

## **0.1 Introduction**

This chapter establishes the theoretical framework and technological infrastructure utilized to construct the Multi-chain Stablecoin Protocol. The methodology is grounded in a rigorous analysis of distributed ledger technologies, cryptographic primitives, and financial stability mechanisms. By synthesizing the high-performance capabilities of the Solana blockchain with the deep liquidity of EVM-compatible networks, the proposed solution adopts a Hub-and-Spoke architecture. This chapter provides a detailed exposition of the selected blockchain platforms, the software engineering principles behind the smart contracts, the mathematical foundations of the cross-chain verification process, and the economic theory underpinning the Collateralized Debt Position (CDP) mechanism.

## **0.2 Blockchain Platforms and Architecture**

### **0.2.1 Blockchain Fundamentals and Distributed State Machines**

Fundamentally, a blockchain is a distributed, immutable ledger that maintains a shared state across a network of unreliable nodes without requiring a central authority. Introduced conceptually by Satoshi Nakamoto in the seminal Bitcoin whitepaper, the technology relies on cryptographic primitives and consensus algorithms to ensure data integrity [1]. While Bitcoin introduced the concept of decentralized currency, the technology evolved significantly with the advent of Ethereum, which generalized the blockchain into a "Transaction-Based State Machine." In this model, the state of the system encompasses not just account balances but also arbitrary code execution logic, known as smart contracts [2].

For a Multi-chain Stablecoin Protocol, the blockchain serves as the trusted substrate for financial agreements. The core requirement is the guarantee of atomicity and state transition validity. When a user creates a Collateralized Debt Position (CDP), the blockchain ensures that the state transition from "solvent" to "liquidated" occurs deterministically based on predefined mathematical rules. However, the "Blockchain Trilemma" posits that a single network typically trades off between scalability, security, and decentralization. This limitation necessitates the architectural decision to separate the system into specialized layers: a high-performance execution layer for state management and secure, high-liquidity layers for asset custody.

### **0.2.2 The Hub: Solana Blockchain Architecture**

To address the scalability limitations inherent in traditional blockchains, this project selects Solana as the central "Hub" for orchestrating the global state of the protocol. Unlike the single-threaded processing model of the Ethereum Virtual Ma-

chine (EVM), Solana introduces a novel architecture optimized for high throughput and sub-second latency, making it ideal for real-time risk management in a CDP system [3].

The distinguishing feature of Solana is Proof of History (PoH), a cryptographic clock that allows nodes to agree on the order of events without waiting for conventional consensus communication overhead. PoH creates a historical record that proves that a specific event occurred at a specific moment in time. This mechanism functions in tandem with the Tower BFT consensus algorithm to achieve finality times of approximately 400 milliseconds. Furthermore, Solana utilizes the Sealevel runtime, which enables the parallel processing of smart contracts. In the context of this thesis, Sealevel allows the protocol to update the Health Factors of thousands of Universal Wallets simultaneously. If market volatility triggers mass liquidations, the parallel architecture prevents network congestion, ensuring that the protocol remains solvent where other single-threaded chains might fail due to transaction backlogs.

### **0.2.3 The Spokes: Ethereum Virtual Machine (EVM) Networks**

While Solana provides the necessary performance for state logic, the Ethereum Virtual Machine (EVM) ecosystem represents the center of liquidity and user adoption in decentralized finance. The EVM is a quasi-Turing complete stack-based virtual machine that executes smart contract bytecode [4]. Chains such as Ethereum, Binance Smart Chain (BSC), Arbitrum, and Optimism all adhere to this standard, forming the "Spokes" of the proposed architecture.

