University College London

Department of Computer Science

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Financial Technology, University College London

Streamlining Derivative Trading: Enhanced Liquidity and Risk Mitigation with Blockchain-based Tokenised Collateral Management

Author
Vincenzo Incutti

Industrial Supervisors
Richard Barker (CEO),
Gerard Banaszkiewicz (COO),
Steve Haigh (CTO),
John Anderson (CPO),
Nazish Zaidi (Lead Business Analyst),
Stephen Ashworth (Head of Platform),
Brendan Bradely (Chairman),
Tokenovate

Academic Supervisor
Prof. Christopher D. Clack
Department of Computer Science
University College London

September 8, 2023

This dissertation is submitted as part requirement for the MSc Financial Technology degree at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

The aim of this study is to address the operational inefficiencies plaguing the collateral management systems in financial derivative trading, which have far-reaching implications for market liquidity and efficiency. To this end, we introduce a blockchain-based solution that automates settlement and reconciliation processes, traditionally managed by intermediary clearing houses. Our methodology involves the creation of a tokenised collateral system on the Bitcoin Satoshi Vision (BSV) blockchain, adhering to industry standards set by the International Swaps and Derivative Association (ISDA) and utilizing sCrypt, a Typescript-based Domain Specific Language to write smart contracts. A proof of concept tokenising crude oil as collateral is also presented. Our research reveals that while the BSV ecosystem is still in its infancy, greater collaboration between financial and tech sectors is essential for a seamless transition to a blockchain-based financial infrastructure. The study also sheds light on the technical hurdles in adapting existing ISDA frameworks to blockchain technology.

Acknowledgments

I wish to express my gratitude to my supervisor, Dr. Christopher D. Clack, for his invaluable assistance throughout the drafting of this dissertation. Additionally, I extend my appreciation to the entire team at Tokenovate Ltd, with a special mention of John, Nazish, and Stephen, for their invaluable business insights and guidance on the practical applications of this work. I would also like to acknowledge the postgraduate students who are working on their own dissertations under Dr. Clack's supervision, especially Finn, for being a source of inspiration and providing valuable feedback on my work. Finally, my heartfelt thanks go to my flatmate, Lukas, for the unwavering moral support during the challenging months of research and writing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Derivatives [100], as intricate financial instruments, rely on a myriad of interconnected systems—spanning finance, economics, law, and technology—for effective trading. The complex interplay between these systems often leads to conflicts that necessitate legal resolution or the cessation of trades to reconcile discrepancies in counterparties' information.

The 2008 financial crisis prompted international bodies to enforce stricter regulations to mitigate systemic risk [50]. However, these expanded regulatory and legislative measures were imposed on an antiquated technological infrastructure not originally designed to accommodate them.

A challenging aspect of derivative trading is the management and payment of collateral. Collateral, assets temporarily transferred between parties, acts as a safeguard against counterparty risk during the derivative lifecycle. The current collateral administration process is fraught with issues. Movement of collateral assets often incurs considerable costs and operational overheads, sometimes necessitating the physical transfer of assets to third-party accounts. According to a report by the European Commission, the European Securities and Markets Authority (ESMA) has identified more than 400 trading data contributors, each currently reporting data in different manners [28]. The reason behind this is that current reporting standards leave discretion in the interpretation of various reporting data fields, creating low quality market data reports and resulting in regulatory reporting arbitrage, sometimes even paving the ground for deliberate mis-reporting of trades (the Financial Conduct Authority (FCA) constantly updates their list of reporting transaction fines [52], from whose frequency and magnitude we can infer the gravity of the issue). These inefficiencies frequently compel third parties to hold positions overnight and bridge unexpected holdings.

The International Swaps and Derivatives Association (ISDA) [68] has been working to

standardize, digitize, and automate several crucial processes of derivative trading. A key focus has been the utilization of Smart Derivative Contracts to enhance infrastructure and mitigate challenges. By leveraging the transparency, accountability, and decentralization inherent in blockchain technology, Smart Derivative Contracts could automate many operational clauses of derivative agreements [24]. This automation could reduce manual input (and thus error), cut reconciliation costs, improve regulatory reporting, and streamline the collateral allocation process. The end results could be diminished balance sheet risk and enhanced capital allocation for liquidity creation.

This dissertation proposes a tokenized collateral system implemented via smart contracts on the Bitcoin Satoshi Vision (BSV) blockchain [13]. The aim is to refine the derivative trading lifecycle while adhering to existing regulatory, legislative, and business standards and processes. It evaluates the advantages of using the BSV blockchain to represent collateral, explores potential future directions for a settlement and payment layer trading such collateral, and analyzes the legal, financial, and economic implications of tokenized collateral. It also compares BSV with more widely-adopted distributed ledger systems like Ethereum [30] and Bitcoin [95].

The research maps the collateral representation from the ISDA Common Domain Model (CDM) [65] — the industry's most widely accepted standard—to one in sCrypt [111], a domain-specific language (DSL) based on TypeScript [91] for writing smart contracts on BSV. It examines the fractionalization of collateral value via satoshis and explores the use of off-chain oracles for providing real-world data streams to smart contracts.

1.1 Objectives and Contributions of the Thesis

The innovative contributions of this dissertation exist within the intersection of blockchain technology and the traditionally structured realm of collateral management for derivative trading. This work illuminates the potential for this technology to streamline, secure, and transform the intricate processes involved in managing collateral.

Firstly, a substantial contribution of this work involves developing a concrete methodology for tokenising underlying collateral assets. Tokenisation, as examined in this dissertation, is a procedure that embodies the translation of the economic and legal rights associated with a real-world asset into a digital token. This mechanism makes the asset easily transferable and divisible, enabling more efficient management and real-time valuations. In essence, the endeavour to precisely define the method and expected value outcomes of tokenisation provides a gateway to greater liquidity, improved accessibility, and the potential for enhanced market depth and breadth.

Secondly, this work outlines a robust representation for these tokenised assets in the digital

space, which is often referred to as a "digital twin". Specifically, this involves demonstrating in code how these digital representations can encapsulate their unique value per asset type. This novel contribution paves the way for an improved system of representing physical assets digitally. It enhances the overall understanding and representation of the asset and, in the process, facilitates more accurate and efficient asset management.

The creation of a Proof of Concept (PoC) forms the third innovative contribution. The PoC, demonstrated in code form, goes beyond the theoretical assertions to offer a hands-on, tangible representation of the proposed solution. It provides a compelling representation of how such a system might operate in practice, adding credibility to the proposed methods.

In addition, the benefits of the proposed solution are explored from a multi-dimensional perspective, encompassing economic, financial, legal, and technological facets. This exhaustive approach enables a comprehensive appreciation of the implications of the proposed solution, ensuring that its application is grounded in practicality and feasibility.

Specific novel techniques have been proposed in constructing the solution. The use of BSV as the blockchain of choice for implementing tokenised collateral management, for example, is a unique choice. The decision to use BSV has been taken considering its microtransaction capability, the scalability it offers, and its adherence to the original Bitcoin protocol, providing a stable platform for building the applications [27].

Moreover, the proposed architecture incorporates oracles (3rd party data providers to communicate off-chain information to the smart contracts) that are secure, transparent, and tamper-evident, ensuring that any financial calculations or regulatory compliance checks are carried out based on trustworthy data. By doing so, the architecture is not only enhancing the automation but also reducing the potential for human errors and biases in decision-making processes.

The derivative lifecycle is a complex system containing a high degree of interconnectivity between the legal, economic, financial and technological layers. We have decided to exclude the following aspects from the scope of this thesis in the interest of concision and clarity, however we do hope that the present work will constitute part of future comprehensive research aimed at capturing the full complexity of the system.

- We do not concern ourselves with determining which parts of the derivative lifecycle are worth automating. ISDA recommendations assume that certain aspects of the legal agreements between counterparties are easier to automate than others [23]. We restrict the mapping of collateral to its tokenised representation to those aspects that have been already codified in a formal representation in the ISDA CDM.

- We do not perform an exhaustive quantitative comparative analysis with existing collateral management and trade settlement systems, e.g. SWIFT [119]. The thesis only presents an overview of how the transactions and obligations could be performed by using the tokenised representations, paving the way for future quantitative research to gauge the effectiveness of the system.

1.2 Structure of the Thesis

The remainder of this thesis is structured as follows.

We provide the necessary background and technical preliminaries in Chapter 2. It is divided into several sections, starting with an overview of financial derivatives (Section 2.1). The life cycle of derivative trades (Section 2.1.1) and the specific challenges associated with collateral management are then elaborated (Sections 2.1.2 and 2.1.3). Further sections in Chapter 2 delve into existing industry and regulatory standards (Section 2.2), such as the ISDA Common Domain Model and ISDA Create. Towards the end of the chapter, we also introduce the technical aspects of blockchain technology (Section 2.3), focusing on smart contracts (Section 2.3.1), Bitcoin Satoshi Vision (BSV) (Section 2.3.2), and oracles (Section 2.3.3). The chapter concludes with a regulatory and legal analysis of the status of tokenised assets (Section 2.4) and a literature review of current academic and industry advancements in the space (Section 2.5).

Chapter 3 presents the tokenised collateral system built on BSV. The chapter opens with an explanation of the key components that make up the system and discussion of the high-level architecture (Section 3.1). This is followed by a comprehensive discussion on how ISDA's Common Domain Model has been translated and adapted to the constraints imposed by BSV and sCrypt, highlighting the interoperability and taxonomy challenges encountered (Section 3.2). Finally, the flow of operations in a standard collateral re-evaluation process is described through a sequence diagram (Section 3.3).

Chapter 4 brings forth the results and discussions based on the proposed architecture. This chapter is designed to provide a nuanced understanding of how the theoretical underpinnings translate to practical considerations. It discusses the role of custodians in this new architecture in Section 4.1, as well as critically evaluating the benefits of the proposed solution from financial, economic, legal, and technological perspectives (Section 4.2). Furthermore, it analyses the proposed tokenisation mechanism from a legal standpoint (Section 4.3).

Chapter 5 provides the conclusions and suggests avenues for future work. These suggestions encompass, among other things, enhancing the regularity of collateral re-evaluations, broadening the spectrum of supported asset categories, and integrating bespoke risk mod-

els into the computations. Ultimately, the importance of future collaboration between financial institutions such as ISDA and technology providers like sCrypt is highlighted as a means to advance the ecosystem in the future.

Chapter 2

Background and Preliminaries

This chapter lays the foundation by covering essential background information and technical details. The chapter is organized into multiple sections, beginning with an introduction to financial derivatives in Section 2.1. This is followed by an exploration of the life cycle of derivative trades in Section 2.1.1, and a discussion on the complexities of collateral management in Sections 2.1.2 and 2.1.3. Section 2.2 examines established industry and regulatory frameworks, including the ISDA Common Domain Model and ISDA Create. The latter part of the chapter shifts focus to the technological underpinnings, specifically blockchain technology in Section 2.3. Here, we discuss smart contracts in Section 2.3.1, BSV in Section 2.3.2, and oracles in Section 2.3.3. The chapter wraps up with an analysis of the regulatory and legal considerations for tokenized assets in Section 2.4, as well as a review of recent scholarly and industry developments in Section 2.5.

2.1 Financial Derivatives

Financial derivatives are complex financial instruments whose value is contingent upon or derived from the value of another asset, referred to as the "underlying" asset [100]. These derivatives act as contracts between two or more parties, and their price fluctuates based on changes in the underlying asset, which can be virtually any item of value. Common examples of underlying assets include stocks, bonds, commodities (like gold, oil, or agricultural products), currencies, interest rates, and market indices.

Derivatives can take numerous forms, however they all fundamentally function as a way of shifting risk or opportunity between parties. Some of the most common types of derivatives include futures contracts, forward contracts, options, and swaps.

- **Futures and Forwards**. These are agreements to buy or sell an asset at a specific future date at a pre-determined price. The key difference between the two is that

futures are standardized contracts traded on an exchange, while forwards are privately traded over-the-counter (OTC) and can be customized to fit the needs of the parties involved.

- **Options**. These give the holder the right, but not the obligation, to buy (call option) or sell (put option) an asset at a specified price within a certain timeframe.
- **Swaps**. These are agreements to exchange one stream of cash flows for another. For example, in an interest rate swap, one party might agree to pay a fixed interest rate in exchange for receiving a variable rate from another party.

Derivatives play a crucial role in financial markets for three primary reasons:

- **Risk Management and Hedging**. Companies and individuals alike use derivatives to reduce exposure to various risks. For instance, a manufacturing firm might use commodity futures to stabilize volatile input prices, or an international corporation might use currency swaps to mitigate the risk of exchange rate fluctuations. What sets derivatives apart in risk management is their ability to provide tailored solutions to hedge specific risks. Unlike traditional financial instruments, derivatives can be customized to match the exact duration, amount, and nature of the underlying exposure [60]. This precision allows entities to effectively neutralize their risk without over-hedging or under-hedging.
- **Speculation**. Traders and investors can use derivatives to profit from their predictions about changes in the price of the underlying asset. By correctly anticipating these price movements, they can potentially earn substantial returns. The leverage provided by derivatives is unparalleled. With a small initial margin or premium, investors can gain exposure to a much larger position in the underlying asset [11]. This means that even small movements in the price of the underlying can result in significant percentage returns (or losses) on the derivative position.
- **Arbitrage**. Derivatives can also be used to exploit price differences in different markets. Arbitrageurs aim to purchase an asset in one market and simultaneously sell it in another at a higher price, profiting from the price discrepancy. Derivatives offer a wider array of arbitrage opportunities due to their inherent complexity and the variety of contracts available. For instance, an arbitrageur can exploit mispricings between a stock and its futures contract, or between two different expiration dates of the same option. Additionally, derivatives often require less capital than purchasing the underlying asset directly, making arbitrage strategies more accessible [98].

2.1.1 Derivative Trade Lifecycle

The derivative trade lifecycle refers to the process that a derivatives contract goes through from its initiation to its final settlement (Figure 2.1). This lifecycle consists of several stages, each critical to the completion of the trade [72].

- **Trade Initiation**. The lifecycle of a derivative trade begins with the initiation of a contract. This involves the agreement on various aspects such as the type of derivative being traded (e.g., futures, options, swaps), the underlying asset on which the derivative is based, the size of the contract, the price, and the expiry date of the contract. This stage is crucial as it establishes the primary terms and conditions that will govern the trade.
  - **Payment of Upfront Premium**. In some derivatives like options, an upfront premium is often required. This is a payment made at the beginning of the contract to secure the rights provided by the derivative. The premium is usually non-refundable and paid by the option buyer to the option seller. The payment of this premium is crucial as it can affect the profitability and risk profile of the trade. Failure to pay the premium may result in the cancellation of the contract.
- **Trade Execution**. After the contract terms are decided upon, the trade is executed. The execution creates a legal obligation between the parties involved. This means they are now legally bound to uphold their end of the agreement. This can occur on an exchange, where standardized contracts are traded, or over-the-counter (OTC), where contracts can be customized to fit the needs of the parties involved.
- **Trade Confirmation**. Following execution, trade confirmation takes place. This involves the communication between the trading parties to verify the terms of the deal. The aim is to ensure that both parties agree on the details and understand their obligations. This process minimizes the risk of a trade failing due to miscommunication or misunderstanding.
- **Trade Clearing**. After trade confirmation, clearing occurs. A clearinghouse or a central counterparty (CCP) becomes involved at this stage. The CCP stands between the buyer and seller, essentially becoming the buyer to every seller and the seller to every buyer. This reduces counterparty risk — the risk that one party may default on its contractual obligations. The clearinghouse also handles the administration of the trade, from validation of the transaction to maintaining records.
- **Trade Settlement and Reporting**. Upon reaching its expiration date, the derivative contract is settled. The exact nature of the settlement depends on the type of derivative and the terms of the contract. It could involve a cash settlement or the

physical delivery of the underlying asset. The process concludes with the reporting of the transaction for record-keeping and regulatory purposes. Any profit or loss is realized at this stage.

  - **Multiple Payments Over Time**. For derivatives like interest rate swaps or certain types of structured products, there may be multiple payments that occur over a potentially very long time, sometimes spanning years. These payments are usually outlined in the initial contract and are subject to variables like interest rates or asset performance. The timing and amount of these payments are critical elements in the risk and return profile of the derivative.

- **Post-Trade Events**. After the trade is settled and reported, any post-trade events are handled. This might include the monitoring and management of any remaining risk, the calculation and payment of taxes, and any necessary compliance reporting.

Each stage of the trade lifecycle involves a variety of market participants, including the original trading parties, brokers, exchanges, clearinghouses, and regulatory bodies. Each of these actors plays a crucial role in ensuring that the trade is completed efficiently, transparently, and in compliance with all applicable rules and regulations.

2.1.2 Collateral

Collateral refers to the assets or pool of assets that are offered up by one party as a safety net, a guarantee against their contractual obligations [62]. Collateral can take many forms, ranging from cash and government securities to corporate bonds, equities, or any other asset deemed acceptable by the counterparties. The role of collateral is inherently protective, a safeguard designed to mitigate risk. Should a party default on their obligations, the collateral can be seized by the counterparty to counterbalance any financial loss resulting from the default.

The process of allocating collateral within derivative trades is multifaceted and dynamic. It's not simply a matter of assigning assets at the outset and leaving them be. Rather, it's a continuous recalibration, responding to shifts in the derivative position's market value, changes in the creditworthiness of the counterparties, and the constant ebb and flow of market volatility. This allocation process commences with the posting of an initial margin. This is essentially a percentage of the contract's total value, posted by both parties, acting as the first line of defense against potential market-induced losses.

In addition to the initial margin, counterparties are also required to post a variation margin [71]. Unlike the initial margin, the variation margin is not static. It's adjusted daily to accurately reflect the ever-changing market value of the derivative contract. Should market movements negatively impact a party's position, they may be called upon to post additional collateral, a process known as a margin call.

