Saline Network

The Intentional Blockchain Network

info@risingsealabs.com

June 2024 [Draft]

Abstract

This paper describes the design and implementation of the Saline Network, a new intent-driven blockchain with built-in cross-chain interoperability whose purpose is the introduction and implementation of verification based computing. Saline's universal intents are precise user-supplied rules that describe to our Proof of Stake network's validators what constitutes valid transactions against their wallets. By combining our rich intent and atomic transaction toolbox with on-chain data slots, trustless interoperability infrastructure and native support for zero-knowledge proofs (ZKPs), users and institutions get the ability to safely hold and use any combination of digital assets according to their own dynamic rules, without having to worry about dApps, smart contracts and permissions. From specifying threshold signature requirements to fully trustless automated market makers and exchanges, builders and developers can use the Saline network to enhance the user experience of existing wallets on other networks, or offer various centralised or decentralised services directly on Saline. Signatureless intents give rise to a dynamic and trustless intent matching ecosystem where arbitrary third parties can be rewarded for generating economic activity on the network, without ever compromising the safety of any user's assets.

Contents

1 Introduction

User experience in crypto is perhaps the strongest obstacle to the kind of mass adoption that we have been promised for over 15 years. Despite numerous flavours, iterations and refinements, blockchains still have not solved the decentralised finance problem. The crypto ecosystem is plagued by three major downfalls: complexity, fragmentation and trust assumptions.

We present the Saline Network, an intent-driven blockchain network placing simplicity and the user experience at the very top of its priorities. In contrast to other systems relying heavily on Smart Contracts - and trying to fix them - Saline gets rid of them altogether. The versatile, powerful and expressive intent system replaces most of their usage, with Off-Chain & Zero Knowledge Proofs filling the remaining gaps. Saline emphasises the "express what you want not how to get it" approach, making any user operations easy if not trivial. Users can express their intent, such as "I'd like to swap my SOL for ETH for no less than price X", and broadcast it to the chain, without ever having to think how or where to actually perform the trade themselves.

Fragmentation is solved by supporting foreign blockchains natively: Saline allows you to reason about other L1 or L2 chains thanks to its network of Observers. The user onboards their funds from any chain to their Saline Wallet, takes part in any intent-backed interactions, and withdraws their funds to any chain, natively. We aim to reduce interactions with other chains as far as possible. The need for trust in the current crypto ecosystem seems more and more misaligned with the initial promises: back in 2008, Bitcoin's philosophy was about avoiding the need for central banking systems and enabling fully decentralised, peer-to-peer finance. 15 years later and the ecosystem relies in great part on centralised exchanges, centralised apps and centralised services that need people to trust them and relinquish custody of their funds to opaque third parties.

Saline aims at correcting this by bringing utility and custody together, with the user remaining in full custody of their funds throughout. Efficient and low cost trustlessness is achieved mathematically using Zero Knowledge Proofs everywhere possible, from Observers watching the foreign chains down to the heart of the intent system itself.

This paper explains and presents the Saline Network in detail, justifying our choices and laying out the groundwork of the technical stack. The first section presents Intents, the core concept at the heart of the Saline Network. The second section presents and details the main individual components of the Saline Network: the intent language itself, the intentional wallets, the blockchain, the validators and the cross-chain capabilities. The third section showcases some application examples, the fourth section explains the security and the game theory of the system. The fifth and last section talks about the future works.

Intent-based solutions are a growing and ever more popular field in crypto, because they allow for so much more flexibility that a standard transaction-based one. Our approach is novel in that we are not trying to upgrade an existing system with (a form of) intents, which usually requires downgrading the benefits that intents can bring. Instead we made two crucial observations: 1. to maximise the benefits and strength of intents, they must be as general as possible and 2. the existing crypto ecosystem is already enormous and it would be foolish to try and replace it, so we must integrate with it.

These observations led us to design a network from the ground up: a blockchain, a set of validators, a set of economic rules, reward system, etc. all built-around (vs. made compatible with) around intents, that natively integrates very well with foreign blockchains. This paper goes through these individual components comprising our network.

2 On Intents

2.1 What are intents, anyway?

Intents playing such a crucial and central role needs proper definition. If we consider a blockchain as an (extremely) distributed database, acting as a ledger, then our primary means of acting with this database is by submitting transactions which mutate the current, global state. The state that is being maintained is global and (usually) visibly accessible to anyone. A condition for this system to work is that not everybody can modify the state through arbitrary transactions: a set of rules enforced by all nodes running the chain will validate that a transaction is correct and thus can be incorporated in the blockchain (and thus mutates the state). The important question becomes thus: what makes a transaction valid?

Details will obviously differ from blockchain to blockchain, but one of the arguably most important checks that must be made is that for a transaction that mutates the state by changing an entry's value (e.g. spending or allocating funds to a wallet) the owner of said wallet is the author of that transaction. A system where anyone could change anyone else's balance would be pointless, so how can the system know the change is intended? The typical ("old way") of doing this, used by virtually every blockchain, is through the use of cryptographic signature: every actor has a pair of public and private keys which they use to sign a transaction they intend the chain to incorporate. In the same spirit where the use of a credit card to purchase goods is protected with a pin code that (supposedly) only the owner knows, a blockchain restricts using an account only to those who know the private key for it: every transaction on the blockchain is then signed by the wallet's private key, and the validators can easily check whether the signature passes or not.

This system obviously works as hundreds of blockchains are based on this. It is not, however, the only choice, and through this paper we hope to showcase that neither it is the ideal one. The fundamental issue with signature-based validation is that, while it works (it proves the owner of the wallet agrees with the transaction), it actually gives stronger guarantees (and thus imparts more limitations) than intended: when the author of the wallet signs a transaction, what it really proves is that they are the author of the transaction. Being the author of a transaction obviously means they intended the transaction (otherwise why submit it at all?), but it doesn't have to be the other way around. If we had a way for the owner of a wallet to prove they agree with a given transaction, even if they did not submit it, then the system should still work.

That's exactly what Intents achieve: they give a way for wallets owners to express what they intend, what they agree with, and then any transaction (submitted by them, or not!) that matches is considered valid and can thus be incorporated in a block (without necessarily needing their signature!). This is truly a paradigm shift, because it unlocks entirely new and vastly broader operations: suddenly, signatureless transactions become not only possible, but the norm. In practical terms, a wallet owner will install an intent (or more) on their wallet, and broadcast them to the chain: the installed intents will be kept in the ledger and synchronised among validator nodes, exactly like any other data, and will be used as part of the transaction verification process. In Saline, we go even further by saying that intent verification is the transaction verification process. The beauty of this is that it is trivial to implement the old way of validating transactions (should this be needed) with the intent system: the owner of the wallet just had to install an intent that says transactions need his signature to be valid.

We will detail every individual components and aspects of the chain in further sections, but the key part is that everything is integrated and part of the same ecosystem, allowing Saline Network unprecedented synergy and field of action for its first-class intents.

2.2 A universal intent abstraction

Intents being fairly new in the crypto system, they come in varied forms, shapes and flavours. If the overall goal of simplifying blockchains interactions is the common factor, the spread of what they allow and restrict differ vastly from projects to projects. In Saline Network, we are aiming to be as generic and as abstract as possible, without compromising on usability and cost. Contrary to a good portion of the competition, our intents are not specialised to perform one or two types of operations (usually swaps, exchanges) because they operate at a lower level. Our intents are only about providing constraints to restrict what is deemed acceptable.

Constraining on inbound addresses allows users to set up allow and deny lists to receive funds, while

constraining on outbound addresses allows users to target customers and make custom payment rules. Constraining on amounts can leverage risks, set up levels of responsibilities. Constraining on time allows making positions expirable and protects against drifting too far from the market. And perhaps more importantly, the ability to compound transactions, i.e. allow a transaction to happen only if (a set of) another happens simultaneously opens entirely new dynamic approaches that could only be dreamed of by completely closed and centralised instances or overly complicated Smart Contracts.

That is the sense that we put into our universal approach: Saline doesn't create a narrow and restricted use-case when intents can be used, instead they make intents part of the rules so that anyone can create arbitrarily complex constraints, as long as they can express them in terms of intents.

3 The Saline Network

3.1 Wallet at the Core

The Saline Network is comprised of several components, the most important and innovative being the Wallet. In almost all other blockchain networks, the wallet is an implementation detail, reduced to a pair of public and private keys. In Saline, the Wallet is the central part.

They are natively multi-currency so all of the funds onboarded in Saline, from all the L1 or L2 chains can be held in the same wallet, without the need for over-complicated setups. They greatly facilitate swaps and trades when they are realised on the Saline chain.

Wallets are also where intents are installed. Contrary to some other projects, we do not need a separated, permissioned and centralised intent pool. Instead, intents are completely integrated to the system, first-class citizen directly attached to a wallet. This allows blockchain explorers to display the list of installed intents on every wallet. This implies transparency and consistency, and is part of the User Experience first mantra: no need to register to an external service, no need to link the wallet and relinquish permissions or control. This also makes it easier and more efficient for the validators to check transactions against intents as everything is on-chain.

In a sense, one can view Saline as a custody project, providing on-chain committed, universal intents and native cross-chain compatibility. The cross-chain aspect comes from the fact that we can think of every Saline Wallet as a scattered part of a huge decentralised and trustless cross-chain bridge: to onboard funds into Saline, users simply transfer their funds on a Smart Contract on their L1 chain and their funds directly appear in their wallets ; there is no intermediate bridge. In the same spirit, to redeem funds out of Saline, users only have to submit a "redeem transaction" to the chain, which once finalised, will issue the Smart Contract on their L1 chain to credit their wallet with the funds: the process is direct, immediate and doesn't involve third-parties.

