UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

Secure, Distributed Financial Exchanges: Design and Implementation

Chan Nam Ngo

Advisor

Prof. **Fabio Massacci**

Università degli Studi di Trento

External Reviewers

Prof. **Ian Goldberg**

University of Waterloo

Prof. **Ivan Visconti**

Università degli Studi di Salerno

Thesis Commitee

Prof. **Ian Goldberg**

University of Waterloo

Prof. **Florian Kerschbaum**

University of Waterloo

Prof. **Ivan Visconti**

Università degli Studi di Salerno

October 2019

Abstract

Blockchains and Byzantine Fault Tolerance form the basis of decentralized currencies and ledgers, such as Bitcoin*,* Ripple*,* ZeroCash*, and* Ethereum*. Several studies have focused on the currency aspects (e.g. authenticity, integrity, anonymity, and independence from central banks). In this thesis, we start by exploring to understand the security challenges and practical solutions for building simple payment networks. Then, we leverage such understanding in identifying the security challenges of more advanced and complex systems, in particular Futures Exchanges. The decentralization of a Futures Exchange poses new security challenges: i) the interplay between the security and economic viability, i.e. using the Price Discrimination Attack one can strategically force a trader out of the market when the trader's anonymity is broken; ii) the non-monotonic security behavior of an Exchange, i.e. an honest action may invalidate security evidence; and iii) the proportional burden requirement in the presence of high-frequency participants. Our goal is to enucleate the non-trivial design principles to resolve these challenges for building secure and distributed financial exchanges. We demonstrate the application of the distilled design principles by building a cryptographic reference for a futures exchange called* FuturesMEX*. We also simulate the performance of a* FuturesMEX *Proof-of-Concept with the Lean Hog market data obtained from the Thomson Reuters Ticks History DB. The results show that the obtained protocol is feasible for a low-frequency market such as Lean Hog. Furthermore, we investigate an extension of public markets, i.e. dark pools (private markets), in which the order book information is conditionally visible to some (financially) suitable parties. We propose a new cryptographic scheme called Witness Key Agreement that makes dark trading possible by probing prices and volumes based on committed financial information Finally, we evaluate the theoretical and practical performance of the new scheme; using a simulation of the dark pool data collected from the aggressive Bloomberg Tradebook, we obtain positive results.*

Keywords security challenges; design principles; distributed ledgers; zero-knowledge; financial exchanges; dark trading

Acknowledgment

First and foremost I would like to thank Prof. Fabio Massacci for being such a great supervisor and colleague over these past four years.

I am grateful to my co-authors of Chapter 4 of my thesis (Jing Nie and Julian Williams contributed to the business domain knowledge, i.e. the description of the Futures Market, the Price Discrimination Attack and the TRTH Lean Hog data set; Fabio Massacci and Daniele Venturi set the general framework and direction of the protocol construction and the security proof; I coordinated the co-authors in all aspects and contributed to the protocol design, optimization and implementation); and Prof. Florian Kerschbaum, my advisor during my visit to the University of Waterloo, without whom Chapter 5 of my thesis would not be possible (I discussed thoroughly with Florian Kerschbaum to construct the Witness Key Agreement scheme; then Fabio Massacci helped me adapt the scheme into the dark trading scenario; finally Julian Williams contributed to the business domain, i.e. the description of the Dark Pool and the Bloomberg Tradebook data set). I am very glad that I had the chance to work with you and learn from your expertise.

I would like to thank Prof. Ian Goldberg and Prof. Ivan Visconti for agreeing to be my thesis's external reviewers, without whom my thesis would not arrive in its better form today (Ian Goldberg pointed out many weaknesses, both technical and presentation, in Chapter 2, 3 and 5 and helped me clarify unclear points in Chapter 4 and 5; Ivan Visconti suggested some clarifications, especially to put a comparison between the Witness Key Agreement scheme and the Hash Proof System in Chapter 5).

I would like to thank the Department of Information Engineering and Computer Science of the University of Trento for offering me the chance to pursue my PhD.

Thank you to my friends (especially Andrea Mariello) and colleagues in Trento and Waterloo. Without you, working in Trento and Waterloo would not be what it was. Especially I want to thank Pierantonia Sterlini, Irma Carolina Fung Escalante, and Andrea Stenico for your great support in administration.

I am grateful to Mai Trang, my dearest wife, as well as my parents, brothers and other relatives for their support and encouragements during these past four years.

Contents

1 Introduction **1**

1.1 Motivation and objectives 1

1.1.1 Understanding the Security Challenges and Solutions for Distributed Financial Exchanges 1

1.1.2 Cryptographic Implementations of Secure, Distributed Financial Exchanges 3

1.2 Research Contribution 5

1.3 Structure of the Thesis 6

2 Payment Transaction Networks **9**

2.1 A Primer on Payment Transaction Networks 9

2.1.1 Traditional Digital Payment Transaction Networks 11

2.1.2 Crypto-based Payment Transaction Networks 11

2.2 Security Challenges and Solutions for Payment Transaction Networks . . . 16

2.2.1 Integrity and Availability 17

2.2.2 Confidentiality and Anonymity 20

2.3 Beyond Payment Transaction Networks 22

2.4 Summary 24

3 Security Challenges and Design Principles for Distributed Financial Exchanges **25**

3.1 An Introduction to Futures Markets 25

3.1.1 Informal Security Requirements 29

3.2 Security Challenges and Design Principles 30

3.2.1 Protect against Discrimination 30

3.2.2 Ensure Responsible behavior 33

3.2.3 Manage Non-Monotonic (Honest) Evolution 34

3.2.4 Account for a Large Number of Parties 36

3.2.5 Guarantee Proportional Burden 37

3.2.6 Ensure Drop-Out Tolerance . . . 38
3.3 Summary . . . 39

4 FuturesMEX: A Secure, Distributed Futures Market Exchange 41
4.1 Formal Futures Market Definition . . . 41
4.2 The Ideal Reactive Functionality . . . 44
4.3 Assumptions and Crypto Building Blocks Overview . . . 49
4.4 Solution Overview . . . 51
4.5 FuturesMEX Crypto Building Blocks . . . 54
4.5.1 Futures Market Relations . . . 56
4.6 Protocol Construction . . . 59
4.7 Security Analysis . . . 67
4.7.1 Proof sketch . . . 67
4.7.2 Security Proof . . . 68
4.8 Protocol Optimization . . . 74
4.8.1 Optimized Building Blocks . . . 75
4.9 Beyond Security-With-Abort . . . 76
4.10 Implementation . . . 77
4.10.1 Evaluation . . . 78
4.11 Related Work . . . 84
4.12 Summary . . . 85

5 Dark Financial Intermediation with Witness Key Agreement 87
5.1 Possible Solutions for Dark Financial Intermediation . . . 87
5.2 Dark Financial Intermediation with Witness Key Agreement . . . 90
5.2.1 Limitations of our WKA construction . . . 91
5.3 Recap of Technical Background . . . 91
5.3.1 Summary of NIZK, SoK, AKE, WE and MPC . . . 91
5.3.2 Formal Definitions for NILP and QAP . . . 94
5.4 Witness Key Agreement . . . 100
5.4.1 Witness Key Agreement Definition . . . 100
5.4.2 Split Designated-Verifier Non-Interactive Linear Proof . . . 101
5.4.3 Construction of WKA . . . 104
5.4.4 Security Analysis . . . 106
5.5 Witness Key Agreement for Quadratic Arithmetic Program . . . 108
5.6 Instantiation . . . 108
5.7 Performance Evaluation . . . 110
5.7.1 Theoretical Performance Evaluation . . . 111

5.7.2 Practical Performance Evaluation 112
5.8 Summary 114

6 Conclusion **115**
6.1 Research Contribution 115
6.2 Future Work 117

Bibliography **119**

List of Tables

2.1 Loss of value . . . 17
2.2 Loss of value countermeasures . . . 21
2.3 Information disclosure countermeasure examples . . . 21

3.1 Key Compositions and Characteristics of Futures Market . . . 26
3.2 Samples of Market Activity . . . 28
3.3 Informal Security Requirements . . . 29

4.1 Market Indicators for the current round t of the Futures Market . . . 43
4.2 Value $\mathsf{cash}(v)$ to liquidate an inventory of volume v . . . 43
4.3 Formal Security Requirements . . . 44
4.4 FuturesMEX Notation . . . 51
4.5 Futures Market Relations . . . 52
4.6 Merkle Tree's supported operations . . . 53
4.7 Standard *NP* relations for commitment schemes . . . 56
4.8 zk-SNARK Simple and Opt. Circuits Performance . . . 79
4.9 MPC Performance . . . 79
4.10 Runtime of Individual Market Operations with Merkle Tree of depth 10 . . . 81
4.11 Runtime of Individual Market Operations with Merkle Tree of depth higher than 10 . . . 81

