ALGORAND CO-CHAINS*

By Silvio Micali

A QUICK SUMMARY

The public, permissionless blockchain enables all organizations, individuals, and governments to transact with efficiency, security, and transparency, both internally and with each other. It also enables private enterprises to expand their addressable markets, increase revenue, generate new product opportunities, and more.

Such a blockchain however, may not satisfy the privacy and regulatory requirements that traditional public and private institutions (e.g., financial, central banks, governments, health care, etc.) must follow. Such institutions may need to both know and carefully select those who run their networks and validate their activities. A private, permissioned blockchain, ideally that is interoperable with the public chain is better suited to implement these goals. However, without this interoperability, it also risks isolating its users, by making it very hard for them to interact with other private blockchains and the world at large.

At Algorand, we have been developing our co-chain architecture to enjoy the advantages of both types of blockchain. Essentially, a co-chain

1. Is totally independent from the public chain, shields its transactions from all outsiders, chooses its own validators, and runs its own Algorand consensus algorithm;
2. Interoperates with the Algorand main chain to transact with other co-chains, and everyone else, with the same ease and security with which the members of Algorand's permissionless chain transact with each other; and
3. Enjoys, both in its internal and external interactions, the same atomic transactions, layer-1 smart contracts, and all other primitives and tools offered by the permissionless Algorand protocol. In fact, it automatically inherits all the improvements and upgrades that will be added to Algorand's permissionless protocol.

The world needs both permissionless and permissioned blockchains. Algorand provides best-of-breed implementations for both types of blockchains. More importantly, Algorand guarantees their *synergy*. No matter how scalable, distributed, and secure a permissioned chain may be, its members may not want to interact only with each other, but also with other chains and with the world at large.

The Algorand *co-chain architecture* enables the members of a permissioned chain to work most securely and efficiently not only with each other, but also, while retaining maximum autonomy, with the Algorand permissionless chain and members of other chains as well.

** The Algorand Co-Chain architecture has been developed by Sergey Gorbunov, Maurice Herlihy and Silvio Micali. Their full technical paper will be published in the near future.*

Algorand™

1. THE PERMISSIONLESS VERSION OF ALGORAND

BLOCKCHAIN PROMISES AND BLOCKCHAIN CHALLENGES

Transparency, immutability, and trust among players who do not know each other are fundamental to public, permissionless blockchains. Without proper technology, however, blockchains have remained aspirational for several years.

THE ALGORAND PERMISSIONLESS BLOCKCHAIN

Algorand provides a permissionless blockchain that is truly decentralized, scalable, and secure. It is truly decentralized, because every single token can participate in the consensus protocol, with the same power as any other token. It is truly scalable, because it allows even billions of users to generate a block in seconds, using a trivial amount of computation. And it is secure, because it cannot be subverted by a few miners or delegates, or by the owners of a small percentage of the tokens. Indeed, the Algorand blockchain is guaranteed to work correctly so long as most of its tokens are in honest hands.

The Algorand protocol relies on brand-new technology, such as its unique cryptographic sortition and its super-efficient Byzantine agreement. (For a more detailed discussion of Algorand's consensus protocol see X.)

In addition to full decentralization, scalability, and security, the Algorand permissionless blockchain enjoys the following notable properties.

- ***No Forks and Transaction Finality.*** The Algorand blockchain does not fork. Each new block is separately agreed upon and is guaranteed to remain on the Algorand chain forever.[1] Thus, its users can immediately rely on the transactions contained in a new block without having to wait for the block to become sufficiently deep in the chain.
- ***Standard Assets and Smart Contracts at Layer 1.*** A blockchain handles different transactions at different *layers*. Layer 1 is the most direct and secure layer. Traditionally, layer 1 only handles ordinary payments and the consensus protocol itself, while the issuance of new assets, smart contracts, and everything else are handled at layer 2. Layer-2 protocols, however, are notoriously slow, expensive, and prone to errors. By contrast, Algorand also handles at layer 1 the issuance of standard assets and a large class of smart contracts, including ***asset tokenization, atomic transactions***,[2] and ***collateralized loans*** combined with the ability to quarantine and claw-back disputed transactions if necessary. In fact, Algorand satisfies at layer 1, most of the current use cases for smart contracts, and with the same security and efficiency reserved for ordinary payments.

