // иначе уйдём в бесконечную рекурсию и кончится gas

        if (rounds < maxRounds && address(target).balance >= 1 ether) {

            rounds += 1;
```

```
            target.withdraw(1 ether);

        }

    }

    function attack() external payable {

        require(msg.value >= 1 ether, "Send at least 1 ETH");

        // кладём 1 ETH на баланс Attacker внутри target

        target.deposit{value: 1 ether}();

         // запускаем первый вывод (дальше receive сделает ещё maxRounds
раз)

        target.withdraw(1 ether);

    }

}
```

6. Secure Rollup Contract

After identifying the vulnerability, a secure version of the rollup contract was developed. The fix includes:

- updating the user balance before transferring Ether,
- introducing a reentrancy lock mechanism.

As a result, repeated entry into the withdrawal function becomes impossible, and the attack transaction is reverted.

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract RollupSecure {

    mapping(address => uint256) public balances;

    bool private locked;

    modifier nonReentrant() {

        require(!locked, "Reentrancy blocked");

        locked = true;
```

```
        _;
        locked = false;
    }
    function deposit() external payable {
        balances[msg.sender] += msg.value;
    }
    function withdraw(uint256 amount) external nonReentrant {
        require(balances[msg.sender] >= amount, "Not enough");
        balances[msg.sender] -= amount;

        (bool ok,) = msg.sender.call{value: amount}("");
        require(ok, "Transfer failed");
    }
}
```

7. Security Analysis and Comparison

A comparative security analysis was conducted between the vulnerable and secure versions of the rollup contract. The vulnerable contract allows unauthorized fund withdrawal via reentrancy, while the secure version correctly blocks the attack.

Additionally, the contract logic was analyzed from the perspective of automated security tools such as Slither and Mythril. The vulnerability corresponds to well-known reentrancy patterns typically flagged by such tools.

This confirms that the applied fix aligns with established smart contract security best practices.

8. Summary of Contribution

The contribution of Participant 3 demonstrates the importance of secure smart contract design in rollup systems. By combining off-chain batching, Merkle Tree state commitments, and practical security analysis, this work highlights both the scalability benefits and security challenges of rollup architectures.

ANNEX:

GitHub: blockchain-project
ReadMe:
https://blockchain-challenges.readme.io/update/page/report-for-research-and-project-theme-blockchain-technology-challenges-scalability-security-privacy

CONCLUSION

This project examined how **Layer-2 ZK-Rollup architectures** can address the fundamental challenges of **scalability, privacy, and security** in modern blockchain systems. By combining theoretical analysis with a research-oriented prototype, the study demonstrates that these challenges are best addressed through an integrated architectural and cryptographic approach rather than isolated optimizations.

The work builds on foundational research in zero-knowledge proofs and cryptographic state commitments. Zero-knowledge principles, formalized in early cryptographic literature and later realized through succinct non-interactive arguments, enable correctness verification without revealing sensitive data. Merkle tree–based state representations, widely used in blockchain systems, provide compact and verifiable commitments to off-chain state. Together, these mechanisms form a rigorous basis for scalable and privacy-preserving computation.

At the architectural level, the prototype illustrates how **off-chain transaction execution combined with on-chain verification** can reduce computational and economic load on the base layer while preserving trustlessness. Transaction batching improves scalability, Merkle roots ensure state integrity, and zero-knowledge proofs enable confidential yet verifiable state transitions. This design reflects the core principles of ZK-Rollups described in contemporary Layer-2 research.

The project also highlights the importance of security beyond cryptography by demonstrating a reentrancy vulnerability and its mitigation. This analysis reinforces existing research showing that secure smart contract development is essential even in systems protected by advanced cryptographic techniques.

Rather than aiming for production readiness, the project intentionally focuses on clarity, correctness, and analytical depth. As a result, it provides a coherent research prototype that both explains and validates key ZK-Rollup concepts. The work contributes an accessible foundation for further academic study and future extensions toward more complex and fully featured Layer-2 systems.

References

1. Goldwasser, S., Micali, S., & Rackoff, C. (1985). *The Knowledge Complexity of Interactive Proof Systems*.

2. Groth, J. (2016). *On the Size of Pairing-Based Non-interactive Arguments*.

3. Ben-Sasson, E., Chiesa, A., Tromer, E., & Virza, M. (2014). *Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture*.

4. Merkle, R. C. (1988). *A Digital Signature Based on a Conventional Encryption Function*.

5. Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*.

6. Wood, G. (2014). *Ethereum: A Secure Decentralised Generalised Transaction Ledger (Yellow Paper)*.

7. Buterin, V. (n.d.). *An Incomplete Guide to Rollups*.

8. L2BEAT. (n.d.). *Layer-2 Research and Analytics*.

9. Eyal, I., & Sirer, E. G. (2016). *Bitcoin-NG: A Scalable Blockchain Protocol*.

10. The DAO. (2016). *Post-Mortem / Analysis of the DAO Exploit (Reentrancy)*.

11. Luu, L., Chu, D.-H., Olickel, H., Saxena, P., & Hobor, A. (2016). *Making Smart Contracts Smarter*.

12. Trail of Bits. (n.d.). *Slither: Solidity Static Analysis Framework*; Mythril. (n.d.). *Security Analysis Tool for EVM Bytecode*.