5.1 Dark Pool Example Relations . . . 89
5.2 Theoretical Performance evaluation (non-deterministic case) . . . 112
5.3 Specific circuit evaluation . . . 112
5.4 Market data samples for low and high days of Bloomberg Tradebook . . . 113

List of Figures

2.1 Banking models 10
2.2 The *simplified* Bitcoin Transactions 13
2.3 The Ripple Network Components and Gateways with the External World . 15
2.4 The Systemic Risk of Blockchains: Forked and Orphaned Transactions . . 19

3.1 Order Book 27
3.2 The Order Pathway 28
3.3 Forcing Alice out of the market 32

4.1 Market State Transition Diagram 45
4.2 The operations of the ideal functionality $\mathcal{F}_{\mathsf{CFM}}$ for posting, cancelling and marking to market 46
4.3 The operations of the ideal functionality $\mathcal{F}_{\mathsf{CFM}}$ for margin settlement and order fulfillment 47
4.4 Inventory flags state transition diagram 53
4.5 Sub-protocols Π_{put} and Π_{get} 55
4.6 Hybrid Implementation of the Ideal Functionality 59
4.7 Sub-protocols Π_{valid}, Π_{net}, Π_{match}, and Π_{backup} 61
4.8 The **Initialize** phase of protocol Π_{DFM} 63
4.9 The **Post Order** and **Cancel Order** phase of protocol Π_{DFM} 64
4.10 The **Margin Settlement** phase of protocol Π_{DFM} 65
4.11 The **Mark To Market** phase of protocol Π_{DFM} 66
4.12 An evaluation of the memory requirements of different MPC functionalities over time 80
4.13 Crypto Protocol Evaluation on Q1 of Lean-Hog 84
4.14 Total Burden of Computation by Retail Traders 85

5.1 Non-interactive linear proof [102] 96
5.2 NILP for QAP [102]) 99
5.3 Witness key agreement definition 101

5.4 Security of Witness Key Agreement Scheme 102
5.5 Split designated-verifier NILP 103
5.6 Split DV NILP for QAP 104
5.7 Construction of Witness Key Agreement 105
5.8 Witness key agreement for QAP 109
5.9 The Scheme 3 of the Paillier cryptosystem [148] 110
5.10 WKA Evaluation on Bloomberg Tradebook 113

Chapter 1

Introduction

In this thesis, we seek to understand the security challenges and develop design principles for building distributed financial intermediation. We start by investigating the existing (both traditional and crypto-based) simple financial intermediaries, i.e. Payment Transaction Networks (PTNs). Then, we explore the general financial intermediation such as financial exchanges owing to the interesting aspects of these concepts. In particular, the thesis chooses Futures Exchange as an example because it is one of the landmark financial institutions. Following the distilled design principles, this thesis constructs a concrete cryptographic and distributed implementation securely replicated the ideal functionality of a centralized Futures Market Exchange (with rigorous security proof). This thesis also investigates distributed dark trading (private markets) where the previously public information is now conditionally visible.

1.1 Motivation and objectives

1.1.1 Understanding the Security Challenges and Solutions for Distributed Financial Exchanges

A traditional PTNs classification distinguishes between token-based and account-based systems [3]. A good example of the token-based systems is E-cash [53], where parties exchange electronic tokens that represent value, whereas Visa [178] and MasterCard [135] exemplify account-based systems in which money is stored only as numbers in bank accounts.

Token-based systems, compared with the account-based ones, offer potentially stronger anonymity as it is hard to identify the payer of the token in a transaction. For example, the original owner of an E-cash coin is untraceable [56] unless the coin is spent twice in off-line

transactions.[1] Many previous works focused on the security aspects of the token-based payment systems, which include authenticity, integrity, anonymity, etc. [44, 53, 56, 163]. However, operational electronic token-based systems are centralized as both payer and payee must open an account in the *same* E-cash *bank* to initiate a funds transfer.

New decentralized payment systems, e.g. Bitcoin [144], ZeroCoin [139], ZeroCash [164], Ethereum [77], Ripple [157] or RSCoin [94], have been widely adopted as the systems are not simply managed by *a single financial institution* but *a distributed set of nodes world-wide*.[2] The decentralization of these payment networks relies on the consensus protocols, such as Proof-of-Work [80, 144, 167] or the practical Byzantine fault tolerant algorithm [49].

Research Question 1. ***(RQ1)*** *What are the basic security challenges and possible solutions for building a secure, distributed payment network?*

Compared with simple payment, general financial intermediation such as futures trading is more complex. A futures contract is a standardized agreement between two parties to buy or sell an underlying asset, at a price agreed upon today with the settlement at some future date [169]. The parties "promise" to buy or sell an underlying asset, and this "promise" can itself be traded. Such trading is carried out in a double auction market operated by a centralized clearinghouse called Futures Exchange [5], such as the Chicago Mercantile Exchange (CME) [59].

Futures trading poses advanced security challenges compared with simple payment, and it is non-trivial to address these challenges. In a preliminary observation, the first challenge is the *interplay between security and economic viability* [133]. Whereas, integrity is obviously needed for payments (see the Ethereum DAO mishaps [63] [3]), confidentiality seemed less critical for exchanges [57]; one can trace all transactions to a Bitcoin's ID using public information in the blockchain, yet this hardly stopped Bitcoin from thriving [164]. In a futures market, disclosing a trader's account allows attacks based on Price Discrimination. A market vulnerable to those *price discrimination attacks* will collapse as safe trading is not possible in such market and traders will just flock away.

Another challenge, w.r.t other crypto applications such as auctions [162], is the simultaneous needs to i) make publicly available offers by all parties, ii) withhold information on who made the quote and iii) trace *honest* consequences of an anonymous public action

[1] In online E-cash; the double-spent coin will be rejected immediately.

[2] Bitcoin, ZeroCash and Ripple are operational, whereas RSCoin is still a laboratory experiment.

[3] In the DAO (Decentralized Autonomous Organization) framework 1.0, there existed a generic vulnerability called "recursive call vulnerability", which allowed a malicious recipient contract to call *executeProposal()* recursively for money to be *transferred multiple times out of the DAO*. This vulnerability affects the reward account mechanism of the existing DAOs based on the framework 1.0. The issue provoked the DAO hack incident, in which the attacker managed to drain more than 3.6 million ETH from the DAO on 18th June 2016, by a race attack using a recursive call in the DAO's implementation. Such incident was clearly an integrity violation. In economic term, the DAO suffered from money pumps as it proceeded with account clearance prior to ledger update.

to the responsible actor. The prototypical example is posting a public, anonymous buy order while accruing the revenues from the sale (without even the seller knowing the actual buyer and vice-versa). The Futures Exchange must also guarantee that iv) actors do not offer beyond their means, an issue related to double spending [11], double-voting [24], or groundless reputation rating [186]. E-voting offers traceability for one's own vote but not the ensemble of agents. Applications of e-cash and privacy-preserving reputation systems guarantee anonymity for honest actions but traceability only in case of malfeasance and dishonest behavior.

Popular security protocols in the past were authentication protocols. A protocol could have various degrees of complexity (e.g. Kerberos [140] vs TLS [72] vs IKE [105]), but essentially two parties wanted to authenticate each other, possibly with the help of a trusted third party. The common design principles for such authentication protocols were: each party's main goal was to receive the other party's security evidence and they participate equally in the protocol.[4] With an emergence of financial protocols [139, 144, 164] and practical deployment of secure multi-party-computation (MPC) [38], the *number of parties* participating in the protocol has massively increased. The parties do not talk to each other, but to an ensemble of agents. Furthermore, some parties might be more active than others in communication. Yet, they potentially share the same burden in generic MPC computational effort [156]. This leads to some interesting questions: *Is security evidence always monotonic as the number of honest parties increases?*; *What type of failures can materialize if this is not the case? If some application requires non-monotonic security, are there design implications if some parties are more active than others?*

Research Question 2. ***(RQ2)*** *What are the advanced security challenges for building distributed financial exchanges and how can we address them through new design principles?*

1.1.2 Cryptographic Implementations of Secure, Distributed Financial Exchanges

To fully decentralize a futures exchange, the main technical difficulty we need to face is the fact that futures markets are fully stateful systems where the functionality changes the internal state at each round due to a valid move performed by an agent that updates the public information and her own private information. The global constraint is that an agent's legitimate move can unpredictably make another agent's state invalid owing to a

[4]Obviously the server would have had more load than a client, but this only happens because the server participates in several authentications with several clients at once.

change in the public information. As a whole, the market must transit to a new state where the legitimate move is accepted, and the invalid state is fixed.

This intrinsic non-monotonic feature prevents the protocol designers from improving the communication complexity by replacing interactive MPC [156] steps with independent non-interactive proofs [21]. While the satisfaction of individual constraints could be solved using a standard "commit-and-prove" approach between the concerned individuals [164], this is inapplicable for the global constraints. The alternative is to implement the entire functionality via general-purpose MPC [156]. However, this solution is unacceptable given a large variance in trading activity. For example, some traders only make a few large orders, others make several trades every millisecond [130]. This leads to an efficiency requirement, informally stating, each computation should be mainly a burden on the trader who benefit from it, which cannot be met by a naïve MPC implementation.