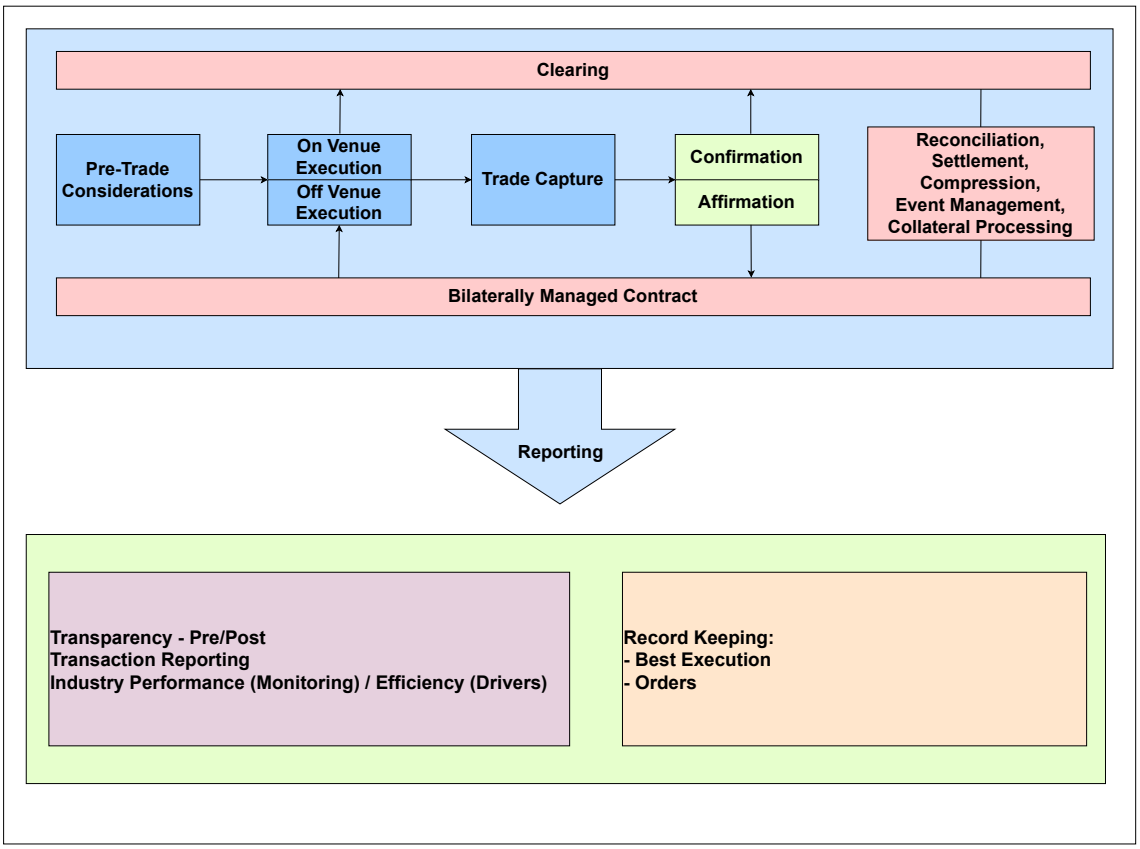


Figure 2.1: Adapted from [72]. The steps involved in the lifecycle of a derivative trade as per ISDA definition [72]. Starting with *pre-trade considerations*, parties assess market conditions, risks, and regulatory implications to align their trading strategies. The *trade execution* can either be on-venue, on regulated platforms, or off-venue, directly between parties. Once executed, the trade's details are meticulously captured, followed by a thorough *confirmation and affirmation* process to ensure accuracy and mutual agreement. This leads into a multifaceted phase of *portfolio reconciliation, settlement*, *compression* (reducing the number of derivative contracts in a portfolio without altering its net risk profile), *event management*, and *collateral processing*, ensuring both parties have matching records, finalizing the trade, managing lifecycle events, and mitigating credit risks. The *clearing* stage introduces a central counterparty, guaranteeing trade terms and reducing default risks. Comprehensive §*reporting* to authorities ensures transparency and compliance, while diligent *record-keeping* tracks trade history and adherence to best execution practices.

Moreover, it's worth noting that in certain circumstances, collateral can be subjected to reuse or rehypothecation. This essentially allows the receiving party to use the collateral for other purposes, such as pledging it for their own derivative trades. This practice, however, is subject to regulatory constraints and must be explicitly permitted in the collateral agreement.

In the world of derivative trades, the accurate valuation of collateral is paramount. From the moment it's posted, the value of collateral must be closely monitored and revalued regularly, often daily, to ensure it remains commensurate with the exposure. The quality

of collateral also holds significant weight, with high-quality, highly liquid assets being the preferred choice because they can be swiftly sold if a counterparty default occurs.

2.1.3 Collateral Management Challenges

The collateral management process is currently tainted by several challenges, spanning various stages of the lifecycle and posing significant impediments to the efficient operation of financial markets. ISDA provides a breakdown of the most significant elements of friction in the process in their *Blueprint for the Optimal Future State of Collateral Processing* [62] whitepaper (Figure 2.2). These can be broadly categorised in the following areas:

- **Asset Selection**. The process of asset selection, which determines what constitutes eligible collateral, is guided by a combination of regulatory stipulations and market conventions. However, the ultimate decision on what will be deemed eligible collateral is left to the discretion of the trading partners. This lack of standardization presents a significant challenge. The free-form nature of eligible collateral, typically defined in a Credit Support Annex (CSA), further compounds this issue. Moreover, difficulties arise in determining common definitions for certain asset types, such as high-quality liquid assets (HQLAs) [9];

- **Margin and Interest Calculation**. The current process of margin and interest calculation is largely manual, making it prone to processing errors. Counterparties independently calculate their interest based on the terms of their agreements, which include agreed rates and day count. This independent calculation often leads to potential discrepancies thereby leading to settlement risk. The manual matching of interest calculation payments and the manual reconciliation process further exacerbate these challenges;

- **Trade Transaction Management**. Mismatched and unmatched trades are primary drivers of disputes in the margin and collateral process. High volumes of new trades and amendments result in daily volatility in portfolios, making the management of these trades a complex task. Portfolio reconciliation is managed on the day following execution, which limits the ability to resolve trade-matching issues prior to issuing margin calls.

- **Record Keeping and Reconciliation**. Record keeping and reconciliation are integral to the collateral lifecycle, but they are also time-consuming and costly. This fragmentation and the resulting inaccuracies pose significant challenges to the efficient operation of financial markets. For instance, the existence of up to 18 trade repositories (entities that centrally collects and maintains the records of OTC derivatives) globally, which are mostly regional or national, limits the scope of trades captured and hampers market transparency [36]. These repositories are ill-equipped

to offer the comprehensive data needed for policymakers to monitor and mitigate systemic risk effectively. As a result, the OTC derivatives market is likely to become less transparent in the future, further complicating record keeping and reconciliation efforts.

- **Operational Challenges**. The operational challenges associated with the collateral lifecycle are also significant. Principals often face difficulties in releasing assets as collateral due to regulatory rules and operational challenges. When certain assets are used as collateral, it involves physical movements of those assets to 3rd party accounts. This physical movement of assets incurs significant costs and operational overheads.

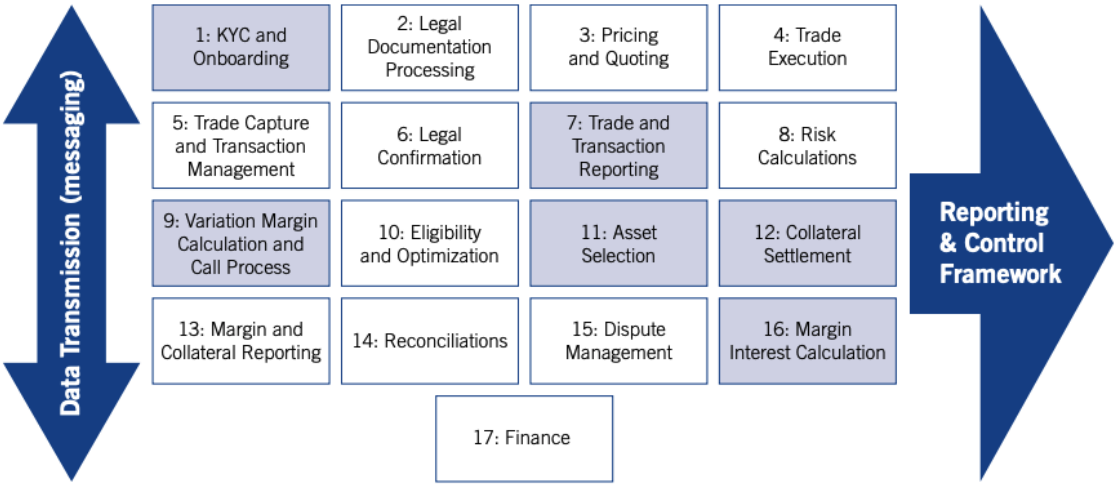


Figure 2.2: Reproduced from [62]. Stages of Collateral Management: Highlighted steps are discussed in the ISDA Blueprint [62]. *KYC & Onboarding* often grapple with intricate due diligence requirements and data inconsistency across global jurisdictions. *Trade and Transaction Reporting* faces the challenge of ensuring timely and accurate data submissions amidst diverse regulatory standards. *Variation Margin Calculation and Call Process* can be complex due to fluctuations in market prices and differing contractual terms. *Asset Selection* confronts the dilemma of balancing optimal returns with counterparty acceptability and liquidity constraints. *Collateral Settlement* experiences delays due to multi-party involvement and reconciliation discrepancies. Lastly, *Margin Interest Calculation* struggles with diverse rate agreements and the intricacies of time-bound computations.

2.2 Industry and Regulatory Standards

2.2.1 ISDA Common Domain Model

The ISDA Common Domain Model (CDM) [65] is a groundbreaking initiative aimed at bringing uniformity to the financial markets. It addresses the challenges posed by the increasing intricacy of the industry and the pressing need for streamlined operations. The model functions as a comprehensive framework that is both "machine-readable" and "exe-

cutable." In simpler terms, "machine-readable" means that the data and rules within the CDM can be easily understood by computers without human intervention. "Executable" means that the model can automatically carry out transactions and processes, further reducing manual effort. Being an open-source project, the CDM is freely accessible and can be modified to suit the specific needs of different organizations in the financial sector. This adaptability is particularly beneficial for industry participants looking for tailored solutions.

One of the primary objectives of the CDM is to enhance operational efficiency in financial markets. It does this by setting a digital standard for various trading activities and events. This uniform standard makes it easier for different companies and platforms to interact with each other, thereby minimizing the discrepancies that often require time-consuming reconciliation.

Transparency is another cornerstone of the CDM. It ensures that regulatory bodies and market participants are aligned, which is crucial for consistent reporting to authorities. The model is built on foundational design principles such as "abstraction-based normalization," which means simplifying complex elements to their most basic forms; "composability," or the ability to combine different components seamlessly; and "modularization," which allows for easy updates and modifications. These principles contribute to the robustness and flexibility of the CDM.

Governed by a structured set of guidelines, the CDM is applicable to a broad spectrum of financial products, including but not limited to Over-The-Counter (OTC) derivatives and cash securities. The model comprises various elements, including the publicly accessible ISDA CDM Distribution [54], the specialized Rosetta Domain-Specific Language (DSL) [102], and a supportive ecosystem of applications built on the CDM framework [103].

2.2.2 ISDA Create

ISDA Create [75] is a digital platform developed to automate the negotiation and execution of documentation used in the derivatives and securities financing markets. By transitioning from manual, paper-based processes to a more streamlined digital approach, ISDA Create aims to enhance efficiency and reduce the time taken to finalize agreements. The platform provides users with the ability to negotiate and execute multiple documents simultaneously, offering a centralized location for storage and access. This centralized approach simplifies the tracking of negotiations, document version control, and the overall management of legal agreements. Additionally, ISDA Create supports the integration of standardized data structures, facilitating easier data extraction and analysis.

2.2.3 ISDA Clause Library

The ISDA Clause Library [74] is an initiative aimed at providing clarity and consistency in the documentation of derivatives trade agreements. Recognizing the complexities and nuances inherent in these contracts, the library offers a structured approach to categorizing and defining standard clauses used in ISDA documentation. By doing so, it facilitates a more streamlined negotiation process, reducing ambiguities and potential misunderstandings between parties. The library serves as a reference tool, enabling legal professionals and contract negotiators to quickly identify and understand the implications of specific clauses. This systematic approach not only simplifies the drafting process but also aids in ensuring compliance with regulatory requirements.

2.2.4 Industry Participants

The drive from ISDA towards increasing standardization and digitization of the derivative trading lifecycle has been echoed by both established financial institutions and emerging startups, leveraging cutting-edge technologies like AI and blockchain. J.P. Morgan, a banking behemoth, has been actively exploring the potential of blockchain in streamlining derivative transactions. Their Liink product (formerly Interbank Information Network or IIN) aims to reduce friction in the information exchange process, ensuring smoother and more efficient derivative trades [86]. Barclays, another banking giant, has also shown interest in blockchain's potential to revolutionize the derivative trading landscape. They believe that the technology can address the challenges of transparency and efficiency that have long plagued the sector [55]. On the startup front, Digital Asset is making waves with its smart contract language, DAML, which seeks to automate and digitize complex derivative contracts, making them more accessible and transparent [40]. Clause [82] is another innovative startup that integrates AI and blockchain to automate real-time derivative contract performance. Their platform can interpret and execute clauses in derivative contracts, ensuring that all parties adhere to the agreed terms. Lastly, Komgo [79] is a blockchain-based platform that focuses on commodity trade finance, a subset of derivative trading. By offering a decentralized solution, Komgo aims to reduce fraud and operational errors, ensuring a more secure and efficient trading environment.

2.3 Blockchain

Blockchain is a decentralized and distributed ledger technology that allows multiple participants to maintain a shared and tamper-evident record of transactions or information in a secure and transparent manner [92]. It was originally introduced as the underlying technology for cryptocurrencies like Bitcoin, but its potential applications have expanded beyond digital currencies.

At its core, a blockchain is a chronological chain of blocks, where each block contains a batch of validated transactions or data. Each block is linked to the previous one through a cryptographic hash, creating a chain of information that is extremely difficult to alter. Throughout the rest of this work we use the term "immutable" to describe this chain, however it's important to note that blockchain blocks can technically be changed. However, any alteration to a previous block would become immediately evident when the cryptographic hashes are checked, requiring the modification of all subsequent blocks. This makes the blockchain highly resistant to tampering and ensures the integrity of the data.

Blockchain is particularly suited to address the challenges encountered in the collateral management lifecycle described in section . In particular,

- **Asset Selection**. By incorporating blockchain technology, the process of asset selection can be significantly standardised. This is achievable by establishing a decentralised protocol for defining eligible collateral, thereby eliminating the discretion and free-form nature of the selection process. This protocol could also include a consensus mechanism to agree on definitions for asset types such as HQLAs, reducing misunderstandings and potential disputes.
- **Margin and Interest Calculation**. Blockchain's smart contracts, which are programmable contracts that automatically execute when certain conditions are met, could automate the manual, error-prone process of margin and interest calculation. By setting the terms of agreements, including rates and day count, as the conditions in the smart contract, the calculations would be automatically performed and agreed upon by all parties involved, thereby eliminating discrepancies and reducing settlement risks.
- **Trade Transaction Management**. Blockchain's immutability and real-time transaction records can significantly enhance trade transaction management. By storing every new trade and amendment on the blockchain, all parties can monitor the portfolio volatility and address trade-matching issues immediately, rather than waiting for the following day. This would minimise mismatches and unmatched trades, reducing the disputes in the margin and collateral process.
- **Record Keeping and Reconciliation**. With blockchain technology, record keeping and reconciliation could be streamlined and automated. Each transaction and adjustment made during the collateral lifecycle would be recorded in real-time on the blockchain. This permanent, transparent ledger would provide a single source of truth for all participants, thus eliminating data fragmentation, reducing the inaccuracies in trade reporting, and cutting down the time and cost of reconciliation.
- **Operational Challenges**. The use of blockchain could also mitigate the opera-

tional challenges involved in the collateral lifecycle. The movement of assets used as collateral could be represented as token transfers within the blockchain, bypassing the need for physical transfer of assets. This could expedite the process, reduce costs, and make it easier to comply with regulatory rules. By tokenizing assets, principals could also easily release assets as collateral, enhancing operational efficiency. The benefits of efficient release and re-allocation of collateral are described in detail in Section 4.2.1.

2.3.1 Smart Contracts

The term "smart contract" was first coined in the 1990s by computer scientist and cryptographer Nick Szabo [121], long before the rise of blockchain technology, as we know it today. Szabo envisaged smart contracts as computerized transaction protocols that execute the terms of a contract, aiming to deliver highly efficient, automated solutions to contract law.

Today, in the backdrop of blockchain technology, the term "smart contracts" has become used to refer to programmable, self-executing, and self-enforcing protocols that run on the blockchain. They are essentially predefined rules and conditions encapsulated in the form of a digital contract. Once these predefined conditions are met, the smart contract automatically triggers a transaction or a specific action, making the whole process tamper-evident and leaving no room for default.

2.3.1.1 Smart Legal Contracts

Taking the concept of smart contracts a step further are "smart legal contracts" – these are conventional legal contracts interwoven with smart contract functionalities [106]. A smart legal contract is essentially a legal agreement that has some or all terms and conditions represented and executed by software. These contracts are not only legally binding agreements but also contain provisions that are capable of being automated and carried out by a machine, providing the best of both worlds – the solidity of legal contracts and the automation of smart contracts.

2.3.1.2 Smart Derivative Contracts

One specific application of smart contracts in the finance industry is the emergence of "smart derivative contracts" [23]. Smart derivative contracts have the potential to revolutionize the derivatives market by embedding the terms of a derivative contract on a blockchain. These self-executing contracts can automate the life cycle of a derivative trade, from creating the contract, performing valuations, managing margins, and all the way to the final settlement.

When applied to collateral management in derivatives trading, smart derivative contracts can automate collateral posting and maintenance, which are often complex and resource-intensive processes. These contracts can be programmed to monitor the value of the collateral and automatically issue margin calls when necessary. The blockchain's immutability ensures that the smart derivative contracts are tamper-evident and their execution can be audited, thereby improving the security and transparency of the process. This use of blockchain and smart derivative contracts promises to transform collateral management, significantly reducing the risks, costs, and inefficiencies associated with the process.

2.3.2 Bitcoin Satoshi Vision

Bitcoin Satoshi Vision (BSV) is a cryptocurrency and blockchain network that emerged from a hard fork split from Bitcoin Cash (BCH) in November 2018 [13]. It was created in response to disagreements within the Bitcoin Cash community over proposed changes to the protocol. BSV proponents, led by Craig Wright and Calvin Ayre, sought to adhere more closely to what they consider to be the original vision of Bitcoin's anonymous creator, Satoshi Nakamoto. This vision encompasses principles of larger scalability, minimal transaction fees, and robust data handling, making BSV uniquely suited to address the challenges in the collateral management lifecycle. In particular, the benefits offered by BSV in the collateral management process become even more evident when compared to alternative distributed ledger platforms such as Ethereum Bitcoin:

- **Scalability**. BSV's main advantage is its scalability, which is primarily achieved by significantly increasing the block size limit as compared to Bitcoin and Ethereum. The original Bitcoin and Ethereum networks have opted for smaller block sizes (1 MB, fixed-size blocks for Bitcoin [12] and variable-sized blocks for Ethereum — up to a ceiling of 30 million "gas" units [42], where "gas" is the unit of measurement for computational effort that is required to perform various operations, such as executing smart contracts, making transactions, or storing data [43]) to maintain decentralization and security. Smaller block sizes are easier for individual nodes to process, thereby encouraging more participants and maintaining the decentralized nature of the blockchain. However, the trade-off is a limited throughput of transactions, which can slow down the network and increase transaction fees during periods of high demand [116]. BSV, on the other hand, has decided to prioritize scalability by allowing much larger block sizes (up to 4GB at the time of writing [15]), aiming to accommodate more transactions per block. This enables quicker processing of transactions, creating a responsive system that is ideal for managing daily portfolio volatility and facilitating trade transactions. Fixed block sizes are generally used as a compromise between scalability and network health; variable block sizes could lead to inconsistencies in block propagation and validation times, potentially

compromising the security and integrity of the network [116].

- **Data Handling**. BSV provides robust data handling capabilities, allowing for comprehensive transaction data to be stored directly on the chain. While Ethereum also supports data storage, its higher transaction fees for complex data operations may limit efficiency. BSV's capabilities ensure an efficient, transparent, and immutable record of collateral lifecycle events, which dramatically improves record keeping, reconciliation, and reporting accuracy.

- **Smart Contracts**. BSV's smart contracts offer an economical approach to automating processes in the collateral management lifecycle. Unlike Ethereum or Bitcoin's smart contracts, which can incur high gas fees, BSV has significantly lower transaction costs, though it is not entirely without fees — at the time of writing (5th September 2023), the average transaction cost is $0.81 on Ethereum [134], $1.40 on Bitcoin [133] and $0.00000367 on BSV [128]. This makes it more cost-effective for executing complex automation tasks, such as margin and interest calculations and asset selection. Additionally, BSV, similarly to Bitcoin, incorporates specific safeguards such as opcode limits and script size restrictions to minimize the risk of "runaway processes" like infinite loops [14]. This contrasts with Ethereum, where the presence of high gas fees often acts as a de facto limitation against such inefficiencies but doesn't provide built-in safeguards in the protocol [120]. BSV's approach ensures a more streamlined and reliable experience for executing smart contracts, without relying solely on transaction costs as a deterrent against poorly optimized code.