3.2 The Saline Architecture

Figure 1. gives an overview of the Saline Chain architecture. The main steps have been annotated for clarity.

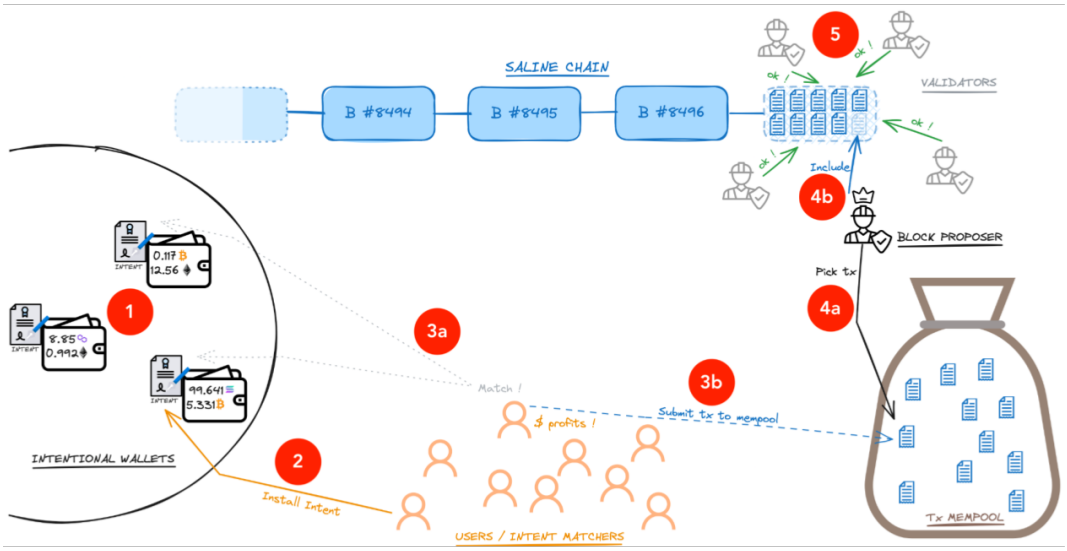

Figure 1: The Saline Architecture

- Step 1: the Smart Wallets (also called Intentional Wallets) are the centrepiece: they hold users funds, are multi-currency and most importantly, they receive intents installed by users.
- Step 2: most interactions will start by a user installing an intent on his wallet. As a reminder, an intent is a set of rules and restrictions that dictates what a valid transaction should look like to act on this wallet ; this can be a swap, a transfer, a loan, an AMM, it can be loosely restricted

or on the contrary very restricted so that nothing but a very precise transaction goes through. Constraints can be amounts, time, counterparties, rates, yields, etc.

- Step 3:
  - 3a: a user (can be any user, or a program, a bot, etc.), called a matcher notices that two (or more) user intents are compatible with one another, i.e. the realisation of one would realise the other
  - 3b: that matcher submits a transaction (or a set of bundled transactions) that matches those intents to the transaction mempool
- Step 4:
  - 4a: the current block proposer ("miner"), sees this transaction in the mempool.
  - 4b: they decide to include this transaction (or set of) in the current proposed block
- Step 5: the current block (with the transaction matching the two (or more) intents) has been included in the block, and the blockchain's validators vote on the block to finalise it.

A very important aspect to understand, which sets the Saline Chain apart from the others, is that Intent Verification is the validation process: this is not a process that comes "on top of" anything, or as "an addon". The validators' job is to verify that transactions included in the current proposed block correctly match intents (and a few boilerplate formatting and versioning checks).

In other words, the Saline Chain acts as a verification network, unlike the Ethereum Virtual Machine, which is a network for computation.

The following sections will look at the different components of the Saline Network in more detail.

3.2.1 Core concepts and abstractions

A few concepts found throughout the Saline stack are described in this section: addresses, transactions and intents. Those three concepts alone have the responsibility of replacing the entire usual Smart Contract based paradigm, both when it comes to implementing programmatic financial interactions ("code is law"), but also as cross-chain endpoints, for locking and unlocking assets under specific circumstances..

Addresses

Addresses are either Saline addresses, sal:xxx, committed Saline addresses salc:xxx, data addresses (data:xxx), or foreign addresses that live on another blockchain (eth:xxx, btc:xxx).

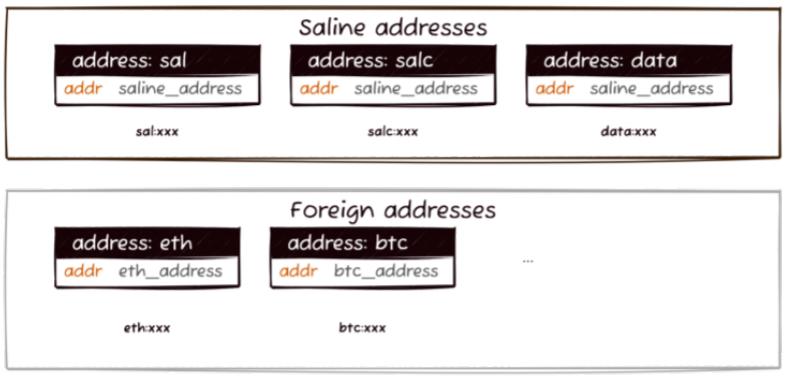

Figure 2: Addresses Representation

Saline addresses can hold and have intents related to any combination of assets from all supported networks

Foreign addresses can only hold and have intents related to the assets supported on their network

Transactions

Saline transactions are where a lot of its power resides. In addition to usual transfer transactions, Saline also comes with atomic swap transactions as a primitive, as well as different flavours of compound transactions. The additional structure allows us to express and enforce complex interactions or "protocols" between various parties, without the use of a shared distributed computer. The Saline network could instead be viewed as a shared distributed verifier.

Primitive transactions

Primitive transactions are transactions that affect the balance or installed intents of some address(es).

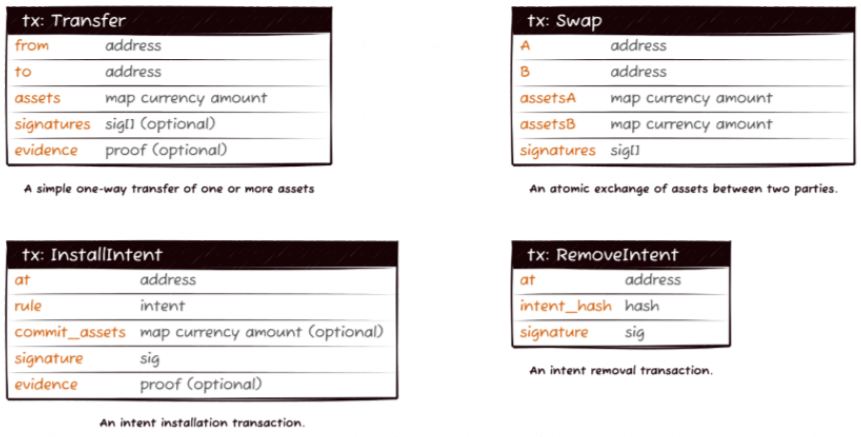

Figure 3: Primitive Transactions

These are the bread and butter of Saline transactions and are at the heart of most of the activity.

Simple transfers are the usual one-way movement of assets from a sender to a recipient, but are also the backbone of asset onboarding and offboarding. Transfers:

- between two Saline addresses: do not require any evidence.
- where the from field is a foreign address and the to field a Saline address: asset onboarding transactions; should come with valid evidence of the funds having been locked in the Smart Contract of the foreign chain, produced by the cross-chain infrastructure.
- where the from field is a Saline address and the to field is a foreign address: asset offboarding transactions, do not require any evidence, will get propagated to the corresponding network to unlock the appropriate funds and send them to the to address.

The reader will notice that onboarding and offboarding (bridging in and out of Saline) is a native operation in Saline, and is generally much easier and requires much less trust than in any other network.

Swaps are the expected atomic exchange of assets between two parties, where on Saline both parties can bundle any combination of assets together in their leg of the swap. Swaps can only occur between two Saline addresses, or two foreign addresses from the same network.

Intent installation and removal transactions are the ones affecting the list of intents attached to a user's wallet: the former adds an intent to the list while the latter removes one. A user can simply install an intent but also optionally commit their assets, which means they should be considered locked by the blockchain (think held in a programmatic escrow), and not allowed to be used in other transactions. Intents can also be attached to foreign addresses, hence the evidence field. The rule field contains a compact binary encoding of the intents described in the next section.

Transaction sequences

Sequences of transactions let you tell Saline that it should validate those transactions as an atomic (all or nothing), ordered whole instead of transaction-by-transaction.

Figure 4: Transactions sequences

They can be used to eliminate counterparty-risk (bundling several legs of a larger operation together), install-and-fulfil intents and more. Sequences can only be made of primitive transactions, preventing arbitrary nesting to keep intent verification efficient.

Gated transactions

Gated transactions let you express non trivial request-response patterns that will be validated atomically.

