Solana Grant Proposal for an NFT[[1]](#footnote-1)/ RSA Program

# Why do we need another NFT program on Solana?

NFT/RSA is a non-fungible token program that uses one token account and N tracking accounts to batch NFT mints, transfers, and burns, on the Solana blockchain. The Solana program library already provides a non-fungible token program but it uses one account per token, and N tracking accounts per token account, to track single NFT mints, transfers, and burns, on the Solana blockchain.

The NFT/RSA program uses RSA accumulators[[2]](#footnote-2) and zero knowledge proofs to batch hundreds, even thousands, of NFT mints, transfers, and burns in constant space and time. Each batch mint, transfer, or burn requires only 3 to 6 transactions to track NFT ownership. Uploading NFT public or private URI’s to the Solana ledger is optional and not part of NFT mint, transfer, and burn transactions.

# The NFT/RSA Program

The Solana blockchain provides a tamper-proof, and non-repudiable, ledger for tracking NFT ownership transactions. The Solana blockchain also serializes the execution of transactions globally so the NFT/RSA program only needs to enforce two transaction validation rules.

* No duplicate tokens.
* No double transfers.

An NFT has a unique URI that references something of value. This URI is not stored in account state but can be uploaded to the Solana ledger so satisfy data availability requirements. An NFT URI is hashed to a prime number. Prime numbers are batched by computing the product of the prime numbers, and they are inserted into or deleted from token and tracking account RSA accumulators using modular exponentiation operations. Clients use small, constant sized, zero knowledge set commitments, set membership proofs, and set non-membership proofs to prove knowledge of exponents because it is not practical to transfer these large exponents to the NFT/RSA program. The NFT/RSA program updates the account RSA accumulator state if these proofs of exponentiation can be verified.

The NFT/RSA program creates two types of account.

* Token account. Token mints and burns are tracked by inserting and deleting prime numbers into the token account accumulator. Inserts and deletions can be performed as single or batched operations.
* Tracking account. Token ownership is tracked by inserting and deleting prime numbers into tracking account accumulators. Inserts and deletions can be performed as single or batched operations.

The following table illustrates proof validations and RSA accumulator updates for program transactions.

* Initialize transactions store empty RSA accumulators ({}) in Token (TKA) and Tracking (TRA) account state. The NFT/RSA
* Mint, transfer, and burn ownership transactions update RSA set accumulators ({}) stored in a token (TKA) account, and tracking (TRA) accounts with exponentials (E) that are the product of NFT prime numbers. Insert: {u^E}. Delete: {u^1/E}.
* Upload transactions store NFT prime number (P) elements and public or private URI strings on in the Solana ledger.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transactions | Validations | TKA | TRA1 | TRA2 | Ledger |
| Initialize |  | {u} | {u} | {u} |  |
| Mint | TKA: Verify non-membership proof(E)TKA: Verify membership proof(E)TRA1: Verify membership proof(E) | {u^E} | {u^E} |  |  |
| Transfer | TRA1: Verify non-membership proof(E)TRA2: Verify membership proof(E) |  | {u^1/E} | {u^E} |  |
| Burn | TKA: Verify non-membership proof(E)TRA2: Verify non-membership proof(E) | { u^1/E} |  | {u^1/E} |  |
| Upload |  |  |  |  | [P,URI] |

Clients use their wallet accounts to initialize tracking accounts, and to fund their tracking accounts so that they can pay for program transactions. The NFT/RSA program credits tracking accounts and debits the program account as payment for transaction execution.

# Solana Design Proposals

A prototype of the NFT/RSA program has been developed to test the feasibility or executing NI-POKE2 membership and non-membership proofs on the Solana blockchain. The NFT/RSA program exceeds the transaction compute unit limit (200 000) with the current Solana program costing model due to the high cost of big number operations and hashing operations required to implement the RSA set membership and non-membership proofs. The Solana team would need to accept the following design proposals to make it possible to run the NFT/RSA program on the Solana blockchain.

* Big number design proposal.
* RSA hash design proposal.

## Big Number Design Proposal

The proposal is to add big number system calls to Solana validators so that programs can perform big number operations. The proposal is to use the big number library from the [rust openssl crate](https://github.com/sfackler/rust-openssl) and a costing model similar to that implemented in the EIP-101 and the EIP-2565 precompiles.

## RSA Hash Design Proposal

RSA set membership and set non-membership proofs are implemented using the NI-POKE2 protocol. There are three hash operations in the protocol. The proposal is to add these hash operations as system calls so that programs that use RSA accumulators can execute on the Solana blockchain.

* *Hash to generator*. The hash to generator operation implemented in [accumulator-rs](https://github.com/mikelodder7/accumulator-rs) derives a HMAC key from a Blake3 digest of a proof transcript.
* *Hash to prime*. The hash to prime operation is a loop that generates a blake3 hash of an incrementing proof transcript until the digest is a prime number. The prime number test uses the Miller-Rabin algorithm from the openssl library. This is the most expensive hash in the NI-POKE2 proof and the loop count is non-deterministic.
* *Hash to digest*. Solana has a system call for generating a SHA265 64 byte digest of a proof transcript but the NI-POKE2 algorithm only requires a Blake3 32 byte digest of a proof transcript.
1. Non-fungible token. [↑](#footnote-ref-1)
2. [A Deep Dive on RSA Accumulators (gakonst.com)](https://www.gakonst.com/deep-dive-rsa-accumulators) [↑](#footnote-ref-2)