2. THE PERMISSIONED VERSION OF ALGORAND

A main advantage of permissioned blockchains is the ability to shield transactions from outsiders.[3]

In the permissionless version of Algorand, every native token (in addition to be a unit of the native currency, the Algo), can participate (with the same power of every other token) in the consensus protocol. In a permissioned version of Algorand, however, an enterprise E may use the given pool of 10B tokens for consensus only, and partition it among its own chosen set of validators, V, in any way it wants. For instance, E may choose V to consist of only five validators and assign to each of them 2B consensus tokens. By so doing, E gives each of the five validators the same power to generate new blocks. As for another example, E may choose to have 55 validators, assigning 1B tokens to each of the first 5 and 100K tokens to each of the other 50. This way, E assigns the first 5 validators 10 times the power of block generation than the other 50 validators.

A permissioned version of Algorand enjoys a tremendous level of granularity to assign different weights to different validators.

By permissioning the Algorand blockchain, instead of building its own permissioned chain from scratch or adopting another permissioned chain, E gains two main advantages:

(a) Weighted Decentralization on Demand. The option of choosing an arbitrary number of (arbitrarily weighted) validators is crucial. Indeed, E may want to exercise this option in order to increase the security of its own blockchain, or to enlarge the community it serves. A blockchain initially serving a few financial institutions may start with a few validators. But what if, later, it wants to welcome medium-size banks, small banks, and credit unions, all of which wish to participate to block generation? A consensus protocol working for a few participants may fail to work efficiently with hundreds or thousands of participants. And changing horses in the middle of the race may be quite challenging!

By permissioning consensus protocols capable of scaling to billions of validators, E is guaranteed to be able to enlarge the set of validators at any time and without any problem. Scaling down is easy. Scaling up is harder.

(b) Transaction finality and Layer-1 Smart Contracts. Transaction finality is a crucial property for any blockchain, private or public, permissioned or permissionless. And, so is the ability to handle most of the needs for smart contracts at layer 1.

By permissioning Algorand, E obtains a permissioned blockchain that automatically inherits these crucial and rarely enjoyed properties.

(c) Upgradeability and Continued Innovation. Using a permissioned version of the Algorand protocol automatically provides E with future upgrade improvements and innovations whenever they are added to the core permissionless Algorand main chain.

---

[1] Algorand consensus is not a drawn out process. The fact that more and more blocks are attached to given block B makes more and more probable that consensus on B has been reached. Algorand separately reaches agreement on a new block. When this is done, it reaches agreement on the next block. And so on.

[2] An atomic transaction enables multiple users to exchange assets, or to execute multiple payments in multiple currencies, via a single transaction. Accordingly, no participant in an atomic transaction can cheat another participant, and no one is afraid to "go first."

[3] Another often-cited reason for opting for a permissioned blockchain is security. This reason, however, misses the point that decentralization is itself a main source of security.

3. ALGORAND CO-CHAINS: DEFINITION AND CHALLENGES

THE DEFINITION

An Algorand co-chain is a special, permissioned version of Algorand. Accordingly, it is a scalable permissioned chain that is decentralizable on demand and enjoys transaction finality and layer-1 smart contracts. But it also satisfies an additional crucial property:

(c) Interoperability with other co-chains. A permissioned blockchain enables the members of a given group to safely interact with one another. But it may *not* enable them to interact with other entities and individuals. This is a big limitation, because the world "outside" is larger than that "inside," and we may want to interact with this larger world. A group of financial institutions may want to set up their own permissioned chain. But a group of health care institutions may also want to do so. Since health care represents a major part of the economy, it is not realistic to assume that the first chain does not care about interacting and exchanging assets with the second chain. Without outside interoperability, the members of a permissioned chain risk remaining trapped in their own castle.