- **Micropayments**. BSV's low transaction fees make it highly effective for micropayments, allowing for flexible interest payment schedules and streamlined settlements. This directly contrasts with Ethereum, where high gas fees make small transactions economically inefficient. BSV's affordability in this context significantly minimizes the need for manual reconciliation processes, such as matching transactions, verifying records, and resolving discrepancies, thereby reducing both time and labor costs.

- **Tokenization**. Both BSV and Ethereum platforms offer the capability to tokenize assets, thereby streamlining the collateral management lifecycle. Tokenization on these platforms eliminates the need for physical asset transfer, enhancing operational efficiency. However, the two differ in terms of transaction costs. BSV offers a more cost-effective solution for asset tokenization and transfer, while Ethereum, although equally functional in allowing tokenization, comes with higher transaction fees [38]. This makes BSV a more economical choice for businesses looking to manage operational challenges in collateral management without sacrificing functionality.

2.3.2.1 UTXOs

In the BSV blockchain, the Unspent Transaction Output (UTXO) model serves as the cornerstone for enabling transactions. Unlike traditional account-based models, where the state of an account is updated continuously, UTXOs are discrete data units that represent a specific amount of cryptocurrency [6]. Each UTXO consists of its value in satoshis (the smallest unit of a Bitcoin) and a ScriptPubKey — a cryptographic script that outlines the conditions for spending that particular UTXO. UTXOs are created as outputs from transactions and can be thought of as virtual "coins" that can be consumed or spent in subsequent transactions. When a UTXO is spent, it becomes an input for a new transaction, and the process creates new UTXOs as outputs, which in turn can be spent in the future.

The ScriptPubKey associated with each UTXO specifies the "locking script" that must be "unlocked" by a corresponding "unlocking script" or ScriptSig for the UTXO to be spent. Different types of ScriptPubKeys exist to facilitate various transaction types. Pay-to-Public-Key-Hash (P2PKH) [104] is one of the most common types, designed for simple, one-to-one transactions. In a P2PKH output, the ScriptPubKey locks the UTXO with the hash of the recipient's public key. To spend it, the recipient must provide both the public key and a digital signature that proves ownership of the corresponding private key.

Pay-to-Script-Hash (P2SH) [105] offers more complexity and flexibility, enabling outputs to be locked with a hash of a script rather than a public key. P2SH allows for more complicated locking conditions, like multi-signature requirements or time-locked releases. In a P2SH transaction, the person who sets the conditions is not necessarily the one who fulfills them. This type enables functionalities like multi-signature wallets, where multiple parties must sign off on a transaction, or smart contracts that execute automatically under predetermined conditions.

Together, these UTXO features and ScriptPubKey types make up the rich ecosystem of transaction possibilities in the BSV blockchain, offering both robust security measures and versatile programmability.

2.3.2.2 sCrypt

sCrypt [111] is a high-level scripting language purpose-built for writing smart contracts on the BSV blockchain. It was conceived as a more developer-friendly alternative to Bitcoin's low-level scripting language, Bitcoin Script. While Bitcoin Script has proven to be highly secure and robust, it is also complex and cumbersome for developers, especially those new to blockchain development. To address this, sCrypt was developed to provide a statically-typed, C-style syntax that is both expressive and familiar to developers, thereby improving the readability, development efficiency, and bug detection in BSV smart contracts.

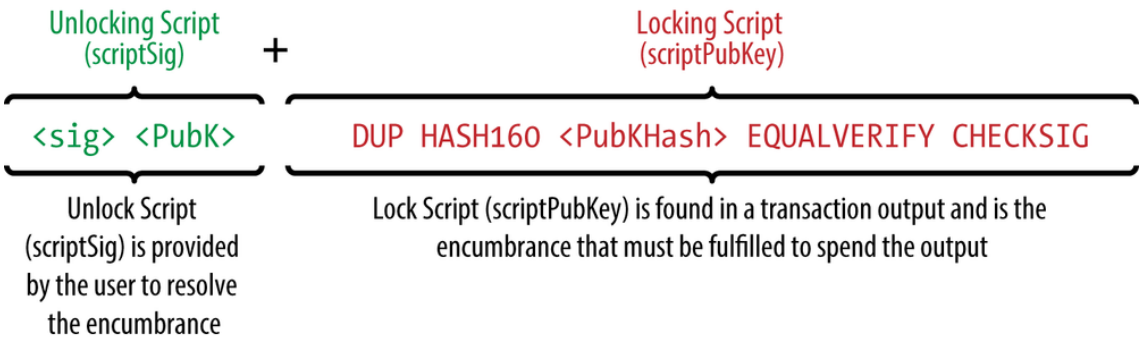


Figure 2.3: Reproduced from [6]. Unspent Transaction Output, with a focus on the ScriptSig and ScriptPubKey components. ScriptSig serves as the 'unlocking script,' providing the necessary credentials to spend a UTXO. The ScriptPubKey acts as the 'locking script,' setting the conditions under which the UTXO can be spent. Together, these cryptographic scripts form the basis for secure and programmable transactions on the BSV blockchain.

In our project to tokenize collateral assets on BSV, we are using sCrypt for several key reasons:

- **Efficiency and Readability**. The syntax of sCrypt is similar to popular programming languages such as TypeScript [91], making it easier to write, read, and maintain. This improves efficiency and reduces the potential for errors in the development process.
- **Security**. sCrypt is engineered to tap into the native security protocols of Bitcoin Script by directly cross-compiling from its high-level language to Bitcoin Script. Through the process of cross-compilation via Typescript, sCrypt allows for the development of smart contracts that are not only secure but also reliable and robust. This direct translation ensures that the smart contracts are tightly integrated with Bitcoin's established security features, guaranteeing the integrity of our tokenized collateral assets without introducing extraneous vulnerabilities.
- **Static Typing**. sCrypt is a statically typed language, which enables the detection of many errors at compile-time rather than at run-time. This is particularly important in blockchain development where contracts are immutable once deployed.
- **Built for BSV**. sCrypt is specifically designed for the BSV blockchain. This ensures smart contracts can fully leverage the unique features of BSV, such as its scalability and microtransaction capabilities, which are integral to managing tokenized collateral assets at scale.

2.3.3 Oracles

Oracles, within the context of blockchain and smart contract technologies, serve as intermediaries that provide the bridge between blockchain systems and the external world

[3]. They relay real-world data to smart contracts on the blockchain, enabling these decentralized applications to interact and integrate with off-chain data and systems. Given that blockchains are deterministic systems, they can't directly interact with the external world because they operate based on pre-defined rules and consensus mechanisms that ensure security and immutability within the network. Since blockchains are designed to be isolated systems to maintain their integrity and trustworthiness, any interaction with external data sources poses the risk of introducing vulnerabilities or inaccuracies. Thus, they need a trusted channel to provide them the necessary data, and that's where oracles come in.

Oracles can be of various types, such as input oracles, output oracles, cross-chain oracles, and compute-enabled oracles [19]. Input oracles fetch real-world data and deliver it to the blockchain for smart contracts to use. Output oracles, on the other hand, enable smart contracts to dispatch instructions to external systems, prompting them to carry out specific actions. Cross-chain oracles facilitate data and asset movement between different blockchains, thus enabling interoperability. Compute-enabled oracles provide off-chain computation capabilities that are impractical to perform on-chain due to various constraints.

Oracles can significantly enhance the efficiency and robustness of collateral management systems in several ways:

- **Real-time Price Feeds** [18]. The foundation of collateral management is the accurate valuation of collateral assets. Oracles, especially input oracles, deliver real-time market data relating to the prices of assets that serve as collateral in smart contracts. As an example, in scenarios where a specific cryptocurrency or token is put forth as collateral, an oracle can supply its latest market price, ensuring that the valuation of the collateral is as precise as possible. In the context of decentralized finance (DeFi), such data streams can stimulate automated reactions when prices hit predetermined thresholds - for instance, it may launch a margin call when the collateral's value dips below a certain level, thus ensuring prompt action and prevention of potential risks.
- **Risk Management** [21]. Proficient collateral management hinges on robust risk management protocols, and oracles play an instrumental role in this area. They have the capacity to supply data concerning the volatility and liquidity of assets under consideration as collateral. This information aids in assessing associated risks - for example, an asset with high volatility brings with it a higher risk of a sharp decline in its value, which could result in the collateral's value being insufficient. Oracles deliver real-time data feeds that can support these risk assessments and guide better informed, proactive decision-making.
- **Automation of Collateral Calls**. Oracles and smart contracts together contribute

to automating collateral calls, each serving distinct roles within the system. In this setup, oracles function as real-time data feeds that continuously monitor market conditions and the valuation of the counterparty's collateral. When these oracles detect that the market circumstances have changed unfavorably, thereby devaluing the collateral to an insufficient level, they pass this critical information to the smart contract. On the other hand, the smart contract holds the pre-defined logic for executing a collateral call based on the information received from the oracle. It determines if the collateral devaluation warrants an automatic call, calculates the additional amount of collateral required, and then triggers the collateral call to the counterparty. The smart contract also facilitates the secure, trustless transaction of posting additional collateral, making the entire process more efficient and transparent. This is the use case the we propose as part of system described in Chapter 3.

- **Regulatory and Compliance Checks**. Oracles can play a critical role in not just retrieving data related to regulatory obligations and compliance checks, but also in ensuring that smart contracts remain up-to-date with real-time changes in regulations and industry-standard practices. This is especially significant given that regulations and compliance standards are often subject to change, and smart contracts, once deployed, are immutable by nature [135]. To address this, smart contracts must be designed with "upgradability" in mind, allowing for the modification of their logic without requiring the redeployment of the entire contract. Various techniques exist to achieve upgradability, such as using external libraries, data separation, or the proxy pattern [90]. In particular, the proxy pattern [20] is a technique wherein the main smart contract delegates its logic to another contract (often termed the "implementation contract"). The proxy contains the state variables and remains consistent, while the logic and data manipulation are conducted in the implementation contract. When regulations change, a new implementation contract can be deployed and linked to the existing proxy, ensuring that compliance-related data is always current and that collateral evaluation criteria align with the latest regulatory requirements. At the time of writing and to the best of our knowledge, no specific oracles solely dedicated to providing information about standards and regulations exist. However we envision, and in fact suggest, ISDA launching its own oracle service to provide requirements about, for example, the eligibility of different types of collateral [67].

- **Cross-chain Interoperability** [17]. In a context where multiple blockchains are in operation, the same asset could exist on several chains. Cross-chain oracles in such scenarios can be invaluable, enabling smart contracts to identify and engage with these assets across different chains. This significantly improves the flexibility

and scope of the collateral management process.

- **Settlement and Reconciliation**. Oracles can also expedite the settlement and reconciliation process inherent to collateral management. Output oracles, for instance, can engage with traditional banking systems to initiate payments based on the stipulations of a smart contract, thereby seamlessly aligning traditional and decentralized finance systems [26].

2.4 Asset Tokenisation

Asset tokenization is the process of converting the rights to a physical or intangible asset into a digital token on a blockchain. These tokens can represent a wide variety of financial instruments, including equities, bonds, real estate, commodities, and even artworks or intellectual property. There are several types of tokens, each with their unique properties and applications [63]. Table 2.1 provides a breakdown of the distinguishing features of each model.

- **Registered Tokens** [29]. These tokens have an identity layer. Ownership of these tokens is tied to a specific identity, which can be verified against a registry. This is particularly useful for assets where legal and regulatory compliance is crucial (Figure 2.4a). The first ever digital bond, denominated in GBP, issued by the European Investment Bank lies in this category [47].

  In the Registered model, ownership of tokenised assets is tied to a specific identity stored in a register controlled by a *Registrar* entity. The Registrar and the *Issuer* of the tokens could coincide. When they do not, the Issuer submits instructions to the Registrar when issuance operations have to be performed, as well as transferring the ownership rights of the tokenised securities to the Participants in the network. Transfer operations are performed by the Participants by issuing transfer instructions to the Registrar. It is important to highlight the fact that in this scenario the power to update and rectify entries in the registers lies exclusively with the *Registrar*, the Issuers and Participants can simply communicate the actions they want to perform. Another important aspect to consider is that in this scenario the tokens merely represent evidence of rights, they are not assets in their own right, and modifying the state of the register simply involves updating the token balances of specific participants. Note that two registers are present: the first, on-chain register stores the mapping between public keys and tokens, i.e. the holding relationships; the second, off-chain register maps public keys to real-world identities. The Registrar also keeps an offline business continuity record, essentially replicating the on-chain records, to ensure operations can proceed in case of network disruptions.

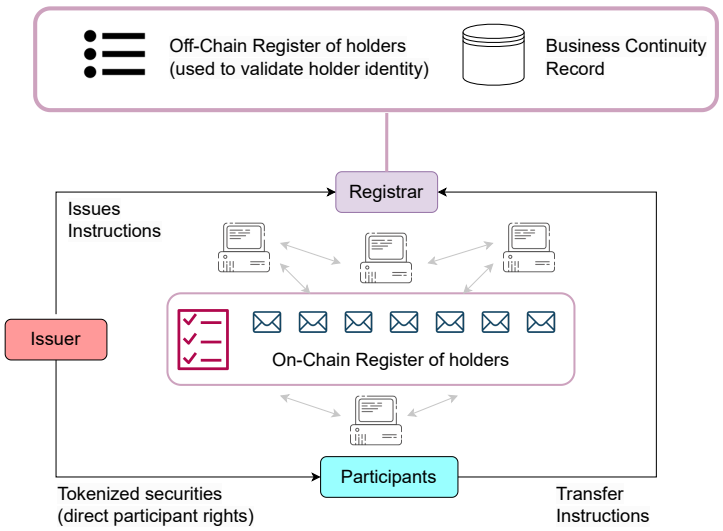
- **Bearer Tokens** [4]. These are tokens that grant ownership to whoever holds them, much like cash or certain types of bonds. They don't have an embedded identity layer that verifies the owner's identity. Whoever controls the private key controls the asset (Figure 2.4b). Stablecoins (dollar-pedgged digital coins) such as USDT [124] and USDC [22] fall into this category.

  In the Bearer model, the tokens are intangible assets in their own right and ownership of tokenised assets relies exclusively on the control of the tokens themselves. In this scenario, no Registrar is needed; the Issuer simply creates the tokens and transfers ownership to the Participants, which are then fully responsible for administering the assets. It is important to highlight that in this scenario the token holder has exclusive control, meaning the token holder has additional operational burdens as well as increased exposure to risk. Moving funds requires updating ownership of the asset themselves (meaning that once the asset is transferred, associated rights and control over it now belong to the new holder, increasing risks related to irreversibility of transactions in case of errors).

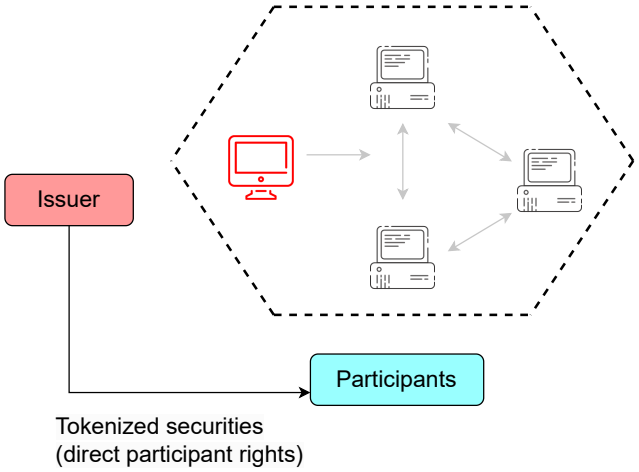
- **Claim Tokens** [25]. These tokens represent a claim or a right against an asset, rather than the asset itself. They're often used in debt issuance or other types of contractual agreements where one party has a claim against another (Figure 2.4c) The Depository Trust & Clearing Corporation's (DTCC) Ion Project, a blockchain-based alternative settlement engine, falls into this category [33].

  In the Claims model, a third-party Operator is responsible for running and maintaining the blockchain system, while also providing the rules for operating within the system. The Claims model resembles the Registered model, however we note a few key differences. Firstly, there is no additional identity layer, ownership is fully determined within the boundaries of the blockchain system by means of cryptography (mapping between public keys and token representations). Secondly, the Issuer and Operator cannot coincide: the Issuer is exclusively responsible for granting ownership rights to the Participant while the Operator's only responsibility is to run the network. Lastly, Participants are directly interacting with each other: ownership changes are still reflected by updates to a ledger of balances, however these are performed directly by Participants by interacting with smart contracts deployed by the Operator (and whose specifications are defined in the system rules). Overall, the Claims model is more flexible than the Registered model in the array of underlying assets it can represent, however this also introduces an extra layer of complexity as the claims must be exercised through appropriate legal channels and contractual arrangements.

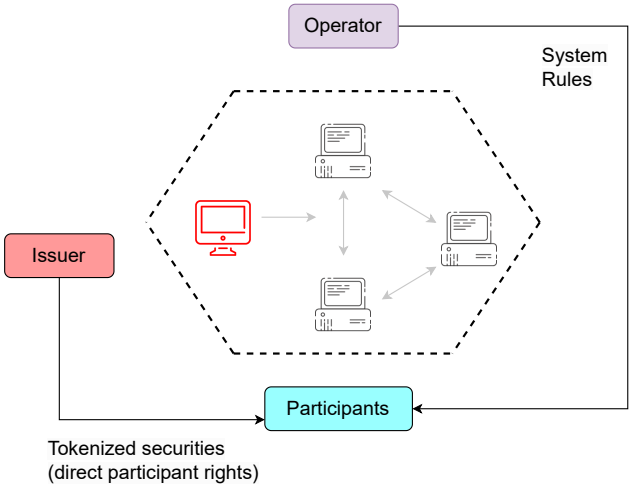
[1]This image is reproduced with permission from Tokenovate Ltd.



(a) Registered



(b) Bearer



(c) Claims

Figure 2.4: Adapted from [126] [1]. Three asset tokenisation models showing roles of *Issuers*, *Registrars*, and *Participants* in the network: Registered (2.4a), Bearer (2.4b), and Claims (2.4c) models.

Tokenisation Model	*Description*	*Control*	*Traits*	*Transfer Mechanism*	*Examples*
Registered	Rights determined by reference to a register controlled by Registrar	Registrar has powers to update/rectify entries	Mere data/evidence of rights	Updating token balances	First EIB Digital Bond in Sterling [47]
Bearer	Rights determined by reference to exclusive control of tokens	Token holder has exclusive control	Intangible asset in its own right	Transfer of control of token	USDT [124], USDC [22]
Claims	Rights determined by reference to entries in a system controlled by a third-party Operator	Third-party operator has powers to update/rectify entries	Mere data/evidence of rights	Updating token balances	DTCC Project Ion [33]

Table 2.1: Properties of different asset tokenisation models.

Tokenisation has significant effects on the collateral management process:

- **Enhanced Liquidity**. Tokenization essentially acts as a liquidity enhancer, particularly for assets traditionally characterized by illiquidity. Assets such as real estate, private equity, or even fine art, can be divided into fractional ownership represented by tokens. This process makes it feasible to sell or trade fractions of these assets, thereby unlocking their value and transforming them into more liquid instruments. This increase in liquidity consequently extends their functionality as collateral. The tokens can be more readily bought, sold, or valued, thereby rendering them more practical as collateral in transactions. Furthermore, their divisible nature implies that exact collateral amounts can be more effortlessly arranged, leading to improved capital utilization.

- **Operational Efficiency**. The potential efficiency gains through tokenization in collateral management are substantial. By transforming assets into digital tokens on a blockchain, the process of transferring these assets, whether for a collateral call, substitution, or release, becomes notably more efficient. The removal of intermediaries and the streamlined, automated procedures made possible by smart contracts can significantly reduce the time and cost of these transactions. This automation reduces manual errors and the overall complexity of collateral management, making it more straightforward and less resource-intensive.