Research Question 3. ***(RQ3)*** *How can we apply the new design principles to build a viable secure, distributed futures exchange?*

In many public markets (e.g. the CME), prices and volumes, which allow buyers and sellers to buy and sell assets, are published in the limit order book [169]. In these markets, information on individual traders' positions and quotes however must be private (see Chapter 3 on the importance of a trader's confidentiality of financial protocols). Other financial arrangements, such as over-the-counter (OTC) markets, used by institutional investors and market makers, e.g. *Dark Pools* [107], have tighter confidentiality requirements as minimal legitimate information leakage can be costly [141], For example, some institutional investors make large quotes that might shift the market in an unfavorable direction. [5]

One of the purposes of dark (but legitimate) markets is to remove most of the available public information [70]. Since some information is required for trades to exist, some public variables (e.g. committed on a ledger in a distributed setting) may describe the trigger for a transaction from a potential counterparty. However, each member of the trading venue may refuse to broadcast their back-stop willingness to pay. They would rather do peer-to-peer adjustments in quotes in an OTC bond or derivatives market, such as MTS [143] or the OTC Foreign Exchange Options (FXO) market, the world fourth largest financial market by volume.

[5] Suppose a high-frequency trader has placed a large order specifying that a price and volume are only executed when those conditions are satisfied with the public order book. When a large investment firm, such as a pension company, submits a large market buy-order at the prevailing prices, the high-frequency trader can quickly cancel his sell orders. So as the large order walks up the limit book, the buyer is paying a much higher price than the expected market price. The high-frequency trader receives more money than expected from their initial standing sell limit orders. Hence, the market appears to 'retreat' from large market orders.

As trades are bilateral[6] an investor may send an offer whose particulars are only available to the traders who have available (e.g. committed on the ledger) a suitable combination of financial instruments (e.g. the sum of the cash available and the evaluation of the holding contracts at the market price are above a cash threshold and in general an arithmetic or boolean combination thereof) while traders cannot disclose that they hold these instruments as this information might be strategically used against them (see the Price Discrimination Attack in §3.2.1). In addition, succinct communication is needed in practice. For instance, zcash [185], the production version of ZeroCash, relies on blockchain [144] to reach consensus, where the block size is limited to only 2 MB. Besides, the committed information is frequently updated, e.g. a trading account in FuturesMEX gets updated with a new quote, while the condition of interest is persistent. This requires a scheme that works efficiently with different instances of the same relation.

Research Question 4. ***(RQ4)*** *How can we design a cryptographic scheme that can support the dark pool scenario?*

Non-goals. The focus of our protocols design is to protect against *digital* attacks on integrity, anonymity and confidentiality. *Physical, economic and social* attacks are, and always will be, possible similarly to centralized systems (e.g. insider trading, cartels manipulating the underlying assets or the availability glitches such as the NASDAQ ones [174]) and they are typically dealt with by ex-post law enforcement [131].

1.2 Research Contribution

To address **RQ1**, we conduct a brief overview of the traditional PTNs, such as the tiered banking versus the correspondent banking model, and a high-level description of the crypto-based PTNs, e.g. E-cash [56], Bitcoin [144], ZeroCoin [139] and ZeroCash [164]. The crypto-based PTNs' new feature, compared with the traditional systems, is the removal of the trusted central bank in a clearing and settlement. Then, we describe the security challenges to build such simple PTNs and demonstrate various systems to address these security challenges using different cryptographic techniques. Such discussions are useful for understanding more complex system designs.

For **RQ2**, to extend simple payment to futures trading, we first summarize the high-level functionalities of a Futures Market [169]. Then, the new security challenges to build a distributed financial exchange and required principles to address these challenges are developed. Noticeably, among the security challenges, we present an economic attack

[6] In an OTC swap market, government-to-government, government-to-bank, bank-to-bank, and bank-to-corporate, all have different average volume sizes.

when a trader loses anonymity, in which other traders can strategically force some traders out of the market.

The distilled design principles are used to build a reference implementation of a futures exchange. To address **RQ3**, this thesis provides the first secure, distributed Futures Exchange called FuturesMEX, which replicates the functionalities of the Chicago Mercantile Exchange Globex specifications manual. This includes each of the main quote types (limit and market orders) to build more complex quotes and standard margin accounting, and market features [62].

To support dark trading (**RQ4**,) we define and construct a new cryptographic scheme called Witness Key Agreement (WKA) that allows a party to securely agree on a secret key with other parties if they own a witness (the secret financial information with the desired properties) to the problem instance (the committed information).

We provide a performance simulation for both cryptographic implementations of public and private trading. For FuturesMEX, we use the CME Lean Hog data (a low-frequency public market). For WKA, we use the Bloomberg Tradebook (an aggressive dark pool).

1.3 Structure of the Thesis

Chapter 2 is dedicated to **RQ1**, consisting of the simple Payment Transaction Networks (PTNs), their security challenges, and practical solutions. For answering **RQ2**, we identify new security challenges and the design principles for more complex systems, such as a Futures Market Exchange in Chapter 3. Chapter 4 addresses **RQ3**, showing how to incorporate the distilled design principles to build our reference implementation called FuturesMEX. We answer **RQ4** using Chapter 5 where we describe an extension of the public markets, i.e. dark trading, with a new cryptographic primitive called Witness Key Agreement (WKA). Finally, chapter 6 concludes the thesis with a summary of contributions, limitations, and future research directions.

We provide a detailed structure for each chapter.

- Chapter 2 starts with the high-level PTNs description in §2.1. We describe the security challenges in building PTNs and possible solutions in §2.2. Finally, we briefly introduce the futures trading scenario as an extension of simple PTNs (§2.3), and we provide a summary of the chapter (§2.4).

- In chapter 3, we introduce the key aspects of futures markets (§3.1). We explore the new security challenges to be addressed when building a distributed futures exchange, and we show how these challenges affect the new design principles in §3.2. Among these challenges, we present an interesting price discrimination attack due

to loss of confidentiality and anonymity (in §3.2.1). Finally, we conclude the chapter in §3.3.

- Chapter 4 starts with a formal centralized futures market model (§4.1). This is followed by the description of the ideal functionality of its secure, distributed version in §4.2. Next, we describe our crypto building blocks (briefly in §4.3, formally in §4.5), the non-monotonic security state of the functionality (§4.4) and in our protocol (§4.6). We provide a proof of its security (§4.7). We also discuss some optimizations (in §4.8) and how to go beyond security-with-abort (§4.9). Our proof of concept and its performance evaluation are presented in §4.10 and §4.10.1. Finally, we survey related work (§4.11) and conclude the chapter (§4.12).

- In chapter 5, we discuss the possible but unsuitable solutions to the dark pool scenario in §5.1 followed by our solution in §5.2. Then, some technical background is covered in §5.3. In §5.4, we provide the main intuition and the generic WKA scheme construction. A security proof also appears in this section (§5.4.4). A WKA construction of quadratic arithmetic programs (QAPs) is provided in §5.5. An instantiation based on a variant of the Paillier cryptosystem and our scheme's evaluation are found in §5.6 and §5.7. Finally, we conclude the chapter in §5.8.

Chapter 2

Payment Transaction Networks

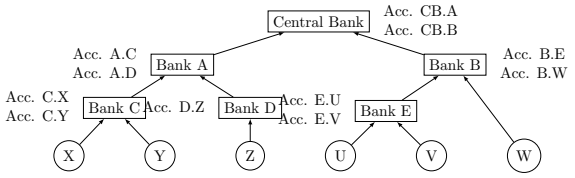
In this chapter, we start with a primer on PTNs. We provide a brief overview of the traditional PTNs and a high-level description of the crypto-based PTNs. Compared with the old centralized E-cash *scheme, a new feature of the recently developed cryptocurrency networks such as* Bitcoin*,* ZeroCoin *and* ZeroCash *is the elimination of the trusted central authority, i.e. the central bank, in clearing and settlement, albeit some systems, e.g.* Ripple *and* RSCoin*, still rely on the bank for value creation. We then describe the security challenges in building PTNs and the strategy to overcome them. The chapter covers the most basic security challenges and the possible solutions to implement simple financial intermediation such as a PTN. Compared with a simple PTN, our main objective, a financial exchange, poses more complex challenges and it is not trivial to address them using those basic solutions applicable to PTNs.*

2.1 A Primer on Payment Transaction Networks

In physical payments without physical cash transfers, the payer gives the payee a cheque,[1] with which the payee can withdraw cash from the payer's bank or lodge the cheque at the payee's bank so that the funds eventually are transferred to the payee's account from the payer's. When a payee lodges the cheque, the funds will be credited after some time, and the cheque is sent to a clearinghouse where the payer's bank will validate the cheque. If the cheque is valid, e.g. funds available and the signature matched, the payer's bank will debit the amount written on the cheque to the payer's account once the two banks have settled the amount by crediting and debiting the respective accounts at their central bank. An invalid cheque will be returned and the payee will not acquire the funds. In cheque

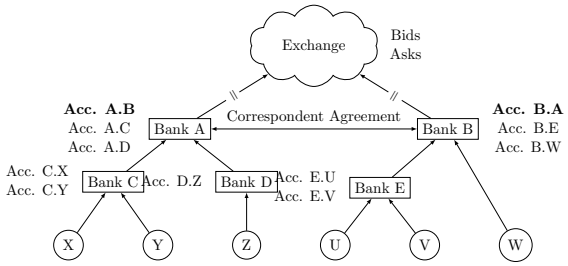
[1] We use the Anglo-French spelling to avoid confusion with the word "check".