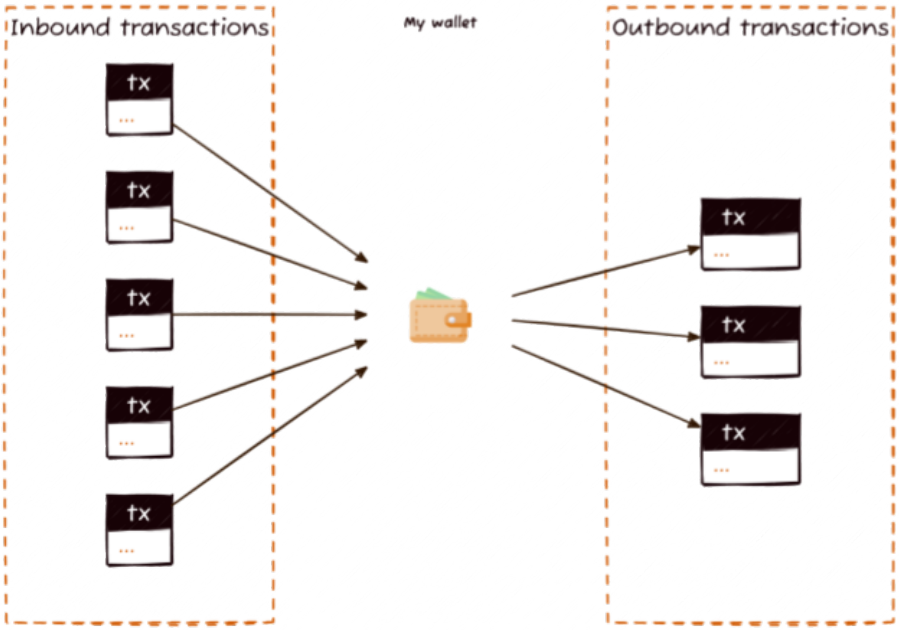

Figure 5: Gated transactions

Such a transaction is used to fulfil a corresponding gated intent, which expresses the wallet owner's approval of outbound transactions satisfying the pattern described in the intent upon receiving inbound transactions satisfying the pattern described in the intent.

By *inbound*, we mean a simple transfer where the wallet owner's address is the to field, or a swap where the wallet owner is one of the parties.

By *outbound*, we mean simple transfers where the wallet owner's address is the from field, or a swap where the wallet owner is one of the parties.

Data transactions

Data transactions are used to create, update and remove on-chain data. Every Saline address can start posting data that can be used in intents to guard the approval of transactions.

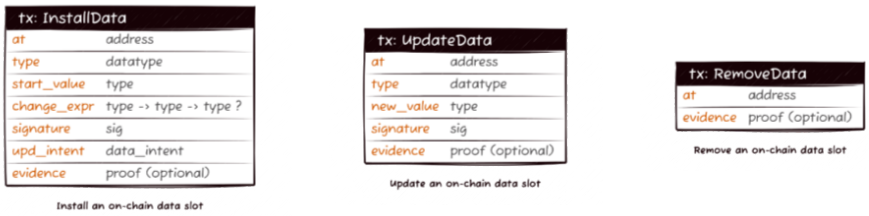

Figure 6: Data transactions

Protocol transactions

Finally, Saline also has transactions related to turning off/on support for onboarding/offboarding and assets from particular networks, announcing upgrades, validator set changes and a few other similar mechanisms.

Extra transaction data

All transaction must specify a submitted-by field. All transactions can additionally specify a nonce. Transfers and swap can additionally specify a for-intent field saying that the tx should only be accepted as part of a gated transaction that fulfils the intent with the hash given by for-intent, and cannot be validated on its own. Transactions sent by matchers should also have a submitted-by field to indicate where to send the matching rewards.

All transactions can come with additional data, and intents are free to place additional expectations on such fields.

Intents

An intent is a user-supplied rule that describes a pattern of transactions that a user approves. More formally, a Saline intent is the given of:

- a **transaction shape** (simple transfer, swap, gate)
- a **set of constraints** relating the values of different fields in the transaction(s) together
- some **metadata** (tip and time validity information)

Transaction shape

- **Transfer:** *"User is willing to accept a simple transfer, if < constraints >"*
- **Swap:** *"User is willing to approve a swap, if < constraints >"*
- **Gate:** *"User is willing to approve some outbound transactions given some inbound transactions, if < constraint >"*

Set of constraints

Constraints can target and relate various fields from the transaction, as well as from different transactions in the case of gates, using arithmetic, comparisons, list and set operations, cryptographic signatures verification, zero knowledge proof verification and much more.

Constraints can also make use of on-chain data, e.g "user accepts any inbound transaction of any asset as long as the total value is above $100", where the price data is taken from an on-chain data slot.

Metadata

Saline intents currently come with two additional pieces of data. The first piece is time validity information, where a user qualifies his intent with:

- **no limit:** the intent can be fulfilled infinitely many times in principle, has no time validity constraints

- **fulfil once:** the intent is automatically uninstalled once fulfilled (e.g a quick swap to gain exposure to a new asset)
- **frequency:** the intent can be fulfilled at a given time frequency (hourly, daily, ...)
- **time window:** the intent can only be fulfilled before, during or after a given time or time window

The second (optional) piece, only applicable for **signatureless intents**, is about specifying a tip that the user is willing to give to a matcher fulfilling this intent.

Example

- any simple transfer transaction signed by that user
  - simple transfer + signature constraint

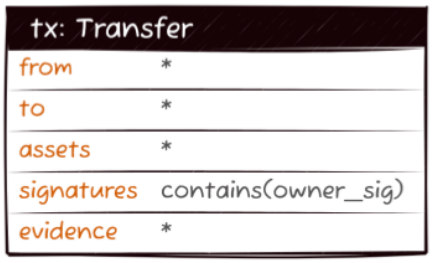

Figure 7: Simple transfer

- any swap transaction taking 2 BTC from the user and giving him at least 36 ETH in return
  - swap + constraints on assets

Figure 8: Swap

- any gate transaction that looks like a verifiable 3-participant lottery (using ZK for verifying that the payout transaction goes to the right participant, given some random generator and seed the organizer commits to)

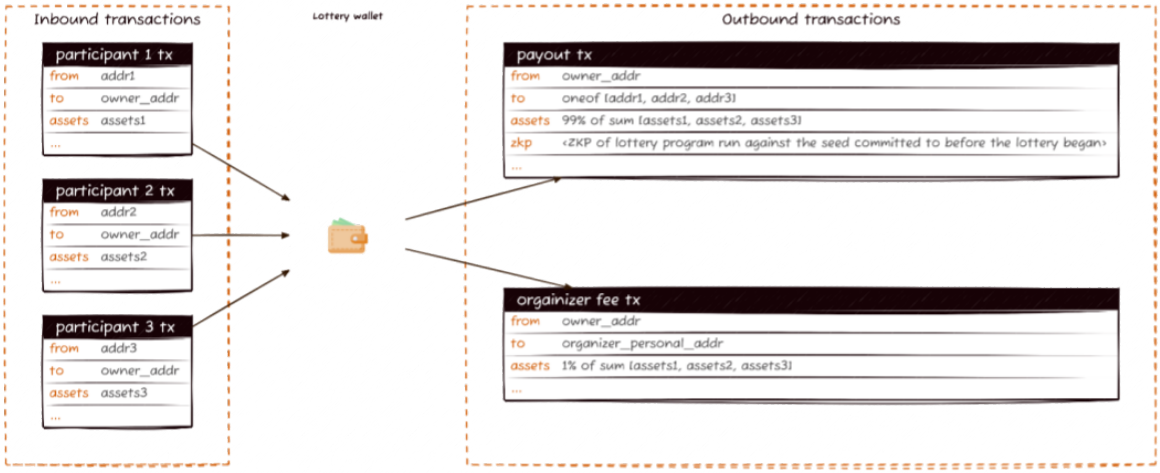

Figure 9: Gated transactions

Intent verification

Intent verification consists in establishing whether a given Saline transaction fits the pattern described by an intent.

In order for a transaction to fit an intent:

1. The transaction's shape must fit the shape required by the intent.
2. The constraints given by the intents must be satisfied when instantiating the constrained variables with the transaction's data + any on-chain data used by the intent with the current values.
3. Potential time validity restrictions specified in the metadata must not be violated.

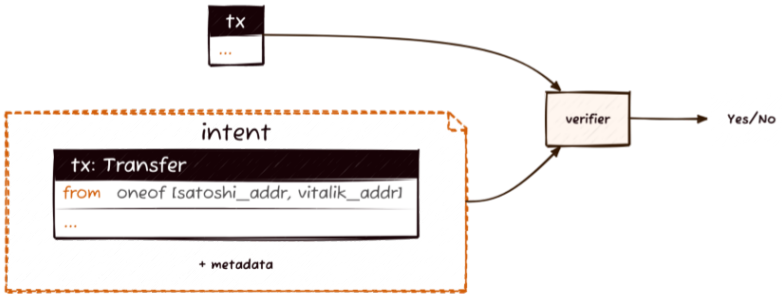

Figure 10: Intent verification

Multiple users, multiple intents

Users of the Saline network maintain a list of intents installed at their address, and can even send them around to third parties who can help them reach the desired outcome.