- **Transparency and Auditability**. The adoption of blockchain technology in asset tokenization introduces an unprecedented level of transparency and auditability within the scope of collateral management [70]. Each token transaction—be it related to a collateral call, release, or substitution—is indelibly recorded on the blockchain in an immutable ledger. Authorized parties can scrutinize this ledger to gain a transparent, auditable record of all transactions. This enhances the efficiency of risk management in collateral operations, as involved parties can easily access and evaluate the necessary data. While the immutable nature of blockchain transactions does aid in regulatory compliance, it is important to note that the technology is not a panacea for all compliance challenges. Legal texts often contain ambiguities inherent to natural language, making some rules and regulations open to interpretation. Therefore, while blockchain records can serve as strong corroborating evidence, they may not guarantee full compliance with all relevant legal requirements. Nevertheless, for rules and regulations deemed fully automatable by regulatory bodies, blockchain adds an invaluable layer of trust and security to the compliance process.

- **Interoperability and Flexibility**. Tokenization also brings about a new degree of interoperability and flexibility to collateral management. In a tokenized world, the same asset can exist on multiple chains, enhancing the flexibility of collateral

operations. Cross-chain interoperability made possible by blockchain technology implies that collateral can be easily transferred across various platforms, catering to the diverse needs of participants in the collateral management lifecycle. This feature, however, also introduces the risk of "double use" or "double spending," where the same asset could potentially be used as collateral on multiple platforms simultaneously. To mitigate this risk, the use of decentralized identity and ownership verification systems [39], smart contracts with built-in "lock-up" features [123], or cross-chain oracle services [17] could be implemented to track and authenticate the status of tokenized assets, ensuring they are not misused.

- **Democratization of Access**. Finally, tokenization potentially democratizes access to certain types of collateral. By breaking down larger, illiquid assets into smaller, more accessible tokens, a wider range of participants can be recipients of these assets for collateral use. This diversification enlarges the pool of available collateral, thereby enhancing risk mitigation by reducing overreliance on a limited set of assets.

2.5 Relevant Literature Review

This literature review aims to provide a comprehensive assessment of the existing academic and professional literature on collateral management within the traditional derivatives market, the challenges and inefficiencies therein, and the advent of tokenized collateral in derivative trading. The chapter also explores the practicalities and legal considerations surrounding the deployment of tokenized collateral in financial markets.

Notably, [5] argues that although there's abundant collateral supply, it is the infrastructural weaknesses that contribute to the immobilization of collateral, which can affect demand in another segment of the financial system. Complementing this perspective, [77] identifies the post-Global Financial Crisis inefficiencies in trade reporting, especially in over-the-counter (OTC) derivatives, positing that blockchain technology could serve as a viable solution to these inefficiencies. The potential for streamlining comes with a hefty price tag; according to [16], the financial industry would need to invest over $53 billion in infrastructure and technology investments to upgrade and source new capabilities to achieve collateral efficiency and operational efficiency. Further complicating matters, the report also reveals that operational preparedness for derivatives clearing and collateralization remains a work in progress for nearly half of the firms surveyed. The increasing financial burden is also evident in the escalating amounts of Initial Margin (IM) and Variation Margin (VM) that need to be collected, which according to a 2022 ISDA report [69], reached $1.4 trillion by the end of that year.

ISDA's report on the cross-border fragmentation of global OTC derivatives [66] amplifies the complexity by pointing to a 77% decline in volumes of cleared euro interest rate

swaps between European and US dealers, highlighting the intricate challenges that arise as markets become more fragmented. These challenges are compounded by outdated methodologies; as Greenwich Associates suggests, the majority of firms still rely on manual methods for trade confirmation and reconciliation, a practice that not only hampers speed but also consumes 60% of the budget for cleared derivatives processing in North America. As a transformative step, [49] propose a paradigm shift towards decentralized Financial Market Infrastructures (dFMIs), even though they acknowledge the associated complexities of complete decentralization.

Contrary to the shortcomings of the traditional system, tokenized collateral in derivative trading appears promising. [92] furnishes a strong business case and a more formal structure for Smart Collateral Contracts, positing that tokenized collateral could be aligned seamlessly with traditional risk management strategies and CCPs. [107] goes even further by signaling the strong potential for tokenized collateral to fundamentally alter derivatives trading through decentralized finance and on-chain asset management. Although [84] provides a generalized overview of asset tokenization, they do not specifically delve into its financial applications. Yet, the promise of tokenized collateral is also countered by legal and regulatory impediments. For instance, [57] and [101] question the legal enforceability of smart contracts, probing the delicate intersection of law and smart contract coding. Similarly, [51] talks about the regulatory landscape that is yet to fully embrace blockchain-based tokens, pointing to the potential hurdles that might arise as regulators begin to closely scrutinize this growing space.

The existing body of literature, while comprehensive in assessing the inefficiencies and challenges of traditional collateral management systems and the prospects of tokenized collateral, leaves a critical gap in terms of implementing these innovations on blockchain platforms, specifically on the BSV network. This oversight is particularly significant given BSV's unique features and capabilities that may offer distinct advantages for collateral management. The literature falls short in providing a detailed roadmap for actualizing tokenized collateral systems on BSV, and there is limited focus on the specific operational, legal, and regulatory considerations that would be unique to this particular blockchain. The present work aims to bridge this gap by not only conducting an in-depth analysis of how tokenized collateral management could be effectively implemented on BSV but also by addressing the idiosyncratic challenges and opportunities that this blockchain platform presents. In doing so, the research contributes a nuanced perspective that could serve as a blueprint for future implementations and regulatory discussions, thereby filling a significant void in the current academic and professional discourse.

Chapter 3

Solution Architecture

This chapter delves into the system of collateral tokenisation on the BSV blockchain. The chapter begins by outlining the essential elements and overarching structure of the system in Section 3.1. It then moves on to a detailed examination of how the ISDA's Common Domain Model has been adjusted to fit within the limitations set by BSV and sCrypt, with a focus on issues related to interoperability and taxonomy in Section 3.2. Concluding the chapter, Section 3.3 illustrates the steps involved in a typical collateral re-evaluation process, using a sequence diagram for clarity.

3.1 Components

A high-level overview of the architecture of the system is provided in Figure 3.1. The system is made up of two components:

1. **Wallet Software**. This is a piece of software acting as a client, interacting with the chain to spend specific UTXOs depending on the required action in the collateral lifecycle. The Wallet contains the Typescript files corresponding to the ISDA CDM: these describe the properties that different objects used throughout the lifecycle must possess, such as legal documentation or different types of events. The Wallet also contains Enumerations, i.e. lists of specific named values that certain properties can take on, e.g. the type of collateral allowed in a trade can only be one described in the Eligible Collateral Enumeration. The Wallet uses the type definitions when a new contract instance is created, populating it with the desired properties agreed upon by the two counterparties (this could either be done by a Wallet held by a custodian or one of the two counterparties, the difference being that a the custodian will have already sorted disputes regarding the specifics of the contract, thus lowering if not eliminating the chances of on-chain disputes that would require additional contract updates). By storing the type definition off-chain, the system allows for

easily updating the type definition should ISDA wish to do so, as opposed to a solution where the types were stored on chain and references to their addresses would have to be updated in all contract instances where they were used. It is true, on the other hand, that this poses problems regarding version mismatches: a party might update their wallet software while another might not, causing issues of interoperability. A fully on-chain solution could solve this problem by having a single source of truth for the type definitions and implementation of this alternative is suggested as future work on this dissertation.

2. **UTXOs**. These represent the states that the collateral is in at different points in time. We define two types of UTXOs, the contract and the balance. Note that this nomenclature is just an abstraction used for explanation purposes, there are no such things as different types of UTXOs, the only differentiating factor between them being the conditions that must be satisifief for the spending to occur.

   (a) **Contract**. This is the 'Smart Contract' representing the collateral. It contains the business logic describing the conditions that must be satisfied to spend that collateral (e.g. that only a transaction whose signature corresponds to the private key of one of the two counterparties or the custodian can be accepted). In Bitcoin terms, this is a Pay-to-Script-Hash (P2SH), a type of ScriptPubKey, i.e. the locking script (see Section 2.3.2.1 for more details), which allows for the spending of bitcoin based on the satisfaction of the script whose hash is specified within the transaction. Additionally, this is where the contract state (e.g. the amount of collateral currently represented by that UTXO denominated in a pre-specified currency, or the timestamp of the last update to the collateral state) is also stored. As every UTXO, it also contains an amount of satoshis, corresponding in this scenario to the value of the collateral. This is computed by taking into account the current USDC-to-Satoshi rate and the Mark-to-Market valuation of the collateral — which introduces an additional exchange rate risk — as explained in further detail on page 42.

   (b) **Balance**. These represent the amount of satoshis that either counterparty has available at any moment in time and from which value can be withrdrawn and transferred to the smart contract in case the valuation of the collateral drops, or conversely to which value can be transffered in the scenario where the valuation of the collateral increase and satoshis must be returned. In Bitcoin terms, these are Pay-to-Public-Key-Hash (P2PKH), a type of ScriptPubKey (see Section 2.3.2.1 for more details) which locks an amount of Bitcoin to the hash of a specific public key and only a message signed with that public key can spend the Bitcoin. The hash of the public key is commonly referred to as an address.

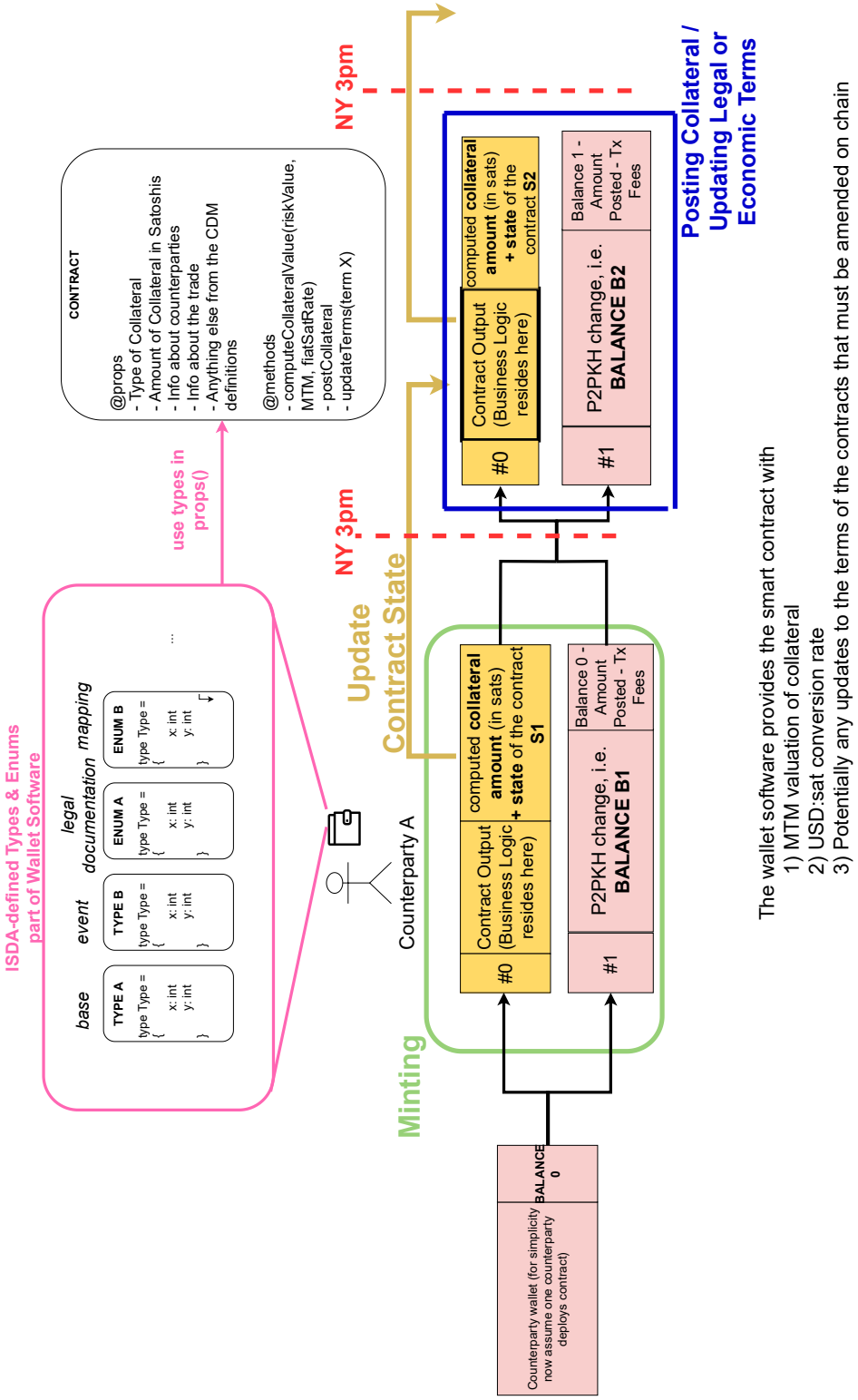


Figure 3.1: The diagram illustrates the key components of the system. The Wallet software incorporates ISDA Common Domain Model type definitions to populate smart contracts with essential properties. UTXOs are categorized as Smart Contracts (yellow) and Balances (pink). Collateral relationships begin with Minting and can be modified through Collateral Posting or Economic Terms updates. Collateral re-evaluation occurs regularly (e.g., at 3pm New York time) to accommodate potential price fluctuations.

As the UTXOs are spent and contract state and balances are updated (details in Section 3.3), a chain of successive lifecycle states is formed, which together represent the entire collateral relationship between the two parties throughout time. Multiple UTXO chains, and thus series of smart contracts, can exist between the same two counterparties at the same time, each for a specific collateral relationship, and we define each one as a *UTXO Set*. It is important to highlight that the system has checks in place to verify that a piece of collateral pledged in one UTXO Set cannot be utilised in another UTXO Set. By inspecting a UTXO Set, the entire history of the relationship can be inspected and thus audited, easing the job of regulatory compliance. Figure 3.2 shows a visual representation of UTXO Sets where a custodian still plays the role of a middleman between the two counterparties and holds assets on their behalf, thus being responsible for interacting with the chain and creating the UTXO Sets. The relevance of the figure of the custodian in this context is dicussed in further detail in Section 3.2.

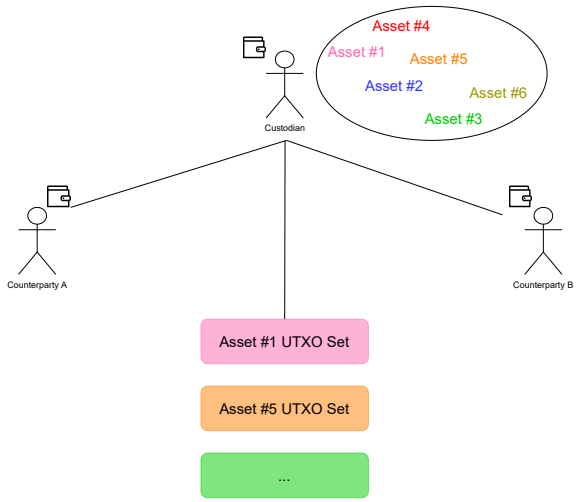


Figure 3.2: The Custodian wallet manages tokenised assets for counterparties. UTXOs linked to the same collateral are organized in UTXO Sets. An unlimited number of distinct UTXO Sets can exist between the same two counterparties. The diverse colors within the image represent the various assets held by the custodian, all of which can be pledged as part of a single UTXO Set at any given moment. Upon termination of a collateral relationship, the pledged assets become available for use in another UTXO Set.

3.2 Mapping Common Domain Model to Smart Contracts

A significant amount of time has been spent on designing the conversion from the ISDA Common Domain Model (see Appendix A.6) to a functionally equivalent representation in Typescript, more specifically in a sCrypt-compliant format. This requirement posed some technical challenges that we will expose in this section. The design principles behind the original Rosetta implementation are described on the official ISDA CDM website [67], however a Typescript implementation is also provided [64]. While a good starting point, we note that the Typescript implementation is heavily outdated and does not reflect the

latest changes in neither the Typescript language style formats, resulting in numerous compiler errors that had to be manually fixed, nor, more crucially, in the CDM definitions themselves, leading to extensive manual reconciliation work between the two sources. Our suggestion to ISDA, as described in chapter 5, is to create an automation tool to update the distributions in the different languages the CDM is provided in (including many of the most popular programming languages such as Python, Java, Scala, C#, Go and Typescript) to reflect the up-to-date changes to a single source of truth. This would require knowing the inner workings of the different languages and is outside the scope of this work .

The first step consisted in creating a visual representation of the type system defined in the Rosetta Domain-Specific Language originally used to publish the CDM. This can be observed in Figure 3.3, showing a tree-like structure where nodes represent types and edges represent connections between types, with a directed edge from type A to type B if type A references type B in its definition.

The visual representation allowed us identify three semantically logical groupings between different types which we define as the *Relationship*, the *Portfolio* and the *Position*, each represented as a distinct contract in the final system design.

- **Relationship**. This represents the overarching trading relationship between the counterparties independent of any specific piece of collateral. It includes references to multiple collateral portfolios, provisions for the collateral agreement (such as the type of eligible collateral, the collateral types accepted by the counterparties, or the substitution provisions) and details on the Independent Amount (IM). This serves as an extra cushion of collateral that the counterparty requires to be posted to mitigate the credit risk associated with the other party. This amount is independent of the mark-to-market valuation of the derivative contract [71]. The most noticeable difference between the original CDM and our solution relates to how the information about the counterparties is represented for the purposes of payments, in this specific context relating to IM but extending beyond that. The CDM utilises two pieces of information, the *Payer/Receiver Account Reference* and the *Payer/Receiver Party Reference* to uniquely identify a counterparty. These have been replaced by the BSV address of the parties' wallets. See Appendix A.1 for the source code of the Relationship contract.

- **Portfolio**. This represents a collection of collateral positions all governed by the same legal agreement. The portfolio contains a list and corresponding identifiers for the individual positions, and it is in turn contained in a Relationship. An aggregate balance is computed by taking into account the valuation of all the individual pieces of collateral. In our solution, this translates to storing and fetching all the UTXOs corresponding to the Positions and summing the amounts of satoshis locked in those

UTXOs. See Appendix A.2 for the source code of the Portfolio contract.

- **Position**. This represents the specifics of a piece of collateral, including its type, any treatments applied to it such as different types of haircuts[1], and the history of movements of that collateral piece (the *Price Quantities* in Figure 3.3). Two noticeable differences must be pointed out here related to the redundancy of specific attributes. Firstly, the *Settlement Status Attribute*, used to specify whether the valuation associated with the collateral includes amounts that have already been settled, amounts that have not been settled yet or both, is not of much use in a blockchain context where settlement is (close to) immediate. In fact one of the key advantages offered by the system is the immediacy with which trades are settled and reduced reconciliation efforts. We deemed the attribute redundant and decided to exclude it from the final system. Secondly, and relatedly to what has been just described, the *Settlement Terms* attribute connected to specific movements of the collateral becomes redundant and have thus been excluded from the system design. Finally, the CDM allows for different asset types to be represented, as shown by the *Underlying Product* attribute. However, due to time limitations with this dissertation and for the purposes of implementing a viable proof-of-concept, we restricted ourselves to one type of underlying, namely commodities, more specifically crude oil, as mark-to-market valuations would be easier to compute compared to more complex products such as interest rate swaps or indexes. Support for other types of underlyings is suggested as future work in Chapter 5. See Appendix A.3 for the source code of the Position contract.

[1] "Haircuts" refer to the percentage reductions applied to the value of collateral provided by a counterparty to account for potential market fluctuations.

