# Klaytn Improvement Reserve

# Auditable Privacy Preserving FT/NFT Transfer on Klaytn Blockchain

# 

Date: 2021.9.1

## ## Summary

We have successfully developed Zklay which is a zero knowledge proof based Klay (and FT) transfer scheme to support privacy and auditability in Klaytn at the first KIR. From the lesson, we will propose KIPs for privacy, auditability, and zero-knowledge supporting. We improve the proof generation performance and reduce the gas consumption for Zklay. In addition we extend the proposed scheme to support NFT (non-fungible token).

In detail, we will propose a new KIP (auditable privacy preserving fungible token) to allow privacy and auditability similarly to KIP-7. Moreover, to reduce the gas cost in the current version of the proposed scheme, we will propose a new native instruction called “MiMC” hash in Klaytn as KIP. The MiMC hash has been proposed to accelerate the hash computation in zk-SNARKs proof generation since the SHA256 is too slow in zk-SNARKs proof generation. However, since it is not supported as a native instruction, its gas cost is too high compared with the existing SHA256 that is supported as a native instruction.

To improve the proof generation performance which determines the transaction generation performance in a user side, and the proof verification performance that affects the gas consumption in a smart contract, we devise a new faster and more efficient membership proof which is the major performance bottleneck in zk-SNARK based on RSA accumulator.

Finally, we will extend Zklay to support NFT with auditability and privacy, and define a new KIP (auditable privacy preserving non-fungible token) similar to KIP-17 (non-fungible token) and KIP-37 (multi token). For NFT Zklay, we will define the transaction format, the wallet interface with a wallet example, and so on. And we wil implement and show the NFT Zklay demo.



1. Previous Zklay Project Summary

## ## Team(Individuals, Corporation) Introduction

The project is performed by Zkrypto Inc[[1]](#footnote-2) which is specialized in a zero-knowledge proof technology.



2. Zkrypto Inc

|  |  |  |
| --- | --- | --- |
| **Position** | **Name** | **Short Bio.** |
| Principle investigator | Hyunok Oh | B.S./M.S./Ph.d./ in Computer Engineering, Seoul National University  Professor, Hanyang University.  CEO, Zkrypto Inc.  Expert in cryptography, zero-knowledge proof, and blockchain.  Research on zk-SNARK, snark-friendly encryption schemes, zk-SNARK based applications, etc.  Working in international zk-SNARK standardization. |
| Principle investigator | Jihye Kim | B.S./M.S. in Computer Engineering, Seoul National University  Ph. D. in information and computer science, UC Irvine  Associate Professor, Kookmin University.  CTO, Zkrypto Inc.  Expert in cryptography, zero-knowledge proof, and blockchain.  Research on zk-SNARK, snark-friendly encryption schemes, , zk-SNARK based applications, etc.  Working in international zk-SNARK standardization. |
| Investigator | Gweonho Jeong | B.S/M.S in information systems engineering, Hanyang University  Ph.D student, Hanyang University.  Expert in cryptography, zero-knowledge proof.  Research on anonymous transfer (Zklay) |
| Investigator | Nuri Lee | B.S in electrical engineering, Kookmin University  M.S student, Kookmin University.  Expert in cryptography, zero-knowledge proof, and blockchain.  Research on anonymous transfer (Zklay) |
| Investigator | Thomas Ekow | M.S/Ph.D in information systems engineering, Hanyang University  Principal engineer, Zkrypto Inc.  Expert in zk-SNARK system design and optimization |
| Investigator | Seungpyo Cho | M.S in information systems engineering, Hanyang University  Principal engineer, Zkrypto Inc.  Expert in zk-SNARK system design and optimization |
| Developer | Sungjoo Kim | B.S/M.S in electrical engineering, Kookmin University  Engineer, Zkrypto Inc.  Expert in cryptography, zero-knowledge proof. |
| Developer | Ingeun Lee | B.S/M.S in electrical engineering, Kookmin University  Engineer, Zkrypto Inc.  Expert in cryptography, zero-knowledge proof |
| Developer | Donghae Yang | B.S student in S/W engineering, Hanyang University  Intern engineer, Zkrypto Inc.  Developing a blockchain application |