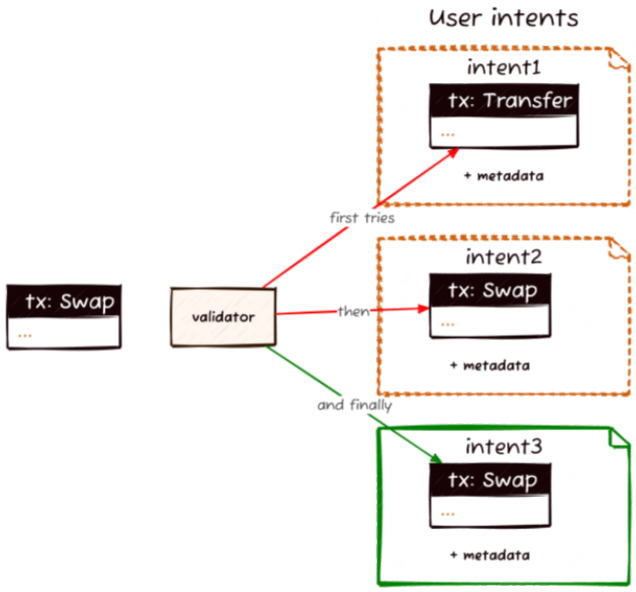

Figure 11: List of intents installed

When a transaction is submitted to the network, anyone who wants to check the transaction's validity needs to lookup the installed intents of the parties involved in that transaction, and check whether the transaction fits any of the installed intent.

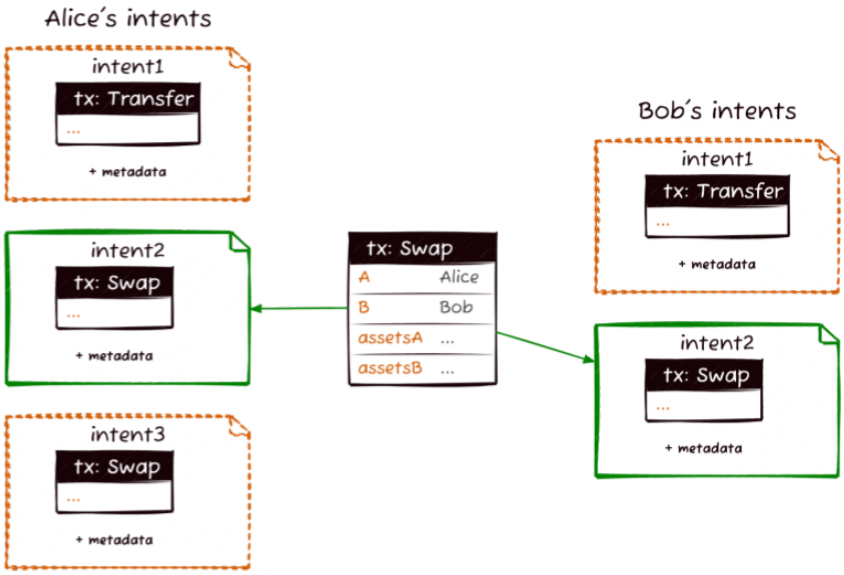

Figure 12: Check whether the transaction fits any of the installed intent

Unless the transaction has a non-empty for-intent field (hash), *the first intent of the list that allows the transaction in* is considered to be the one fulfilled by the transaction.

Onboarding and offboarding of assets

Conceptually, onboarding and offboarding of assets is represented by simple transfer transactions whose from (respectively to) address field is a foreign address.

An onboarding transaction must additionally provide valid evidence, to certify that the corresponding assets have been locked in the source blockchain.

Figure 13: Onboard and Off-board transactions

3.2.2 The wallets

The technological choices for a blockchain's wallet can be tricky and is generally the result of a careful balance between security, usability and the set of features that is required for the chain (is it needed to hide the public key and generate keys on the fly, like Monero does? Is it needed to support HD wallet for sub-wallet generations?).

Our choice was influenced by several factors:

- We want to account for quantum computing gaining power in the next few years;
- We want users to be able to recover all of their wallets with a single passphrase for a better user experience, which implies a key-derivation scheme.
- We believe in the extensive use of threshold signatures scheme made trivial to deploy by intents: we needed an efficient way to aggregate signatures.

These decisions led us to choose BLS12-381 for the curve governing Smart Wallet keys, following more or less closely EIP-2333.

Security

This choices gives us quantum backup though the Lamport signatures, it gives us a tree-structure for generating keys from a master seed/passphrase and it gives us constant space key aggregation, which coupled with a small-size indexing scheme allows us to efficiently and cheaply verify threshold signatures, which we believe will become very widely used in Saline.

Flexibility

Saline Intents are installed and attached to a Wallet, this allows the greatest amount of integration and flexibility. Depending on their constraints and phrasing, intents attached to a wallet can greatly modify their behaviour. This encourages using service-specific wallets: users in Saline will routinely generate and handle several Smart Wallets ; potentially dozens as their use cases will expand (think having a shared-custody wallet, a wallet acting as an AMM, a wallet for limit position, a wallet to receive payments for a business, etc.). The goal is to make it very simple for Saline users to spin a new address and attach a few intents to them. It would be hugely impractical if they had to remember as many passphrases to secure, migrate or recover all their wallets. With this choice following EIP-2333, only a single passphrase is needed to restore the master wallet and all subsequent sub-wallets.

User experience

In traditional blockchain systems, the wallet's private/public key pair is used for every transaction (because the user signs every transaction with the private key and the validators verify the signature against the wallet's public key). In Saline however, it is likely the key pair will be used much less: thanks to the signatureless intents, the user should need to sign (and the validators need to verify the signature) much less often. Creating and installing an Intent does require signature and verification, but besides that, signatures are not needed, unless specifically required by the user (through an intent that explicitly requires signatures).

3.2.3 The blockchain

The Saline blockchain is the backbone of the entire system and as such requires careful consideration and design. This section details the inner working and design choices of the Saline blockchain as a decentralised intent verification engine and goes through its major components.

Blockchain nodes and validators

As almost everything relies on Intents, the details of the chain itself (as a protocol) need not be particularly complex. The Saline chain implements its own application layer on top of CometBFT for consensus. Validator nodes will gossip among themselves to vote and validate for inclusion blocks of transactions, as usual. CometBFT gives us very fast finality so transactions matching intents are not held pending for extended periods of time. The desirability of this property comes from the time-sensitivity of some intents along with the general desire from users of being able to manage and use their assets as promptly as possible.

The number and repartitions of (full) nodes is an important decision in any serious blockchain design. Saline will be a utility blockchain first, not a speculative one. As such we opted for an initial fixed and relatively small number of nodes: about a hundred. For bootstrapping and stability reasons, we will most likely run the majority of nodes at first all while being very transparent to the community about it. Once adoption is high enough, we will be gradually releasing validators seats among those first 100 to community members, partners, investors, customers, etc. in the aim of reducing our control over the chain: we aim to reach decentralisation as soon as possible to prevent anyone (including us) exercising any actual control over the chain but without compromising on security. We plan to lower our stake in validators seats and cap it to between 10 and 20%: not high enough to be able to exert control over the chain, but high enough to earn revenues from validator rewards. After having successfully managed to lower our position to about 10-20 seats of the 100, we will hold auctions every year or so to sell additional validator seats, thus allowing new validators to secure (and earn from) the network. The plan is thus to have a growing network of validators, but at a lower pace when compared with networks such as Ethereum with an open pool. It is also our wish to keep validator yield high, higher than in other, traditional blockchains to incentivise faithfulness, uptime and this security.

Where are the smart contracts?

Smart Contracts are the tools that gave blockchains the ability to perform computations whose results can be guaranteed and secured with the same properties as those for the transactions. This is why Ethereum was called "the world computer". It enabled this distributed system to act not only as a database, but also as a computing environment. Smart Contracts come with their own set of issues, however. The peculiarities of the validation system and the guarantees that they promise led the design of the Smart Contract languages. They are, by nature, restricted and not easy to use. Furthermore, the immutability of deployed Smart Contracts and the stakes involved make the possibility of failure a real challenge to designing non-trivial programs. Finally, each node needs to run the Smart Contract to come to the same answer (to be agreed upon via consensus next), which is inefficient.

In the Saline network, there are simply no Smart Contracts. While a good deal of what would call for a Smart Contract in traditional blockchains can be realised with an intent, the remaining portion is replaced by proofs of correct execution, specifically Zero Knowledge Proofs (ZKP) of verified computation. In the design of the Saline Network and chain, Zero-Knowledge Proofs are first-order citizens, which means an intent can specify to take as input a ZKP. From this, we naturally replace Smart Contracts' costly and lengthy executions by an off-chain Zero Knowledge program execution whose proof of execution is posted on-chain.

The on-chain computing environment only needs to verify those ZKPs, which can be done quickly, efficiently and cheaply. This enables users to run complex, costly and lengthy-to-execute programs off-chain by simply posting a ZKP of their execution to the chain, thus getting around the efficiency, cost and complexity limitations of traditional Smart Contracts. The burden on the chain is greatly reduced: instead of having the validators run the program (or Smart Contract) entirely, they only will verify the proof.

Intent as model, which Saline implements

Saline intents can be thought of as "models", shaping a desired execution which is carried out separately. The user defines the space of future transactions which they are happy to accept, without

having to worry about the actual execution, such as how to handle complex permissions that secure digital assets, or finding the best path to navigate. This presents several advantages:

- Users get to define the rules for their assets directly, instead of whitelisting and trusting arbitrary dApps and smart contracts that claim to have implemented something that looks like their intent.
- Users retain custody of their assets instead of handing them over for the duration of the action (and stepwise transaction flow) required.
- More can be done thanks to the greater expressivity inherent in not having to define implementation of every component of a model.
- Because the chain is merely a verification layer, rather than a full fledged execution environment, it is simple, fast and cheap to use, whilst being more secure thanks to a smaller attack surface and greater intuitiveness.