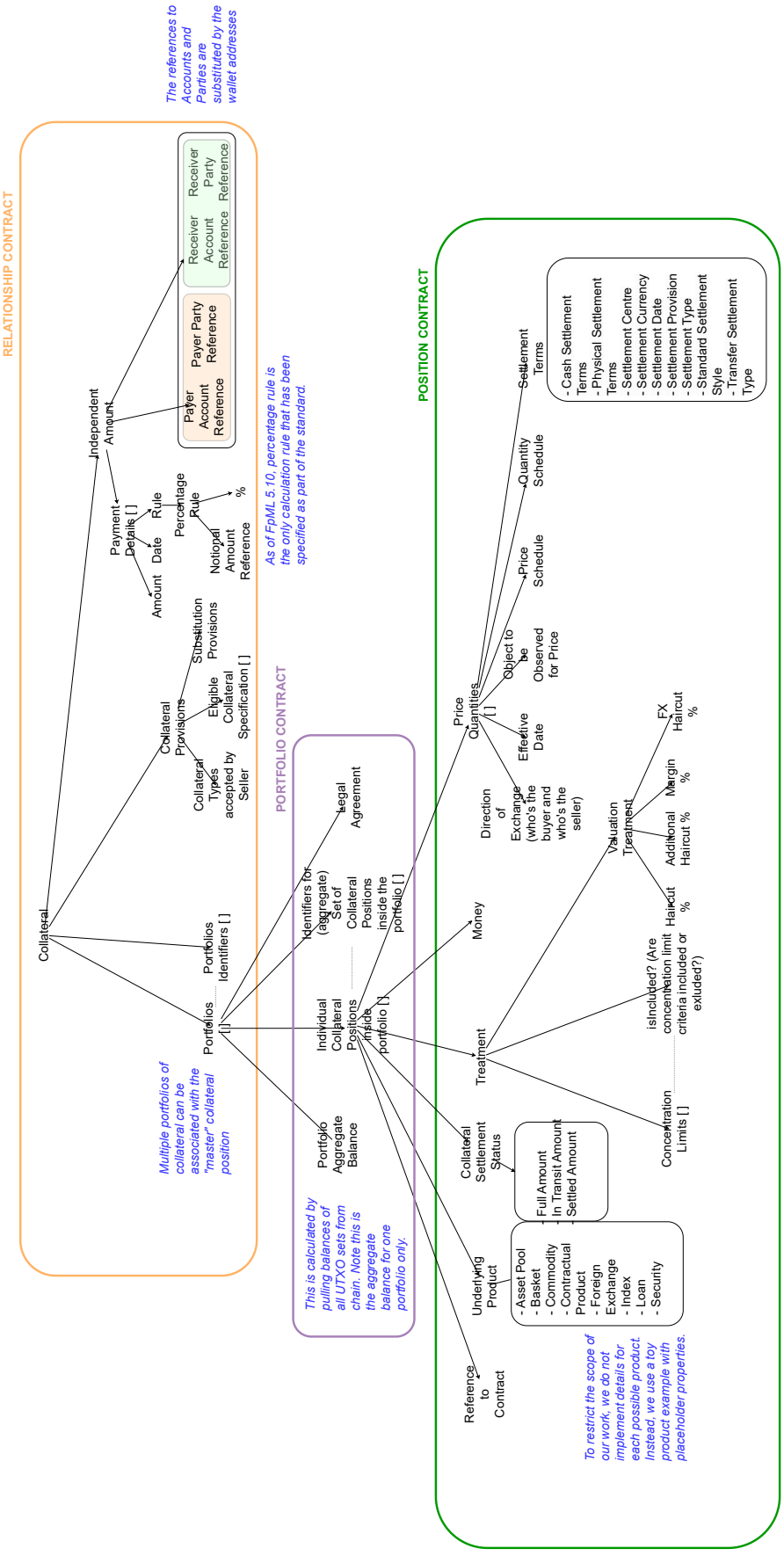


Figure 3.3: Illustration of the ISDA Common Domain Model (CDM) in a tree-like arrangement. Incoming edges denote containment of one type within another. The CDM is organized into three tiers of smart contracts. The Relationship contract signifies the trading relationship between counterparties, unrelated to specific collateral. The Portfolio contract oversees a group of collateral positions under a shared legal agreement. The Position contract outlines collateral details like type, treatments, and history of movements.

A key aspect of the dynamics between the different contracts is the ability to reference each other in multiple directions (e.g. fetching individual collateral positions starting from the relationship as well as retrieving the relationship details from any one position). Two approaches have been considered to include this property in our system.

1. Our original approach envisioned using arrays to store references to other contracts in the form of UTXO addresses. A challenge was quickly encountered in this scenario due to the fact that sCrypt's array implementations does not allow for variable-sized arrays, only fixed-size [108]. This would make it impossible to append new portfolios to a relationship or positions to a portfolio. As a workaround to this issue we originally considered storing only references from 'children' contracts to 'parents', removing the *Individual Collateral Position Portfolio* array from a Portfolio and the *Portfolios* array from a Relationship. This way, deploying a new Position, for example, would only require knowing the address of the containing Portfolio at deployment time and include it in the Position, and no arrays would have to be modified. Implementing the bi-directional retrieval capability would have required, however, some complex reverse-indexing [130] taking place off-chain. This solution appeared cumbersome and complex and was therefore discarded.

2. We developed an alternative solution, which is the one currently adopted by the system, after support talks with the main sCrypt developers. This solution uses HashedSet to store references to UTXO addresses. An HashedSet [110] is defined as a special type of HashedMap [109], a data structure that efficiently stores key-value pairs by using a hash function to compute indexes for quick retrieval and insertion. A HashedSet is a subtype of a HashedMap where values are identical to their corresponding keys and are thus omitted. A key aspect of the sCrypt implementation of HashedSets is that only the hash values of the keys are saved on the chain, meaning that off-chain copies of the items in the set must be stored locally to be able to deserialize the set into an intelligible representation (hash functions are designed to be one-way functions, making it extremely difficult to retrieve the original input from the hash value [118]). While adding some technical complexity to the final solution, this approach achieves the bi-directional referencing currently supported by the CDM and we deemed this design to be more feasible and elegant than the original one involving arrays

As already mentioned, mapping the CDM to sCrypt has proven particularly challenging for a number of technical reasons described below:

- **Data types supported by sCrypt**. sCrypt supports three main basic data types, namely *boolean*, *bigint* and and *ByteString*. While the CDM also uses the boolean type, it relies on the *number* and *string* primitives. All references to these two

data types had to be converted to *bigint* and *ByeString*, respectively. The rationale behind the choice of these two particular data types to represent numbers and text is not provided in the sCrypt documentation, and while requiring some manual work to convert the CDM, we speculate on the utility of these two data types specifically in a blockchain context:

- **Bigint** (represents numeric values which are too large to be represented by the number primitive in Typescript [93]).
  - **Precision**. BSV operates on very large numbers, especially when dealing with satoshis. Regular *numbers* have limited precision, while *BigInt*s can accurately represent and perform arithmetic operations on integers of arbitrary size without loss of precision (and only a 5.30% performance difference as measured in arithmetic operations per second at the time of writing [89]).
  - **Consistency with Bitcoin Protocol**. The Bitcoin protocol itself uses 64-bit integers for various purposes, therefore BigInt is a better match for representing these integers in a way that aligns with the underlying protocol.
  - **Security and Trust**. Using BigInt helps minimize the risk of bugs caused by floating-point inaccuracies and ensures that the code behaves as expected.
- **ByteString** (represents arbitrary binary data, ensuring this is handled as raw bytes rather than interpreted characters)
  - **Script operations**. BSV transactions often involve manipulating raw binary data (like public keys, hashes, signatures) rather than traditional text data. Using ByteString can help represent this data more accurately and efficiently.
  - **Hexadecimal Representation**. in BSV data is often represented in hexadecimal format (base16). Using a ByteString or binary data representation can make it easier to work with and manipulate these hex-encoded values.
  - **Data (De)serialization**. Data serialization and deserialization efficiency[2] is pivotal due to constraints such as block size limitations. Binary data

[2]Efficiency here encompasses both temporal and spatial considerations. Compact data representations are crucial because blockchain memory is a limited resource; using it judiciously minimizes transaction fees. Additionally, quick data operations are imperative for supporting the high throughput of transactions in the financial derivative contexts.

representation can facilitate more compact serialization formats, thereby optimizing both transaction size and associated fees.

  - **Data Integrity**. Using binary data representation can help ensure data integrity by preventing unintended character encoding or transformations that could affect calculations.

- **Floating point numbers**. While the BigInt datatype ensures higher precision with integer calculations, it does not support floating point calculation, which in our context would be required. For example, the spot price of an asset or the exchange rate between USDC and BSV are all floating point numbers. The current solution truncates the decimal part by taking the floor of the floating point number. Possible workarounds to this barrier are discussed in Section 5.

- **Dates**. The CDM relies on the built-in Date object in Typescript to represent points in time [94]. Consistency with this approach would have required translating part of the Typescript implementation itself to utilise the basic sCrypt data types, which we considered out of the scope for this work. A suggested workaround is to simply use a BigInt value that represents the UNIX epoch time (the number of milliseconds since the midnight at the beginning of January 1, 1970, UTC [61])

- **Generics**. These are reusable and flexible code components in Typescript that allow types to be parameterised, enabling functions, classes, and interfaces to work with different data types while maintaining type safety and preventing the need for code duplication [127]. For example, the $FieldWithMeta\langle T \rangle$ interface defined in the *metatypes.ts* file of the CDM allows to encapsulate a value of any generic type T along with accompanying metadata within a single structure. Generics are not supported by sCrypt, the compiler yielding an *Invalid Type* error. A possible workaround to to this obstacle would be to create specific variations of the generic interface for all possible types to be supported (sCrypt does support user-defined types [112]), effectively violating the very purpose of generics. This has not implemented in the current version of the system as it would have required significant additional manual work that would have not added any major benefits for the purpose of the proof-of-concept and is suggested as future work.

- **Arrays**. As already mentioned on page 37, sCrypt only supports fixed-size arrays. This poses an issue when a type contains arrays that could potentially be empty at deployment and will need to be filled in later. For example, the *EligibleCollateral-Criteria* interface contains an array of *IssuerCriteria[]*, specifying requirements or qualifications that an issuer of a financial instrument must meet in order for that instrument to be considered acceptable as collateral, which could be potentially empty at the beginning in case no criteria apply to the type of issuer of the instrument.

Assuming the arrays are fixed-size, it makes sense that the compiler would not allow to use empty arrays as these could not be modified while at the same time occupying block size. We originally tried to apply the same workaround described on page 37 utilising *HashedSet*s instead of arrays, however the compiler would still throw errors in case the HashedSet was empty at deployment. A possible solution — not implemented in the system, which currently simply ignores empty arrays — would be to redeploy a new contract when the values to populate the array are known, ignoring the UTXO corresponding to the previous state.

- **Circular dependencies**. In a strongly typed language like TypeScript, a circular dependency between types occurs when two or more types depend on each other directly or indirectly. This can lead to a situation where the types reference each other in a way that creates a loop, making it difficult for the compiler to properly infer and resolve the types [96]. In the CDM, many circular dependencies are present. For example, the *Underlying Product* for the collateral position could be of type *Basket*, which contains multiple *Basket Component*s that could potentially be *Basket*s themselves. Such dependencies would prevent compilation from succeeding, therefore we had to manually amend the types that were generating the problem. While far from optimal, we opted for this workaround as a more thorough solution would have required a deeper overhaul of the CDM itself which was outside the scope of this work. As discussed in 5, we suggest close collaboration between sCrypt developers and CDM should be fostered to increase the level of interoperability between the two tools.

3.3 Flow of Events

The lifecycle of a piece of collateral, and its UTXO Set on-chain, can be broadly split into two stages (refer to Figure 3.1):

1. **Minting**. This is the initiation of the UTXO set. The Wallet software creates the smart contract by using the CDM type system, and consumes an existing UTXO representing the balance of the party minting the collateral to create two new UTXOs: the contract and the remaining balance.

2. **Posting/Updates**. At regular time intervals the state of the UTXO Set is updated to reflect changes in collateral valuation (note that this functionality is not currently implemented but could easily be achieved by running a *"cron job"* [3] process in the background). The updating transactions consumes two inputs, the current contract state and balance of the counterparty, and produces two new UTXOs: the updated

[3] A "cron job" process is a scheduled task or automated script in Unix-like operating systems that runs at specified intervals without the need for manual initiation [78].

contract state (potentially containing a different amount of satoshis) and the updated balance. These updates can be of two types:

(a) **Posting/Receiving Collateral**. This type of update refers to movement of satoshi to either the contract or the party's balance depending of the new valuation.

(b) **Updating Contract Terms**. this type of update allows the custodian to modify the business logic of the contract (e.g. the eligibility of an existing or new type of collateral could be updated). Recounting that legal and economic terms from both the ISDA Master Agreement and the CSAs can be updated in the current sysytem, supporting this functionality is core to a smooth transition the on-chain solution.

In the current implementation 3pm New York Time (ET) has been arbitrarily chosen as the time to update the UTXO Set, however this level of granularity is quite rough and would not account for intra-day valuation variations. An additional piece of functionality that constantly monitors the state of the market and the valuation of the collateral and updates the UTXO Set whenever a specified threshold valuation is passed would be a good solution to reflect large swings in the market. This is suggested as future work on the this thesis.

A more granular view of the flow of events is provided in Figure 3.4, where a sequence diagram of the system is described. The two sections of the sequence diagram correspond to the Minting and Updating stages described above.

1. **Minting**. Let us assume that a custodian is in charge of interacting with the chain. The act of minting requires three smart contracts to be written, the Relationship, the Portfolio and the Position (details on the contents of each are described in more detail in Section 3.2). The custodian will first deploy the Relationship contract. Only after this has been deployed on-chain can the custodian include the address of its UTXO in the Portfolio contract and deploy the latter. A key aspect of mechanism of the current system is that all objects (within the collateral context as well as beyond it) contain references to each other, and such interconnectedness allows for easy retrieval and inspection beginning with any starting point (one could go back to the trading relationship between parties from a specific collateral position or vice versa, fetch all the positions contained within a relationship). Adapting this mechanism on-chain involves reflecting any change to a '*sub-contract*' (hereby defined as a contract referenced by another one) by updating the *'parent'* contract(s) (the referencing contracts). In the Minting scenario just described, the fact that the Portfolio contract has been deployed must be propagated upwards to the corresponding Relationship within which the Portfolio is contained. In a similar manner,

when the Portfolio has been deployed, the Position contracted can be filled in with the Portfolio UTXO's address, published on chain and the deployment information (which takes the form of a UTXO address itself) propagated up the stack to the Portfolio and the Relationship.

2. **Updating**. The type of update described here corresponds to the movements of satoshis following a change in collateral valuation. Updates of the legal and/or economic terms have not been implemented as the core value proposition of this work is to streamline the valuation and margin-posting elements. While contract term updates would be a required feature of a fully CDM-compliant system, we leave this implementation as future work. Margin calculation and posting follow the steps desribed below:

   (a) The Wallet must compute the Mark-to-Market (MTM) valuation for the asset. Depending on specific type of asset, the MTM calculation may vary (e.g. for commodities this is simply the spot price multiplied by the quantity of the asset held as collateral, while for interest rate swaps more detailed calculations involving the fixed and floating rates, as well as the market rates for the corresponding fixed and floating rates, are required). For the purpose of a proof-of-concept, this thesis uses a commodity, crude oil, as collateral. Note that the system is easily extendable to support other types of commodities simply by plugging in the MTM valuation models that a counterparty may already use. The commodity spot price is fetched from the party's preferred traditional financial API, e.g. the Bloomberg terminal. This work used the free and open-source Yahoo Finance library [131, 132] (see Appendix A.4. In the initial version of the system we planned to use an oracle to provide the spot price and/or MTM, since relying on a traditional API introduces the risk of valuation mismatches should the parties use different sources for their calculations. We were unable to find public oracles providing the service and we attribute this to the low level of maturity of the technical ecosystem, however adapting to changes in 3rd party services provided in this context is suggested as a direction of future work.

   (b) The valuation of the collateral should be expressed in satoshis in order to facilitate the movement of collateral. Denominating the asset in satoshis require converting the preferred fiat currency of the parties to satoshis, and this could be accomplished by fetching the exchange rate from an oracle service. In this thesis we use the WitnessOnChain oracle service [129] to fetch the exchange rate between USDC and BSV (see Appendix A.4. USDC is a stablecoin, i.e. a digital token whose value is pegged to USD, in the case of USDC specifically via reserve-based pegging. Each USDC is redeemable for one dollar, and is backed by one dollar or a dollar-denominated asset with equivalent value held in ac-

counts at regulated U.S. financial institutions. Those accounts are audited by U.S. accounting firm Grant Thornton LLP, which issues monthly attestations on the reserves backing USDC [56]. While we initially wanted to convert from USD to BSV (as USD is the currency most widely adopted in current collateral relationships), we were not able to find an oracle service providing this specific conversion rate. While relying on USDC is not optimal, the system can be considered safe as long as the stablecoin maintains its peg. An extra layer of security to protect against the fluctuations in the USDC-BSV exchange rate would be to apply an additional haircut to the CSA of the collateral agreement, effectively treating the conversion as a traditional foreign exchange rate. A haircut is a reduction applied to the value of an asset expressed as a percentage of its value [44]. For example, if the average fluctuation of the USDC-BSV exchange rate over the previous week was 5%, a 5% discount haircut could be applied to the collateral. The full implications and implementation of this solution are left as future work on this dissertation.

(c) The Wallet then fetches the current collateral value stored in the Position contract. This is required to determine whether the collateral has increased or decreased in value.

(d) The Wallet then computes the new value of the collateral by using the MTM valuation and the USDC-BSV exchange rate. Note that at this stage any additional factors traditionally used in asset valuation, e.g. specific risk models such as Value-at-Risk (VaR) [59], SPAN (Standard Portfolio Analysis of Risk) [76] or custom Monte Carlo Simulations [97], can be incorporated. Performing these calculations off-chain introduces an additional risk factor since a counterparty must trust the results that the other party provides, fundamentally nullifying one of the core value propositions of this thesis, i.e. the transparency of the margin calculations. An ideal solution would work entirely on-chain, with the calculations, or at least the parameters used in them, publicly auditable. While outside the scope of this work, a thorough analysis of the theoretical and technical aspects of implementing risk calculations at least partly on-chain represents an essential component in fully delivering the benefits that a transition to blockchain purports to offer. Details on how this solution could potentially look like are provided in the Suggestions for Future Work 5.

(e) The Wallet then updates the Position contract to reflect the newly computed collateral valuation. This information is stored in a property of the smart contract so that it can later be used to verify whether the difference in valuations computed in the successive steps is correct.

(f) The Wallet computes the difference in satoshis between the previous and current valuations of the collateral.

(g) Depending on whether the valuation has risen or dropped, an amount of satoshis corresponding to the difference is transferred to either a P2PKH balance UTXO for the counterparty, in the former case, or to a new contract UTXO in the latter (see Appendix A.5). Note that whenever UTXOs are spent or updated, changes must be propagated to all referencing contracts (as described in 1), hence the *updatePortfolio* and *updateRelationship* functions that appear in the sequence diagram.

Figure 3.4: Sequence diagrams depicting the steps in the collateral smart contract lifecycle. The upper diagram displays the initial deployment of three smart contracts: Relationship, Portfolio, and Position. The lower diagram illustrates the process of reassessing collateral value due to market shifts, incorporating data from oracles and financial APIs. It also showcases the adjustment of satoshi amounts to align with collateral valuation changes.

