Side Protocol:

A Financial Infrastructure for Bitcoin

Side Labs

v1.1, March 27th, 2025

Table of Content

Abstract

Bitcoin is widely recognized as both a store of value and a payment medium; however, its potential remains underutilized as demand for DeFi applications leveraging Bitcoin continues to grow. Much like traditional banks, which provide a spectrum of financial services based on fiat currencies and real-world assets, there is a need for decentralized infrastructure that offers diverse financial services for Bitcoin, with lending as a primary focus. Despite Bitcoin's decade-long presence, no project has yet fully realized this potential. Side Protocol addresses this gap as an extension layer designed specifically for Bitcoin, enabling a new generation of decentralized financial applications within a Bitcoin-centered internet.

1. Introduction

At the core of Side Protocol is a non-custodial, pool-based lending system for Bitcoin that eliminates the need for third-party custody of BTC collateral. This lending system is implemented as a cross-chain solution, leveraging an alt-chain architecture via Side Chain—a fully Bitcoin-compatible appchain purpose-built for this protocol. This paper focuses primarily on the technical architecture and operational mechanics of the lending system.

2. Preliminary

The core principles of the lending system are straightforward:

- Non-custodial: While BTC can be used as collateral to borrow other assets, the collateralized BTC cannot be arbitrarily spent by the counterparty under any circumstances—unless a liquidation event is triggered. The protocol enforces this by locking all BTC collateral in a designated 2-of-2 multi-signature vault, where funds can only be moved based on pre-signed conditions.
- Pool-based: A few peer-to-peer non-custodial lending protocols have already emerged and are in production. Side's peer-to-pool model offers a more efficient, permissionless, and automated lending market with enhanced DeFi composability. It is designed to evolve into a backend infrastructure for Bitcoin DeFi, enabling apps—including wallets and exchanges—to integrate lending functionality without the need to build their own products, source liquidity, or manage Bitcoin custody.

- Alt-chain risk remains with alt-chain participants: The protocol is designed to minimize risk for borrowers. BTC collateral is securely stored in a dedicated collateral vault created for each borrower—without the use of any wrapped or synthetic BTC. The spending conditions of the vault are pre-signed by the borrower, ensuring that collateral cannot be moved without their consent. This creates a trust-minimized environment where alt-chain risks are isolated to alt-chain participants, without negatively impacting borrowers.

Side leverages established Bitcoin native technologies, including Schnorr-based Adaptor Signatures, Taproot, and FROST, integrated with Discreet Log Contracts (DLCs) and secure decentralized oracles. The combination of these technologies into Scriptless Scripts enables Side to provide smart contract-like functionality on Bitcoin today, without requiring any changes to the underlying Bitcoin protocol's opcodes.

2.1 FROST

FROST (Flexible Round-Optimized Schnorr Threshold Signatures) is a threshold signature scheme based on Schnorr signatures. It enables a subset of t participants out of n to jointly produce a single Schnorr signature that is indistinguishable from one generated by a single signer. FROST improves upon earlier threshold schemes by requiring only two rounds of communication, reducing latency and improving suitability for asynchronous or partially connected environments.

The protocol supports distributed key generation and allows signers to operate non-interactively once nonce commitments have been exchanged. These properties make FROST particularly applicable to blockchain-based systems, including threshold wallets, oracle networks, and smart contract control. Its compatibility with existing Schnorr-based cryptographic systems, such as Bitcoin Taproot, enables efficient integration without changes to verification logic.

2.2 Threshold Adaptor Signatures

Adaptor signatures [1] are a cryptographic construction that augment standard digital signatures, such as those used in Schnorr or ECDSA schemes, by embedding a conditional structure into the signing process. An adaptor signature can be thought of as a "pre-signature"—a valid signature that remains incomplete until a secret value, known as the adaptor, is revealed. Once the adaptor is known, the pre-signature can be transformed into a valid final signature. Crucially, knowledge of both the pre-signature and the final signature allows any observer to extract the adaptor itself. This property enables the design of advanced protocols such as atomic swaps, escrow mechanisms and Discreet Log Contracts, where revealing the adaptor serves as an implicit signal of certain conditions being met. Adaptor signatures therefore serve as a powerful tool for enabling conditional execution and cross-chain interoperability in trust-minimized environments.

A threshold adaptor signature [2] allows a group of signers to collaboratively produce a pre-signature that commits to a secret value (the adaptor), and can only be turned into a full signature once the secret is revealed. Importantly, this is done in a threshold manner, meaning no single signer has full control.

