Explaining Algorand's Layer-1 Strategy (Part 1): Fungible Tokens and Atomic Multi-Party Transfers

By Silvio Micali

It's been exciting to see the different ways people are using Algorand since the Algorand Foundation launched a public mainnet on June 19th, and so many ideas and contributions to the open source platform.

Thanks to the unique security, scalability, and decentralization of the Algorand protocol, a block is produced in less than 5 seconds, and all transactions are immediately final. These fundamental properties have led many in the community to experiment with the technology in different ways.

One of the common use cases we have seen is using the protocol to create new tokens and assets, including a diverse array of tokenized assets such as derivatives, options, swaps, securities, and more. The recent announcement of Algorand joining ISDA was based on the idea of allowing for flexible templates and asset creation across a large library of known asset types.

On other chains, the creation of these new assets requires either smart contracts or other Layer-2 approaches. This has led to a host of innovations, but also carries security, efficiency, and scalability issues. In addition, we have observed many use cases where smart contracts are too complicated for the task at hand.

At Algorand Inc, we are committed to being an ongoing contributor to the open source community and are proposing two new features that eliminate the drawbacks of these Layer-2 approaches.

These new features, *both implemented in Layer 1*, are:

1. The generation of new fungible tokens and
2. The execution of *atomic multi-party transfers* (AMPTs), a new construct that we will explain in a moment.

By being implemented in Layer 1 of the blockchain, both features become part of the Algorand consensus protocol and are made very secure, efficient, and easy-to-use.

THE BENEFITS OF LAYER-1 IMPLEMENTATION

In common blockchain parlance, Layer 1 handles basic payments and the consensus protocol, while Layer 2 includes smart contracts and any other functionality not directly connected to the generation of new blocks.

Algorand has chosen to implement core functionalities at Layer 1 for three main reasons:

1. ***Security.*** Any functionality implemented at Layer 1 enjoys the highest level of security: that of the consensus protocol itself. Our Layer-1 transactions are designed by experts, tested thoroughly, verified by auditors, and inspected by contributors everywhere. Attack vectors for any asset created in Layer 1, such as double spending, denial of service, etc. are countered by the same mechanisms that exist for the Algo, the native currency of the Algorand public blockchain.

2. *Compatibility.* Layer-1 functionality is interpreted in the same way by all blockchain participants. Indeed, although the core consensus protocol can be augmented in a decentralized consensual manner, all participating software works with the same consensus protocol. This promotes compatibility. If two users use the same Layer-1 implementation to launch their own fungible tokens, then an application developed for one token will also work for the other.

By contrast, if two users launch their fungible tokens via two separately designed smart contracts, the two implementations may differ in subtle ways and a DApp correctly designed for one may fail to work for the other.

3. *Efficiency.* Core functionalities implemented at Layer 1 can take direct advantage of the system to optimize their performance in computation, storage, and communication.

By contrast, relying on virtual machines and other Layer-2 apparatus often results in unnecessary complexities and costs.

Note: Not all new features are suitable for Layer-1 implementation. In particular, any features implemented in Layer-1 must not significantly slow down the consensus protocol; should be important enough to merit such prominence in the platform; and should be designed in a way that simplifies checking their correctness.

LAYER-1 FUNGIBLE TOKENS

A token is fungible if any two of its units are interchangeable. One example of a fungible token is a currency, because any two units have the same purchasing power. Another could be "points" in a loyalty program, or a unit of stock[1].

Algorand lets users generate their own fungible tokens, thus becoming their *token managers*. A token manager can

- Generate a supply of new tokens,
- Increase the circulating supply of the new tokens,
- Freeze the tokens (generated by her) in a given account, and
- Transfer her abilities to a new token manager.

These properties are similar to those enjoyed by Ethereum's ERC-20 tokens. In Algorand, however, these properties are implemented at Layer 1, rather than via smart contracts as in Ethereum.

The benefits of a Layer-1 approach are especially valuable when fungible tokens are used to launch a new currency. Were a currency to be launched via a smart contract, the slightest error in the contract, or the slightest compatibility issue when applying an existing app to the new currency can and has caused losses on many occasions.