## ## Motivation

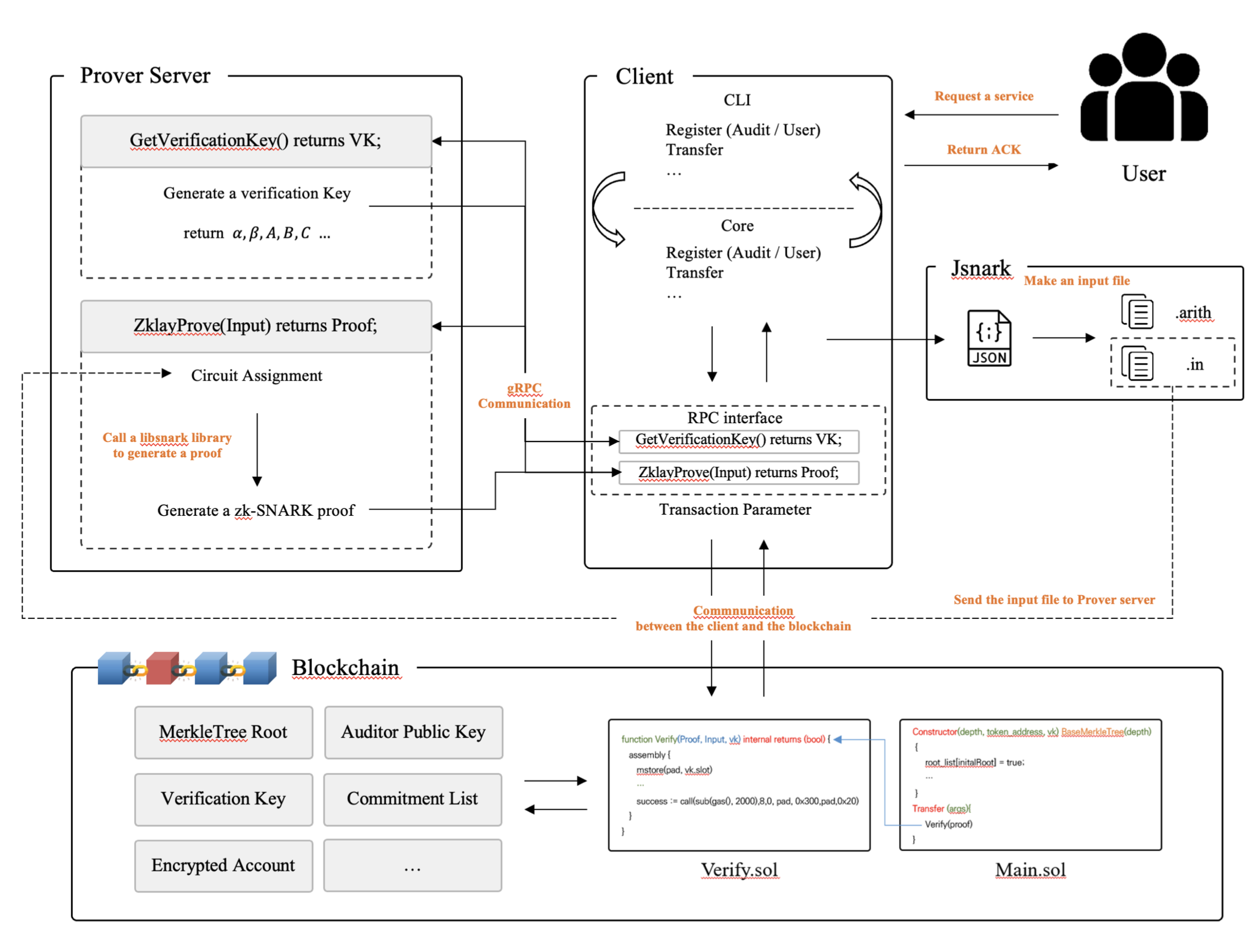
In the public blockchain, since the stored data is available publicly, it becomes more crucial to provide the privacy in the blockchain. A simple encryption of a transaction is not a solution to this privacy problem in the blockchain since miners cannot verify the validity of the encrypted transaction publicly. To solve the problem, zero-knowledge proof approaches have been widely adopted in the blockchain. Zcash, Zeth, BlockMaze, and Zklay are zero-knowledge based solutions to provide an unlinkability (or anonymity) by utilizing a join-split approach in which an encrypted transaction (or commitment) is sent to so called a mixer to break the transaction linkability. To spend the received money without revealing which commitment is used, a membership proof is used. Currently most approaches use a merkle hash tree to show the membership. In the merkle tree, hash functions are performed many times. For instance, to support anonymity among 2^32 transactions, 32 times hash operations should be evaluated in zero knowledge proof generation. Since a hash such as SHA is a bit based function, it builds a very large zero-knowledge circuit which decreases the proof generation performance. To accelerate the proof generation, more ZKP (zero knowledge proof) friendly hash algorithms are developed and utilized such MiMC, Poseidon, and Ajtai hashes. Especially, Zklay is also utilizing MiMC hash algorithm for the membership proof. However, since these hash algorithms are not native instruction in blockchains, they require many computations in smart contracts. For instance, a single MiMC computation consumes 30K gas in Ethereum and Klaytn. Moreover, due to the maximum operation limitation in Klaytn, which is devised to provide the fast block creation, a limited number of hash functions can be executed in a smart contract. To solve the problem, we will propose a new KIP to support MiMC hash as a native instruction like SHA256 since the MiMC hash will be frequently utilized in future blockchains supporting zero-knowledge proofs. In addition, we will develop a new fast and light membership proof for the Zklay, which accelerates the proof generation and reduces the required gas cost.