Adaptor signatures offer a cryptographic alternative to hashlocks for conditional payments in Bitcoin. Unlike hashlocks, which require publishing preimages on-chain, adaptor signatures embed the condition directly into the signing process. The final signature reveals the secret implicitly, enabling trustless payment execution with improved privacy and no reliance on on-chain scripts. This makes them well-suited for atomic swaps and contract protocols like DLCs.

2.3 Discreet Log Contracts (DLCs)

Central to Side's infrastructure is the Discreet Log Contracts (DLCs) [2], an oracle-based Bitcoin smart contract framework designed to enhance Bitcoin's programmability. DLCs allow for conditional payments based on off-chain events, all without involving a third-party custodian to manage funds.

These contracts leverage multiple cryptographic methods, including multi-signature transactions, Schnorr signatures [4], and adaptor signatures, ensuring the security and non-custodial nature of the system.

A DLC begins with a funding transaction that locks funds from two parties into a 2-of-2 multisignature output. Pre-signed Contract Execution Transactions (CETs) [5] are created for different potential outcomes of the external event, but these CETs remain inactive until a real event outcome occurs.

The oracle, which is independent of the contract parties, publishes a public key or nonce at the contract's initiation. Participants use this key to generate adaptor signatures for the CETs. When the oracle reveals the outcome by disclosing a signature, the participants can finalize the relevant CET and broadcast it to the Bitcoin network, redistributing the locked funds according to the contract's terms.

DLCs provide a secure, trust-minimized method to execute conditional payments and smart contract logic, pushing Bitcoin's capabilities for DeFi applications.

3. Architecture

Side Chain, the Side Protocol blockchain, is a fully-Bitcoin compatible appchain that leverages Cosmos-SDK [6] and CometBFT [7], a high-performance consensus engine that serves as its backbone. This architecture facilitates fast transaction finality and high throughput, making it ideal for applications that require quick confirmation times.

3.1 Bitcoin Compatibility Layer

Side Chain is specifically crafted with Bitcoin-compatible features to provide a seamless and frictionless user experience:

- Bitcoin Address Compatibility: Unlike many other Bitcoin L2s or sidechains, Side Chain supports native Bech32/Bech32m address formats, enabling users to interact with Side Chain using their existing Bitcoin mainnet Taproot or Native SegWit addresses. This eliminates the need to create new wallets or manage multiple addresses.
- Bitcoin Wallet Compatibility: All transactions on Side Chain can be signed directly using popular Bitcoin wallets such as OKX Wallet, Unisat, or hardware wallets like Ledger.
- Embedded On-chain Bitcoin Light Client: Side Chain includes a built-in SPV (Simple Payment Verification) client that verifies Bitcoin mainnet transactions with 6 confirmations

3.2 Oracle++

In our DLC-based lending system, an oracle must cryptographically sign periodic event attestations to enable automated liquidations of BTC collateral when necessary. Oracle++ is Side Protocol's decentralized oracle system designed to fulfill this requirement through two core components:

Data Provider Network (DPN)

The primary participants in the Data Provider Network (DPN) are Side Chain validators, who perform two key functions. First, they synchronize Bitcoin block headers by implementing a voting power–weighted majority decision mechanism, requiring consensus from validators representing at least two-thirds of total network voting power. This ensures accurate and consistent tracking of Bitcoin block heights. Second, they provide aggregated price feeds by sourcing data from major cryptocurrency exchanges and computing a weighted average price based on validator voting power, thereby improving reliability and mitigating manipulation risk. These off-chain data inputs are submitted via Cosmos Vote Extensions [8], which offer a robust framework for incorporating off-chain data into on-chain consensus using ABCI++ enhancements.

Event Signer Network (ESN)

Complementing the Data Provider Network is the Event Signer Network (ESN), which is specifically designed to support DLCs. Oracle++ introduces a robust security architecture that minimizes reliance on any single oracle. At its core is the FROST scheme, which enables threshold-optimized Schnorr signatures and provides Byzantine fault tolerance with a 15-of-21 signing threshold. The 21 participants are randomly selected from the top 50 active validators.

Unlike traditional DLC oracles that rely on fixed public keys, Oracle++ uses Distributed Key Generation (DKG) to produce two ephemeral keypairs for each event: one for the one-time oracle signing key, and one for the nonce. The private keys are never known by any individual; instead, they are secret-shared among 21 participants, each holding one share. These keys are used to collectively sign the event attestation. DLC participants use the adaptor point to construct the CET signature, which can later be completed upon receiving the corresponding attestation signature.

