



KONTOR

Kontor is a **next-generation Bitcoin metaprotocol** and interoperability layer with a powerful smart contracts system and a scalable, Bitcoin-native file storage system.

Team

The Kontor team has been at the **forefront of the blockchain industry** from the launch of Counterparty in 2014 through the deployment of Symbiont's technology into production in 2019.



Evan Wagner

Evan has over 10 years of experience working in the cryptocurrency and blockchain industry. He studied philosophy and mathematical logic at Hampshire College before going on to co-found Counterparty. He was co-founder and COO of Symbiont, the world's first enterprise blockchain technology company.



Adam Krellenstein

Adam studied physics and philosophy at Hampshire College. He was the primary architect and lead developer of Counterparty. He co-founded Symbiont and served as its CTO, managing a team of up to 75 engineers and heading the development of Assembly, the groundbreaking permissioned blockchain platform.

Bitcoin Can't Scale *Horizontally*

Layer-2s are **economically parasitic**, siphoning transaction volume and fees away from Bitcoin, so they live in the shadow of token metaprotocols like Ordinals and BRC-20.

Whereas Ethereum 2.0 was centrally planned and designed to accommodate rollups, the **Bitcoin network is heterogenous, decentralized, and resistant to change.**

Metaprotocols Build *Up* Not *Out*

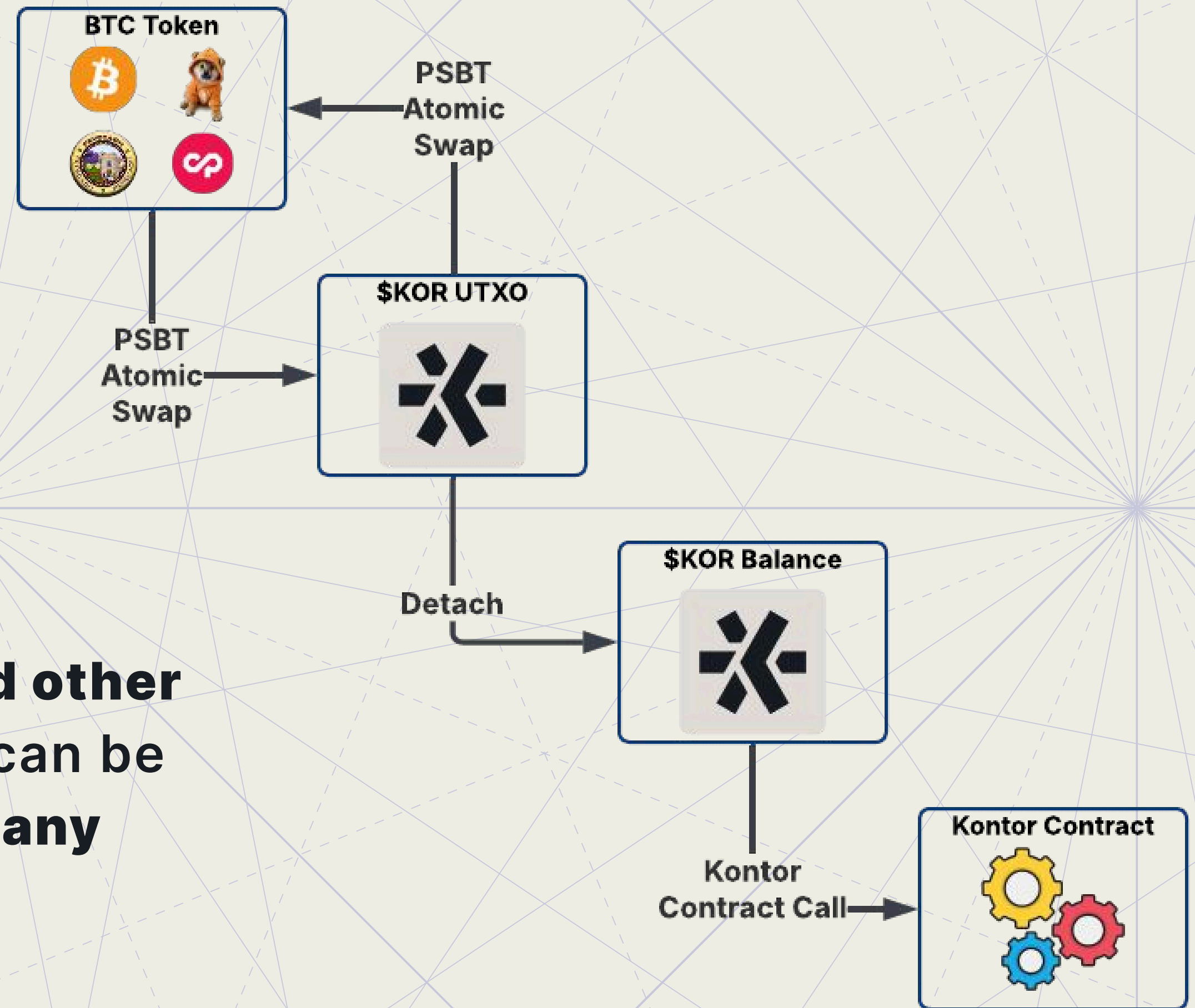
Unlike L2s, **metaprotocols are Bitcoin-native**, increasing the functionality of the network while keeping all value **on-chain**. Metaprotocols don't require new op codes or changes to Bitcoin's security model.