Co-chains are Algorand permissioned chains that guarantee efficient and safe interoperability with the Algorand permissionless chain and other co-chains.

A FIRST CHALLENGE: SECURITY

Interoperability among permissioned chains is easy to claim but harder to guarantee. Consider a simple example. A user a owns an asset x that she wants to exchange with an asset y owned by another user b.
This problem can be solved in less than 5 seconds, with finality and security, if a and b belonged to the Algorand permissionless chain or to the same Algorand co-chain. Indeed, they could use an atomic swap, a main tool available in Algorand as a layer-1 transaction. But what if a is a member of a co-chain A and b a member of a different co-chain B?

Asset exchange across different chains is typically approached by hash- and time-lock protocols. This approach, however, is quite problematic. Besides requiring multiple and logically complex steps, it is vulnerable to a denial of service attack. Such an attack enables a cheating party to keep his own asset while getting possession of the other party's asset. To avoid it, the protocol may need to last for a long time, to make denial of service more costly than the value of the assets in question. (For details see W.)

A SECOND CHALLENGE: CLEAR TITLE

But a different problem looms large and applies to any protocol that involves only x and y and their respective blockchains A and B. Namely, since A and B are permissioned and private, at most their members may know that x and y have exchanged their original assets, and thus that b is now owned by a member of chain A. If chain B were corrupted, nothing prevents y from selling b or exchanging it for other assets, over and over again, with members of different blockchains! In essence, this is the asset-exchange equivalent of double spending.

A permissioned chain is corrupted if (the majority of) its validators are malicious or their secret keys have been compromised. Within a corrupted chain, original blocks can be replaced by fake ones, and it is no longer clear who owns what. (This is why decentralization is crucial to security, why hundreds of validators are better than dozens of validators!) The corruption of a permissioned chain is particularly insidious, because its very privacy prevents outsiders from noticing such a corruption. The corruption of a chain should be a rare event, but when it occurs, it should affect only the members of that chain, rather than affecting also honest chains! No one can guarantee that another chain will remain honest. But...

Chain interoperability should guarantee clear title for any asset acquired by a member of an honest chain.
Even for an asset acquired from a member of a corrupt chain.

4. ALGORAND'S (SIMPLIFIED) CO-CHAIN ARCHITECTURE

Let us now provide a high-level view of how Algorand co-chains interoperate. To simplify our task, let us initially ignore privacy.

PREAMBLE

We denote Algorand's mainnet, which is permissionless and public, by $MAIN$. Accordingly, every co-chain monitors the blocks of $MAIN$. For each co-chain C, $MAIN$ maintains

- An updated list, $VALIDATORS_C$, of C's validators, and
- An updated list, $ASSETS_C$ of all assets owned by members of C that are transferable to other chains.

Initially, when a co-chain is formed, both lists may be included in what essentially is C's "genesis block in $MAIN$".
(This genesis block is different from the original genesis block of C, which specifies which are C's initial public keys and which assets such keys initially own.)

Over time, both $VALIDATORS_C$ and $ASSETS_C$, are kept up-to-date by C by posting in $MAIN$ suitable transactions signed by (a proper majority of) C's latest list of validators.

Let us stress that $MAIN$ has not only no idea about the transactions that occur in a co-chain C, but also about the actual public keys of C, never mind about the actual users behind those keys! In fact, $ASSETS_C$ does not reveal any information about public keys in C which control the assets in $ASSETS_C$.