The transactions between X and Y only need Bank C. In the case of the transactions between X and Z, Bank A needs to settle the account A.C and A.D to transfer money from Bank C to Bank D, and Bank C will deduct the account C.X, whereas Bank D will credit the account D.Z. More sophisticated transactions between Y and V will involve more banks, including the Central Bank, and this accrues more transaction fees.



(a) The tiered-banking model

Bank A and Bank B hold an account of each other in their ledger, i.e, A.B and B.A, respectively. The transactions between X and V requires a cooperation of both Bank A and B to settle the amount. Obviously, the settlement also needs to be propagated down to Bank C, B, and E. The transactions could also involve currency exchange. In this case, Bank A and Bank B will go through the Bid/Ask orders to sort out the exchange.



(b) The correspondent-banking model

Figure 2.1: Banking models

payment, the payee suffers from such reversible payment and usually is on a disadvantage side (e.g. if the payee, usually a merchant, has shipped the goods upon receiving the cheque). Essentially, electronic transfer eliminates the physical transfer of cash and their risk of delayed authentication, e.g. signature matched, and delayed authorization, e.g. funds available.

Fig. 2.1a illustrates the tiered-banking model. A central bank is at the top, whereas multiple dependent banks and the clients are at the lower levels. The central bank may also control the monetary supply, and its dependent banks must hold an account in its ledger. The clients are at the leaves of the hierarchy. In the correspondent-banking model of Fig. 2.1b, Bank A and Bank B form a direct relationship with each other, possibly through an exchange to settle the transactions without a third party. In an exchange transaction, a single bank may have accounts in two currencies. The clients may incur some fees for currency conversion. Alternatively, both the payer and payee's banks may have a correspondence account with a *multinational bank*, and the exchange is carried out as an internal transfer by the *multinational bank*.[2] Thus, the transaction may pass through multiple levels and two central banks that have correspondent agreements (Fig. 2.1a and Fig. 2.1b). The more third parties the money flows through, the more transaction fees the final clients pay.

[2]A correspondence account is held by a banking institution to receive deposits from, make payments on behalf of, or handle other financial transactions for another financial institution, usually overseas, referred as "Nostro" and "Vostro" accounts in international finance. A "Nostro" (ours) account is an account of our money, held by the other bank. A "Vostro" (yours) is an account of the other bank money, held by us.

2.1.1 Traditional Digital Payment Transaction Networks

As discussed by Abrazhevich [3], traditional electronic payment systems typically either replace the direct transfer from payer to payee, i.e. token-based systems, e.g. E-cash [56], and electronic cheque [7], or the indirect form of transfer, i.e. account-based systems, e.g. Debit Cards and Credit Cards.

PayPal [150] processes payments via the Internet using the client's cards detail. Recently, innovations in payment technologies offer some wrappers to the existing payment infrastructures, e.g. Google Wallet [100] and Apple Pay [8] allow users to pay with their authenticated phone.

Payments across intermediaries are unnecessarily instantaneous. In a Differed Net-Settlement System, the bank intermediaries may settle the outstanding differences at the end of the day [12]. A Real-Time Gross-Settlement System (RTGS) provides the function of both "real-time" and "gross-settlement", meaning low-volume, high-value, and immediate payments. Due to the securely authenticated parties, RTGS is a low-risk transaction environment as payments are final and irreversible at the execution point [26]. An example of RTGS is CHAPS [51] in the UK where its counterpart is BACS [10] that handles Net-Settlement. Other examples include FedWire [84] and CHIPS [173] in the US. The Eurosystem is in the process of implementing the Target2 [78] system, which provides RTGS for banks across the European Union.

For international payments, financial institutions around the globe make use of SWIFT [170] (Society for Worldwide Interbank Financial Telecommunication) to send the payment orders in a secure and standard environment. Then, the payments are settled using their correspondent accounts. SWIFT identifies the financial institutions via a Business Identifier Code (the SWIFT code).

The proposal of "programming languages for financial contracts" (e.g. the Ethereum smart contract [77]) is far from new. It was first proposed by J. M. Eber and was marketed by the company LexiFi for the management of traditional financial contracts where the contracts were expressed and automated (centrally) using Modeling Language for Finance (MLFi) [113].

2.1.2 Crypto-based Payment Transaction Networks

The concept of a digital currency dated back to 1982 when David Chaum introduced the E-cash scheme [56].

The E-cash Scheme E-cash takes a "central bank" approach with a completely flat hierarchy where no intermediary institution exists. The scheme provides such strong anonymity

that the E-cash coin issuing bank and the merchant cannot identify the spender of the coin, even if they cooperate. The scheme allows both online and offline verification of the coin. In the case of offline verification, the payer's identity could be revealed if the coin is spent twice [56].

At the core of E-cash it lies the Blind Signature scheme [53], which allows the bank to sign a serial number (Serial#) without knowing its content. However, with the basic E-cash scheme one can only create fixed value coins. To allow different value coins, the bank can introduce various coin signing keys; upon signing, the coin will have the specific pre-defined value.

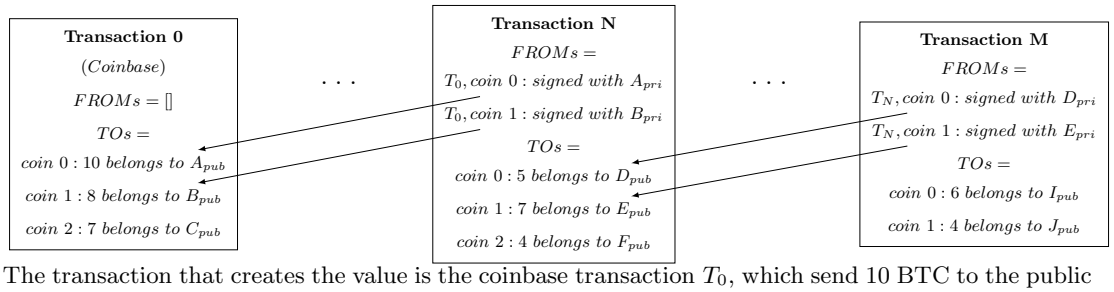
In E-cash, a client needs to out-of-band deposit some fiat currency into the bank to open a E-cash bank account. Then, the client can *withdraw* the E-cash coins from the bank by sending a *blinded* Serial# to let the bank blind sign the Serial# using the Blind Signature scheme [53]. This prevents the bank from seeing the original Serial#. Then, the blind signature on the Serial# can be unblinded by the client to get the signature on the original Serial#. The original Serial# and the unblinded signature on it form an E-cash coin, later used in an E-cash transaction.

To initiate an E-cash transaction, the Payer obtains the Payee ID from the Payee. Then, the Payer creates a transaction with some E-cash coins and the Payee ID. The transaction is sent to the Payee. However, it is first encrypted with the E-cash bank's public key to prevent the Payee from seeing the coins' Serial#. This requires the original minting bank to perform the consolidation of the transaction. Upon receiving the encrypted transaction from the Payer, the Payee forwards the encrypted transaction to the minting bank. The bank will decrypt the transaction with its private key. The Serial# will be checked against the *spent* Serial# database to prevent *double-spending*, i.e. a Payer might attempt to use the same coin in multiple transactions. If everything is correct, the bank will accept the coins and credit the Payee's account with the corresponding amount.[3] The deposited coins' Serial# will be added into the *spent* Serial# database.

In parallel to this "mostly monolithic" architecture, the new architectures aim to transfer cash quantities without centralized clearing and settlement.

The Bitcoin Network Subsequent attempts at a purely digital currency were B-Money [67], hashcash [81], and BitGold [171], which were the first to use "Proof-of-Work" (PoW) [112], a hard cryptographic computational puzzle, as a mean of mitigating Sybil attacks [74] and eventually determining the inherent value for the medium of exchange. These conceptual ideas later matured into the Bitcoin-esque approach, which is free from a trusted central bank. The idea of a public ledger predates Bitcoin. However, the original idea was not

[3]The Payee can now act as a Payer to withdraw some E-cash coins and to spend in another transaction later.