As a **next-generation metaprotocol**, Kontor inherits Bitcoin security model, uses standard BTC addresses, and requires minimal integration work for infrastructure support.

Hybrid UTXO Model

With Kontor's novel **hybrid UTXO model**, assets are attachable to UTXOs for direct interoperability with Bitcoin, but Kontor is also capable of rich, account-based DeFi functionality.

Kontor offers **full interoperability with Bitcoin and other Bitcoin metaprotocols** — trustless atomic swaps can be chained directly to Kontor contract calls, allowing **any Bitcoin user** to interact with Kontor immediately.



KOR Tokenomics

Kontor respects **BTC's primacy as digital money** and the network's unit of account, while KOR's economic model was designed from the ground up for data storage and smart contract execution — **BTC is the gold; KOR is the gas.**

KOR emissions scale based on network health to ensure files remain available forever. User fees are burned so that as network activity increases, inflation remains under control.

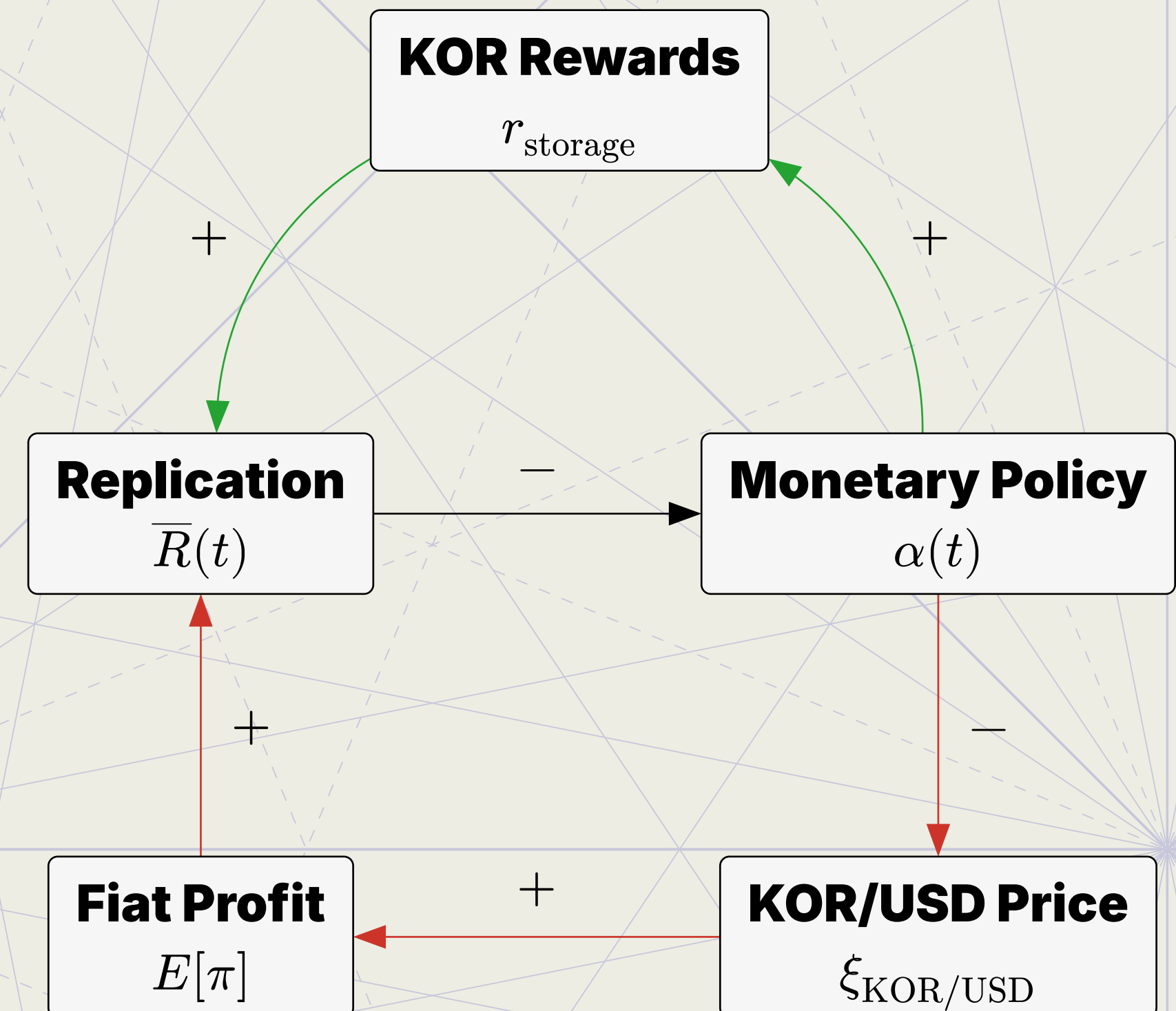


Figure 1: Unified macroeconomic feedback system. Signs show $\frac{\partial y}{\partial x}$ for edge $x \rightarrow y$. The **monetary policy** is stabilizing to counteract the naturally destabilizing **price channel**.

Smart Contracts

Sigil is a **next-generation WebAssembly** framework that enables developers to develop, deploy and run rich smart contract applications directly on the Bitcoin blockchain.

Sigil leverages the WASM Component Model to offer strong compile-time type safety, powerful language built-ins, an expressive storage interface, rich types, **native cross-contract calls** and best-in-class **developer ergonomics**.