This design ensures that only actively participating validators are involved in key generation, while maintaining fault tolerance by allowing up to one-third of signers to be offline during attestation.

Event Announcement

Price Event: Price events represent continuous numerical outcomes, such as BTC/USD prices, which cannot be discretely enumerated. To handle this, the system employs Numeric Outcome Compression, a mechanism that maps price values into fixed-length intervals. For example, with a compression length of 100, the price map would include ranges such as (0, [0,100)), (100, [100,200)), …, (99900, [99900,100000)), where each entry defines a valid range for price resolution. Upon finalizing a price attestation, the system automatically generates a new price event for the next monitoring interval, ensuring continuous and timely tracking of market data.

Date Event: ESN maintains a 365-day rolling event calendar aligned with maximum loan tenures. The oracle executes automated daily updates to advance the coverage window, ensuring continuous temporal attestation availability.

Loan Event: Initialized during loan origination, repayment-triggered events create cryptographic proof bindings between mainchain DLCs and Side Chain states. This mechanism enables non-custodial collateral redemption without DCM coordination.

Event Attestation

Upon event maturity, the oracle generates threshold signatures attesting to canonical outcomes. These cryptographically signed attestations are published on-chain, enabling DLC participants to deterministically unlock CETs through adaptor signature decryption.

3.3 Trustless Relayer

The trustless relayer enables automated cross-chain transaction relays between Bitcoin and the Side Chain by leveraging cryptographic state verification. The relayer continuously scans Bitcoin blocks with at least 6 confirmations to ensure transaction finality and security. It filters and indexes only transactions that involve predefined vault addresses. It ensures transaction integrity and validity through:

- Merkle Inclusion Proof generation and verification
- Stateless transaction validity checks

It is designed with two fundamental properties: permissionlessness and verifiability. Being permissionless means that any participant can operate a relayer client, as the software is fully open-source and adheres to a standardized set of proof generation rules. This design ensures that there are no centralized points of control or bottlenecks in the relaying process.

Verifiability is achieved through the requirement that every relay action must be accompanied by a Simplified Payment Verification (SPV) proof. These proofs allow the Side Chain to independently verify the authenticity and finality of Bitcoin transactions without relying on a trusted third party. This architecture upholds the trust-minimized nature of the protocol by ensuring that only valid, confirmed Bitcoin transactions can trigger events or unlock assets on the Side Chain.

3.4 Native BTC-Collateralized Lending System

3.4.1 Participants

Borrower: The party receiving the loan by using native BTC as Collateral

Liquidity Provider: Users supplying the loan through the Lending Contract

DCM (Distributed Collateral Manager): A decentralized network of operators organized into a threshold adaptor signature scheme. DCM operators sign Bitcoin 2-of-2 multi-sig transactions on behalf of the Lending Contract, with the counterparty being the Borrower. Additionally, the DCM manages Liquidated Assets in the event of a liquidation.

Lending Contract: A smart contract deployed on the Side Chain that automates the operations of the lending pool. This contract enables liquidity providers to supply assets for lending and earn rewards in return. While the Lending Contract itself does not have the capability to sign transactions, it delegates this function to the DCM, which signs transactions on behalf of the contract and in collaboration with the borrower.

3.4.2 Glossary

Collateral: The BTC assets pledged by the Borrower to borrow the loan

Collateral Vault: A Bitcoin Taproot address where borrowers send their BTC as collateral for lending. Each loan has its own unique vault address designated for its collateral

Principal: The original amount of assets borrowed

Maturity Time: The deadline by which the Borrower must repay the loan in full. If the Borrower fails to repay by this date, the DCM may liquidate the Collateral to recover the outstanding debt

Liquidated Assets: Collateral liquidated and sent to the DCM due to the borrower's failure to fulfill payments on the principal and interest of the loan before Maturity Time

Loan Default: The failure to repay a loan by the Maturity Time, which can result in the liquidation of Collateral

Liquidation Price: The BTC price at which a loan becomes undercollateralized, triggering the execution of CETs

Final Timeout: The specified time after which the Borrower is entitled to reclaim the collateral if the DCM or Side Chain becomes unresponsive.