3.2.4 Going cross-chain

Whilst this section technically is part of the design of the chain, it is sufficiently important to warrant exploring in depth. The existence of other ecosystems and their assets is baked into our network's design. The core mechanism through which information about other chains is passed to Saline is via zero knowledge proofs of correct computation, which allow provers to run off-chain and still demonstrate that a node of the remote chain was run exactly as requested and that the transaction on that chain was indeed included in the canonical history of said chain.

Zero knowledge proofs are a data type in Saline intents and are the means of off-loading complex computation from the chain. Because the proof cannot be faked, just as with intent verification at a higher level, it does not matter where the code that produced the proof was run.

Aside from remote chains' nodes, examples of programs you might want to run off-chain then include in your intents are:

- A centralised exchange's margining, proving to a user that the exchange has the right to margin call their wallet for a certain amount, by computing the user's P&L and margining calculation in a provable manner verified on Saline. This enables decentralised custody.
- Automated trading or financial services infrastructure, because any corruption of the code thereof will generate a different proof when run. This protects institutions against insider attacks, supply chain attacks and other unauthorised changes to the code that makes their financial decisions.
- A video game tournament might make an on-chain prize system conditional on proof of winning, which can be obtained by playing in a proof-emitting environment. This has the added benefit of acting as an anti-cheat mechanism - players can only run the exact game code provided by the tournament, as any modifications would void the proof emitted.

Remote transactions can thus be authenticated with different mechanisms, each coming with some payload that the prover needs to include, such as signatures or aforementioned zero knowledge proofs.

In case a remote network splits into various forks, special governance transactions could be used to pause on- and off-boarding of assets related to that network.

Increasing trustlessness (signatures vs. proofs)

The network will initially launch with trusted watchers for cross-chain communication. The watcher will sign the Ethereum data ("transaction T took place on Ethereum in block B") and the signature will be verified on Saline.

We will then move to using proofs to send the same data in a more trustless manner: "Here is a cryptographic proof that transaction T took place on Ethereum in block B". As mentioned above, because the proof cannot be faked, it does not matter if the prover is trusted. The proof will then be verified on Saline.

Crucially, both schemes can coexist, because both simply reflect a different type of "evidence".

Block data vs self-contained proofs

The simplest approach to prove transaction inclusion, common to many protocols today, consists of proving that the transaction can be found in a specific block on the remote chain that we commit to. Saline nodes only have to synchronise block height and hash information to verify the claim.

We can go further and provide a self-contained proof, whose sole verification proves transaction inclusion. This requires encoding the remote chain's transaction and block validity logic in a ZK program or circuit, which is very expensive, but provides the greatest safety and does not require Saline to keep a cache of block data.

Onboarding assets onto Saline

The system is agnostic to the cross-chain mechanism (trusted watcher, or zero knowledge prover).

1. Send assets to the relevant smart contract or wallet on the remote chain. Include the local (Saline) address in which you wish to have the assets credited.

2. The cross-chain mechanism will observe this transaction, and create the relevant evidence - signature, zero knowledge proof - before sending it to Saline via RPC.

3. Saline verifies the evidence on-chain, then credits the assets to the target local address.

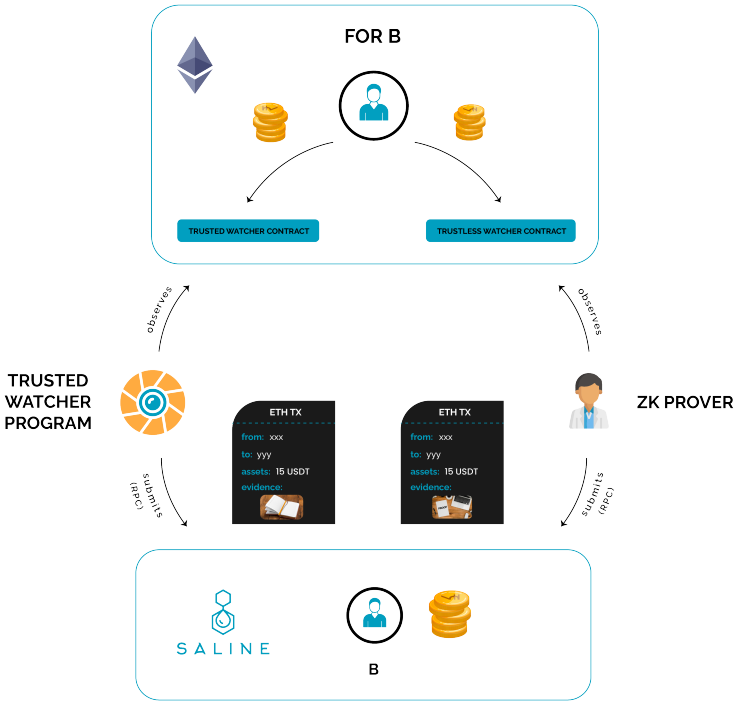

Figure 14: Onboard asset onto Saline

Redeeming assets from Saline

The process is even simpler, and once again mechanism-agnostic.

1. Make a local to remote address transaction on Saline.
2. The cross-chain mechanism will observe, create evidence, and take action on the remote network to credit your assets to your remote address, from the (Saline) remote address where they were locked.

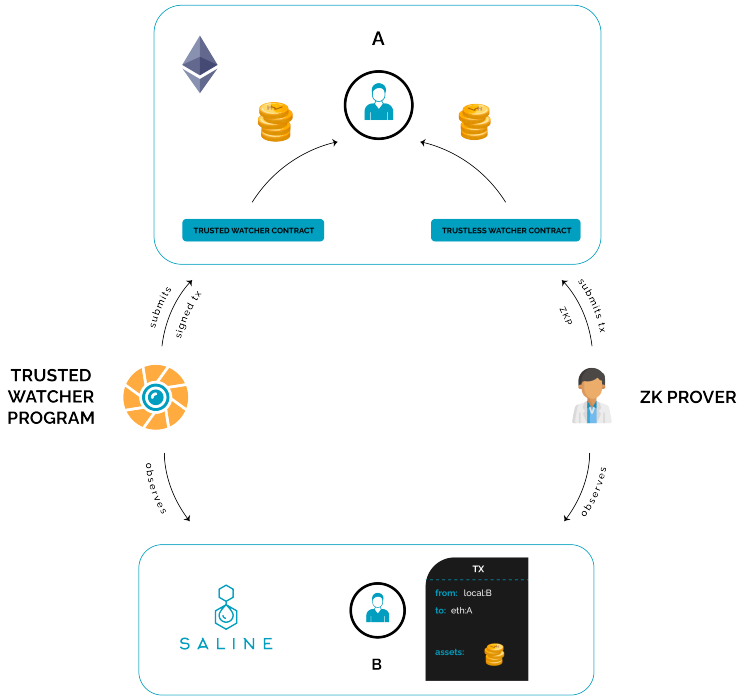

Figure 15: Redeeming asset from Saline

3.2.5 Intent matching and validation

Intent matching is somewhat equivalent to what other protocols call solving (with matchers replacing solvers), and perhaps mimics market-making on exchanges.

Matchers

Because assets can be onboarded or redeemed in Saline transactions, matchers are able to leave the intent pool concept behind and match intents with liquidity sitting outside Saline, for example on centralised exchanges like Binance. This also allows anyone to be a matcher, regardless of inefficiency or capital.

Thus matchers can look either like "solvers" matching intents within Saline or like liquidity providers, bringing assets onto Saline to match intents that do not yet have a counterpart within the network.

Matchers receive a fee which exceeds the cost of submitting the transaction. The more complex the service, the higher the fee. The fees are collected from the matching tips specified by the users who created the intents that matchers fulfil.

Matchers are free to decide what subsets of intents they are looking into fulfilling and tailor their RPC queries to nodes with filters in order to only discover the relevant intents.

Finally, it should be noted that there is absolutely no trust assumption made in any of the interactions between users, matchers and blockchain nodes. Users can make their assets available for some operation and matchers can generate such operations, without at any point asking users to hand over the custody of their assets to a third party. The only valid use that matchers can make of user intents is to fulfil them, since the blockchain will reject everything else.

Validators

As a transaction request is sent to validators, they verify that the transaction matches the intents on the wallets involved in the transaction. For this service, they are also paid a fee, which exceeds the cost of running the relevant verification procedure.

Transactions also send a small fee to the chain itself, which goes towards chain infrastructure, development and maintenance costs.

4 Applications

4.1 Simplifying cross-chain asset management

4.1.1 Cross-chain Trustless Settlement

OTC cross-chain transaction settlement has poor efficiency and security. Traditional mechanisms often hinge on the negotiating power of large market players, raising concerns regarding the initiation of (irreversible, on-chain) transactions (e.g. "taker pays first" when dealing with market makers).

Saline enables a transformative approach to settling trades, allowing two parties to autonomously complete transactions without reliance on third-party trust. This method eliminates the question of payment sequence and counterparty withdrawal midway through a transaction.