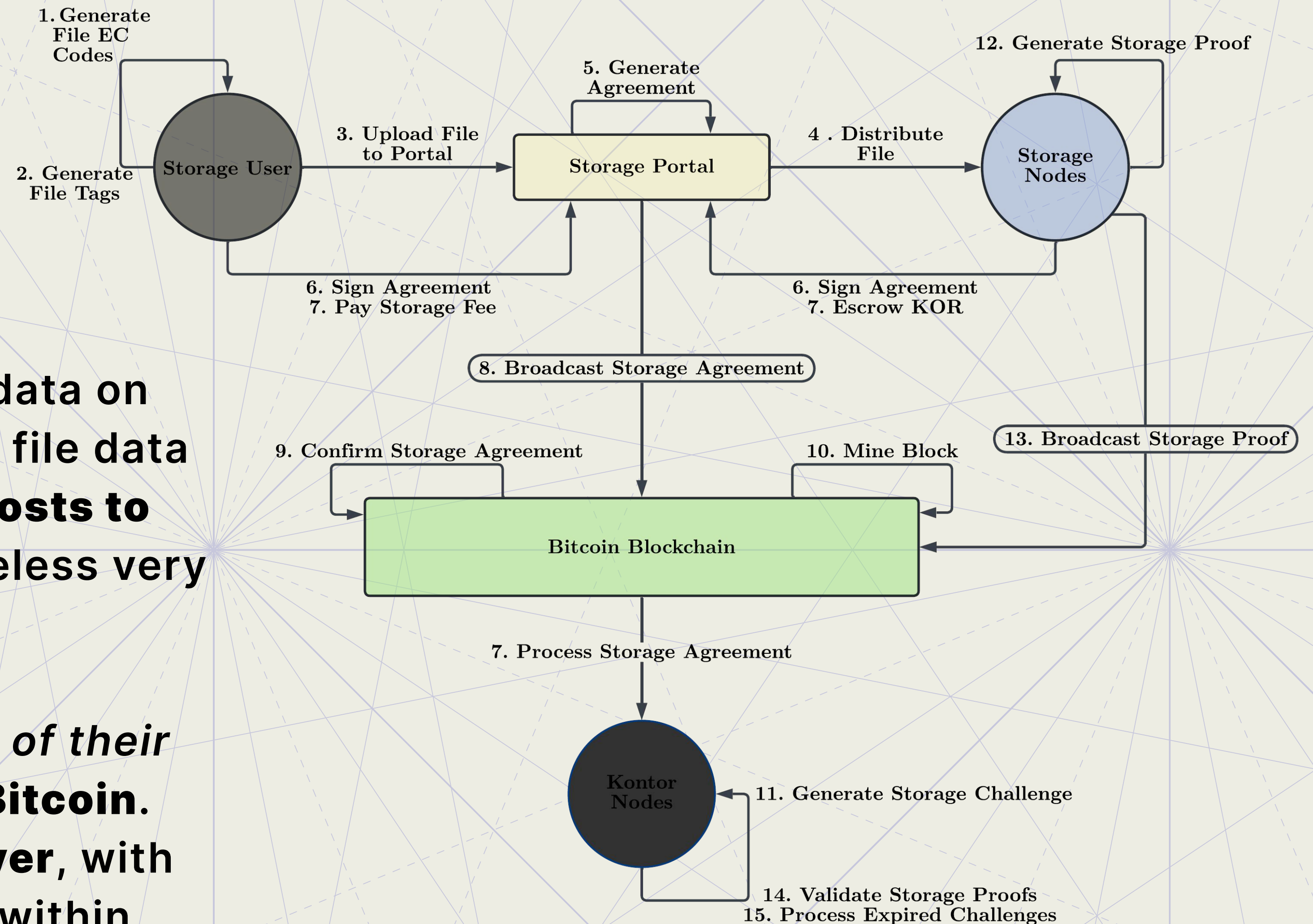
Writing contracts in Sigil **feels like writing normal Rust**, requiring developers to learn a minimum number of new concepts and to jump through as few hoops as possible to write, deploy and run their contracts on Bitcoin.

```
1  use stdlib::*;
2
3  contract!(name = "token");
4
5  #[derive(Clone, Default, StorageRoot)]
6  struct TokenStorage {
7      pub ledger: Map<Account, Integer>,
8  }
9
10 #[init]
11 fn init(){
12     TokenStorage::default().init();
13 }
14
15 #[proc]
16 fn mint(n: Integer) {
17     let to = ctx.signer();
18     let ledger = storage().ledger();
19
20     let balance = ledger.get(&to).unwrap_or_default();
21     ledger.set(to, balance + n);
22 }
23
24 #[proc]
25 fn transfer(to: Account, n: Integer) -> Result<(), Error> {
26     let from = ctx.signer();
27     let ledger = storage().ledger();
28