The transaction that creates the value is the coinbase transaction T_0, which send 10 BTC to the public key address A_{pub}, 8 BTC to B_{pub} and 7 BTC to C_{pub}. To spend those coins, in transaction T_N, the owners must sign with the private keys A_{pri} and B_{pri} for the output index 0 and 1 of the transaction T_0, respectively. This also applies to the transaction T_M with D_{pub}, D_{pri} and E_{pub}, E_{pri}.

Figure 2.2: The *simplified* Bitcoin Transactions

decentralized [163]. The Bitcoin protocol introduces a decentralized transaction database, namely blockchain [144], shared by all nodes participating in the Bitcoin system. Every executed transaction is included in the blockchain so that any node can track the balance of each address at any point in the system and later validate a transaction in the network.

In the Bitcoin network, upon solving a PoW [32], the solver is rewarded with some bitcoin (BTC). The PoW puzzle is hard to solve but easy to check. It is also a progress-free puzzle, in which the chance of solving does not increase with effort. This process is referred to as "mining", and the individual nodes are commonly referred as "miners" [30]. Once they successfully solve a block puzzle, they will receive a reward and all transaction fees [34] included in the block. The BTC is bound to the owner of an address, i.e. a public key (or its hash) whose private key is only known to the owner.

Fig. 2.2 outlines the *simplified* process of conducting transactions in the Bitcoin network. Two types of transaction are Coinbase [29] and Regular [33]. While the regular transaction indicates BTCs are transferred from an address [28] to another address; the Coinbase is used for the introduction of new coins into the system circulation (value creation). As such, it is a Coinbase transaction that distributes a miner's reward. For the Bitcoin system, this reward is 50 BTC in the first instance. Then, it will be halved every 210.000 blocks. By the year 2140, the reward will be reduced to 10^{-8} BTC.

A blockchain is a sequence of applications of a hash function [115] to a sequence of transaction blocks. Every block contains the information of the current transactions and a reference to its previous block header's hash. Each block in the blockchain is included with a set of transactions. Each transaction consists of a vector of inputs [35] and a vector of outputs [36].[4] Each input of a transaction T spends a certain amount of BTCs from a previous *unspent* output of a concluded transaction T_0. To spend the previous output,

[4]These new outputs are called *unspent* transaction outpus (UTXOs). When they are used as inputs to a new transactions, they become *spent* and cannot be used again as inputs to another transactions.

the input references a tuple of $(hash, n)$. A double-SHA256 is performed on the T_0's raw data for the $hash$, whereas the n is the index of the output in T_0. Each output of a transaction T specifies an amount that will be transferred to a Bitcoin address. A block also hashes all its transactions into a Merkle Tree [134] to maintain integrity.

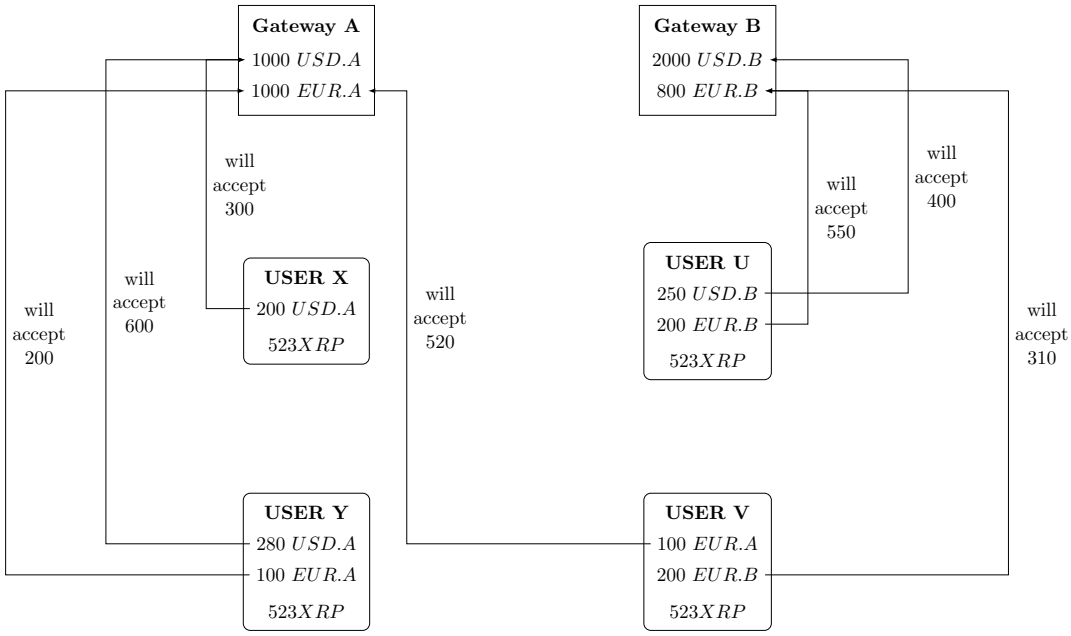
A payment in Bitcoin starts with the Payer creating a transaction signed with the private key to unlock the funds associated with the public address. The transaction will be broadcast to the entire Bitcoin network to notify all miners to verify the correctness of the transactions, i.e. inputs are UTXOs, input amounts are greater than or equal to output amounts and the signature is correct, before adding them into blocks. After solving the PoW and creating a block that includes the transactions, the miners will broadcast the block into the entire network. Other nodes, upon receiving it, will perform the validation for the signature, the PoW, the transactions' correctness, and the integrity of the block before adding it into their blockchain and broadcast their acceptance.

Bitcoin's variants Since the launch of Bitcoin in 2009, many cryptocurrencies have been developed. Some use different hash functions than Bitcoin. Dogecoin [132], Litecoin [127], and PotCoin [154] uses the hash function of Scrypt [152] which is more memory costly to deter the use of hardware-based mining devices. Dash [75] uses X11, consisting of 11 different hashing algorithms in a chain, whereas Primecoin [118] require finding a prime chain composed of Cunningham chain and Bi-twin chain. Nxt [146] applies the Proof-of-Stake mechanism [31] where the miners must prove the ownership of a certain amount of cryptocurrency. Many new cryptocurrencies offer extended functionality. At an extreme, this is represented by *Ethereum* [77] that uses "programming languages for financial contracts."

> "A built-in fully fledged Turing-complete programming language that can be used to create "contracts" that can be used to encode arbitrary state transition functions, allowing users to create any of the systems described above, and as well as many others that we have not yet imagined, simply by writing up the logic in a few lines of code."

Interestingly, cryptocurrencies can also be used to incentivize correct behavior and fairness in MPC [121] by holding crypto tokens in escrow in the claim-and-refund transaction flavor where each party makes "conditional" public transactions with each other. The parties that finalize the computation can claim the fund; otherwise, it is refunded to the original payer [122, 123]. An alternative solution is to commit and lock the crypto tokens in safe deposits before the computation, only released at the end [119].

Among the variants, Ripple takes a relatively different approach from Bitcoin. Ripple [157] provides the functions of an RTGS and currency exchange of a traditional PTN. Fig. 2.3 shows the components of the network of the gateways A and B, usually carried out



Gateways A and B correspond to financial institutions and provide liquidity, i.e. creating issuances that represent the external tokens transferred in the network. The clients X, Y, U, and V declare their trust-line to the gateways A and B. For instance, user X accepts at most 1000 USD.A. The actual deposit of external values at the various gateways is external to the system as in the E-cash protocol.

Figure 2.3: The Ripple Network Components and Gateways with the External World

by financial institutions such as banks that introduce liquidity into the systems. They create "issuances" that represent the external tokens transferred in the network. The gateway A issues USD.A and EUR.A, whereas the gateway B issues USD.B and EUR.B. For the integrity of the issuances, the gateways will sign the issuances with their private keys. Out-of-band, other nodes of the network (the clients) deposit some "real world" money at the gateways to declare a trust-line with a gateway and to accept a maximum value of their digital currency for transactions with other parties.

An interesting direction stems from a Bank of England report [13] in 2015, which covers several salient questions for the financial industry, regarding the development of the technology. One of the primary concerns that central banks may have is losing control of money-supply as an instrument of policy. This specific question is tackled directly by RSCoin [94]. This network delegates the monetary supply to a central authority, such as a bank; however, it utilizes a distributed network of other parties to perform transaction validation.

RSCoin simply delegates the value creation to a Central Bank since the Central Bank is a Central Authority assumed to be honest. To generate value, the Central Bank signs a transaction to allocate the value to an address. The RSCoin network provides strong

transparency and audit capacity of a central monetary provider and the attractive features of a distributed transaction system (chiefly decentralized ownership). However, it suffers from relying on the Central Bank to merge the high-level blocks, creating a central point of failure let alone a computational bottleneck.[5]

RSCoin can be considered as a traditional digital PTN with cryptography extension borrowed from Bitcoin. The Central Bank is still on the top of the hierarchy and maintains the ultimate ledger.