3.4.3 Lending Workflow Overview

Side defines a liquidity pool-based lending protocol that provides loans to BTC holders and generates returns for loan providers. The loan assets are pooled in smart contracts on the Side Chain and offered to borrowers without involving any third party holding the collateral during the loan period. At a high level, the protocol proceeds as follows:

1. Loan Assignment
    a. To initiate the loan process, the borrower submits a request to the Lending Contract, including a specified Maturity Time. Upon receipt, the Lending Contract assigns a

    dedicated Collateral Vault for the borrower, which will be used to lock the BTC collateral.

    b. The Lending Contract then computes the applicable interest rate and liquidation price, derived from real-time BTC market data. It subsequently reserves the corresponding loan amount from the available liquidity pool for the borrower.
    c. To claim the loan, the borrower must deposit BTC into the assigned Collateral Vault and submit two pre-signed CETs to the Lending Contract. The first CET enables liquidation in the event that the BTC collateral falls below the defined liquidation threshold. The second CET is executed in the case of loan default, i.e., if the borrower fails to repay by the maturity date.
    d. In parallel, the DCM pre-signs a third CET. This CET is executed only upon verified loan repayment and facilitates the release of the collateral back to the borrower.
    e. Once the CETs are verified and the collateral funding transaction is confirmed on Bitcoin, the borrower receives the loan on the Side Chain.
2. Repayment
    a. To avoid default liquidation, the borrower must repay the loan before the Maturity Time. Upon repayment, a repayment transaction is sent to the Lending Contract covering both principal and interest.
    b. Event Signers attest to the repayment and publish a signature on the Side Chain. Using this attestation, anyone can reveal the corresponding CET adaptor signature and broadcast it on the Bitcoin network to release the collateral back to the borrower.
3. Liquidations
    a. Collateral may be liquidated in two cases: if the BTC price drops below the liquidation threshold or if the borrower fails to repay by the Maturity Time.
    b. In both scenarios, Event Signers publish attestations that unlock pre-signed CETs, allowing the DCM to take custody of the collateral.
    c. The DCM then sells the BTC at market value, repays the loan to the pool, and returns any surplus to the borrower.
4. Final Timeout
    a. A `final_timeout` protects the Borrower in the event that the Lending Contract or Side Chain becomes unresponsive.

In the following sections, we will provide more details of each step involved in the lending process as outlined above.

3.4.4 Loan Assignment

The Borrower sends a loan request to the Lending Contract to initiate a loan.

```JavaScript
{
  "borrower":"bc1p5d7rjq7g6rdk2yhzks9smlaqtedr4dekq08ge8ztwac72sfr9rusxg3297",
  "maturityTime": "2024-10-10T00:00:00z",
}
```

The Lending Contract then assigns a unique vault, the Collateral Vault, for the loan. Expressed in a Miniscript-like pseudocode, the lock script for this vault is as follows:

```JavaScript
or(
    // case 1 - the DCM and Borrower can collaborate to create CETs
    and(
        pk(borrower),
        thresh(T, pk(DCM0),pk(DCM1),...pk(DCMn))
    ),

    // case 2 - collateral reverts to Borrower after Final Timeout(15 days
after Maturity Date),
    and(
        pk(borrower),
        after(final_timeout)
    )
)
```

The Collateral Vault serves as the foundation for all collateral-related operations. In this paper, we will examine each spending condition in detail. For now, here are some initial observations to help build understanding.

The DCM multi-sig participants are indexed from `0..n`, with a threshold `T` of signatures required to spend.

The aggregated adaptor signature setup in the script above merits further examination. We can formally describe the Adaptor Signature process as follows. Let `λ` represent the secret scalar, and `G` be the base point of the elliptic curve. The adaptor point `A` is computed as `A = λG`. Let `m` denote the message, which is pre-signed signature of Bitcoin transaction: `m = sig_hash(psbt)`

Given a private key `sk` and the adaptor point `A`, we define the adaptor signature `σ_A = SignAdaptor(sk, m, A)`.

The function `SignAdaptor` represents the cryptographic operation to create an adaptor signature. To redeem the signature, we use the secret `λ` to adapt `σ_A`, producing the final signature `σ = Adapt(σ_A, λ)`.

In this formulation:

- `λ ∈ Zq` (the scalar field of the curve)
- `A, G ∈ E(Fq)` (points on the elliptic curve)
- `m ∈ {0,1}^256` (256-bit hash output)
- `σ_A`, `σ` are elements of the signature space

In summary, from the Bitcoin chain's perspective, the resulting signature appears as a standard Schnorr signature. However, this scheme modifies the signing process to produce a special adaptor signature, which can embed a secret to be revealed at a later stage. This adaptor signature enables trust-minimized loan repayment by leveraging *Scriptless Scripts* within the Bitcoin framework.