Unlike conventional platforms that employ escrow services, potentially subjecting transactions to the behaviour of the escrow, Saline enables users to directly set transaction intents within their wallets. These intents detail the transaction amount, rate, expiry, and other necessary parameters, which the counterparty can simply review and accept

Illustrative Example:
Alice wishes to settle 10 BTC for 100 ETH with Bob, to be settled within the next 2 hours.

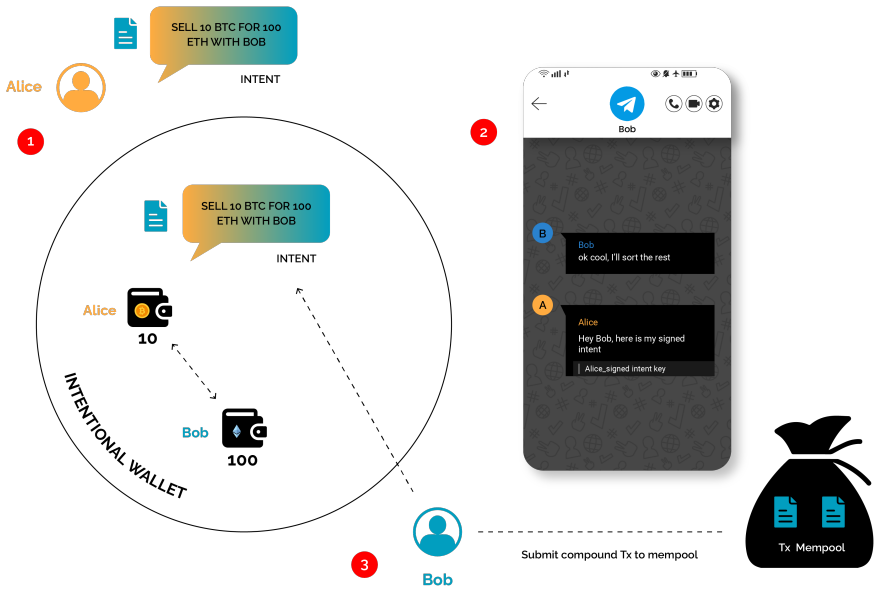

Figure 16: Cross-chain settlement

1. **Intent Creation:** Alice formulates an intent specifying the desired exchange pairs, the amount, and the counterparty — in this case, selling 10 BTC for 100 ETH to Bob. She signs this intent and forwards the signed intent data to Bob.

2. **Intent Execution:** Upon receipt, Bob's responsibility is to facilitate the transaction:

   - He instals the signed intent onto Alice's wallet, which enables him to transfer the 10 BTC from Alice's wallet without requiring her manual approval.
   - Subsequently, Bob initiates a swap transaction from his wallet, exchanging his 100 ETH for the 10 BTC from Alice's wallet.
   - He then submits this compound transaction to the blockchain's mempool for processing.

Extended Applications:
Saline's framework is not limited to transactions with known counterparties but also extends to group settings. For example, users can create private intents accessible only within specific groups, such as a private Telegram chat. This ensures that the intent remains confidential and is only executable by approved participants within the group, enhancing privacy and security until the transaction is fully processed.

4.1.2 Cross-chain asset deployment

Saline not only simplifies the transaction process across different blockchain networks but also enhances the strategic deployment of assets across these networks. This capability allows users to concentrate liquidity effectively and utilise Saline's innovative intent logic for seamless cross-chain asset deployment.

Example:
Consider scenarios where market opportunities arise from price discrepancies or maintenance margin requirements:

- If the price of ETH in the ETH-USDC liquidity pool on Uniswap deviates from its corresponding central exchange (CEX) price by more than 2%, Saline can autonomously deploy assets to capitalize on this arbitrage opportunity.
- Similarly, if the maintenance margin for an ETH-USDC position on DYDX reaches 6%, Saline can automatically top up the user's ETH wallet with 10,000 USDC to maintain the position.

Technical Insights:
These operations are contingent upon the availability and reliability of data from oracles, which Saline integrates to obtain real-time financial information. Despite the complexity that might arise from varying data sources, Saline's intent abstraction remains simple and chain-agnostic. This simplicity is key to enabling users to set conditions or 'triggers' that activate specific asset deployments when predefined criteria are met.

By leveraging Saline, users gain a powerful tool for managing their digital assets more proactively across multiple ecosystems. Saline acts as a central platform where assets can be strategically deployed to exploit financial opportunities, manage risks, and optimize returns in real-time.

4.2 Trustless liquidity pools, AMMs, index funds, exchanges

4.2.1 Liquidity pools & AMMs

In the DeFi landscape, liquidity pools and AMMs play a crucial role by enabling the permissionless and automated trading of digital assets. While Saline's swap intent mechanism, facilitated by solvers constantly listening to intents, might reduce the necessity for traditional AMMs, integrating liquidity pools directly within the Saline ecosystem provides an alternative solution for both matchers and users.

Example: USDC-ETH liquidity pool

1. **Swaps using constant product AMM model:** In scenarios such as a USDC-ETH liquidity pool, the AMM employs the constant product formula

$$x * y = k \tag{1}$$

ensuring that the product remains constant post-transaction where

$$k = \frac{balance(ETH)}{balance(USDC) + balance(USDC_{swap})} \tag{2}$$

For a user swapping USDC for ETH, an intent verifies whether the swap preserves this constant product. If the criteria are met, the transaction is executed accordingly.

Intent:

$$y = \frac{balance(ETH)}{balance(USDC) + x} * x \tag{3}$$

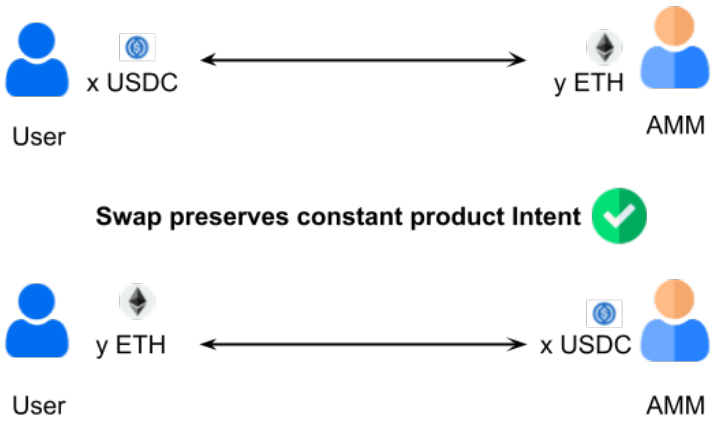

Figure 17: Swaps using constant product AMM model

2. **Added Liquidity:** Users can add assets to a liquidity pool, such as USDC and ETH, with the intent to preserve the balance proportionally. A token is minted representing the liquidity provided. The amount minted, z, is calculated based on the existing supply and the amounts added

   Intent:

   - Constant product

   $$balance(ETH) * x = balance(USDC) * y \tag{4}$$

   - z token minted

   $$\begin{aligned} z = if supply(z) == 0 \\ \text{then } \sqrt{x * y} \\ \text{else } min(x * \frac{supply(LP)}{balance(USDC)}, y * \frac{supply(LP)}{balance(ETH)}) \end{aligned} \tag{5}$$

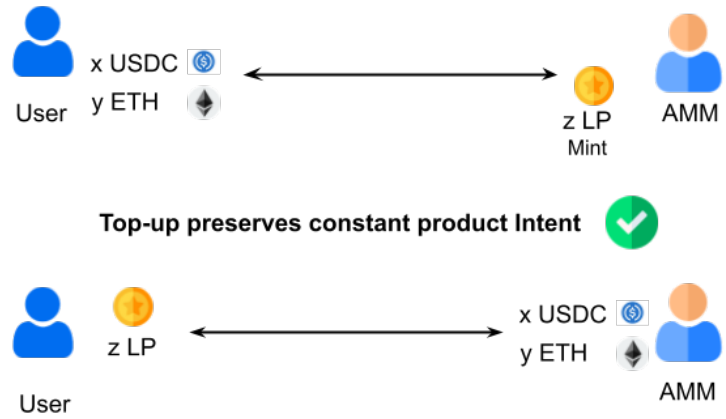

Figure 18: Added liquidity

3. **Remove Liquidity:** When users decide to remove liquidity, they redeem their LP tokens for the underlying assets. The calculations determine the exact amounts of USDC and ETH to be returned Following the redemption, the supply of LP tokens need to be adjusted

Intent:

$$x = balance(USDC) * \frac{z}{supply(LP)} \tag{6}$$

$$y = balance(ETH) * \frac{z}{supply(LP)} \tag{7}$$

$$supply_{new}(LP) = supply(LP) - z \tag{8}$$

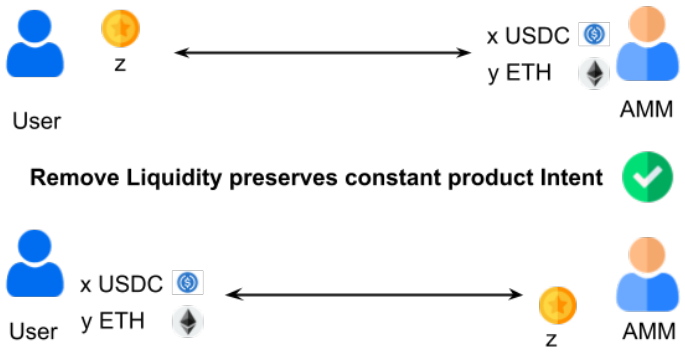

Figure 19: Removed liquidity

Simplification Through Intent Libraries:
Although the process involves some calculations and intent configurations, Saline simplifies these operations by encapsulating them within a single, configurable intent library. Users can effortlessly set the properties of their liquidity pool and launch it, streamlining the management of decentralized liquidity on the platform.