Privacy preserving cryptocurrency Bitcoin and its previous variants only provide pseudonymity, i.e. identity in disguise, but not full anonymity, since the transactions can still be linked by analyzing the transaction graph [25, 136].

In contrast to the preceding types, Monero offers Payer and Payee stronger privacy by hiding the connection between the Payer and the Payee's addresses and applies Cryptonote [160] that utilizes a traceable link signature [86] to confirm the validity of the transactions.

ZeroCoin [139] is a combination of the E-cash scheme and Bitcoin. ZeroCoin is based on the Bitcoin protocol; however, its protocol provides additional anonymity for the network users by allowing the user to "mint" the basecoin (BTC) into a ZeroCoin to hide the origin of the Payer in a subsequent transaction.

ZeroCash [164], a protocol that succeeded ZeroCoin, offers stronger anonymity and privacy. ZeroCoin is only a "coin washing service" where the origin of the Payer is hidden; but, the Payee and the transacted value are still open, whereas ZeroCash is an actual privacy-preserving payment scheme, in which both Payer, Payee, and the transacted value are shielded. ZeroCash was deployed as zcash [185]; however, the system (Sapling) required an initial setup to generate an almost 800 MB data to store the proving key and to verify the key for zero-knowledge proof [19]. Such setup ceremony is costly but achievable [20,40].

2.2 Security Challenges and Solutions for Payment Transaction Networks

Integrity and Availability are obvious needs.[6] However we should also consider two additional security properties: *Confidentiality and Anonymity.*[7]

[5] In Europe only German and Italy's central banks have the computational horsepower to perform the task.

[6] Integrity in a PTN is preserved if the correct transaction validation rules are followed. In the DAO's code (Chapter 1), the validation rule was wrong: it proceeded with account clearance prior to ledger update, therefore integrity was broken. Availability refers to the degree to which the PTN is in an operable and committable state when handling a transaction.

[7] With Confidentiality, the value is unknown. With Anonymity, the owner is unknown.

Table 2.1: Loss of value

Notes: The first threat to PTN integrity and availability is the possibility of a party to lose money. A key distinction is whether such losses are systemic failures or due to individual actions, either by a victim's lack of care or by malicious activities of other parties. From the table, decentralization of PTNs make individual thefts harder as it requires changing the value of the global ledger. However, the Payee can still incur a loss in the transaction owing to lack of care. Technological failures to achieve consensus (e.g. Bitcoin forks) introduce the systemic risk parallel to the failure of a central bank.

Threats	**Others Benefit**	**Victim Actively Involved**	**Examples**
Systemic Loss	-	-	*CA* fails and all values are lost; Blockchain forks.
Individual Loss	-	yes	Lose the Serial#, private keys.
Fraud	yes	yes	Over-drafting; Over-spending.
Theft	yes	-	Unauthorized-spending.

We categorize the actors of a PTN into Payer, Payee, Central Authority (CA), and Broker.

Payer and Payee A typical PTN transaction must involve at least the main actors: a *Payer* (who pays) and a *Payee* (who is paid).

Central Authority A step in a payment system is an individual responsibility if we need *a specific trusted party*, e.g. a CA, to perform a particular action (e.g. the central bank signs the coin-base transactions and broadcasts it to the miners in RSCoin). Without the CA's actions, the step is invalid, no matter what the others do (e.g. the miners cannot generate a coin-base transaction without the CA's signature in RSCoin). In a PTN, there might be more than one CA, e.g. in Ripple, each (issuing) gateway is a CA. However, they are independent and have ultimate authority in their related transactions. In Fig. 2.3, the gateway A that issues USD.A can only sign the issuance transactions in USD.A, whereas the gateway B that issues USD.B can only sign the issuance transactions in USD.B [158].

Brokers A step in a payment system is collective responsibility if the authority is shared among multiple *untrusted but verifiable* actors, i.e. the Brokers, who converge on a decision using a pre-defined rule. For example, Bitcoin transactions are aggregated as a new block into the blockchain by the miners using the longest chain rule [144].

2.2.1 Integrity and Availability

The criteria for *integrity and availability threats* classification are whether others benefit from the victim's loss and whether the victim is actively involved in one's loss. We identify three high-level integrity and availability threats: loss (systemic and individual), fraud, and theft as revealed in Table 2.1.

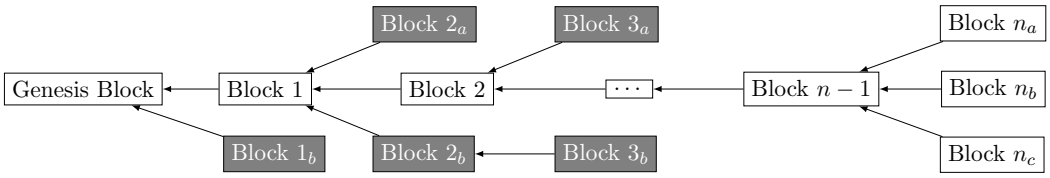
Systemic Loss Systemic Loss in a PTN is not caused by an individual but rather by some intrinsic fragility of some of the system's components. In the presence of CAs, the obvious sources of fragility are the (un)trustworthiness of the CAs. An example is the failure of Central Banks. By design, CAs can violate integrity by arbitrarily printing money into the system; thus, protection against their failure must be dealt with outside the protocol.

While all token-based payment systems suffer from systemic risk such as attacks on crypto, e.g. quantum computing, collective responsibility PTNs are amenable to an additional different type of systemic risk, namely the *failure to reach consensus* [126][8]: the blockchain forks and some legitimate transactions are excluded from the main chain. This threat is *not present* in traditional PTN.

We illustrate this in Fig. 2.4 for the case of Bitcoin. At some point, several nodes are generating new blocks or proposing transactions at the same time. The members of the PTN closer to the promising payers may create different blockchains. In that case, the blockchain is forked and every node will maintain the fork until conflict resolution mechanisms kick in. In Bitcoin, the consensus mechanism is based on the presence of a "longer" chain. The longer chain becomes the main chain, and the other tentative chains are considered invalid. The blocks in the invalid chain are "orphaned" blocks. From the PTN's perspective, they are all potentially valid but not consolidated payment promises. Every transaction in the orphaned blocks should be returned to the transaction queue to be added in the new block. As shown in Fig. 2.4, the transactions in blocks 1_a, 2_a, 2_b, 3_a, and 3_b are orphaned transactions because of previous fork resolutions. At block $n-1$, the chain suffers from three forks (blocks n_a, n_b, and n_c) and must wait for the next fork resolution.

The classic distributed consensus problem is described as a set of n participants, out of which t are possibly faulty, try to agree on a single value v drawn from a set of values V proposed by the participants. The most trivial solution is majority voting. The classic paper on Byzantine agreement on synchronous system [126] focuses on three properties that an agreement protocol must hold: (i) Validity: The single value v must be initially proposed by non-faulty participants. (ii) Agreement: Every non-faulty participant must agree on the same value v. (iii) Termination: The agreement must be reached in finite time. The robustness of a Byzantine protocol is measured in terms of the number of faulty participants that it can tolerate. It is proven that Byzantine agreement is only possible with $n \geq (3t+1)$ [126].

[8]We do not consider the scenario of dishonest majority here since the notion of "correct" evolution may be open to debate. When Ethereum changed its rules to retroactively undo the DAO issue [63], many networks were acting "dishonestly" according to the original rules; this did not result in the complete breakdown of the distributed system. The dissenting party just divided the community. The same could happen here.



Notes: After $n-1$-blocks, three blocks (n_a, n_b, n_c) have been proposed concurrently. The global chain is forked until some miners append a new block to one of the three alternatives. Some previous forks in the past at block 2 resulted in orphaned transactions (in grey). In most blockchains (e.g. Bitcoin, ZeroCash), those transactions are *not consolidated* in the system, no matter whether they are valid from the perspective of a "normal" payment system. The situation is particularly dire for those in the forked chain $3_b : 2_b : 1$: they have added transactions along what they thought was an eventually legal chain. The payers must re-play some transactions along the entire subchain (both 2_b and 3_b) if they want the transactions to be inserted in the final ledger. Some new blockchains (e.g. IOTA [110]) are more flexible since they allow smaller branches to merge with the bigger ones.

Figure 2.4: The Systemic Risk of Blockchains: Forked and Orphaned Transactions

A further FLP impossibility result, applicable to deterministic and asynchronous system [82] has proven that even with a single faulty-participant, the distributed consensus cannot be achieved in asynchronous settings because the nodes fail to differentiate a failure from a delay. To circumvent the impossibility results, researchers introduced techniques such as randomization [155], failure detectors [50] or assume partial synchrony [76]. In such systems, safety is always prioritized over liveness.