LAYER-1 ATOMIC MULTI-PARTY TRANSFERS

Algorand allows multiple users, who hold possibly different fungible tokens and have negotiated a set of transfers of these tokens, to execute all of these transfers by posting on chain a single transaction. That is, if

- $T_1, T_2, \ldots$ are potentially distinct fungible tokens, including the Algo;
- $a, b, \ldots$ are potentially distinct users; and
- a transfers to b x_1 units of T_1; c transfers to d x_2 units of T_2; etc. are individually valid transfers;

then, after each user digitally signs her own transfer, either

(i) none of the transfers is executed, by a given time, or

(ii) all of them are executed, by that time.

That is, an AMPT guarantees the simultaneous execution of multiple transfers of all kinds of fungible tokens. Simultaneity is a most valuable property, but it has been hard to achieve up to now. Consider the above set of transfers. A user a can individually transfer to user b x_1 units of token T_1. However, having done her transfer, user a has no confidence that b and all other users will also execute their own promised transfers. This explains why smart contracts are typically used to ensure the all-or-nothing nature of AMPTs. But smart contracts are grungy, error-prone, and costly Layer-2 constructions. So, Algorand implements AMPTs at Layer 1 instead.

The use cases of AMPTs abound. Here are just five of them.

[1]By contrast, in a set of non-fungible tokens, two tokens can be very different from one another and can be valued very differently. Think of apartments in the same complex, where each apartment has its own planimetry and light exposure.

Example 1: *Bilateral Exchange*

Let USDX be a USD-backed stable coin on the Algorand blockchain. Assume that a is a user owning sufficiently many algos and b a user owning sufficiently many USDXs. Then, an AMPT enables b and a to trade ---say--- 1000 algos for 1,500 USDXs.

Such an exchange is *immediate* and *unmediated*. If each party has the right amount of money, a bilateral exchange can be executed within 5 seconds, by atomically posting the proper data in the next block. The ability to execute a bilateral exchange by a single Layer-1 transaction ensures that no user can cheat the other by failing to post its own transfer, and that the exchange is fast.

Such speed and true simultaneity are main advantages of the Algorand blockchain over other blockchains. Bilateral exchanges are in fact fundamental primitives for the creation of decentralized financial systems.

By contrast, bilateral exchanges in other blockchains are *not* simultaneous *nor* fast. They are implemented via multiple steps and over multiple blocks, and, as we shall explain later on, some of these steps require a long waiting time.

Example 2: *Circular Trades*

The speed and simultaneity of AMPTs do not apply to just two transfers, but also to multiple and interdependent transfers, such as the following "circular trade:"

User a transfers x units of token T_1 to user b *if and only if*
b transfers y units of token T_2 to user c *if and only if*
c transfers z units of token T_3 to a.

Example 3: *All-or-Nothing Funding*

Layer-1 AMPTs simplify a host of joint funding applications. For instance, they make it trivial for multiple users to split the bill for a dinner or a shared taxi ride. As for another example, they simplify crowdfunding, by enabling a group of users to contribute pre-specified amounts of possibly different currencies to the same project (e.g., a given charitable organization) so that each user in the group ends up making her payment if and only if everyone else in the group also does so.

Example 4: *Decentralized Exchanges*

Layer-1 AMPTs provide powerful building blocks for constructing fast and secure decentralized exchanges, where users trade with each other without having to trust any intermediaries.

Example 5: *Internal Units of Account*

Internal units of account can be represented as a fungible token, and AMPTs make it possible for several companies to settle their accounts with one another in a *single swift go!* For instance, this would be very advantageous for supply chains, where delays in payments are a massive friction for companies.

EXPECT MORE LAYER-1 FUNCTIONALITIES!

As important as new fungible tokens and AMPTs may be, they are just the first examples of the functionalities Algorand plans to bring to Layer 1. Our plan is to propose a set of Layer-1 *templates* that capture many functions that currently require general purpose smart contracts. Each template will be carefully designed and highly optimized. And the provided set will be rich enough so that even a functionality not having its own ready-made template can be easily implemented by assembling the available templates.

We consider non-fungible tokens (NFTs) an obvious choice for such templates, and propose NFTs in Layer 1 next.

Q&A

What about NFTs now?

It is of course possible to represent a non-fungible token as a class of fungible tokens with a single element. However, such an approach would not meet the performance and ease of use requirements we feel are necessary. We are readying a proposal that would bring non-fungible tokens to Layer 1 in a convenient, efficient, and secure manner.

Is an AMPT-based implementation of a Bilateral Exchange an "atomic² swap", in the sense announced in Algorand's Forthcoming Technology?