4.2.2 Index funds

Saline empowers users and institutions to create and manage index funds using a variety of assets stored in their Saline wallets. This capability is facilitated by Saline's support for multiple assets and chains, allowing for the creation of custom index compositions based on user-defined criteria.

Operational Overview:

Example: A user possesses BTC, ETH, and USDC in their Saline wallet and decides to create an index fund named "Top3," which allocates 70% to BTC, 20% to ETH, and 10% to USDC.

1. **Index creation:** Users can easily set up an index fund by utilizing Saline's pre-configured intent templates. The user specifies the composition of the index, and a new token representing the index is created. This token's value is based on the aggregated value of the included assets.

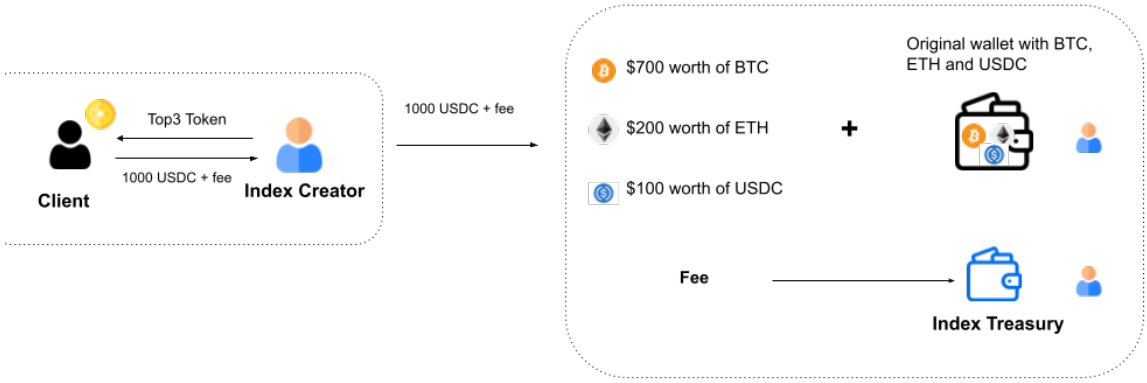

Figure 20: Index creation

2. **Participation in the Fund:** To join the fund, users purchase index tokens that reflect the combined value of the assets in the index. For example, an investment of $1,000 in the Top3 index results in asset allocation proportional to the defined weights: $700 in BTC, $200 in ETH, and $100 in USDC. Transactions are facilitated through intents that automatically execute the necessary swaps on the Saline chain, managed by matchers and solvers who process these intents.

3. **Index Maintenance and Fee Structure:** While simplified here, it's important to note that a portion of the investment (e.g., from the initial $1,000) also covers index fees. These fees compensate the index maintainer for the costs associated with rebalancing and maintaining the index, forming a revenue stream for institutions leveraging idle assets.

4. **Rebalancing the Index:** As market values fluctuate, the index requires periodic rebalancing to maintain its target asset ratios. The index creator sets the rebalancing frequency and thresholds, and Saline automates this process by posting new intents for asset swaps necessary to uphold the index composition.

5. **Exiting the Fund:** Redemption of index tokens is as straightforward as their purchase. Selling the index tokens triggers a conversion of the corresponding assets back to the user's wallet, effectively reversing the initial asset allocation process.

4.2.3 Exchanges

You can run a program in an environment that will generate a proof that this very specific program has run, to the bit, like you wanted, including with the inputs you wanted. This proof is expensive to generate of course (about a million fold slowdown vs running your program on a normal CPU) but it is cheap to verify.

A program might be, for example, the risk engine of a centralised exchange, determining a customer's P&L and therefore their next margin call. Now, what if such proofs were a data type for intents? Then, the exchange would not need to custody customer assets. Instead, a customer could install an intent on their wallet that says "if the exchange proves I owe them margin, they can withdraw it".

Crucially, the customer can no longer drain their wallet ahead of bad P&L, because they also need to provide proof that they still own the assets. So, the exchange does not need to withdraw assets into their own wallets, and can instead custody directly on millions of customer wallets.

This has a lot of interesting implications aside from removing counterparty risk; in particular, parallelizing custody to such an extent allows substantial performance improvements at scale. In fact, in our discussions with a team that worked on a 10-100 million transactions per second exchange, we found that parallelizing custody (in a centralised manner as they had not found us yet) was one of the key tricks to achieving their massive throughput.

From a user perspective, this means the appealing combination of performance and low costs of a CEX with risk management approaching that of a DEX.

4.3 Decentralised liquidity made accessible

In the transformative landscape of DeFi, the concept of enabling every individual wallet to act as a liquidity provider is not just revolutionary but now achievable with Saline. This capability heralds a new paradigm of decentralised, deep liquidity where the limitations of traditional financial systems are transcended.

Operational Overview:

Saline harnesses the power of 'intents'—specific instructions set by wallet owners that dictate how their funds can be used. These intents empower matchers (entities or algorithms that facilitate transactions) to operate on behalf of wallet owners, adhering strictly to the set rules.

Example: Consider the scenario where Alice wishes to exchange 10 BTC for 100 ETH:

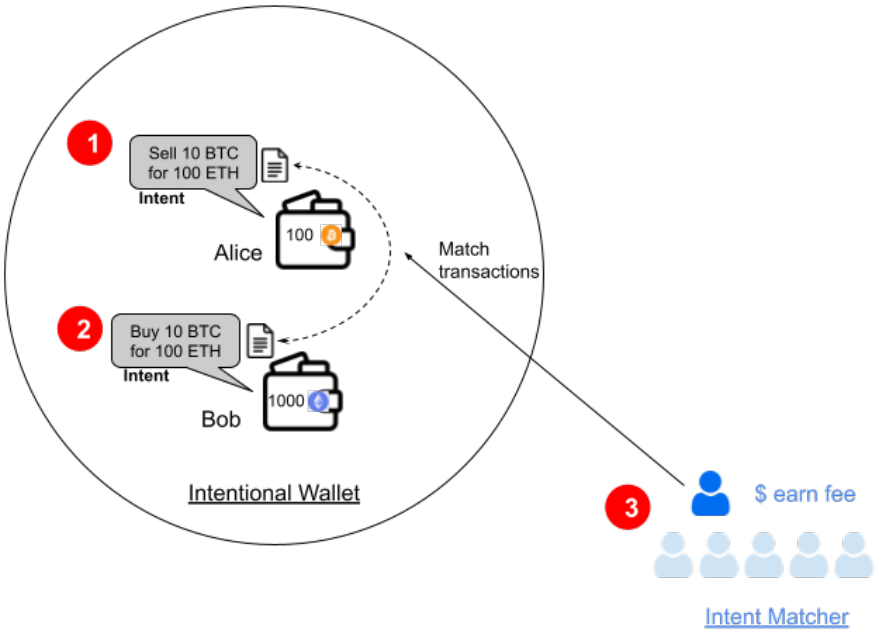

Figure 21: Intent order matching

1. **Intent Creation:** Alice sets an intent in her Saline wallet: "Exchange 10 BTC for 100 ETH." Although not discussed here, the inclusion of a transaction fee or a tip for the matcher can be configured within the intent, offering options like Fast, Standard, or Low, to accelerate the processing based on the user's preference.
2. **Counterparty Intent:** Simultaneously, User B, who desires to swap 100 ETH for 10 BTC, sets a corresponding intent in their wallet.
3. **Intent Matching:** Matchers, operating through RPC connections, monitor these intents. Upon identifying a match, they facilitate a transaction that swaps 10 BTC from Alice's wallet with 100 ETH from Bob's wallet, effectively completing the exchange.

Scalability and Ecosystem Impact:

Envision a scenario where not just two, but thousands of wallets across the blockchain set similar intents. This scalability transforms every participating wallet into a node of liquidity provision, vastly enhancing the depth and accessibility of decentralized liquidity throughout the ecosystem. Matchers play a crucial role in this architecture, as they can aggregate multiple intents from various wallets into single transactions. This not only optimizes transaction efficiency but also potentially maximizes transaction throughput and revenue generation.

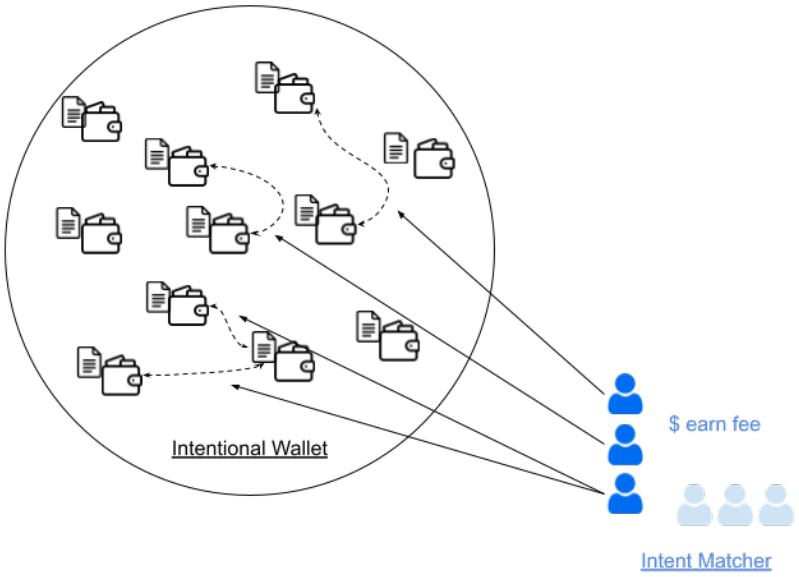

Figure 22: Decentralised liquidity network

Community Involvement:
The Saline platform empowers anyone with access to its RPC to become a matcher, thus democratizing the role of liquidity matching and enhancing the ecosystem's resilience and inclusivity.