The Bitcoin network offered a solution to a variant of the Byzantine agreement problem. The Bitcoin network's consensus problem is different from classic ones as the number of nodes in the network is exceptionally large and dynamic. The consequence is that nodes cannot decide on the value of the majority. The nodes will vote for the "chain with most work" according to the blockchain by referencing to the last block of that chain in the new block. Additionally, the block generation rate is technically maintained at 10 minutes for the propagation of block across the entire network so that every node shares the same blockchain. This solution practically allows the tolerance of faulty-nodes up to 50%. However, a consequence is that the transaction clearing speed is unacceptably slow for any financial implementation (e.g. the Chicago Mercantile Exchange marks to market millions of trades in less than a minute), even though several variants of the protocol are introduced to improve the transaction clearing speed [79, 167].

Individual Loss, Fraud and Theft *Individual Loss* happens when the E-cash or ZeroCoin owner loses the serial numbers. Another possibility is when the Bitcoin or RSCoin network users lose their private keys. The threat of losses due to individual sloppiness or misfeasance (*Fraud and Theft*) that are present in traditional and decentralized PTNs can be broadly characterized in the following classes:

Over-Drafting When the Payer sends a transaction whose value exceeds one's available funds. An example would be the E-cash Payer withdraws a 10$ coin from the bank when the balance is less than 10. Another example is that a transaction in Bitcoin, ZeroCoin, or RSCoin network consists of outputs whose value is larger than the total inputs. In the Ripple network, it is a transaction that pays 1000 USD.A when the payer only has 800 USD.A of credit available.

Double-Spending When the Payer can spend a token twice. The simplest double-spending case in E-cash where Payer uses the same Serial# twice.

Unauthorized-Spending A Payer spends another one's money or changes the value in another's account.

In Table 2.2 we present the basic categorical countermeasures for the mentioned fraud and theft threats as well as the example of countermeasures from some existing PTNs.

2.2.2 Confidentiality and Anonymity

Threats to *confidentiality and anonymity*, i.e. information disclosure, involves answering three questions below. The example countermeasures against loss of either confidentiality or anonymity are summarized in Table 2.3.

Instantaneous Networth At the time t, can an attacker identify the total value v of a nominal identity I?[9]

Transient Value At the time t, can an attacker know about a transaction of total value v between two nominal identities I_1 and I_2?

Persistent Identity Can an attacker link two nominal identities I_1 at time t and I_2 at time $t' > t$?

E-cash relies on the *trusted* bank to preserve Instantaneous Networth and Transient Value. The Persistent Identity of the Payer in a transaction is guaranteed because the bank cannot trace back the original owner of the coin from the Serial# owing to the Blind Signature scheme [53].

Bitcoin, ZeroCoin and RSCoin do not address Transient Value as every transaction is recorded publicly in the blockchain, making their Instantaneous Networth dependent on Persistent Identity.

[9]For the avoidance of doubt, in this scenario we do not consider the "physical" identity of the entity behind the nominal identity in the PTN. Taken on its own, a Bitcoin address has no less (and no more) anonymity than a traditional bank account number. It is only the information held by the bank in its enterprise information system, usually *outside* the interbank payment network, that allows to associate a 27 alphanumeric international bank account number to a human or a legal entity.

Table 2.2: Loss of value countermeasures

To mitigate Over-Drafting, Double-Spending, and Unauthorized-Spending, validation rules are needed while Individual Loss is usually prevented by multiple (and partial) backups of the authentication secrets. In E-cash, if a client loses the signatures but still keeps the serial numbers and the blinding factors, E-cash bank can resend the blind signatures for the client to recover the lost coins. If the system *itself* fails to provide a countermeasure the clients may rely on some external mechanisms to counter the threat, e.g. a Bitcoin user may back up his private keys storage to prevent the loss of the private keys.

Threats	**Over-Drafting**	**Double-Spending**	**Individual Loss**	**Unauthorized-Spending**
Counters	CA or Brokers must check the balance of the Payer or the total amount of the spending coins.	CA or Brokers must check the status (spent/unspent) of the token.	Backup mechanisms.	CA or Brokers must authenticate the Payer or the payment token.
E-cash	*CA* checks *Payer*'s balance before signing the blinded Serial#. *CA* checks coins value in a transaction.	*CA* verifies the Serial# against the spent Serial# DB.	*CA* stores the last n batches of blinded Serial# and the corresponding blind signatures, send back to *Payer*. *Payer* unblinds the blinded Serial# and the blind signatures to recover the coins.	*CA* checks the signature on the Serial#.
Bitcoin	*Brokers* check that input $\geq$ output.	*Brokers* check the inputs of outputs of previous transactions are unspent.	Not Provided	*Brokers* check the signature.
ZeroCoin	Same as Bitcoin.	*Brokers* check the Serial# are unspent and belong to the set of minted ZeroCoins.	Not Provided	Same as Bitcoin.
Ripple	*Brokers* verify balance $\geq$ output.	Not Present	Not Provided	Same as Bitcoin.
RSCoin	Same as Bitcoin.	Same as Bitcoin.	Not Provided	Same as Bitcoin.

Table 2.3: Information disclosure countermeasure examples

System	**Instantaneous Networth**	**Transient Value**	**Persistent Identity**
E-cash	The trusted *CA* keeps the balance DB private.	The trusted *CA* keeps the transaction private.	Guaranteed for *Payer*. *CA* can link the Payee but it is trusted.
Bitcoin	Dependent of Persistent Identity due to lack of Transient Value.	Not Provided	*Payer* and *Payee* use a different key pair per transaction but only obtain Pseudonymity.
ZeroCoin	Only applicable for *Payer*. For *Payee*, it is dependent of Persistent Identity owing to lack of Transient Value.	Not Provided	Only applicable for *Payer* as s/he applies minting and redeeming. *Payee* uses a different key pair per transaction but only obtain Pseudonymity.
Ripple	Not Provided	Not Provided	Not Provided
RSCoin	Same as Bitcoin.	Not Provided	Same as Bitcoin.
ZeroCash	Fully supported, using Commitment Scheme, Merkle Tree and Zero-Knowledge		.

Bitcoin and RSCoin provide only pseudonymity (i.e. identity in disguise, a weaker form of anonymity) in terms of Instantaneous Networth as this property is dependent on

Persistent Identity for both the Payer and the Payee. The Payer and the Payee obtain pseudonymity by using different key pairs as pseudonyms in transactions. However, those transactions can be linked through transaction graph analysis [25, 136].

ZeroCoin provides anonymity for Persistent Identity (and Instantaneous Networth) for only the Payer. To mint a ZeroCoin, a client generates a random serial number and encrypts it into a coin with a second random number. Particularly, ZeroCoin uses Pedersen commitment [151] for minting. Then, miners added the coin to a cryptographic accumulator [22]. The minting operation requires a Bitcoin transaction that spends some BTCs to obtain some ZeroCoins. To spend a ZeroCoin, a Payer must first redeem it into BTC using a zero-knowledge proof [165]:[10] (i) the spending coins belong to the set of minted ZeroCoins, and (ii) the Payer knows the serial number and the random number corresponding to the spending coins.

ZeroCash fully supports confidentiality and anonymity requirements. However, Ripple fails to provide any countermeasures for confidentiality and anonymity threats. Similar to ZeroCoin, ZeroCash uses a Commitment Scheme [95] for confidentiality, a Merkle Tree [137] for anonymity in conjunction with the zero-knowledge proofs [21] for integrity. To create a transaction in ZeroCash, the Payer broadcasts the payment information and some zero-knowledge proofs. Then, the miners create the zero-knowledge proofs: (i) the spending coin belongs to an unspent set of coins maintained as a Merkle Tree; (ii) the Payer knows a secret parameter to unlock the coins; and (iii) the coins' amount is within available funds.

2.3 Beyond Payment Transaction Networks

A futures contract is a standardized agreement between two parties to buy or sell an underlying asset, at a price agreed upon today with the settlement occurring at some future date [169]. A futures contracts is a "promise" to buy or sell, and this "promise" can be traded. Such trading is carried out in a double auction market operated by a centralized clearinghouse called Futures Exchange [5], such as the CME. On the CME, futures contracts range from bushels of corn to Euro/US$ exchange rates. Recently, the Chicago Board Options Exchange (CBOE) and CME launched Bitcoin futures markets. These are 'cash-futures', as they are settled in cash. Eurodollars futures are the largest world market by notional volume: in quadrillions of dollars per year [181].

From a cryptographer's perspective, a Futures Exchange is just an instance of a *reactive ideal security functionality.*[11] In that manner it is very similar to a PTN except that

[10] Informally speaking a zero-knowledge proof is a method one party can prove to another a given statement, without conveying any additional information apart from the statement is indeed true. For further detail see §5.3.

[11] A reactive functionality is a functionality that keeps states.

futures trading introduce advanced functionalities that are more complex than withdraw and deposit hence it requires further investigation.