29     let from_balance = ledger.get(&from).unwrap_or_default();
30     let to_balance = ledger.get(&to).unwrap_or_default();
31
32     if from_balance < n {
33         return Err(Error::new("insufficient funds"));
34     }
35
36     ledger.set(from, from_balance - n);
37     ledger.set(to, to_balance + n);
38     Ok(())
39 }
40
41 #[view]
42 fn balance(acc: Account) -> Option<Integer> {
43     storage().ledger().get(acc)
44 }
```

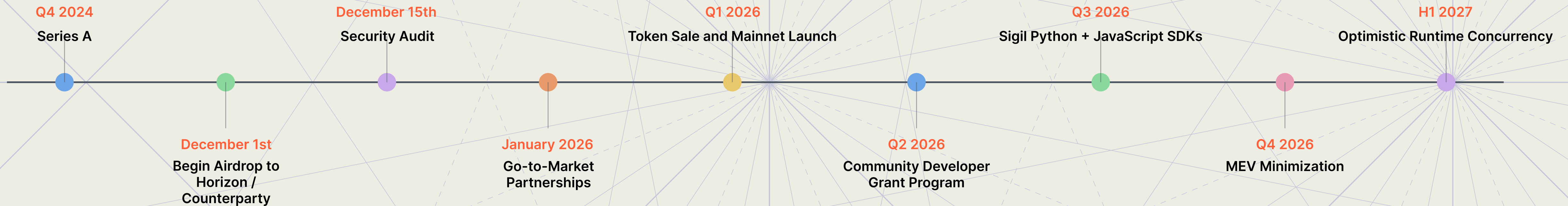
Persistent File Storage

Users desperately want to store their data on the Bitcoin blockchain, but embedding file data directly in transactions **externalizes costs to the whole network** — and it's nevertheless very expensive.

Kontor allows Bitcoin users to *store all of their data off-chain* but still **secure it with Bitcoin**. Data is **paid-for once** and **stored forever**, with the Kontor protocol operating entirely within Bitcoin itself.