The Lending Contract also needs to compute the interest and liquidation price based on the current price and pre-allocate funds for this loan, representing it as the Loan Request:

```JavaScript
{
 "borrower":"bc1p5d7rjq7g6rdk2yhzks9smlaqtedr4dekq08ge8ztwac72sfr9rusxg3297",
 "maturityTime": "2025-10-10T00:00:00z",
 "maturityEvent": "9f33f577811da09fc90ad46586308e8c75acef9201fb1a66c",
 "borrowAmount": "20000",
 "vaultAddress":
"bc1p6r4u4qlajya6feu337gtngksgeyf4nf0mf4q35gtcj5v0ja9m00q3eagh3",
 "Oracle": "86308e8c75acef9201fb1a66c9f33f577811da09fc90ad465",
 "currentPrice": "50000.00",
 "liquidate_price": "30000.00",
 "liquidate_price_event": "6586308e8c75acef9201fb1a66c9f33f577811da09fc90ad4",
 "collateralAmount": "1",
 "createAt": "2024-10-10T10:10:10z"
}
```

If the Borrower accepts the loan terms, they proceed by transferring the BTC collateral to the assigned Collateral Vault. Once the deposit is confirmed and verified on the Side Chain, the Lending Contract generates three CETs, which are then co-signed by both the Borrower and the DCM. These CETs correspond to the possible settlement outcomes of the loan agreement:

- Default Liquidation CET: executed if the loan reaches maturity without repayment.
- Price Liquidation CET: triggered if the attested BTC price falls below the liquidation threshold.
- Repayment CET: executed upon successful loan repayment, releasing the collateral back to the borrower.

In code, the adaptor signature generation process is as follows:

```javascript
// hide CET's signature with adaptor point in the event announcement
let message = sig_hash(psbt);
let adaptor_signature = sign_adaptor(seckey, message, event.adaptor_point);
...
// Later, reveal CET's signature by oracle's signature
let redeem_signature = adaptor_signature.adapt(signature_of_attestation);
```

The Borrower then submits the CETs and corresponding adaptor signatures to the Lending Contract on the Side Chain in order to claim the loan assets (e.g., USDC).

At this point, the collateral is securely locked in the Collateral Vault, and the Borrower has successfully received the loan. The Borrower may choose to repay the loan at any time prior to the Maturity Date.

3.4.5 Repayment

The borrower must repay the loan before the Maturity Time to avoid default liquidation. Assuming the loan is denominated in USDC, the repayment flow proceeds as follows:

1. The borrower submits a USDC transfer transaction to the Lending Contract to repay both the principal and accrued interest.
2. Event Signers sign an attestation of the repayment event, which is triggered by the Lending Contract, and publish the signature on the Side Chain.
3. Anyone (including relayers) can use the Event Signers' attestation signature to reveal the corresponding CET adaptor signature, and broadcast the CET on the Bitcoin network to release the collateral back to the borrower.

3.4.6 Liquidation

Collateral can be liquidated under two scenarios:

1. Collateral Value Depreciation: Price liquidation is governed by a DLC established during the Loan Assignment phase. If the Event Signers publish a signed attestation confirming that the BTC price has reached the liquidation threshold, the DCM uses the attestation signature to unlock the pre-signed CET. This allows the DCM to spend the BTC collateral according to the terms of the CET, thereby initiating the liquidation process and transferring the collateral to the DCM address for sale.
2. Default: The Event Signers regularly provides signed attestations of the date event at predefined intervals (e.g., every 24 hours). If the loan reaches its Maturity Time without repayment, the attestation corresponding to the loan's expiration date will be used to reveal the default CET's adaptor signature. The collateral is then transferred from the Vault to the DCM for liquidation, as dictated by the loan contract terms.

Liquidation Process

The liquidation process is designed to recover as much of the loan value as possible when a position becomes undercollateralized. The DCM plays a central role in managing risk for Liquidity Providers during this process.

The DCM performs the following tasks:

1. Receives Liquidated Assets: When liquidation is triggered, the DCM takes custody of the BTC collateral from the Collateral Vault.
2. Selling Collateral: The DCM lists the collateral for sale at current market value. A liquidation bonus (e.g., 5%) is offered to incentivize liquidators. A portion of this bonus is captured by the protocol via a reserve factor, generating protocol revenue.
3. Repaying the Pool: Proceeds from the sale are sent to the Lending Contract to repay the loan principal and accrued interest.
4. Surplus or Deficit: If there is a surplus from the sale, it is returned to the borrower to minimize the impact of liquidation. However, if the sale proceeds are insufficient to cover the loan principal, the liquidity providers incur the loss.