In Klaytn, not only FT but also NFT becomes popular while no privacy is supported. Currently, every information is publicly known. We will develop an auditable privacy preserving NFT scheme by extending the Zklay.

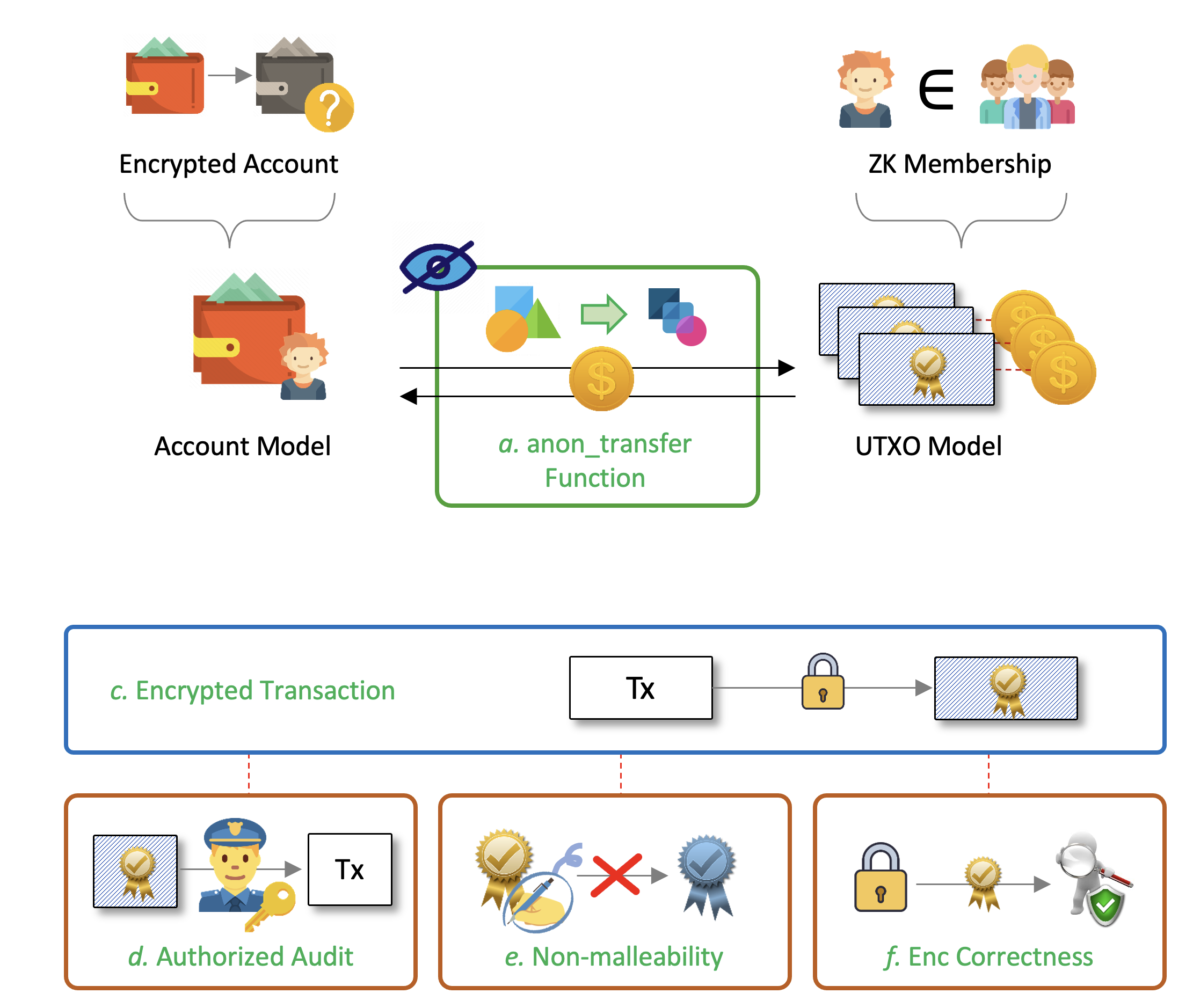
## Background

Zcash [BCG+14] is a well-known privacy-preserving blockchain system using the zero-knowledge proof. In Zcash, a sender makes a new confidential coin (or commitment) that hides all information of the transaction (i.e value, sender, receiver) and the information is decrypted by a receiver. The sender attaches the correctness of the commitment using the zero-knowledge proof. The zero-knowledge proof proves that the commitment has ever created and available in the blockchain, the sender has a privilege to use the commitment, and the newly created commitment is correctly constructed. When the proof is verified, the receiver can use the transferred value. Zether [BAZB20] accomplishes the privacy-preserving account-based blockchain using the zero-knowledge proof (bulletproof) and ElGamal encryption scheme. In Zether, when the sender wants to transfer some value to the receiver, the sender makes a ciphertext of the value with the receiver’s public key. The ciphertext can be added to the receiver’s state, and the sender proves the correctness of the ciphertext and whether the remaining value of the sender’s account is positive or not. However, the sender should generate a zero-knowledge proof for the large user set for anonymity. Specifically, the sender generates dummy ciphertexts with 0 value except the receiver’s ciphertext. It increases the proof size in case of the bulletproof. Zeth [RZ19], and Blockmaze[GWY+20] are account model version of Zcash.

In the previous proposal, we have successfully developed Zklay which supports the Klay transfer with preserving the privacy and the auditability. In the Zklay, the proof generation application is written in C++, the client application is in python (with C), and the smart contract is in solidity. Zklay provides an anonymous transaction which includes public sender/receiver accounts, an encrypted sender account, a nullifier denoting a used confidential coin, and a new confidential coin. In a single transaction, all possible combinations are allowed such as sending among public accounts, encrypted sender account, using a confidential coin, and creating a confidential coin. Moreover, it supports the auditability in which an auditor can trace all transactions by decrypting ciphertext. Note that it is required to include the encryption algorithm in a zero-knowledge circuit to guarantee the decryptability of the ciphertext in a proof to support the auditability where the encryption is excluded in most Zcash style approaches such as Zether and Blockmaze to shorten the proof generation time.