Yes. AMPT-based bilateral exchanges of fungible tokens are a special case of atomic² swaps. Indeed, the latter swaps can handle arbitrary assets, including non-fungible tokens.

Note that fungible tokens are inherently easier to handle than non-fungible ones. When the supply consists of billions of tokens, the fungible tokens owned by a given user can readily specified by just a few bytes: namely, by her balance. But specifying which non-fungible tokens a user owns may require a long list. To efficiently handle non-fungible tokens, Algorand will thus resort to new technology, such as the *self-verifying transactions* announced in Algorand's Forthcoming Technology.

Will Layer-1 primitives totally replace smart contracts in Algorand?

No! Algorand will bring to Layer 1 a set of primitives that (a) have been identified as crucial and (b) have been engineered to enable the Algorand blockchain to generate and finalize new blocks within seconds. Not all primitives the community desires to have on the platform may be suited for Layer-1 implementation. We are thus readying a new generation smart-contract technology in order to meet the efficiency, security, and ease of use standards that Algorand users expect and deserve.

Will Algorand enable "cross-chain AMPTs?"

Algorand will soon provide protocols enabling users of other chains to execute multiple transfers of fungible (and non-fungible) tokens with users of Algorand. We wish to clarify, however, that these cross-chain protocols do not guarantee the simultaneity enjoyed by Algorand AMPTs!

Moreover, such protocols must be expected to be less efficient than AMPTs among just Algorand users. Even an implementation of a bilateral transfer, which involves just two users, is slow when one of them is not an Algorand user.

Why are cross-chain implementations of bilateral exchanges slow?

Because they are implemented via multiple steps, some of which require long waiting periods, in order to ensure users have sufficient time to take some necessary actions.

Assume that one of the two users of a *multi-step* bilateral exchange tries to cheat. Then, to avoid transferring money without receiving any in return, the cheated party must, in some step, post a "STOP transaction" on the relevant blockchain. To enable her to do so on a blockchain that is slow in finalizing its blocks, however, the protocol must allocate a long time to that step. (Indeed, posting a STOP transaction in a non-finalized block is insufficient, since that block may "disappear" due to a fork.)

But this waiting time is not enough. The protocol must also budget for the possibility that the cheated user may be

the target of a denial of service attack that prevents her from posting a STOP transaction during the attack. Thus, to be robust, a bilateral exchange protocol must pre-allocate "enough" time to ensure that a denial of service attack becomes too expensive and will be suspended. Only then will the cheated party be able to post her STOP transaction. How much is "enough?" One, two, and three days have been recommended in the literature. But even three days may be insufficient if the monetary value of the exchange is very high. After all, the cost of a denial of service attack is decreasing rapidly, and a malicious party may be willing to sustain it for a long time in order to extract a substantial amount of money from the other party without having to pay anything himself!

By contrast, since a bilateral exchange between two Algorand users is executed via a *single transaction*, the time allocated to posting this transaction can be as short as, say, 50 seconds, that is, the time to produce 10 blocks. More precisely, two Algorand users may set up their bilateral exchange so that the single transaction completing it is valid only if it appears in any of 10 pre-specified blocks: e.g., blocks n through n+9. If the single transaction appears in one of these blocks, then the exchange is immediately and fully completed, without any user being able to cheat the other. If it does not (because blocks are saturated due to an unusually high transaction traffic, because of the presence of a denial of service attack, or because of any other reason), then *no user* transfers any money to the other, no one has been cheated, and *either user* is free to engage in whatever other transfer she wants with someone else. This is the advantage of a truly atomic (i.e., an atomic[2]) exchange, versus an exchange "atomic" in name only!

Silvio Micali

Founder

Silvio Micali has been on the faculty at MIT, Electrical Engineering and Computer Science Department, since 1983. Silvio's research interests are cryptography, zero knowledge, pseudorandom generation, secure protocols, and mechanism design.

In 2017, Silvio founded Algorand, a fully decentralized, secure, and scalable blockchain which provides a common platform for building products and services for a decentralized economy. At Algorand, Silvio oversees all research, including theory, security and crypto finance.

Silvio is the recipient of the Turing Award (in computer science), of the Goedel Prize (in theoretical computer science) and the RSA prize (in cryptography). He is a member of the National Academy of Sciences, the National Academy of Engineering, and the American Academy of Arts and Sciences.

Silvio has received his Laurea in Mathematics from the University of Rome, and his PhD in Computer Science from the University of California at Berkeley.