Chapter 4

Results and Discussion

This chapter offers an in-depth analysis of the outcomes associated with the proposed system for tokenizing collateral in derivative trading on the BSV blockchain. Firstly, we discuss how the custodian's role is transformed but still critical (Section 4.1). We highlight the benefits of a hybrid model that combines automation through smart contracts with human oversight for tasks like risk assessment and compliance. Secondly, we examine the advantages across four domains: financial, economic, legal, and technological (Section 4.2). This includes discussions on operational cost savings, enhanced market liquidity, regulatory compliance, and how BSV's capabilities align with the system's requirements. Finally, we address the legal aspects of tokenizing collateral, classifying our system as a hybrid model (Section 4.3). This section highlights how the system manages to meet regulatory requirements while also enabling direct interactions through smart contracts.

4.1 On the Role of the Custodian

The role of custodians in our proposed system architecture remains integral but is substantially altered. While functions like record-keeping could be automated via the blockchain's immutable ledger, responsibilities such as risk assessment and compliance assurance still warrant human expertise. The blockchain's capacity for automating record-keeping may reduce operational costs and minimize manual errors, enhancing the efficiency of audits [34]. However, the custodian's role continues to be important in functions like dynamic risk assessments, which necessitate human judgment.

In terms of compliance, smart contracts may automate certain regulatory requirements but fall short in addressing the complexities of varying international laws or rapid legal changes. Thus, custodians act as a safeguard in ensuring all-encompassing regulatory adherence [125].

A new paradigm in custodianship may arise in this setting, characterized by a blend of automation and human oversight. While smart contracts could facilitate instantaneous asset transfers, custodians would retain a role in supervising the initiation and completion of transactions, particularly for handling exceptions or contingencies [73].

This hybrid model of custodianship offers the best of both worlds: it couples the reliability of automated processes with the expertise of human oversight. For instance, a study from the conventional financial sector found that incorporating automated processes along with human supervision decreased transactional errors by nearly 30% [1]. In this context, smart contracts would handle routine operations like immediate post-trade collateral transfers, freeing custodians to concentrate on complex tasks such as liquidity evaluation and risk management [32]. Consequently, this combination streamlines the initiation and settlement processes without compromising on the informed decisions that custodians contribute, resulting in a more efficient and secure trading infrastructure.

4.2 Benefits of Solution

The benefits of implementing the proposed collateral management system are described in this section. We examine the advantages across four key dimensions: financial (Section 4.2.1), economic (Section 4.2.2), legal (Section 4.2.3), and technological (Section 4.2.4). From significant reductions in operational costs to improvements in market liquidity and efficiency, we outline how the solution aims to address existing challenges. We also explore risk mitigation strategies and how blockchain technology can aid in regulatory compliance and dispute resolution. Moreover, the chapter discusses the technological attributes that make the BSV blockchain a suitable choice for this application.

4.2.1 Financial

- **Reduction in operational overheads**. The operational overheads and costs associated with derivatives trading in the realm of collateral management can be substantial. The financial services sector could save more than €4 billion annually in collateral management costs by addressing operational inefficiencies [1]. These costs stem from the need for specialized software, manpower, and compliance measures to manage the complex collateral requirements associated with derivatives contracts. Additionally, the advent of regulations like Dodd-Frank in the U.S. [117] and EMIR [46] in Europe has led to increased reporting and margin requirements, further escalating costs [2]. Firms also incur opportunity costs by tying up capital as collateral that could otherwise be invested. A report by Deloitte estimates that collateral requirements could tie up as much as $1.9 trillion in high-quality assets [32]. Therefore, the operational overheads in collateral management are not just a cost center

but also a strategic concern that impacts liquidity and capital efficiency. Our solution promises to relieve trading entities of the operational overheads by proposing a standardized format to tokenize collateral and automating the movement of assets.

- **Enhanced liquidity**. Enhanced liquidity in derivatives trading, particularly through the ease of trading and transferring assets, has a profound impact on the market. By making previously illiquid assets more accessible, the market experiences a boost in trading volume and efficiency. For instance, the securitization of illiquid assets like mortgages or loans into tradable derivatives allows a broader range of investors to participate in markets that were previously inaccessible. According to statistics reported by the Bank for International Settlements (BIS), the global derivatives market has grown to an estimated $640 trillion in notional amount outstanding, partly fueled by the increased liquidity of assets [31]. The advent of digital assets has further augmented this trend. Cryptocurrencies and tokenized assets, being easily tradable on various digital platforms, have introduced a new layer of liquidity. They have enabled even more participants to engage in the market, making assets like real estate or art, once considered highly illiquid, more accessible through tokenization [80]. This enhanced liquidity not only fosters market stability by allowing for more seamless price discovery but also promotes economic growth by facilitating capital allocation. Furthermore, it reduces the cost of capital for issuers and provides investors with diversified investment opportunities.

  While the benefits of enhanced liquidity through asset tokenization and digital trading platforms are significant, it's crucial to consider the complexities involved. For instance, the mere act of making a traditionally illiquid asset more tradable does not automatically translate to increased market liquidity. The Central Limit Order Book (CLOB) systems, commonly used in asset trading, may not be well-suited for illiquid assets, as Market Makers often require a wide spread to mitigate the risks associated with the asset's volatility [87]. To address these challenges, we propose two alternative approaches to consider. One such strategy is to restrict the trading window for illiquid assets, thereby concentrating market activity and potentially stabilizing prices. Another approach could be the adoption of Automated Market Maker (AMM) systems, a concept borrowed from decentralized finance (DeFi). AMMs operate on a liquidity pool model where participants deposit assets into a smart contract. These pools can then facilitate trades directly between buyers and sellers without the need for a traditional Market Maker. In the context of illiquid assets, an AMM system allows participants to provide liquidity directly, making the market more robust and efficient [7]. These nuanced strategies aim to foster a more stable and liquid market, facilitating better price discovery and capital allocation.

4.2.2 Economic

- **Increased market efficiency**. Traditionally, the settlement of trades often operates on a "T+2" basis, meaning that the transaction is finalized two business days after the trade is executed. This delay introduces a range of inefficiencies and risks, including counterparty risk and the need for more extensive collateral management [8]. Our solution promises to revolutionize this paradigm by enabling "T+0" settlements—same-day settlement of collateral operations. The acceleration to "T+0" is not merely a technological feat but a transformative shift towards increased market efficiency. According to a report by McKinsey [88], shorter settlement times generate significant savings in high-interest-rate environments such as at the time of writing [10]. For investors, these savings may be the greatest near-term impact and the main reason why the business case for tokenization is specifically now ripe for delivering advantages.

- **Risk mitigation**. In addressing the inefficiencies and vulnerabilities in collateral management, our solution offers substantial improvements in risk mitigation. Firstly, trust between parties is enhanced; the transparency and immutability of blockchain transactions eliminate the need for intermediaries, thereby reducing information asymmetry and counterparty risk, as highlighted by Deutsche Bundesbank [37]. Operational risks — such as the risk of delayed settlements or human errors — are also mitigated as the BSV'S low latency enables real-time settlement, thus increasing liquidity. A study by the Depository Trust & Clearing Corporation (DTCC) found that 30% of trades have discrepancies due to manual reporting and human errors [34]. Lastly, the cryptographic security inherent in blockchain technology mitigates the risk of fraud and unauthorized transactions.

4.2.3 Legal

- **Regulatory compliance**. In the traditional system of collateral management for derivatives trading, regulatory compliance has often been a cumbersome process, fraught with inefficiencies such as manual record-keeping and auditing. The cost of compliance is substantial; according to a 2020 report by Thomson Reuters, financial firms spend approximately $180 billion annually on compliance and regulatory obligations [125]. The immutability of the blockchain ensures that once a transaction is recorded, it cannot be altered or deleted. This serves as a robust mechanism for audit trails, aiding in compliance with regulations that require firms to maintain historical data for several years. For example, the Dodd-Frank Act [117] requires swap dealers to keep records for as long as the swap is active and for five years thereafter [115]. The blockchain's immutable nature inherently satisfies this requirement. Additionally, smart contracts can be programmed to automatically enforce compliance

rules, such as minimum collateral requirements or maximum leverage ratios. This reduces the manual effort involved in ensuring compliance and minimizes the risk of human errors.

- **Dispute resolution**. The absence of a single, transparent source of truth in traditional systems often leads to discrepancies in collateral valuation, margin calls, and other contractual obligations, thereby causing disputes that are costly and time-consuming to resolve. The BSV blockchain serves as an immutable ledger, recording all transactions and collateral adjustments, thus eliminating the possibility of data manipulation and reducing the scope for such disputes. This feature is particularly relevant given the recent remarks by ISDA on the necessity to harmonize data reporting rules across jurisdictions and ensuring consistent data sets for regulators [73].

  Moreover, the traditional dispute resolution process is bogged down by bureaucratic inefficiencies and delays, exacerbated by the involvement of multiple parties with disparate record-keeping systems. Smart contracts on the BSV blockchain can be programmed to automatically execute actions like margin calls based on predefined conditions. This automation minimizes human errors and ensures compliance with the terms of derivative contracts, thereby reducing the likelihood of disputes arising from non-compliance. Such automation aligns with the recommendations made in DTCC's "The Changing Face of Derivative Reporting" to improve efficiency [35].

4.2.4 Technological

- **Scalability**. The choice of BSV as the underlying blockchain technology for the proposed collateral management system is not arbitrary but is informed by a set of unique technological advantages that it offers. First and foremost is the issue of scalability. BSV is designed to handle a high transaction throughput, a critical requirement for the fast-paced, high-volume nature of derivative trading. Unlike other blockchains that struggle with scalability issues, BSV can handle larger block sizes, thereby facilitating more transactions per second [136]. To provide a quantitative perspective, BSV can process up to 2,000 transactions per second (tps) [122], compared to Bitcoin's 7 tps [58] and Ethereum's 30 tps [85]. This increased throughput is facilitated by BSV's capacity for larger block sizes—up to 2GB as opposed to Bitcoin's 1MB and Ethereum's variable, but smaller, block size — as discussed on page 17. The larger block size not only allows for more transactions per block but also reduces the likelihood of transaction backlog, ensuring faster processing times. In terms of overhead, BSV offers lower transaction fees, with average fees orders of magnitude smaller than Ethereum's or Bitcoin's, as mentioned on page 18.

- **Security**. Security is another cornerstone. BSV offers a robust security protocol that can withstand various types of attacks, making it a reliable platform for managing financial assets. Its proof-of-work consensus algorithm and the cryptographic techniques employed ensure the integrity and immutability of data [95].
- **Interoperability**. The system is interoperable with other financial systems. This is particularly important for derivative trading entities that operate across different blockchain ecosystems. For instance, a functionally-equivalent representation of an ERC-20 fungible token from the Ethereum blockchain [41] could be implemented on BSV [113], thereby broadening utility and adoption of both ecosystems. Details on how ERC-20 tokens could be implemented on BSV are outiside the scope of this project and are left as future work.

4.3 Legal Consideration of Tokenised Collateral

In Section 2.4, we delineate between various tokenisation models. We now classify the system proposed in this study as a hybrid model that combines features from both the Registered and Claims paradigms. From the Registered model, our system adopts an identity layer, thereby anchoring the ownership of assets to verified identities maintained in a regulated registry. This element is pivotal for adhering to the rigorous legal and regulatory compliance standards often encountered in collateral agreements. Conversely, the system incorporates elements from the Claims model by enabling participants—whether they are the counterparties or custodians acting on their behalf—to engage in direct interactions through the utilization of operator-deployed smart contracts. Additionally, it is important to note that within the proposed system, the tokenised collateral does not constitute an asset per se. Rather, it serves as a digital equivalent, the ownership and control rights of which are established through contractual agreements.

To ensure legal and regulatory compliance, the operator of the proposed system must adhere to a set of key requirements, outlined here in a non-exhaustive list. These include Anti-Money Laundering (AML) regulations such as the United States' Bank Secrecy Act (BSA) [53] and the United Kingdom's Money Laundering Regulations 2017 [99]. Furthermore, Know Your Customer (KYC) procedures are obligatory under these frameworks to verify the identities recorded in the registry. Smart contracts deployed by the operator must also comply with laws governing electronic signatures, such as the Electronic Signatures in Global and National Commerce Act (E-SIGN Act) [48] in the U.S. or the EU's eIDAS regulation [45] for electronic identification and trust services. In terms of establishing control and ownership rights over digital and tokenized assets, the Uniform Commercial Code (UCC) Article 9 [83] serves as a relevant legal framework.

Chapter 5

Conclusions and Future Work

This dissertation has undertaken a comprehensive exploration into the operational inefficiencies in the collateral management systems within financial derivative trading. The primary focus has been on the development and evaluation of a blockchain-based solution, specifically on the BSV blockchain. The research has been conducted in alignment with industry standards set by the ISDA and has utilized sCrypt, a TypeScript-based Domain Specific Language, for the creation of smart contracts.

The study has made several key contributions. Firstly, a concrete methodology for tokenizing underlying collateral assets has been developed. This tokenization process allows for the efficient management and real-time valuation of assets, thereby potentially increasing market liquidity. Secondly, the research has outlined a robust digital twin representation for tokenized assets, demonstrating how these digital representations can encapsulate unique value per asset type. Thirdly, a hands-on Proof of Concept tokenizing crude oil as collateral has been presented, offering a tangible representation of how the proposed system might operate in practice. Lastly, the dissertation has examined the proposed solution from economic, financial, legal, and technological perspectives, providing a comprehensive understanding of its implications.

The research has shown that while the BSV blockchain is still in its early stages, it offers promising features such as microtransaction capability and scalability. However, there are technical hurdles in adapting existing ISDA frameworks to blockchain technology, which necessitate further research and development. On the regulatory and legal front, the study has found that the tokenization of collateral assets can be aligned with existing regulatory frameworks, although the transition would require a multi-stakeholder approach involving both financial and technological sectors. Economically and financially, the proposed system has the potential to reduce operational overheads, improve regulatory reporting, and streamline collateral allocation processes. However, the full economic benefits can only be

realized through greater collaboration between the financial and tech sectors.

We provide the following suggestions for future work and alternative implementation based on the system hereby presented:

- **Type System On-Chain**. As discussed on page 30, implementing the CDM type system directly on the blockchain could address version mismatch issues by centralizing type definitions. This approach would enhance interoperability but would require a strategy for updating on-chain type definitions.
- **Intra-day Valuation Updates**. As discussed on page 41, the current system updates the UTXO Set at a fixed time, 3pm ET, which may not capture market volatility. Future work could include a dynamic update mechanism that triggers when collateral valuation crosses a predefined threshold.
- **Updates of Legal and/or Economic Terms**. As discussed on page 42, while the current system focuses on valuation and margin-posting, future work could include the development of on-chain mechanisms for updating legal and economic terms in compliance with the CDM. This would enhance the system's adaptability to changing market conditions and regulatory requirements.
- **Additional Asset Types**. As discussed on page 35, future work could focus on implementing a comprehensive type system on the blockchain. This would allow for more granular control and validation of collateral types, extending beyond commodities to include complex financial products like interest rate swaps or indexes.
- **Oracles for Spot price and/or MTM**. As discussed on page 42, the initial design aimed to incorporate oracles for more accurate and consistent spot price and MTM data. Future work could focus on developing or integrating with emerging oracle services as the technical ecosystem matures, to mitigate the risks associated with valuation mismatches from traditional APIs.
- **FX Haircut**. As discussed on page 42, the system currently relies on the WitnessOnChain oracle service for USDC-BSV exchange rate conversion. Future work could explore the integration of multiple oracles for redundancy and improved accuracy, as well as the development of a dynamic haircut model that adjusts in real-time based on exchange rate volatility.
- **On-chain Risk Calculations**. As discussed on page 43, future work could explore the integration of more sophisticated risk models like VaR, SPAN, or Monte Carlo Simulations directly into the blockchain. This would address the issue of trust in off-chain calculations by making the parameters and results publicly auditable. Research could focus on optimizing these complex calculations to be more efficient on-chain or investigate hybrid models that perform part of the calculations on-chain

and part off-chain while maintaining a high level of auditability. A possible starting point in this direction would be akin to zero-knowledge rollups in Ethereum. By leveraging zero-knowledge technology, computations are performed off-chain (thus not encountering the costly computational limits of performing operations on-chain), while inheriting the security and auditability of the blockchain by posting the proofs of the calculations on-chain [81]. While the level of maturity of the Ethereum ecosystem in this regard is far more advanced than Bitcoin's, solutions built natively on BSV are starting to emerge, for example as described in [114].

- **Tool to update CDM distributions in different languages**. As discussed on page 34, the existing Typescript implementation of the ISDA Common Domain Model (CDM) is outdated and requires manual reconciliation. Future work could focus on developing an automated tool that updates CDM distributions across multiple programming languages, ensuring they are in sync with the latest definitions and language-specific standards.

- **Fix for Generics**. The current system lacks support for generics in sCrypt, as detailed on page 39. Future work could focus on enhancing the sCrypt compiler to natively support generics, thereby maintaining type safety and reducing code duplication, or alternatively, developing a pre-compiler tool that automatically generates type-specific versions of generic interfaces.

- **Circular Dependencies**. Given the challenges posed by circular dependencies described on page 40, future work could focus on a comprehensive refactoring of the CDM to eliminate these issues. This would not only improve type inference but also enhance the system's overall maintainability and compatibility with sCrypt.

Bibliography

[1] Accenture. Inefficiencies in Collateral Management Cost the Financial Sector More than €4 Billion Annually. https://newsroom.accenture.com/news/inefficiencies-in-collateral-management-cost-the-financial-sector-more-than-four-billion-annually-according-to-accenture-clearstream-survey.htm. Accessed on 26th August 2023.

[2] P. Aditya. European Markets Infrastructure Regulation (EMIR) and Dodd-Frank Wall Street Reform and Consumer Protection Act (DFA): A New Revolution of OTC Derivatives Towards Transparency. *Available at SSRN 2314998*, 2013.

[3] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic. Trustworthy blockchain oracles: review, comparison, and open research challenges. *IEEE access*, 8:85675–85685, 2020.

[4] J. G. Allen, M. Rauchs, A. Blandin, and K. Bear. Legal and regulatory considerations for digital assets. 2020.

[5] R. W. Anderson and K. Joeveer. The economics of collateral. *Available at SSRN 2427231*, 2014.

[6] A. M. Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies*, page 124. " O'Reilly Media, Inc.", 2014.

[7] J. Aoyagi. Liquidity provision by automated market makers. *Available at SSRN 3674178*, 2020.

[8] A. Baig, S. Breeze, J. Cox, and T. Griffith. Settling down: T+ 2 settlement cycle and liquidity. *European Financial Management*, 28(5):1260–1282, 2022.

[9] Bank for International Settlements. High-quality liquid assets. https://www.bis.org/basel_framework/chapter/LCR/30.htm. Accessed on 5th September 2023.

[10] Bank of England. Bank Rate increased to 5.25%. https://www.bankofengland.co.uk/monetary-policy-summary-and-minutes/2023/august-2023#:~:text=A

t%20its%20meeting%20ending%20on,maintain%20Bank%20Rate%20at%205%25. Accessed on 28th August 2023.

[11] S. M. Bartram. Corporate hedging and speculation with derivatives. *Journal of Corporate Finance*, 57:9–34, 2019.

[12] Bitcoin Magazine. What is the Bitcoin block size limit? https://bitcoinmagazine.com/guides/what-is-the-bitcoin-block-size-limit. Accessed on 5th September 2023.

[13] Bitcoin Satoshi Vision. Global Infrastructure On The Original Bitcoin Blockchain. https://www.bitcoinsv.com/. Accessed on 4th September 2023.

[14] Bitcoin Stack Exchange. What are Bitcoin's transaction and script limits? https://bitcoin.stackexchange.com/questions/117594/what-are-bitcoins-transaction-and-script-limits#:~:text=Standardness%20rules%20only%20allow%20for%203600%20bytes%20for%20the%20script,be%20pushed%20per%20stack%20element. Accessed on 5th September 2023.