ASSET TRANSFERS OF FROM AN ALGORAND CO-CHAIN TO THE MAIN CHAIN

A user x of an Algorand co-chain A may want to transfer an asset a she owns via a public key t_x to $MAIN$. User x may have several reasons to do so. For instance, x may want to auction off a, and "the larger the set of bidders, the higher the fetched price." Accordingly, rather than auctioning a on A, user x may prefer to auction it on $MAIN$, so as to welcome bids not only from members of A, but also from users of $MAIN$ or other co-chains. In fact, any member of a co-chain can easily transfer stable coins to $MAIN$ for the sole purpose of participating in the auction.

As with an ordinary transfer in co-chain A, the transfer of a from t_x to $MAIN$ is authorized by a digital signature of t_x: in symbols, $SIG_x(t_x,a, MAIN)$. Since t_x owns a and the transfer is properly authorized, $SIG_x(t_x,a, MAIN)$ enters a new block X of A, properly certified by A's validators. At this point, all members of co-chain A realize that neither t_x nor any other public key in A owns asset a. Thus (unless A is corrupted), t_x can no longer authorize the transfer of a, within A or outside of A.

As all other blocks of of A, X is structured so as to make it easy to isolate $SIG_x(t_x,a, MAIN)$, and all other transfers of assets to $MAIN$, from all other information that must remain visible only to the members of A. Conceptually,

$$X= (SIG_x(t_x,a, MAIN) , other\ transfers\ to\ MAIN, H)$$

where H is the one-way hash – typically 256-bit long – of all transactions in A that must remain private within A. Notice that this format of X is quite compact. In fact, it contains only 256 bytes in addition to the information that is intended to be conveyed to Algorand's main chain.

The so formatted block X, and its certificate in A, are propagated to the nodes of $MAIN$.

Since co-chain A runs the same consensus algorithm as $MAIN$, and since A's validators are known to $MAIN$, $MAIN$'s validators can parse X's certificate, and thus learn that

- t_x is a key of A that owns asset a, and
- (the owner of key) t_x wishes to transfer a to Algorand's main chain.

Accordingly,

- Asset a is removed from $ASSETS_A$ and
- The key t_x is recorded as the (possibly new) key of $MAIN$ owning (in $MAIN$!) asset a.

NOTE. Step 1, which leverages $MAIN$, is both *public* and *permissionless*. Specifically, the fact that $MAIN$ is permissionless guarantees that t_x will become a key in $MAIN$ without any problems. And, the fact that $MAIN$ is public guarantees that everyone realizes that asset a is now in $MAIN$. This guarantees that y will (in the next step) get clear title to a. In fact, whether or not co-chain A is corrupted, neither x nor any other member of A will be able to transfer a to any member of any other co-chain.

ASSET TRANSFER FROM THE MAIN CHAIN BACK TO THE CO-CHAIN

Having sold a in $MAIN$, t_x may want to transfer to A the stable coins earned in the auction.

More generally, if t_x is a public key of both $MAIN$ and A, t_x may want to transfer an asset b it owns in $MAIN$ to A. Again, such a transfer may be authorized by a digital signature of t_x, in symbols, $SIG_x(t_x,b, A)$, which enters a new block of $MAIN$. Since $MAIN$ is permissionless, the validators of A may see the appearance of $SIG_x(t_x,a, A)$ in a block of $MAIN$, or they may be presented with a suitably compact proof of such appearance by t_x itself. Either way, A's validators will cause t_x, since it was already a key in A, to be the current owner of asset b in A. At the same time, as soon as $SIG_x(t_x,a, A)$ appears in a block of $MAIN$, t_x no longer owns b in $MAIN$, and $ASSETS_A$ is updated to include asset b.

CO-CHAIN INTEROPERABILITY

Let us now illustrate how co-chains interoperate using the same asset-swap example used above. Now A and B are different Algorand co-chains. Specifically, asset a is controlled in A by a public key t_x whose secret key is known to x, and asset b is controlled in B by a public key t_y whose secret key is known to y.

To exchange their assets, x and y leverage $MAIN$ via the following conceptual steps.