Kontor Roadmap



Leverage network of crypto KOLs and influencers
Partner with artists and marketplaces

Organize meetups, conferences and hackathons
Maintain best-in-class DX and documentation

Fundraising and Tokenomics

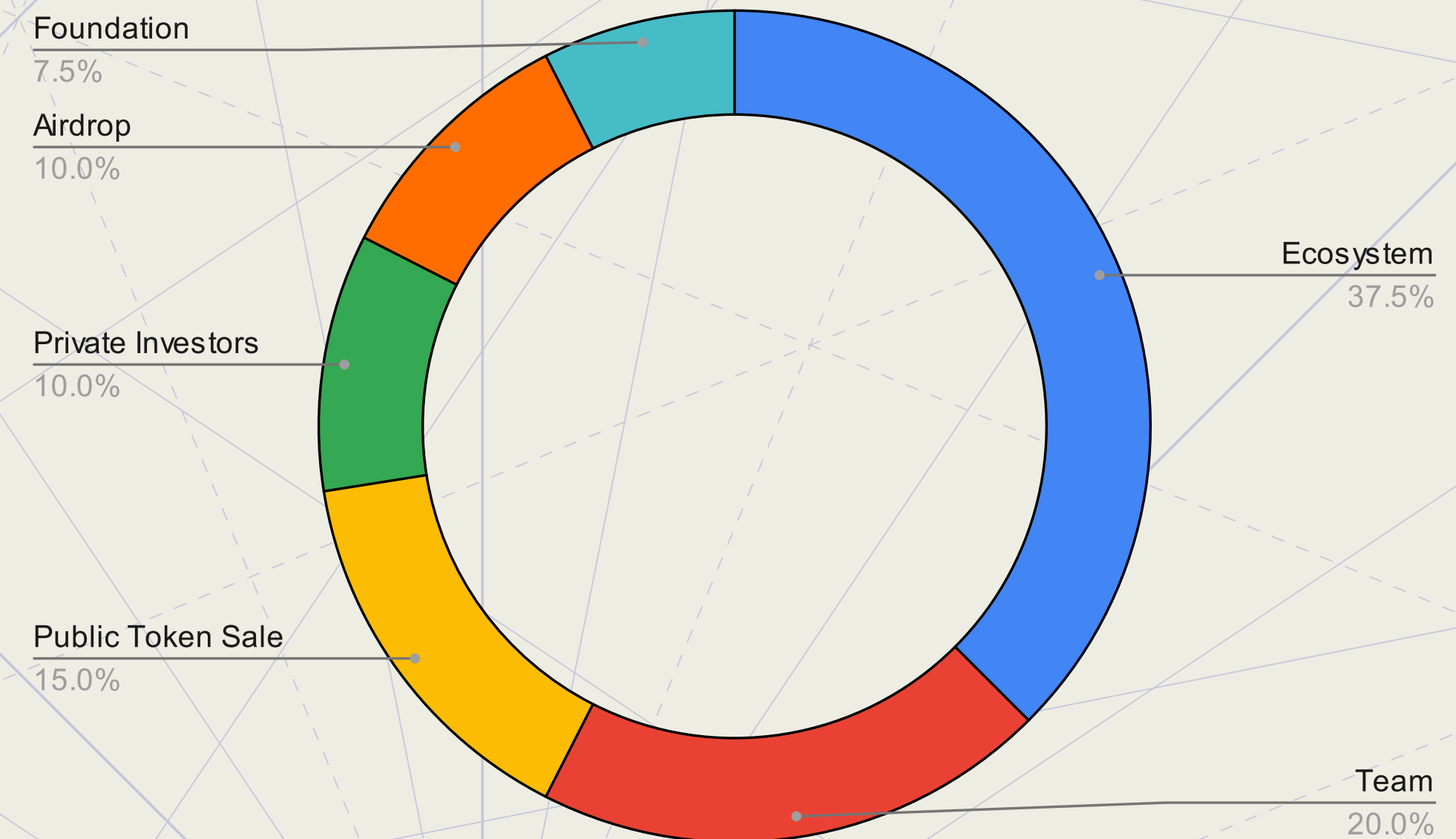
KOR Supply

- Inflation: ~2–5% per year, based on tx volume
- Float: 35% at TGE

Series A

- Raising \$3mm at a \$100mm valuation
- 15 months of runway
- Public token sale shortly before TGE

Initial Token Allocation



Liquid KOR Supply at 5% Net Inflation