[15] BSV Blockchain. TAAL brings 4GB blocks to BSV with breakthrough upgrade – What miners should know. https://www.bsvblockchain.org/news/taal-brings-4gb-blocks-to-bsv-with-breakthrough-upgrade-what-miners-should-know. Accessed on 5th September 2023.

[16] Celent. Maximizing collateral advantage: a survey of buy side business and operational strategies. https://www.celent.com/insights/646280961. Accessed on 1st September 2023.

[17] Chainlink. Cross-Chain Interoperability Protocol (CCIP). https://chain.link/cross-chain. Accessed on 5th September 2023.

[18] Chainlink. Decentralized Data Feeds. https://data.chain.link/. Accessed on 5th September 2023.

[19] Chainlink. Types of Blockchain Oracles. https://chain.link/education/blockchain-oracles#types-of-blockchain-oracles. Accessed on 5th September 2023.

[20] Chainlink. Upgradable Smart Contracts: What They Are and How To Deploy Your Own. https://blog.chain.link/upgradable-smart-contracts/. Accessed on 5th September 2023.

[21] Chainlink. Volatility Oracles: Unlocking New DeFi Risk Management Strategies and Derivatives Markets. https://blog.chain.link/volatility-oracles/. Accessed on 5th September 2023.

[22] Circle. USDC. https://www.circle.com/. Accessed on 29th August 2023.

[23] C. D. Clack and C. McGonagle. Smart Derivatives Contracts: the ISDA Master Agreement and the automation of payments and deliveries. *arXiv preprint arXiv:1904.01461*, 2019.

[24] C. D. Clack and G. Vanca. Temporal aspects of smart contracts for financial derivatives. In *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice: 8th International Symposium, ISoLA 2018, Limassol, Cyprus, November 5-9, 2018, Proceedings, Part IV 8*, pages 339–355. Springer, 2018.

[25] Clifford Chance. Custody of cryptoassets: moving towards industry best practice. https://www.cliffordchance.com/content/dam/cliffordchance/briefings/2023/06/custody-of-cryptoassets.pdf. Accessed on 5th September 2023.

[26] CoinDesk. How Data Oracles Connect DeFi and TradFi. https://www.coindesk.com/video/how-data-oracles-connect-defi-and-tradfi/. Accessed on 5th September 2023.

[27] CoinGeek. BSV's low transaction fees make it enterprises' only blockchain option. https://coingeek.com/bsv-low-transaction-fees-make-it-enterprises-only-blockchain-option/. Accessed on 23nd August 2023.

[28] Council of European Union. Document 52021PC0727: Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL amending Regulation (EU) No 600/2014 as regards enhancing market data transparency, removing obstacles to the emergence of a consolidated tape, optimising the trading obligations and prohibiting receiving payments for forwarding client orders. https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=SWD:2021:0346:FIN:EN:PDF. Accessed on 30th August 2023.

[29] CSIRO Research. Token Registry. https://research.csiro.au/blockchainpatterns/general-patterns/blockchain-payment-patterns/token-registry/. Accessed on 5th September 2023.

[30] C. Dannen. *Introducing Ethereum and solidity*, volume 1. Springer, 2017.

[31] Deloitte. Bank for International Settlements. https://www.bis.org/statistics/derstats.htm. Accessed on 26th August 2023.

[32] Deloitte. Collateral management takes centre stage. https://www2.deloitte.com/content/dam/Deloitte/za/Documents/audit/ZA_CollateralManagement_17042014.pdf. Accessed on 26th August 2023.

[33] Depository Trust & Clearing Corporation. DTCC's project ion platform now live in parallel production environment, processing over 100,000 transactions per day on DLT. https://www.dtcc.com/news/2022/august/22/project-ion. Accessed on 29th August 2023.

[34] Depository Trust & Clearing Corporation. Modernizing the U.S. Equity Markets Post-Trade Infrastructure. https://www.dtcc.com/~/media/Files/downloads/Thought-leadership/modernizing-the-u-s-equity-markets-post-trade-infrastructure.pdf. Accessed on 28th August 2023.

[35] Depository Trust & Clearing Corporation. The Changing Face of Derivative Reporting. https://www.dtcc.com/dtcc-connection/articles/2021/april/23/the-changing-face-of-derivatives-reporting. Accessed on 28th August 2023.

[36] Depository Trust & Clearing Corporation. The future of trade repositories: opportunities, challenges and the pursuit of global market transparency. https://www.dtcc.com/-/media/Files/Downloads/Bylined-Articles/EuromoneyBodsonGTR062013dtcccom.pdf. Accessed on 28th August 2023.

[37] Deutsche Bundesbank. How Can Collateral Management Benefit from DLT? https://www.bundesbank.de/resource/blob/823072/4d14afd4b6dbffa94a46ee52f46e99bd/mL/how-can-collateral-management-benefit-from-dlt-data.pdf. Accessed on 28th August 2023.

[38] M. Di Angelo and G. Salzer. Tokens, types, and standards: identification and utilization in Ethereum. In *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 1–10. IEEE, 2020.

[39] O. Dib and K. Toumi. Decentralized identity systems: Architecture, challenges, solutions and future directions. *Annals of Emerging Technologies in Computing (AETiC), Print ISSN*, pages 2516–0281, 2020.

[40] Digital Asset. DAML. https://www.digitalasset.com/. Accessed on 16th August 2023.

[41] Ethereum Foundation. ERC-20 Token Standard. https://ethereum.org/en/developers/docs/standards/tokens/erc-20/. Accessed on 28th August 2023.

[42] Ethereum Foundation. Ethereum Block Size. https://ethereum.org/en/developers/docs/blocks/#block-size. Accessed on 5th September 2023.

[43] Ethereum Foundation. Gas and Fees. https://ethereum.org/en/developers/docs/gas/. Accessed on 5th September 2023.

[44] European Central Bank. What are haircuts. https://www.ecb.europa.eu/ecb/educational/explainers/tell-me-more/html/haircuts.en.html. Accessed on 22nd August 2023.

[45] European Commission. eIDAS Regulation. https://digital-strategy.ec.europa.eu/en/policies/eidas-regulation. Accessed on 4th September 2023.

[46] European Commission. The European Market Infrastructure Regulation (EMIR). https://finance.ec.europa.eu/capital-markets-union-and-financial-markets/financial-markets/post-trade-services/derivatives-emir_en. Accessed on 26th August 2023.

[47] European Investment Bank. EIB issues its first ever digital bond in pound sterling. https://www.eib.org/en/press/all/2023-030-eib-issues-its-first-ever-digital-bond-in-british-pounds. Accessed on 29th August 2023.

[48] Federal Deposit Insurance Corporation. The Electronic Signatures in Global and National Commerce Act (E-Sign Act). https://www.fdic.gov/resources/supervision-and-examinations/consumer-compliance-examination-manual/documents/10/x-3-1.pdf. Accessed on 4th September 2023.

[49] S. Feenan, D. Heller, A. Lipton, M. Morini, R. Ram, R. Sams, T. Swanson, S. Yong, and D. B. Zalles. Decentralized financial market infrastructures: Evolution from intermediated structures to decentralized structures for financial agreements. *The Journal of FinTech*, 1(02):2150002, 2021.

[50] E. Ferran. *The regulatory aftermath of the global financial crisis*. Cambridge University Press, 2012.

[51] A. Ferreira. Emerging regulatory approaches to blockchain based token economy. *The Journal of The British Blockchain Association*, 2020.

[52] Financial Conduct Authority. Transaction reporting fines. https://www.fca.org.uk/markets/transaction-reporting/transaction-reporting-fines. Accessed on 30th August 2023.

[53] Financial Crimes Enforcement Network. The Bank Secrecy Act. https://www.fincen.gov/resources/statutes-and-regulations/bank-secrecy-act. Accessed on 4th September 2023.

[54] FINOS. Common-domain-model. https://github.com/finos/common-domain-model. Accessed on 5th September 2023.

[55] Futures Industry Association. Blockchain and Barclays: A Structured Approach. https://www.fia.org/marketvoice/articles/blockchain-and-barclays-structured-approach. Accessed on 16th August 2023.

[56] Gemini. USD Coin (USDC) — A Stablecoin Pegged to the U.S. Dollar. https://www.gemini.com/cryptopedia/what-is-usdc-stablecoin-circle-crypto#section-what-is-usdc-bridging-the-gap-between-fiat-and-crypto. Accessed on 22nd August 2023.

[57] M. Giancaspro. Is a 'smart contract'really a smart idea? Insights from a legal perspective. *Computer law & security review*, 33(6):825–835, 2017.

[58] J. Göbel and A. E. Krzesinski. Increased block size and Bitcoin blockchain dynamics. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–6. IEEE, 2017.

[59] C. Gourieroux and J. Jasiak. Chapter 10 - value at risk. In Y. AÏT-SAHALIA and L. P. HANSEN, editors, *Handbook of Financial Econometrics: Tools and Techniques*, volume 1 of *Handbooks in Finance*, pages 553–615. North-Holland, San Diego, 2010.

[60] S. Hammoudeh and M. McAleer. Risk management and financial derivatives: An overview. *The North American Journal of Economics and Finance*, 25:109–115, 2013.

[61] E. Hauser. UNIX Time, UTC, and datetime: Jussivity, prolepsis, and incorrigibility in modern timekeeping. *Proceedings of the Association for Information Science and Technology*, 55(1):161–170, 2018.

[62] International Swaps and Derivatives Association. A Blueprint for the Optimal Future State of Collateral Processing. https://www.isda.org/a/eVKDE/Collateral-Infrastructure-V8.pdf. Accessed on 14th August 2023.

[63] International Swaps and Derivatives Association. Accounting for Digital Assets: Key Considerations. https://www.isda.org/a/88VgE/Accounting-for-Digital-Assets-Key-Considerations.pdf. Accessed on 29th August 2023.

[64] International Swaps and Derivatives Association. cdm-typescript. https://central.sonatype.com/artifact/org.finos.cdm/cdm-typescript/5.0.0-dev.14. Accessed on 23nd August 2023.

[65] International Swaps and Derivatives Association. Common Domain Model. https://www.isda.org/2019/10/14/isda-common-domain-model/. Accessed on 16th August 2023.

[66] International Swaps and Derivatives Association. Cross-Border Fragmentation of Global OTC Derivatives: An Empirical Analysis. https://www.isda.org/a/cSiDE/cross-border-fragmentation-an-empirical-analysis.pdf. Accessed on 1st September 2023.

[67] International Swaps and Derivatives Association. Eligible Collateral Representation. https://cdm.docs.rosetta-technology.io/source/eligible-collateral-representation.html. Accessed on 23nd August 2023.

[68] International Swaps and Derivatives Association. ISDA Homepage. https://www.isda.org/. Accessed on 5th September 2023.

[69] International Swaps and Derivatives Association. Key trends in the size and composition of otc derivatives markets in the second half of 2022. https://www.isda.org/a/wdXgE/Key-Trends-in-the-Size-and-Composition-of-OTC-Derivatives-Markets-in-the-Second-Half-of-2022.pdf. Accessed on 1st September 2023.

[70] International Swaps and Derivatives Association. Legal guidelines for smart derivatives contracts: collateral. https://www.isda.org/a/VTkTE/Legal-Guidelines-for-Smart-Derivatives-Contracts-Collateral.pdf. Accessed on 5th September 2023.

[71] International Swaps and Derivatives Association. Margin Approaches - The Relationship between Independent Amount and Initial Margin. https://www.isda.org/a/zR2gE/Margin-Approaches-New-Edited-August-2022.pdf. Accessed on 23nd August 2023.

[72] International Swaps and Derivatives Association. The Future of Derivatives Processing and Market Infrastructure. https://www.isda.org/a/UEKDE/infrastructure-white-paper.pdf. Accessed on 14th August 2023.

[73] International Swaps and Derivatives Association. Transparency in CDS. https://www.isda.org/2023/03/30/transparency-in-cds/. Accessed on 28th August 2023.

[74] International Swaps and Derivatives Association. What is the ISDA Clause Library? https://www.isda.org/a/YiqTE/ISDA-Clause-Library-Factsheet.pdf. Accessed on 16th August 2023.

[75] International Swaps and Derivatives Association. What is the ISDA Create? https://www.isda.org/a/6ITME/ISDA-Create-Fact-sheet-FINAL-1.pdf. Accessed on 16th August 2023.

[76] Japan Securities Clearning Corporation. SPAN. https://www.jpx.co.jp/jscc/en/cash/futures/marginsystem/span.html#:~:text=The%20SPAN%C2%AE%20(Standard%20Portfolio,clearing%20institutions%20around%20the%20world. Accessed on 22nd August 2023.

[77] O. Jayeola. Inefficiencies in trade reporting for over-the-counter derivatives: Is blockchain the solution? *Capital Markets Law Journal*, 15(1):48–69, 2020.

[78] M. S. Keller. Take command: cron: Job scheduler. *Linux Journal*, 1999(65es):15–es, 1999.

[79] Komgo. Komgo — Trade Finance Solutions for the industry. https://www.komgo.io/. Accessed on 5th September 2023.

[80] P. Laurent, T. Chollet, M. Burke, and T. Seers. The tokenization of assets is disrupting the financial industry. Are you ready. *Inside magazine*, 19:62–67, 2018.

[81] T. Lavaur, J. Detchart, J. Lacan, and C. P. Chanel. Modular zk-rollup on-demand. *Journal of Network and Computer Applications*, 217:103678, 2023.

[82] LawSites. Startup Uses 'Internet of Things' to Enable Contracts to Perform Themselves. https://www.lawnext.com/2016/10/startup-uses-internet-things-enable-contracts-perform.html. Accessed on 5th September 2023.

[83] Legal Information Institute - Cornell Law School. U.C.C. - Article 9 - Secured Transactions. https://www.law.cornell.edu/ucc/9. Accessed on 4th September 2023.

[84] Lesavre, Loïc and Varin, Priam and Yaga, Dylan. Blockchain networks: Token design and management overview. Technical report, National Institute of Standards and Technology, 2020.

[85] W. Li and M. He. Comparative analysis of bitcoin, ethereum, and libra. In *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, pages 545–550. IEEE, 2020.

[86] Liink by J.P. Morgan℠. Unlocking the power of collective intelligence. https://www.jpmorgan.com/onyx/liink. Accessed on 16th August 2023.

[87] O. Maureen. Overview: market structure issues in market liquidity. *This publication is available on the BIS website (www. bis. org).*, page 1, 2001.

[88] McKinsey & Company. Tokenization: A digital-asset déjà vu. https://www.mckinsey.com/industries/financial-services/our-insights/tokenization-a-digital-asset-deja-vu#/. Accessed on 28th August 2023.

[89] MeasureThat.net. JS BigInt vs Number Performance Comparison. https://www.measurethat.net/Benchmarks/Show/19007/0/js-bigint-big-number-performance-v2#latest_results_block. Accessed on 5th September 2023.

[90] S. Meisami and W. E. B. I. au2. A Comprehensive Survey of Upgradeable Smart Contract Patterns, 2023.

[91] Microsoft. TypeScript: JavaScript With Syntax For Types. https://www.typescriptlang.org/. Accessed on 23nd August 2023.

[92] M. Morini. From "Blockchain hype" to a real business case for Financial Markets. *Available at SSRN 2760184*, 2016.

[93] Mozilla Developer Network. BigInt. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/BigInt. Accessed on 23nd August 2023.

[94] Mozilla Developer Network. Date. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date. Accessed on 23nd August 2023.

[95] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.

[96] T. D. Oyetoyan, J.-R. Falleri, J. Dietrich, and K. Jezek. Circular dependencies and change-proneness: An empirical study. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 241–250. IEEE, 2015.

[97] A. Pallavicini, D. Perini, and D. Brigo. Funding Valuation Adjustment: a consistent framework including CVA, DVA, collateral,netting rules and re-hypothecation, 2011.

[98] S. Panayides. Arbitrage opportunities and their implications to derivative hedging. *Physica A: Statistical Mechanics and its Applications*, 361(1):289–296, 2006.

[99] Public General Acts of the UK Parliament. The Money Laundering, Terrorist Financing and Transfer of Funds (Information on the Payer) Regulations 2017. https://www.legislation.gov.uk/uksi/2017/692/made. Accessed on 4th September 2023.

[100] R. Quail and J. A. Overdahl. *Financial derivatives*, volume 127. John Wiley & Sons, 2002.

[101] M. Raskin. The law and legality of smart contracts. *Geo. L. Tech. Rev.*, 1:305, 2016.

[102] REGnosys. Rosetta DSL. https://docs.rosetta-technology.io/rosetta/rosetta-dsl/. Accessed on 5th September 2023.

[103] REGnosys. Rosetta Products. https://docs.rosetta-technology.io/rosetta/rosetta-products/. Accessed on 5th September 2023.

[104] River Financial. Pay-to-Public-Key-Hash (P2PKH). https://river.com/learn/terms/p/p2pkh/. Accessed on 6th September 2023.

[105] River Financial. Pay-to-Script-Hash (P2SH). https://river.com/learn/terms/p/p2sh/. Accessed on 6th September 2023.

[106] G. Rühl. *Smart (legal) contracts, or: which (contract) law for smart contracts?* Springer, 2021.

[107] F. Schär. Decentralized finance: On blockchain-and smart contract-based financial markets. *FRB of St. Louis Review*, 2021.

[108] sCrypt. FixedArray. https://docs.scrypt.io/reference/#fixedarray. Accessed on 23nd August 2023.

[109] sCrypt. HashedMap. https://docs.scrypt.io/reference/classes/HashedMap/. Accessed on 23nd August 2023.

[110] sCrypt. HashedSet. https://docs.scrypt.io/reference/classes/HashedSet/. Accessed on 23nd August 2023.

[111] sCrypt. sCrypt Smart Contracts. https://scrypt.io/. Accessed on 22nd August 2023.

[112] sCrypt. User-defined Types. https://docs.scrypt.io/how-to-write-a-contract/#user-defined-types. Accessed on 23nd August 2023.

[113] sCrypt. UTXO-based Layer-1 Tokens on Bitcoin SV. https://xiaohuiliu.medium.com/utxo-based-layer-1-tokens-on-bitcoin-sv-f5e86a74c1e1. Accessed on 28th August 2023.

[114] sCrypt. ZK-Rollups on Bitcoin. https://medium.com/coinmonks/zk-rollups-on-bitcoin-ce35869b940d. Accessed on 22nd August 2023.

[115] Securities and Exchange Commission. Recordkeeping and Reporting Requirements for Security-Based Swap Dealers, Major Security-Based Swap Participants, and Broker-Dealers. https://www.sec.gov/files/rules/final/2019/34-87005.pdf. Accessed on 28th August 2023.

[116] N. Singh and M. Vardhan. Computing optimal block size for blockchain based applications with contradictory objectives. *Procedia Computer Science*, 171:1389–1398, 2020.

[117] D. Skeel. *The new financial deal: understanding the Dodd-Frank Act and its (unintended) consequences*. John Wiley & Sons, 2010.

[118] R. Sobti and G. Geetha. Cryptographic hash functions: a review. *International Journal of Computer Science Issues (IJCSI)*, 9(2):461, 2012.

[119] Society for Worldwide Interbank Financial Telecommunication (SWIFT). Banks and corporates use Swift for OTC derivatives reporting. `https://www.swift.com/news-events/news/banks-and-corporates-use-swift-otc-derivatives-reporting`. Accessed on 30th August 2023.

[120] Stacks. Advantages of Decidability in Smart Contracts. `https://docs.stacks.co/docs/clarity/security/decidable#advantages-of-decidability-in-smart-contracts`. Accessed on 5th September 2023.

[121] N. Szabo. Formalizing and securing relationships on public networks. *First monday*, 1997.