4. The Zklay Structure for FT



5. The Zklay Features for FT

## Project Description

텍스트이(가) 표시된 사진

자동 생성된 설명

3. Summary of the proposed project

1. Definition of KIP for an auditable privacy preserving fungible token

When we start the Zklay in the previous round, we thought that it may be required to revise the blockchain core to support the privacy in each transaction. Fortunately the completed Zklay does not request any modification of the blockchain core. Most difficult works are concentrated on the zero-knowledge proof generation. From the lesson in the first round KIR, we will define a KIP to support privacy and auditability in FT. The proposal will follow the format of SNIP-20 ( <https://github.com/SecretFoundation/SNIPs/blob/master/SNIP-20.md> ). But the proposed one will be compatible to the proposed Zklay.

In the KIP, messages and queries as following will be defined:

Messages

* RegisterAuditor

Register the auditor public key

* RegisterUser

Register the user address

* zkTransfer

Deposit, withdraw, send and receive the tokens

Queries

* Account

Show the encrypted account information

1. zk-SNARKs friendly hash function

Another lesson from the first round KIR is related with a hash function. To generate a zero-knowledge proof rapidly, it is almost mandatory to adopt a zk-SNARKs friendly hash function like MiMC. However, since the hash is not a native instruction in Klaytn, it requires high gas cost and decreases the Klaytn performance. If the zk-SNARKs friendly hash is natively supported in the Klaytn then it would be beneficial to support zk-SNAKRs efficiently. Hence, we will propose a new KIP to support the zk-SNARKs friendly hash function of MiMC. Note that the proposed MiMC does not mean that we devise a new MiMC hash. We utilize the existing MiMC design and implementation code, and port it in the Klaytn. In addition, we will propose an example how to use the MiMC precompiled contract.

1. Efficient membership proof algorithm

In the current Klaytn, there is the maximum operation limitation to provide 1-second block confirmation. However, as mentioned, since the ZKP friendly hash requires high computation in the smart contract, Zklay is not deployable in the current Klaytn unless the height of the Merkle tree is short. To support Zklay in the current Klaytn without core code modification, we will develop a new fast and efficient algorithm for membership proof which can reduce the gas consumption, too. In this project, we will develop and implement a new membership proof algorithm for fast proof generation and light (less gas consumption) smart contract.

Algorithm overview:

In a traditional membership proof based on RSA accumulator, a user $i$ is assumed to be prime $u\_i$. For set $S = {u\_1, u\_2, \cdots, u\_n}$, an accumulator value is a product of every $u\_j$ in exponent over an unknown order group element ($V$). To prove that $u\_i \in S$, a user produces the membership proof ($W$) by accumulating all the elements in set $S$ without $u\_i$ over $V$. Shortly, $ACC = V^{\prod\_{u\_j \in S} u\_j}$ and $W = ACC^{\frac{1}{u\_i}}$. A verifier simply checks that $W^{u\_i} \overset{?}{=} ACC$. However, the privacy of $u\_i$ is leaked. Through the sigma protocol, it is possible that proving the membership of $u\_i$ without revealing $u\_i$.

Instead of providing $u\_i$ to verifier, prover gives sigma variant proof $k \leftarrow r + u\_i \cdot h$. In detail, a prover chooses random element $r$ and gives $R \leftarrow W^r$. Then, a verifier samples $h$ randomly and passes to a prover. At last, a prover provides $k \leftarrow r + u\_i \cdots h$ instead of $u\_i$. With sigma protocol, the privacy of $u\_i$ can be guaranteed. Nonetheless, it cannot assure the privacy of all the elements in set $S$. That is, in the membership proof $W$, it contains all of the elements except the $u\_i$. Therefore, the privacy of the membership proof $W$ must be considered as well. We will devise a new privacy preserving membership proof algorithm to hide $W.

Compared with the recent membership proof [BCF+19], the proposed scheme will generate the proof more rapidly and the verification (gas) cost is less than [BCF+19].