4.4 Saline Lite vs Saline

Saline's architecture is designed for versatility, aimed at harnessing the full potential of intents within the blockchain ecosystem. However, to accommodate various user needs and technological readiness, Saline is offered in two distinct formats: Saline Lite and Saline Full.

4.4.1 Saline Lite

Saline Lite serves as an introductory platform, enabling users to engage with the power of intents on existing Layer 1 (L1) wallet infrastructures, such as MetaMask. This version is tailored to integrate seamlessly with users' current wallets, providing a straightforward approach to intent-based transactions.

Example: A user wants to swap their USDC for WBTC through an intent interface using their metamask wallet.

1. **Intent Creation:** Users can generate an intent for transactions, such as swapping USDC for WBTC, using the MetaMask snap with a Saline intent user interface (UI).
2. **Intent Signing and Submission:** The user signs the intent and simultaneously initiates a transfer of assets, e.g., 1000 USDC, which are sent along with the intent to a Saline bridge contract.
3. **Intent Processing:** Once received, the intent is activated on the Saline chain. It is registered to an Ethereum address within the Saline network, where matchers can process the swap.
4. **Completion and Bridging Back:** The equivalent WBTC is then bridged back and deposited into the user's original MetaMask wallet.

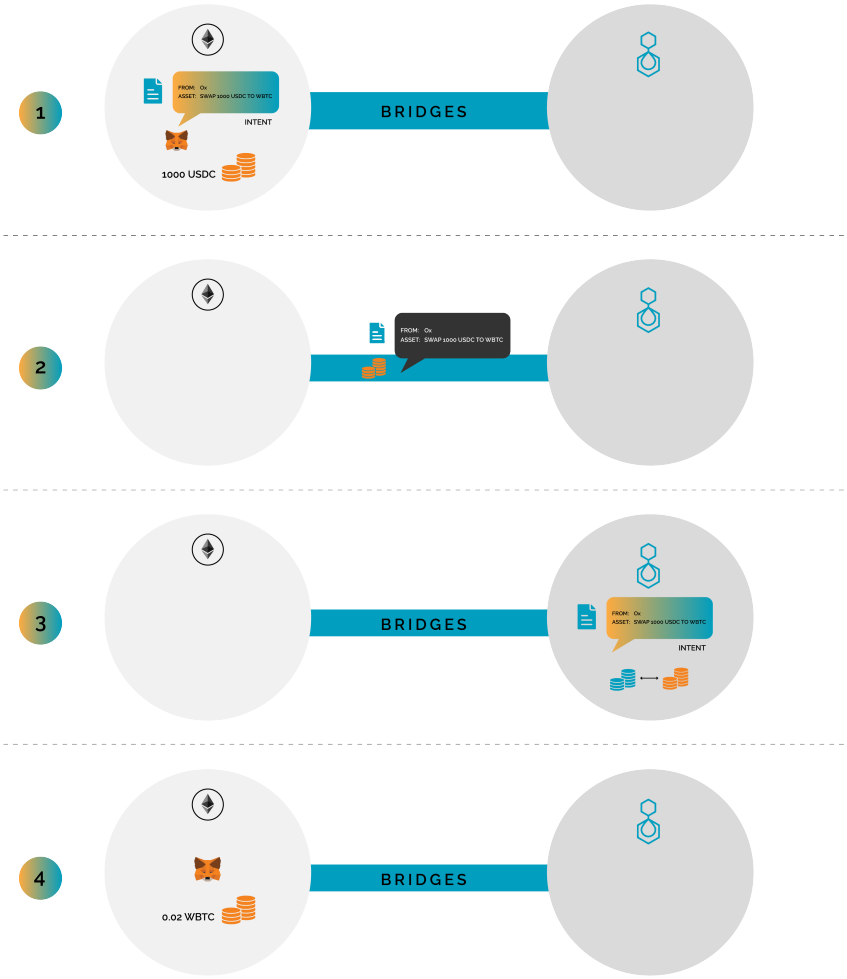

Figure 23: Saline Lite

4.4.2 Saline

Saline: Unleashing Full Functional Capability:

For users seeking deeper integration and broader functionality, Saline offers a comprehensive suite of tools and capabilities. This full version empowers users to leverage advanced intent functionalities, facilitating complex financial operations like token creation, liquidity pool formation, and management of index funds.

1. **Intent Configuration:** Users, through the Saline wallet, set an intent for specific blockchain operations, such as asset onboarding
2. **Asset Bridging:** Assets, such as 1000 USDC, are transferred to a Saline-controlled contract on the Ethereum network, termed "bridged."
3. **Asset Utilisation:** Once bridged, the assets appear as wrapped USDC in the user's Saline wallet. The user can then employ these assets for various functions, including launching new financial instruments or participating in existing ones.

Saline, in its full capacity, thus offers a robust platform for users to engage deeply with DeFi, backed by the innovative intent-based transaction system.

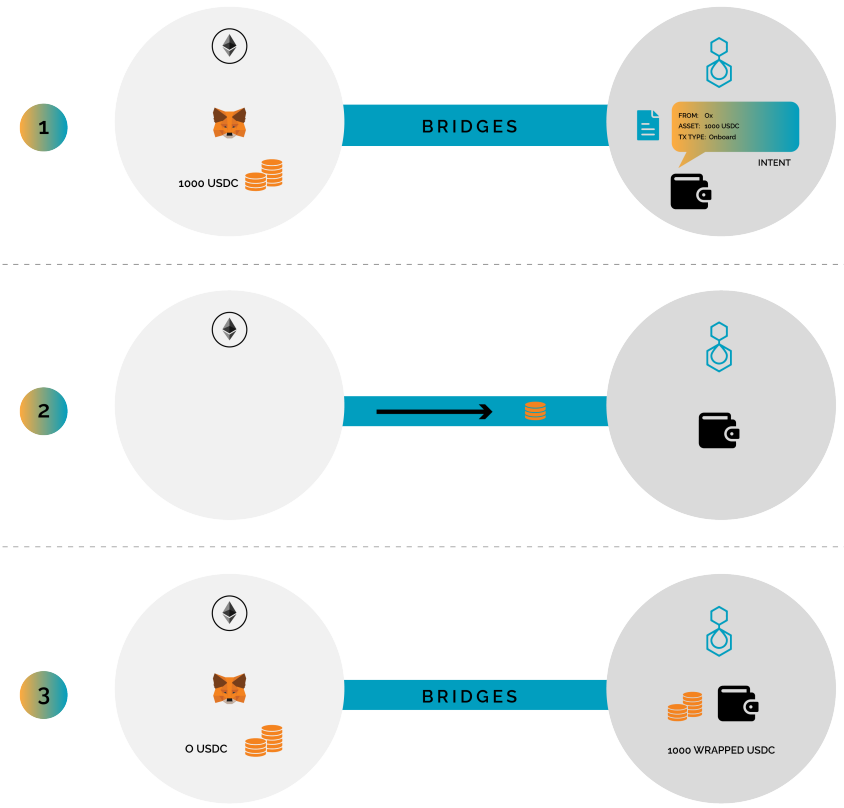

Figure 24: Full Saline

5 The future: verification based computing

This paper focused on how Saline works within and for crypto-based use cases. However, the biggest potential of Saline is its use as the first instance of a verification chain for general computing, bringing about a new paradigm: verification based computing.

Most computing paradigms focus on controlling execution, and successful new paradigms usually offer a novel way to do so. For example, LLMs and video game graphics both benefit from an architecture consisting of large parallel computation on many low throughput processors, instead of having a single core performing with very high throughput. Meanwhile, cloud providers such as AWS and GCP focus on architectures optimised for running a large number of virtual machines with good redundancy and low overheads.

Computing paradigms attempt to control how to get from the current state to the desired state. Object Oriented Programming (Ruby), Logic Programming (Prolog), Functional Programming (Haskell), Procedural Programming (C) all offer different paths of execution, but all focus on controlling that path.

Verification based computing turns this over its head. It offers the output to the market by defining its boundaries, and lets competition tackle how to get there. Verification becomes the only operation performed, with third parties invisibly competing for the cheapest or fastest possible execution to achieve the goals required by verification. This looks a lot like Saline verifying whether a transaction set will match an intent!

You can even pass a set execution path to the verifier, using ZKPs. Because these proofs compress execution into merely the assurance that execution has in fact been performed exactly as asked, it is possible to ask the market to run a program for you, and pass the proof of it to the chain in order for the performer of said execution to get paid.

This is probably the most naive, fastest and thus earliest implementation of verification based computing, but as with other paradigms, we do not yet know just how the market will evolve to take advantage of this unique architecture. It may well make about as much sense to verify execution proofs in Saline, as it does to attempt imperative programming in SQL, and just as SQL developers do not write imperative programs today, we do not expect Saline developers to merely stick to pushing their execution proofs on chain.