The decision to utilize EVM-compatible chains as the Asset Custody Layer is driven by the standardization of token interfaces, primarily ERC-20. The EVM architecture manages state through a Modified Merkle Patricia Trie, ensuring a cryptographically secure mapping of account storage. In this project, Solidity smart contracts deployed on these chains act as "vaults." They leverage the robust security properties of the EVM to lock collateral assets (like ETH or WBTC) while delegating the complex calculation logic to the Solana Hub. This hybrid approach allows the protocol to tap into the deep liquidity and established developer tooling of the EVM ecosystem while bypassing its scalability bottlenecks for high-frequency operations.

## **0.3 Smart Contract Development Frameworks**

### **0.3.1 Solana Program Development: Rust and Anchor**

The implementation of the protocol's core logic on the Solana Hub necessitates a programming environment capable of handling high-concurrency execution

while ensuring rigorous memory safety. For this purpose, the Rust programming language is selected, utilized within the context of the Anchor Framework. Rust provides a distinctive advantage over other low-level languages like C++ through its ownership and borrowing system, which prevents common classes of bugs such as null pointer dereferencing and buffer overflows at compile time.

In the specific context of Solana, raw development using native entry points can be verbose and prone to security oversights regarding account validation. The Anchor Framework addresses this by providing a comprehensive Interface Definition Language (IDL) and a dispatch system that abstracts away the complexities of serialization and deserialization [5]. Crucially, Anchor enforces strict checks on account ownership and mutability by default. For the Multi-chain CDP Protocol, where the state of the Universal Wallet is critical, Anchor's discriminator feature ensures that a malicious actor cannot inject a fake data account into the system to manipulate debt positions. This combination of Rust's performance and Anchor's security constraints constitutes the ideal environment for the system's "State Layer."

### **0.3.2 EVM Smart Contracts: Solidity and Hardhat**

On the EVM Spokes, the priority shifts from high-performance computation to secure asset custody and standardization. Consequently, the Solidity programming language is employed to develop the Gateway and Vault contracts. Solidity is the lingua franca of the Ethereum ecosystem, offering the highest degree of compatibility with existing token standards such as ERC-20. The development lifecycle is managed using the Hardhat environment, which facilitates local network forking, automated testing, and scriptable deployment pipelines.

To mitigate the risks associated with locking user funds, the project integrates the OpenZeppelin library. Rather than implementing cryptographic primitives or access control mechanisms from scratch, the system inherits from battle-tested contracts including `Ownable` for administrative privileges and `ReentrancyGuard` to prevent reentrancy attacks during collateral withdrawal [6]. This strategic choice allows the system to adhere to industry best practices for security while maintaining interoperability with the broader DeFi ecosystem on chains like Arbitrum and Binance Smart Chain.

## **0.4 Cryptography and Cross-chain Verification**

### **0.4.1 Elliptic Curve Mismatch and Solution**

A fundamental technical challenge in implementing a "Universal Wallet" lies in the cryptographic incompatibility between the disparate blockchain networks.

The Ethereum ecosystem relies on the Elliptic Curve Digital Signature Algorithm (ECDSA) utilizing the **secp256k1** curve. In contrast, the Solana blockchain is built upon the **Ed25519** curve, which offers faster signature verification times but different mathematical properties. This discrepancy implies that a standard Solana smart contract cannot natively verify a signature generated by an Ethereum private key without significant computational overhead.

#### 0.4.2 Mathematical Formulation of ECDSA Verification

To bridge this gap, the protocol requires a mechanism to mathematically prove that a transaction request  $m$  originating from an EVM address  $A_{evm}$  was indeed authorized by the holder of the private key  $d$ . The domain parameters of the secp256k1 curve are defined over a finite field  $\mathbb{F}_p$  by the Weierstrass equation:

$$y^2 \equiv x^3 + 7 \pmod{p} \quad (1)$$

When a user signs a cross-chain request, they produce a signature pair  $(r, s)$ . The verification process, which must be executed on the Solana Hub, involves recovering the public key point  $Q$  from the signature. This is achieved using the inverse operation of point multiplication:

$$Q = r^{-1}(sR - zG) \quad (2)$$

Where  $z$  is the hash of the message  $m$ ,  $G$  is the generator point of the curve, and  $R$  is the point with x-coordinate  $r$ . The derived Ethereum address is then the last 20 bytes of the Keccak-256 hash of  $Q$ :

$$A_{evm} = \text{Keccak256}(Q)[12..32] \quad (3)$$

#### 0.4.3 Implementation via Solana Native Program

Implementing the arithmetic operations of Equation 2 directly in a high-level language like Rust would consume an excessive amount of Compute Units (CU), potentially exceeding the block limit. To solve this, the methodology employs Solana's **Native Secp256k1 Program**. This is a precompiled BPF program embedded in the validator software that executes signature recovery at a fraction of the computational cost. By invoking this program via Cross-Program Invocation (CPI), the Main Contract can trustlessly verify EVM signatures, thereby allowing users to control their Solana state using their existing MetaMask wallets without compromising security.

## 0.5 Theoretical Foundation: The CDP Mechanism

### 0.5.1 Economic Model of Stability via Over-collateralization

The financial architecture of the proposed protocol is grounded in the Collateralized Debt Position (CDP) model, a primitive that enables the creation of decentralized stablecoins without reliance on fiat reserves or centralized custodians. Unlike algorithmic stablecoins which often depend on endogenous collateral and reflexive arbitrage cycles, mechanisms historically prone to "death spirals" during market downturns. In this protocol, the CDP model enforces stability through strict Over-collateralization. This economic theory posits that the solvency of the system and the peg of the stablecoin are guaranteed as long as the aggregate value of the locked collateral assets significantly exceeds the value of the issued debt. In this system, the stablecoin is not created out of thin air but is backed by a diversified basket of crypto-assets (such as ETH, WBTC) locked across various EVM chains, serving as a secure liability against the protocol's assets.

### 0.5.2 Borrowing Capacity and Loan-To-Value (LTV) Formulation

A critical component of the CDP mechanism is the determination of a user's borrowing power. This is governed by the Loan-To-Value (LTV) ratio, a risk parameter assigned to each asset type based on its market volatility and liquidity profile. The LTV ratio, denoted as  $\alpha$ , represents the maximum percentage of the collateral's value that can be minted as debt. For instance, a stable asset like USDC might have a high  $\alpha$  (e.g., 0.90), while a volatile asset like ETH might have a lower  $\alpha$  (e.g., 0.75).

Mathematically, the Maximum Borrowable Debt ( $D_{max}$ ) for a user holding a portfolio of assets across multiple chains is the sum of the risk-adjusted value of each asset. For a user depositing  $N$  different asset types distributed across  $M$  blockchain networks, the borrowing capacity is formalized by the following equation:

$$D_{max} = \sum_{k=1}^M \sum_{i=1}^N (Q_{k,i} \cdot P_i \cdot \alpha_i) \quad (4)$$

In this equation,  $Q_{k,i}$  represents the quantity of asset  $i$  locked on chain  $k$ , and  $P_i$  is the real-time market price of asset  $i$  denominated in the reference currency (USD). The term  $\alpha_i$  acts as a dampening factor, ensuring that the protocol accounts for potential price drops before they occur. Consequently, the actual debt  $D_{actual}$  minted by the user must always satisfy the condition  $D_{actual} \leq D_{max}$  at the time of

issuance.

### 0.5.3 The Health Factor Function

The system solvency is mathematically modeled using the Health Factor ( $H_f$ ). This metric is dynamic and is updated in real-time based on oracle price feeds. For a multi-chain protocol, the Health Factor is an aggregate function of all collateral assets  $i$  across all connected chains  $k$ :

$$H_f = \frac{\sum_{k=1}^M \sum_{i=1}^N (Q_{k,i} \times P_i \times \lambda_i)}{D_{total} \times P_{peg}} \quad (5)$$

In this equation:

- $Q_{k,i}$  represents the quantity of asset  $i$  locked on chain  $k$ .
- $P_i$  is the current market price of asset  $i$  denominated in USD.
- $\lambda_i$  is the Liquidation Threshold specific to the asset's risk profile (e.g.,  $\lambda_{ETH} = 0.80$ ,  $\lambda_{USDC} = 0.95$ ).
- $D_{total}$  is the total outstanding stablecoin debt.
- $P_{peg}$  is the target peg price of the stablecoin (e.g., 1 USD).