1. Auditable privacy preserving NFT transfer

There is a similar privacy preserving NFT proposal for ERC-721 NFT called SNIP-721 ( <https://github.com/SecretFoundation/SNIPs/blob/master/SNIP-721.md> ), which does not support auditing. In this project, we will support NFT transfer and propose the KIP- Auditable privacy preserving non-fungible token (NFT).

The anonymous transfer function (zkTransfer) for NFT generates a commitment with NFT id preimage. The NFT transaction will include a ciphertext including NFT id, recipience, and using random key. Using the auditor private key key as well as the recipience private key, it is possible to decrypt the ciphertext to trace the NFT transfer.

## Project Milestones and Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Date(YY.MM.DD) | Project | Details | Etc |
| Start Date + 2 months | - Fast membership | - Devise a fast member proof algorithm  - Implement it in Zklay | **Milestone 1.**  - Technical report  - Experiment results  - Demo |
| Start Date + 4 months | - KIP for FT (privacy preserving KIP-7)  - KIP for MiMC hash  - Auditable privacy preserving transfer NFT | - Propose an auditable privacy preserving FT interface proposal (a new KIP)  - Propose a KIP for a native instruction to support a zero-knowledge friendly hash function (MiMC)  - Design Zklay for NFT | **Milestone 2.**  - KIP for privacy FT  - KIP for MiMC |
| Start Date + 6 months | - MiMC implementation for Klaytn core  - Auditable and privacy preserving NFT transfer | - Implement Zklay for NFT  - Propose an auditable privacy preserving NFT interface proposal (a new KIP)  - Klaytn core code for a MiMC hash instruction | **Milestone 3.**  - Technical report  - Experiment results  - Demo  - KIP for privacy NFT |

## Key Deliverables

Four deliverables are included as below:

1. Technical papers: research paper for fast membership proof. Zklay for NFT
2. Zklay Dapps for FT and NFT with a fast membership proof
3. Zklay smart contracts in Klaytn blockchain
4. KIP proposals
5. MiMC hash code for Klaytn core

## Budget

The zero-knowledge proof based auditable privacy preserving Zklay for FT and NFT requires profound cryptographic knowledge and efforts, which involves a large amount of qualified manpower for research and developments. The project requires investigators and developers for a new zk-SNARK friendly membership proof, KIP proposals, and NFT Zklay.

|  |  |  |
| --- | --- | --- |
| **Classification** | | **Total** |
| Labor Cost | Design/Research : Fast membership proof | 50,000 |
| Implementation : Fast membership proof | 50,000 |
| Design/Research : KIP-27, KIP-MiMC | 30,000 |
| Design/Research : Zklay for NFT | 70,000 |
| Implementation : MiMC hash code for Klaytn core | 30,000 |
| Implementation : Zklay for NFT | 70,000 |

**Table 1: Budget Table (USD)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Classification** | **Monthly wage**  **(per man-month)** | **Estimated**  **man-month** | **Total** |
| Design/Research | 20,000 | 7.5 | 150,000 |
| Implementation | 20,000 | 7.5 | 150,000 |

**Table 2: Labor cost (USD)**

## Attachments

Zkrypto Inc. : <http://www.zkrypto.com>

## Reference

* [BCG+14] Zerocash: Decentralized Anonymous Payments from Bitcoin
* [BAZB20] Zether: Towards Privacy in a Smart Contract World
* [RZ19] Antoine Rondelet and Michal Zajac. ZETH: on integrating zerocash on ethereum. CoRR, abs/1904.00905, 2019.
* [GWY+20] Zhangshuang Guan, Zhiguo Wan, Yang Yang, Yan Zhou, and Bu- tian Huang. Blockmaze: An efficient privacy-preserving account-model blockchain based on zk-snarks. IEEE Transactions on Dependable and Secure Computing, pages 1–1, 2020.
* [BCF+19] Daniel Benarroch, Matteo Campanelli, Dario Fiore, Kobi Gurkan, and Dimitris Kolonelos. Zero-Knowledge Proofs for Set Membership: Efficient, Succinct, Modular.

## Feedback

1. http://www.zkrypto.com [↑](#footnote-ref-2)