1. In chain A, t_x "transfers a to $MAIN$" and proof of this transfer is provided to $MAIN$.
   In chain B, t_y "transfers b to $MAIN$" and proof of this transfer is provided to $MAIN$.
2. In $MAIN$, t_x and t_y exchange a and b via an atomic swap.
3. In $MAIN$, t_x transfers b to A and t_y transfers a to B. Chains A and B see both transfers.

EXPLANATION OF STEP 1

STEP 1 may be implemented by t_x posting $SIG_x(t_x,a, A)$ in a block of $MAIN$, as discussed above. Accordingly, in $MAIN$,

- Asset a is removed from $ASSET_A$ and asset b is removed from $ASSET_B$.
- Key t_x no longer owns a.

Similarly, for t_y .

EXPLANATION OF STEP 2

Since now, in $MAIN$, t_x owns a and t_y owns b, they can swap these assets within a few seconds and with super security. Indeed, via a layer-1 atomic transaction, which is one of the main advantages of the Algorand permissionless chain.

EXPLANATION OF STEP 3

In $MAIN$, t_x transfers b to itself in A as already discussed, because t_x continues to be an approved key of A.

Similarly, for t_y.

ADDITIONAL EXPLANATIONS

Let us note that the whole process is very fast. Indeed, each of the three steps above can be executed within the time necessary to generate a new block. Such time, in Algorand's main chain, takes less than 5 seconds. But block generation in an Algorand co-chain can be much faster. Indeed, in the Algorand protocol, a block can be generated in the time it takes to make sure that most of the validators see it. In a co-chain with superior network speed, this time can be negligible.

Let us also remark that the entire process occurs at *layer 1*, and thus with greater security, whether in the main chain or in a co-chain.

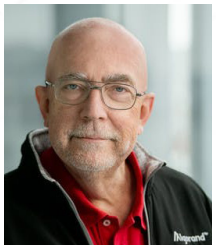
Finally, note that the cumulative value of the assets of a given co-chain may surpass the valuation of Algorand's main chain. However, Algorand's main chain is *not* used to secure the assets of any co-chain. At a given point in time, it is only used to handle a few assets of a given co-chain, and only for a few seconds. Namely, it is used to handle the assets that a co-chain wants to exchange with another chain.

ADDING PRIVACY

The privacy of an exchange of assets between Algorand co-chains can be greatly enhanced.
In particular, t_x and t_y may be temporary keys which x and y use just for the purpose of this asset exchange. That is, prior to starting the three-step process described above, x generates a temporary public key t_x and transfers asset a from whatever public key held a before to t_x. After step 3 is completed and t_x owns asset b in A, x can transfer b from t_x to whatever other public key she chooses. This way, Algorand's main chain never learns which public key in A originally owned asset a and which public key will eventually own b.

SERGEY GORBUNOV | Head of Cryptography

Sergey is an Assistant Professor at the University of Waterloo. His research interests range from foundational cryptography to design of secure large scale systems, computer networks, protocols and blockchains. He received PhD from MIT in '15 where he was the recipient of the Microsoft PhD fellowship. His dissertation was on building advanced cryptographic protocols using lattice-based cryptography for which he received Sprowl's Doctoral Thesis Prize for best PhD thesis in CS at MIT. Prior to joining Algorand, he was the founder and CTO of StealthMine, and spent some time at IBM T.J. Watson Research Centre.



MAURICE HERLIHY

Professor Herlihy is a world expert in Distributed Computation. He is the recipient of the 2003 Dijkstra Prize in Distributed Computing, the 2004 Gödel Prize in theoretical computer science, the 2008 ISCA influential paper award, the 2012 Edsger W. Dijkstra Prize, and the 2013 Wallace McDowell award. He is fellow of the ACM, a fellow of the National Academy of Inventors, the National Academy of Engineering, and the National Academy of Arts and Sciences.

Professor Herlihy holds a Ph.D. in Computer Science from M.I.T.

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, ilvio 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.