[122] C. Tartan, C. Wright, M. Pettit, and W. Zhang. A Scalable Bitcoin-based Public Key Certificate Management System. In *SECRYPT*, pages 548–559, 2021.

[123] Techopedia. Token Lockup. `https://www.techopedia.com/definition/token-lockup`. Accessed on 5th September 2023.

[124] Tether. USDT. `https://tether.to/en/`. Accessed on 29th August 2023.

[125] Thomson Reuters. Cost of Compliance Report 2020. `https://corporate.thomsonreuters.com/Cost-of-Compliance-2020`. Accessed on 28th August 2023.

[126] Tokenovate Ltd. Tokenization Structures. Accessed on 5th September 2023.

[127] Typescript. Generics. `https://www.typescriptlang.org/docs/handbook/2/generics.html`. Accessed on 23nd August 2023.

[128] WhatsOnChain. BSV Average Transaction Fee. `https://whatsonchain.com/block-stat/avg_fee`. Accessed on 5th September 2023.

[129] WitnessOnChain. WitnessOnChain Service. `https://witnessonchain.com/`. Accessed on 22nd August 2023.

[130] Y. Xia, K. He, F. Wen, and J. Sun. Joint inverted indexing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3416–3423, 2013.

[131] Yahoo Finance. Yahoo Finance. https://uk.finance.yahoo.com/. Accessed on 22nd August 2023.

[132] Yahoo Finance. Yahoo Finance Python Library. https://pypi.org/project/yfinance/. Accessed on 22nd August 2023.

[133] YCharts. Bitcoin Average Transaction Fee. https://ycharts.com/indicators/bitcoin_average_transaction_fee#:~:text=Basic%20Info,0.87%25%20from%20one%20year%20ago. Accessed on 5th September 2023.

[134] YCharts. Ethereum Average Transaction Fee. https://ycharts.com/indicators/ethereum_average_transaction_fee. Accessed on 5th September 2023.

[135] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran. An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems*, 105:475–491, 2020.

[136] A. Zohar. Bitcoin: under the hood. *Communications of the ACM*, 58(9):104–113, 2015.

Appendix A

Code Listings

A.1 Relationship Smart Contract

Listing A.1: Relationship Smart Contract

```
1
2  import { assert, ByteString, HashedSet, method, prop, PubKeyHash,
       SmartContract, hash256 } from 'scrypt-ts'
3
4  /*
5      CDM Type Definitions must be translated to sCrypt types (i.e.
           bigint, ByteString, etc ...)
6
7      A Collateral Relationship contains:
8      - An independent amount (see type definition below)
9      - A list of collateral portoflios (implemented as an HashedSet)
10 */
11
12 export type IndependentAmount = {
13     amount: bigint,
14     buyer: PubKeyHash,
15     seller: PubKeyHash
16 }
17
18 export class Relationship extends SmartContract {
19     @prop()
20     independentAmount: IndependentAmount
21
22     @prop(true)
23     portfolios: HashedSet<ByteString>
24
25     constructor(independentAmount: IndependentAmount, portfolios:
           HashedSet<ByteString>) {
```

```
26          super(...arguments)
27          this.independentAmount = independentAmount
28          this.portfolios = portfolios
29      }
30
31      @method()
32      public addPortfolio(newPortfolio: ByteString) {
33          this.portfolios.add(newPortfolio)
34          const outputs: ByteString = this.buildStateOutput(
                this.ctx.utxo.value) + this.buildChangeOutput()
35          this.debug.diffOutputs(outputs)
36          assert(this.ctx.hashOutputs === hash256(outputs), "check
                hashOutputs failed");
37      }
38
39      @method()
40      public updatePortfolioAddress(oldPortfolio: ByteString,
            newPortfolio: ByteString) {
41          this.portfolios.delete(oldPortfolio)
42          this.portfolios.add(newPortfolio)
43          const outputs: ByteString = this.buildStateOutput(
                this.ctx.utxo.value) + this.buildChangeOutput()
44          this.debug.diffOutputs(outputs)
45          assert(this.ctx.hashOutputs === hash256(outputs), "check
                hashOutputs failed");
46      }
47
48  }
```

A.2 Portfolio Smart Contract

Listing A.2: Portfolio Smart Contract

```
1   import { SmartContract, ByteString, HashedSet, assert, prop, method,
        hash256 } from 'scrypt-ts'
2
3   /*
4       CDM Type Definitions exported as basic sCrypt types (i.e. bigint,
            ByteString, etc.)
5
6       A portfolio is a collection of positions.
7       A portfoio must have a reference to an existing collateral
            relationship.
8       Properties:
9           - relationship: TX containg UTXO of corresponding to
                relationship contract
10  */
```

```
11
12
13 export class Portfolio extends SmartContract {
14     @prop()
15     relationship: ByteString
16
17     @prop(true)
18     positions: HashedSet<ByteString>
19
20     constructor(relationship: ByteString, positions:
           HashedSet<ByteString>) {
21         super(...arguments)
22         this.relationship = relationship
23         this.positions = positions
24
25     }
26
27     @method()
28     public addPosition(newPosition: ByteString) {
29         this.positions.add(newPosition)
30         const outputs: ByteString = this.buildStateOutput(
               this.ctx.utxo.value) + this.buildChangeOutput()
31         this.debug.diffOutputs(outputs)
32         assert(this.ctx.hashOutputs === hash256(outputs), "check
               hashOutputs failed");
33     }
34
35 }
```

A.3 Position Smart Contract

Listing A.3: Position Smart Contract

```
1  import { SmartContract, ByteString, Utils,  assert, prop, method, slice
       , hash256 } from 'scrypt-ts'
2  import { RabinSig, RabinPubKey, RabinVerifierWOC } from 'scrypt-ts-lib'
3
4  /*
5      CDM Type Definitions exported as basic sCrypt types (i.e. bigint,
           ByteString, etc.)
6
7      A Collateral Position contains information regarding a specific
           piece of collateral posted.
8
9      From the CDM:
10
11         OMITTED HERE:
```

```
12          - collateralPositionStatus: The collateral positions settlement
                status (OMITTED here as settlment is instantaneous)
13          - priceQuantity [ ]: An array of exchanges between parties of
                one quantity (Quantity) against another (Price) (OMITTED as
                transfer of value happens in satoshis)
14          - tradeReference: reference to an external contract in case the
                product is contractual (OMITTED here for simplicity)
15
16          NOT OMITTED:
17          - treatment: Any treament applied to collateral
18          - cashBalance: This is simply the value of the collateral at a
                specific point in time
19                          * This is represented here by the value of the
                                UTXO at time t after USD -> Satoshis
                                conversion *
20          - product: The product underlying the position (here only the
                Commodity product is used as it is the closest to VCCs)
21
22      From BSV-specific architecture:
23          - portfolio: address of the portfolio of which the position is
                part
24
25  */
26
27  export type ExchangeRate = {
28      timestamp: bigint
29      price: bigint
30      symbol: ByteString
31  }
32
33
34  export class Position extends SmartContract {
35      // Address of the Portfolio to which this position belongs
36      @prop()
37      portfolio: ByteString
38
39      // Oracles (for USDC-SAT) exchange Rabin public key.
40      @prop()
41      oraclePubKey: RabinPubKey
42
43      // The product underlying the position.
44      @prop()
45      productType: ByteString
46
47      // The identifier of the product underlying the position.
48      @prop()
49      productIdentifier: ByteString
50
```

```
51      // The quantity of the product underlying the position.
52      @prop()
53      productQuantity: bigint
54
55      // The current value of the collateral
56      @prop(true)
57      currentCollateralValue: bigint
58
59
60      constructor(portfolio: ByteString, oraclePubKey: RabinPubKey,
            productType: ByteString, productIdentifier: ByteString,
            productQuantity: bigint, currentCollateralValue: bigint) {
61          super(...arguments)
62          this.portfolio = portfolio
63          this.oraclePubKey = oraclePubKey
64          this.productType = productType
65          this.productIdentifier = productIdentifier
66          this.productQuantity = productQuantity
67          this.currentCollateralValue = currentCollateralValue
68      }
69
70      @method()
71      public moveCollateralToP2PKH(delta: bigint) {
72          this.setCollateralValue(this.currentCollateralValue - delta)
73          const contractOutput = this.buildStateOutput(
                this.ctx.utxo.value - delta)
74          const P2PKHOutput = this.buildChangeOutput() // Uses signer's
                address by default (in this case the custodian)
75          const outputs: ByteString = contractOutput + P2PKHOutput
76          this.debug.diffOutputs(outputs)
77          assert(hash256(outputs) == this.ctx.hashOutputs, 'hashOutputs
                check failed')
78      }
79
80      @method()
81      public moveCollateralToPosition(delta: bigint) {
82          this.setCollateralValue(this.currentCollateralValue + delta)
83          const contractOutput = this.buildStateOutput(
                this.ctx.utxo.value + delta)
84          const P2PKHOutput = this.buildChangeOutput() // Uses signer's
                address by default (in this case the custodian)
85          const outputs: ByteString = contractOutput + P2PKHOutput
86          this.debug.diffOutputs(outputs)
87          assert(hash256(outputs) == this.ctx.hashOutputs, 'hashOutputs
                check failed')
88      }
89
90      @method()
```

```
91      setCollateralValue(value: bigint): void {
92          this.currentCollateralValue = value
93      }
94
95      @method()
96      public updateCollateralValue(newValue: bigint) {
97          const amount: bigint = this.ctx.utxo.value
98          this.setCollateralValue(newValue)
99          const outputs: ByteString = this.buildStateOutput(amount) +
                this.buildChangeOutput()
100         this.debug.diffOutputs(outputs)
101         assert(this.ctx.hashOutputs == hash256(outputs), 'hashOutputs
                mismatch')
102         assert(true)
103     }
104
105     // Parses signed message from the oracle.
106     @method()
107     static parseExchangeRate(msg: ByteString): ExchangeRate {
108         // 4 bytes timestamp (LE) + 8 bytes rate (LE) + 1 byte decimal
                + 16 bytes symbol
109         return {
110             timestamp: Utils.fromLEUnsigned(slice(msg, 0n, 4n)),
111             price: Utils.fromLEUnsigned(slice(msg, 4n, 12n)),
112             symbol: slice(msg, 13n, 29n),
113         }
114     }
115
116     @method()
117     public VerifyOracleData(msg: ByteString, sig: RabinSig) {
118         // Verify oracle signature.
119         assert(
120             RabinVerifierWOC.verifySig(msg, sig, this.oraclePubKey),
121             'Oracle sig verify failed.'
122         )
123
124         console.log('Oracle Signature Verified')
125
126         // Decode data.
127         const exchangeRate = Position.parseExchangeRate(msg)
128
129         console.log('Parsed Oracle Data', exchangeRate)
130         assert(true)
131     }
132
133
134 }
```

A.4 Off-chain data (Oracles & Yahoo Finance)

Listing A.4: Fetching USDC-BSV exchange rate via the WitnessOnChain Oracle Service

```
1
2  import { Position } from '../../contracts/Position'
3  import { byteString2Int, toByteString, ByteString, MethodCallOptions,
       bsv } from 'scrypt-ts'
4  import { getDummySigner, getDummyUTXO } from '../../utils/helper'
5  import { RabinPubKey, RabinSig } from 'scrypt-ts-lib'
6  import axios from 'axios'
7
8  export async function getBSV_USDC_ExchangeRate(): Promise<{ rate:
       number; response: any }> {
9    const symbol = 'BSV_USDC'; // Set the trading pair
10   const url = `https://witnessonchain.com/v1/rates/${symbol}`; //
       Create the endpoint URL
11
12   try {
13     const response = await axios.get(url); // Make the GET request
14
15     // Check for a successful response
16     if (response.status === 200 && response.data && response.data.rate)
         {
17       return {
18           rate: response.data.rate, // return the exchange rate
19           response: response.data, // return the entire response
20         };
21     } else {
22       throw new Error('Could not fetch the exchange rate'); // If
           unsuccessful, throw an error
23     }
24   } catch (error) {
25     console.error(error); // Log any errors
26     throw error; // And rethrow them
27   }
28 }
29
30 getBSV_USDC_ExchangeRate()
31   .then(({ rate, response }) => {
32     console.log(`The BSV_USDC exchange rate is: ${rate}`);
33     console.log(`The entire response is: `, response);
34   })
35   .catch(error => console.error(`Failed to get the exchange rate: ${
       error.message}`));
```

Listing A.5: Fetching the price of crude oil via the Yahoo Finance API

```

import axios from 'axios';

export async function getCrudeOilPrice(): Promise<number> {
  try {
    const ticker = 'CL=F'; // Ticker symbol for crude oil futures
    const url = `https://query1.finance.yahoo.com/v8/finance/chart/${ticker}`;

    const response = await axios.get(url);
    const data = response.data;

    if (
      data &&
      data.chart &&
      data.chart.result &&
      data.chart.result.length > 0 &&
      data.chart.result[0].indicators &&
      data.chart.result[0].indicators.quote &&
      data.chart.result[0].indicators.quote.length > 0 &&
      data.chart.result[0].indicators.quote[0].close &&
      data.chart.result[0].indicators.quote[0].close.length > 0
    ) {
      const latestPrice = data.chart.result[0].indicators.quote[0].close.slice(-1)[0];
      return latestPrice;
    } else {
      throw new Error('Failed to parse crude oil price from the response.');
    }
  } catch (error) {
    console.error('Error fetching crude oil price:', error.message);
    throw error;
  }
}

(async () => {
  try {
    const crudeOilPrice = await getCrudeOilPrice();
    console.log(`Current crude oil price: $${crudeOilPrice.toFixed(2)} USD/barrel`);
  } catch (error) {
    // Handle error if necessary
  }
})();
```

A.5 Updating Collateral

Listing A.6: Moving funds beteween Collateral Position and Parties' Balances based on updated collateral valuation

```
/*
    1) Fetch off-chain data and feed it into the contract
    2) Make a contract call to computeCollateralValue()
    3) Create spending TX
        a) If newValue < currentValue, then send the delta to a P2PKH
           for party
        b) If newValue > currentValue, then send the delta Position

*/

import { getBSV_USDC_ExchangeRate } from './getUSDC_BSV_rate'
import { getCrudeOilPrice } from './getCrudePrice'
import { setupSigners } from '../../utils/setupSigners'
import { Position } from '../../contracts/Position'
import { computeNewCollateralValue } from './computeNewCollateralValue'
import { MethodCallOptions } from 'scrypt-ts'

let positionTxId = '8
    da45c34fa1e5646bd1c0cf381d50e3ac1f1844725f000f3960d21f390614b41'

async function crudePrice() {
  return await getCrudeOilPrice();
}

async function exchangeRate() {
    return await getBSV_USDC_ExchangeRate();
}

async function getCurrentCollateralValue(positionTxId): Promise<bigint>
    {
    console.log('Fetching Current Collateral Value')
    const { signerA, signerB, signerCustodian } = await setupSigners();
    Position.compile();
    const positionTx = await signerA.connectedProvider.getTransaction(
        positionTxId);
    const instance = await Position.fromTx(positionTx, 0);
    const currentCollateralValue = await
        instance.currentCollateralValue;
    return currentCollateralValue;
  }

async function calculateNewCollateralValue(): Promise<bigint> {
    const mtm = await crudePrice();
```

```
39      const getUSDC_BSV_rate = await exchangeRate();
40      // Now you can use these values to compute the new collateral value
41      // Remove the *10 below, it's just for testing purposes
42      const newCollateralValue = computeNewCollateralValue(mtm,
            getUSDC_BSV_rate.rate * 10);
43      console.log('New Collateral Value:', BigInt(Math.trunc(
            newCollateralValue)));
44      return BigInt(Math.trunc(newCollateralValue));
45  }
46
47  // Async function to call updateCollateralValue of the Position
        contract with the return value of caluclaateNewCollateralValue()
48  async function updateCollateralValue() {
49      console.log('Updating Collateral Value')
50      const { signerA, signerB, signerCustodian } = await setupSigners();
51      Position.compile();
52      const positionTx = await signerA.connectedProvider.getTransaction(
            positionTxId);
53      const instance = await Position.fromTx(positionTx, 0);
54      await instance.connect(signerA);
55      const newCollateralValue = await calculateNewCollateralValue();
56
57      const current = instance
58      const nextInstance = current.next()
59      nextInstance.setCollateralValue(newCollateralValue)
60
61      const { tx: callTx } = await current.methods.updateCollateralValue(
62          newCollateralValue,
63          {
64              fromUTXO: current.utxo,
65              next: {
66                  instance: nextInstance,
67                  balance: current.balance
68              }
69          } as MethodCallOptions<Position>
70      )
71      console.log('Contract Method Called')
72      console.log('Collateral Value Updated');
73      return callTx.id
74
75  }
76
77  async function moveCollateralToP2PKH(newCollateralValue: bigint, delta:
        bigint) {
78      const { signerA, signerB, signerCustodian } = await setupSigners();
79      Position.compile();
80      const positionTx = await signerA.connectedProvider.getTransaction(
            positionTxId);
```

```
    const instance = await Position.fromTx(positionTx, 0);
    await instance.connect(signerA);

    const current = instance
    const nextInstance = current.next()
    nextInstance.setCollateralValue(newCollateralValue)

    const { tx: callTx } = await current.methods.moveCollateralToP2PKH(
        delta,
        {
            fromUTXO: current.utxo,
            next: {
                instance: nextInstance,
                balance: current.balance - Number(delta)
            }
        } as MethodCallOptions<Position>
    )
    console.log("Moved funds from Position to P2PKH at ", callTx.id)
    return callTx.id
}

async function moveCollateralToPosition(newCollateralValue: bigint,
    delta: bigint) {
    const { signerA, signerB, signerCustodian } = await setupSigners();
    Position.compile();
    const positionTx = await signerA.connectedProvider.getTransaction(
        positionTxId);
    const instance = await Position.fromTx(positionTx, 0);
    await instance.connect(signerA);

    const current = instance
    const nextInstance = current.next()
    nextInstance.setCollateralValue(newCollateralValue)

    const { tx: callTx } = await
        current.methods.moveCollateralToPosition(
        delta,
        {
            fromUTXO: current.utxo,
            next: {
                instance: nextInstance,
                balance: current.balance + Number(delta)
            }
        } as MethodCallOptions<Position>
    )
    console.log("Moved funds from P2PKH to Position at ", callTx.id)
    return callTx.id
}
```

```
126
127
128 const run = async () => {
129     console.log('Before update:')
130     const currentCollateralValue = await getCurrentCollateralValue(
            positionTxId)
131     console.log('Current Collateral Value: ', currentCollateralValue)
132
133     // Compute new collateral value
134     const newCollateralValue = await calculateNewCollateralValue()
135
136     // Move funds from position to P2PKH for custodian
137     if (newCollateralValue < currentCollateralValue) {
138         const delta = currentCollateralValue - newCollateralValue
139         console.log("delta: ", delta)
140         positionTxId = await moveCollateralToP2PKH(newCollateralValue,
                delta)
141     }
142
143     // Move funds from P2PKH for custodian to position
144     else {
145         const delta = newCollateralValue - currentCollateralValue
146         console.log("delta: ", delta)
147         positionTxId = await moveCollateralToPosition(
                newCollateralValue, delta)
148     }
149
150     // positionTxId = await updateCollateralValue();
151     console.log('After update:')
152     await getCurrentCollateralValue(positionTxId).then((value) =>
            console.log('Collateral Value Retrieved after Update:', value))
            ;
153 }
154
155 run();
```

A.6 Common Domain Model

We omit the full Common Domain Model as it would be too large to include in a code listing in this document. We refer to the official Rosetta Workspace for the full type system (note that using this product requires creating a free account), presented both in textual form and through an easy-to-use graphical user interface, as well as the official Typescript distribution of the CDM to download the source code.