The methodology enforces a strict constraint where  $H_f \geq 1$ . If market volatility causes  $H_f < 1$ , the protocol triggers a liquidation event. This deterministic mathematical model ensures that the protocol remains solvent and the stablecoin maintains its peg, fulfilling the core objective of the thesis.

## 0.6 Off-chain Infrastructure

The final component of the methodology is the Guardian Network, which serves as the interoperability layer connecting the deterministic worlds of the blockchain networks.

## 0.7 Off-chain Infrastructure and Guardian Network

### 0.7.1 Architectural Overview

The Off-chain Infrastructure, referred to as the Guardian Network, functions as the cryptographic bridge and synchronization engine between the deterministic environments of the EVM Spokes and the Solana Hub. This component is engineered using **Python**. Unlike passive indexers, the Guardian plays an active role in state transition, specifically in the secure initialization of user identities and the cross-chain relaying of assets.

### 0.7.2 Universal Wallet Initialization via Signature Verification

A critical security function of the Guardian is the verification of user intent during the creation of the Universal Wallet. Since the Universal Wallet acts as the central storage for a user's multi-chain positions, it is imperative to establish a cryptographically proven link between the user's request and their destination wallet on Solana before any on-chain state is modified.

The process is designed as a "Verify-then-Execute" workflow to prevent Denial of Service (DoS) attacks and ensure that only authenticated users can allocate storage on the Solana blockchain. The workflow proceeds as follows:

First, the user constructs a request message  $m$ , which includes the intent to create a wallet, a unique nonce to prevent replay attacks, and the public key of the destination Solana wallet ( $PK_{dest}$ ). To prove ownership of this destination wallet, the user signs the message  $m$  using their corresponding private key, generating a digital signature  $\sigma$ . This signature typically utilizes the **Ed25519** algorithm, which is the native standard for Solana addresses.

Second, the Guardian receives this payload  $(m, \sigma, PK_{dest})$  via a secure API endpoint. Instead of immediately submitting a transaction to the blockchain, the Guardian performs an off-chain verification using cryptographic libraries. The verification function can be formalized as:

$$V(m, \sigma, PK_{dest}) \rightarrow \{\text{True}, \text{False}\} \quad (6)$$

If the function returns **False**, the Guardian rejects the request immediately, ensuring that no gas fees are wasted on invalid transactions.

If the function returns **True**, the Guardian confirms that the requestor possesses the private key controlling the destination wallet. Subsequently, the Guardian constructs a transaction instruction invoking the `initialize_wallet` method on the Solana Main Contract. The Guardian then signs this transaction with its own key (to pay for transaction fees) and submits it to the Solana cluster. Upon successful execution, the Solana state is updated: a new Program Derived Address (PDA) is initialized, serving as the Universal Wallet for that specific user. This mechanism ensures that the heavy lifting of cryptographic validation is shared between the off-chain infrastructure and the on-chain logic, optimizing both security and cost-efficiency.

## **0.8 Conclusion**

This chapter has detailed the methodological approach for the Multi-chain CDP Protocol. By combining the safety of Rust/Anchor for state management, the standardization of Solidity for asset custody, and the mathematical rigor of the Secp256k1 verification for cross-chain identity, the proposed architecture provides a robust solution to the problem of liquidity fragmentation. The economic stability is underpinned by the Over-collateralized CDP model, while the Guardian infrastructure ensures seamless connectivity. These foundational technologies directly enable the implementation of the system design proposed in the subsequent chapters.