3.4.7 Loss of Liveness

A `final_timeout` timelock in the Collateral Vault safeguards the Borrower in the event that the DCM or the Side Chain becomes unresponsive. In such cases, all locked BTC collateral is returned to the Borrower, mitigating liveness risks associated with non-Bitcoin components.

The `final_timeout` must occur after the loan's Maturity Time.

4. Security

Side operates as a non-custodial solution, meaning no third party holds the native BTC on the Bitcoin blockchain throughout the loan's duration.

4.1 BTC Collateral Security

The Collateral Vault is secured using a 2-of-2 multi-signature scheme, ensuring that BTC collateral cannot be spent without the Borrower's authorization. This mechanism enforces strict adherence to Bitcoin-native security principles and eliminates the possibility of unilateral control by any single party.

There are four valid spending paths for the collateral: (1) execution of one of three pre-signed CETs—covering price-based liquidation, loan default, or successful repayment; and (2) a hash time-locked refund mechanism that allows the Borrower to unilaterally reclaim collateral after a predefined timeout if the entire system becomes unresponsive.

At no point can the collateral be spent outside of these conditions. All control paths are strictly encoded within the DLC, ensuring the collateral remains non-custodial and fully governed by cryptographic enforcement.

4.2 Loan Asset Security

The Lending Contract, a smart contract deployed on the Side Chain, governs the management of loan assets according to predefined rules encoded within its framework. Liquidity Providers have the flexibility to enter or exit the liquidity pool at their discretion.

4.3 DCM Security

The DCM acts on behalf of the Lending Contract to co-sign required transactions. However, it's important to emphasize that the DCM has no control over the Lending Contract itself or the Collateral Vault. Its function is strictly limited to managing bad debt on behalf of liquidity providers. Lenders' funds remain non-custodial, securely locked within smart contracts at all times. The DCM only temporarily holds liquidated BTC collateral for the purpose of liquidation. To reduce risk and ensure timely liquidation, a bonus is offered to incentivize third-party liquidators to sell the collateral quickly.

To preserve the trust-minimized nature of the protocol, the DCM and the Event Signers must remain entirely independent entities.

Misconduct or failure by any DCM operator may result in strict penalties, including removal from the role. Furthermore, all DCM operators are required to undergo Know Your Customer (KYC) verification to ensure accountability and regulatory compliance.

4.4 Oracle Security

Side Protocol's Oracle++ architecture separates the traditional role of a Bitcoin DLC oracle into two distinct components: Data Providers and Event Signers. This separation significantly increases the difficulty of fraud, as Oracle++ restricts Event Signers to attesting only to events generated from data provided by Data Providers.

Crucially, Event Signers cannot directly benefit from liquidation events, since only the borrower or the DCM can receive collateral. They also cannot fabricate arbitrary prices, as all valid attestations must be threshold-signed by a predefined quorum of signers. No individual Event Signer can unilaterally manipulate the protocol.

Unlike traditional DLC oracle designs—where a single party holds both the oracle's private key and the λ of the adaptor point—Oracle++ uses DKG to derive these values. This ensures that no single operator knows the full private key and secret λ, eliminating the risk of key leakage or abuse.

In terms of liveness, Oracle++ ensures timely event announcements through randomized selection of active operator candidates. Signing resilience is provided via the FROST threshold signature scheme. Additionally, all oracle operators are KYC-verified to ensure accountability.

References

[1] Bitcoin Ops. Adaptor Signatures. https://bitcoinops.org/en/topics/adaptor-signatures/
[2] Yunfeng Ji. Threshold/Multi Adaptor Signature and Their Applications in Blockchains. https://www.mdpi.com/2079-9292/13/1/76
[3] Thaddeus Dryja. Discreet Log Contracts. https://adiabat.github.io/dlc.pdf
[4] Victor Shoup. The Many Faces of Schnorr. https://eprint.iacr.org/2023/1019.pdf

[5] DLC Specs. Introduction to DLCs.

https://github.com/discreetlogcontracts/dlcspecs/blob/master/Introduction.md

[6] Cosmos SDK. https://docs.cosmos.network/v0.52/build/building-modules/intro

[7] CometBFT. https://docs.cometbft.com/v0.38/

[8] Vote Extension. https://docs.cosmos.network/v0.52/build/abci/vote-extensions