Offchain and Scriptless Mixer

An offchain mixer for account-based blockchains, similar to CoinJoin for Bitcoin, would provide benefits such as increased privacy over on-chain solutions like Tornado Cash. As well as having stronger fungibility and censorship-resistance guarantees, as centralized exchanges can choose to blacklist all accounts that have ever been traced back to using contract-based mixers. It would also be chain agnostic, as the protocol can be implemented for various curves and signature schemes.

We propose a design that utilizes a threshold signature scheme (TSS) to achieve coin mixing without using a smart contract. Instead, multi-party computation (MPC) is used to trustlessly coordinate users off-chain. This provides plausible deniability since, for the outside observer, such behavior could only be explained as if a common “trusted” party helps users to mix their coins.

The downside of such a design is that the comforting on-chain security is no longer within reach. Hence, for an independent protocol targeting a wild realm of permission-less P2P, worst-case scenarios should be considered when evaluating its safety.

This is a joint work with @noot as a part of ChainSafe Solutions R&D. The full proposal paper is available here:Offchain and Scriptless Mixer - HackMD.

Scenario

Assume there are n

parties who wish to participate in a coin mix. To preserve confidentiality, all parties agree to use the same amount - denoted as 1 coin

. Each party has a fresh withdrawAccount_i

, which they will use for the withdrawal transaction.

Assume also a {t,n}-threshold signature scheme defined by a protocol tuple (KeyGen, KeySign)

. It is a synchronous protocol that progresses in rounds each ending with parties broadcasting intermediary values. Commonly used protocols are GG20, CMP, and GKSS20.

1. Parties run the KeyGen

ceremony to produce a distributed secret key (each of them ends up with its own share), a shared public key, and a derived mixAddress

in which all parties will lock 1 coin

. They also run an offline stage of the KeySign

algorithm (e.g. rounds 1-5 of the GG20) that produces all cryptographic materials needed for a quick 1-round message signing. They repeat this n

times for each withdrawal transaction that will be later signed.

1. All parties lock their coins in the mixAddress

. This step needs to be fair and ideally happen in an “atomic” fashion: either all lock or no one at all. We discuss approaches to ensure this atomicity in the following section.

1. Parties jointly sign n

transactions that transfer 1 coin

out of mixAddress

to withdrawAccount_i

with the nonce nonce_i

. Assuming that the previous phase happened atomically and everyone has locked, then each node can simply release all partial signatures at once, otherwise, each node waits for a lock transaction on-chain and only then releases the partial signature corresponding to the locker’s withdraw address.

1. Each party then combines the partial signatures received to create a valid transaction transferring 1 coin

out of mixAddress

, as it is signed by mixAddress

's private key using TSS. Finally, they submit the transactions in order of nonce to withdraw their funds to their fresh accounts.

Fairness & Atomicity

There is the chance that after taking part in the key generation ceremony, users could back out before locking any funds. Since from a single node's point of view, there is no way to know whether others have backed out or not before they lock, they can lock their funds while others don't. And if a threshold couldn't be reached, then their funds are locked forever.

This is a classical instance of the fair exchange problem. As a solution, we propose another MPC algorithm for emulating a special "fair exchange" channel. Each party can broadcast deposit transactions over it and be sure that either all of them receive transactions or none of them obtains anything useful. It is a two-round subprotocol that builds upon a previously performed KeyGen

ceremony.

1. Each party encrypts their deposit transaction m

with a public key pk

generated during KeyGen

. It will then broadcast this ciphertext ct

to all other parties. If some party refuses to broadcast or fails doing so - protocol aborts.

1. Parties will use their key shares sk_i

to compute partial decryptions \alpha_i

, which they then broadcast to each other. The fairness of this step relies on the byzantine agreement that if t+1

parties act honestly then messages m

can be reconstructed through homomorphic addition m = \sum_i^{t+1}\alpha_i

, otherwise no messages are decrypted.

Finally, each party is now able to submit all deposit transactions on-chain. This seemingly redundant interaction is only needed for parties to be sure that even if a portion of them will go offline, those who are still online will be able to lock funds on their behalf and proceed with a mix.

Collusion resistance

Our main challenge is that for a protocol to meet any reasonable degree of privacy it cannot remain collusion-free. Consider a case when n-1

parties collude in creating a "rogue set". They then wait for one honest party that wishes to mix 1 coin

with them. Parties honestly run the KeyGen

ceremony, signing, and all deposit funds into mixAccount

. They then use an alternative implementation of TSS to compute a signature for the transaction that withdraws all funds from mixAccount

into their malicious account. They then front-run honest protocol execution and submit the malicious signature before the "honest" ones.

Malicious parties will be able to pull this attack in any group where they hold a majority, i.e. t+1

. An adversary could even spawn a large number of dishonest "shadow" nodes to render a Sybil attack. At the same time, it is impossible to tell whether some group is honest or not before starting the mix. To address this, we devised a layered defense strategy that includes the following measures:

- Random mix set allocation.

Selecting parties (from a large pool of players) to the mix sets randomly should make it harder for an adversary to coordinate malicious nodes during the computation. On top of that, nodes would have an ephemeral peer ids that are created at the start of the mix and abandoned after.

- Contract-facilitated fraud prevention.

When joining network users would need to lock some collateral collateralAmount

in a smart contract. If some node misbehaves, like signing a malicious withdrawal message, then a “fraud-proof” can be submitted to the contract in the form of the message that was signed by the bad node. Since the cost to participate scales linearly with the number of nodes the attacker controls, collusion quickly becomes economically impractical. The account that locks the collateral doesn’t need to be the same one that participates in the mix, so privacy is not compromised. The node would simply need to provide a signed message from the account that is locked. This way, users still have plausible deniability; even by collateralizing, there is no way to prove they actually participated in a mix.

- Proof of Minimal Capital.

To make Sybil attacks economically less viable to pull off, a requirement can be enforced for each computing party to prove they own a unique address with a sufficient amount of coins (at least for the mix, maybe more akin to over-collateralization). This can be a ZK Range proof that a certain balance of a certain account is included in a certain block’s state root. Alternatively, a ring signature signed out of multiple public keys whose accounts all own enough funds.

- Incentivized nodes.

A special role in the network could be placed on the “guardian” nodes, which “stake” a certain amount of coins (more than joining collateral) and take some commission out of the mix. These nodes can act as a “dishonesty diffusion” lowering the chance of colluding majority appearing in the mix set. They can also act as a “ballot box”, eg. for putting withdrawal addresses without revealing their connection to the participated user.

The threshold tradeoff

There is a tradeoff between risk minimization (t,n

) and liveness/performance. The higher t

we set, the lesser chance for colluding parties to forge a malicious signature, but this also leaves less room for unforeseen failures (peer goes offline) and hurts performance (see performance analysis). Hence, to make collusion and Sybil attacks economically (or computationally) impractical, we want to maximize t

while still keeping it slightly below n

.

We propose deriving threshold value based on the collateralAmount

that is known in advance, such that (t + 1)*collateralAmount > n * mixAmount

, where mixAmount

is the current mix denomination. With such a requirement any adversary, hoping to control the majority of parties in the set, will be at risk to lose much bigger collateral than it can possibly gain from any malicious activity. Yet, it is also important for the collateral to be reasonable enough for users to actually consider using such a protocol.

Privacy analysis

On-chain, all that is seen is n

accounts transferring funds to some account, then this account transferring funds out to other (new) accounts shortly afterward. No withdraw relayers (like in Tornado Cash) are needed, as the withdrawals come directly from the pool’s account. Although such mixes can still be fingerprinted, as they will always look somewhat unique, it would have plausible deniability of mixing, which is not possible at all with explicitly on-chain mixers.

Due to the MPC protocol’s communication complexity, there is a size limit for the supported anonymity sets. Based on the performance analysis we expect to have sets of size 30-50 users. For reference, Monero has a current ring-signature size of 11 and will soon upgrade to ~30. Tornado Cash has anonymity sets on the order of 10^2 - 10^3.

On-chain mixers like Tornado Cash can allow withdrawal after an indefinite period of time. With offchain analog, however, each withdraw needs to happen in a timely fashion otherwise all other withdrawals will be held up because of the account nonce. The only way is for users to make an agreement to withdraw in an interval fashion.

As a potential improvement, it may be possible to allow new sets of users to “mix into” already existing mixAccount

by leveraging KeyReshare

. This subprotocol allows parties from the old set to rotate key shares with new participants without changing the underlying public key. This should make it even less traceable, but more research is needed to consider adopting this into the design.

Closing thoughts

We introduced a cryptocurrency mixer design different from the existing ones. We replaced smart contracts with an offchain MPC, which can increase fungibility and privacy guarantees while making such a solution more censorship-resistant than on-chain analogs. Collision resistance is the biggest concern of such design, hence we would like to validate our defense strategies before moving further.

There is a considerable future work that involves applying key resharing and improving performance and anonymity set sizes (possible with state-based computation).