In particular, *Traders* can 'quote' futures by sending the Exchange a price and a notional volume of assets at which they will buy or sell (a limit order), or initiate a trade by placing an order at the best price from the standing quotes (market order). The Exchange intermediates between buyers and sellers, advertises the orders in a Limit Order Book (price discovery), matches their orders, and ensures that everybody deposits enough money to pay for his promises. For the latter, the Exchange collects an *initial margin* from traders and keeps enough money to keep their promises (maintenance margin) by calling them to deposit more if needed (margin call[12]). If the traders fail to fulfill the margin call, the Exchange will liquidate the open positions (contracts bought or sold) of the Traders and nets them out.

Once traders deposit real money to the exchange, cash becomes numbers on a ledger of the CME. *Thus, digital promises to be settled by digital transfers are particularly suited to illustrate the security challenges of building such an architecture without worrying about crypto-tracing physical barrels of oil.*

Simple PTNs however can be used to bootstrap a futures exchange. For the market initialization, we observe in practice one cannot initialize a market with a self-claimed account. The cash that gets deposited into the market must be backed by a verifiable source where the debit is acknowledged by every market participant, e.g. zcash [185]. We can assume that the initialization of the market takes such verifiable source as input, e.g. we can use a digital cash network that supports a privacy-preserving payment scheme, e.g. zcash, for bootstrapping the market's initial cash.

A key question is whether general financial intermediation [5] as embodied by a Futures Exchange [169] can be effectively replaced by distributed protocols in the same way cryptocurrencies such as Bitcoin [144] or anonymous payment systems such as ZeroCash [164] challenges traditional payment systems.

Decentralized price discovery has been proposed to be one of such differences [57]; however, we can argue that there are *deeper implications for cryptographic protocol design* because *security and economics interact*. Thus, the preferred crypto solution might not be economically viable, and the obvious economic solution in a centralized setting might be a security disaster in a distributed setting. This poses several challenges *other than those we have discussed in this chapter* to implement secure, distributed *and* economically viable financial exchanges. For instance, confidentiality is required to avoid market failures (and not just for 'privacy'), security evidence behaves non-monotonically in the actions of honest parties, and MPC may introduce more problems than it solves. Even a good old

[12]It in the old days of open-cry trading floors; this was a voice call. Now it is an API.

notion such as fail-safe needs to be revised. In the next chapter we will discuss in details these challenges and show how to enucleate the design principles to address them.

2.4 Summary

We have reviewed the traditional payment networks and a high-level description of both old and new crypto-based payment networks. Some interesting security challenges are posed by such systems, and the system designers must rely on various solutions to address these challenges. Such challenges are considered as basic challenges for more advanced financial intermediation such as a futures exchange. To decentralize a futures exchange, we must address more complex (and possibly specific) challenges.

Chapter 3

Security Challenges and Design Principles for Distributed Financial Exchanges

Implementing secure, distributed, and *economically viable financial exchanges radically challenges the traditional constructs, such as zero knowledge and secure multi-party computation (MPC). Confidentiality is required to avoid market failures (and not just for 'privacy'). Security evidence behaves non-monotonically in the actions of honest parties, and MPC may introduce more problems than it solves. Even a good old notion such as fail-safe must be revised. To boost the discussion of such* practical *challenges, we enucleate the design principles to build a distributed Futures Exchange. This chapter starts with an introduction to our chosen example, a Futures Market Exchange, a landmark institution of financial intermediation. Next, we identify the security challenges and distill the corresponding design principles to address the challenges. Noticeably among the challenges, we introduce a new attack based on price discrimination when traders' anonymity is broken.*

3.1 An Introduction to Futures Markets

To illustrate how markets work, we explain the key trading mechanisms and discuss some aspects of the market microstructure of futures contracts [106, 108]. Fundamental participants in a futures market include traders, exchanges, and regulatory bodies as summarized in Table 3.1.

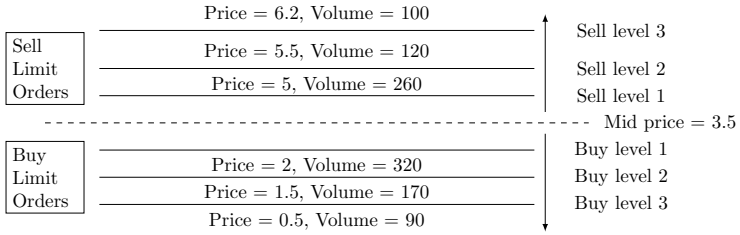
Traders post buy (bid) or sell (ask) orders for a specific futures contract in the market.

Table 3.1: Key Compositions and Characteristics of Futures Market

Traders Characteristics:	
Possible Positions	Buy-side traders holding long positions. Sell-side traders holding short positions.
Possible Actions	Post (Market/Limit Orders) and Cancel (Limit) Orders.
Exchanges Main Functions:	
Price discovery and order matching	Disseminating the real-time market data to market participants by providing a *central limited order book* (see Fig 3.1): an electronic list of all waiting buy and sell quotes organized by price levels and entry time. Matching engines use algorithms to match buy and sell quotes with a price and time priority principle.
Risk management and clearing of orders	Clearing house is responsible for having a *daily/final settlement* by the process of "mark-to-market", so that no pending promise (to buy or sell) and no debt remains unfulfilled. Traders need to deposit an *initial margin* and maintain a minimum funding in the margin account above the *maintenance margin*; otherwise, they will receive a *margin call* for additional funding. Traders failing below the minimum are forced to liquidate their open positions and netted out.
Market fairness and absence of price discriminations	For fairness, *traders are anonymous* as exchanges hold all info about them and never reveal it to others. A trader only see the details of her own orders and not even the ID of the counterparty of an order matching her own order executed through the exchange, as this would allow for *price discrimination*.
Major Players	
Chicago Mercantile Exch.	The largest derivatives market with 3.53 billion of contracts traded in 2015 [87].
Eurex Exchange (Eurex)	The largest European derivatives market with 2.27 billion of contracts traded in 2015 [87].
Regulatory Bodies	Futures markets are regulated by independent government agencies to protect market participants and prevent fraud and manipulation activities, such as the CFTC [176] and the SEC [177].

The *trading position* characterizes a trader as a buyer or a seller: sellers take *short* positions by selling futures contracts; buyers take *long* positions by buying futures contracts. Obviously, buyers prefer to purchase contracts at lower prices and sellers prefer to sell contracts at higher prices. Traders can also cancel orders immediately after having posted them to adapt to fast-changing markets (a heavily used feature). There are two types of trader in a market: the *hedger* and the *speculator*. Hedgers take a position in the market opposite of their physical position as they want to *guarantee physical delivery*, e.g. an airline company participates in oil futures to guarantee they have the budget to buy oil at a future point. Unlike hedgers, speculators have no interest in taking delivery, but instead try to *profit by assuming market risk*, i.e. speculators make a profit by predicting the price fluctuation to buy low and sell high. Cancellations come from those speculators if they think that the position is not profitable any longer but a market exchange *cannot predict in advance* whether an order will be canceled.

The *Exchange* is a centralized intermediary between buyers and sellers that guarantees price discovery, matching, and clearing. It manages risks and guarantees the fairness of the market (see Table 3.1 for the economic perspective).



An order book with limit orders. The dashed line is the average mid-price calculated by the CME as the (unweighted) average of the best sell and the best buy price. Traders' holdings are offset against the mid-price.

Figure 3.1: Order Book

Price Discovery An exchange disseminates the real-time status of market information to traders through an electronic list of all waiting buy and sell orders. This list, called the *central limited order book*, includes (at least) the volume of contracts being bid or offered at each price point, or market depth while keeping anonymous the traders' identities. This list is illustrated in Fig. 3.1.

Matching and Clearing Traders can post new orders (and cancel their pending orders) through the exchange trading platform. Fig. 3.2 illustrates the basic order pathway [65, 101] and the two basic order types: market orders and limit orders [159].[1]

Market orders are buy or sell orders with a specific amount to be executed immediately at current market prices. As long as there are sufficient sellers or buyers in the market, market orders will be matched at the best prices (best bid or best ask) available at the relevant time. Limit orders are orders to buy contracts at no more than specific prices or to sell contracts at no less than specific prices. They may not be executed immediately or even never be executed. Since each order has a unique timestamp, quotes with the same prices entered earliest *must* be matched first. If the opposite order already exists in the order book, market orders are given the highest priority in the matching process. Buy and sell orders at the same prices are matched by the Exchange until the required volume of contracts is reached. Matched orders will go through a clearing and settlement process to complete a transaction [153]. The exchange usually operates its own clearing house responsible for a daily settlement for each futures contract by the process of "mark-to-

[1]The limit order is the most basic order type of a market where both price and volume are explicitly specified. A market order is a special case of the limit order where the trader only needs to specify the volume and the price is implicitly the current best (buy or sell) price. For the exposition, these two are sufficient. Other types of order, e.g. a stop-loss order where setting a stop-loss order for 10% below the price at which you bought the asset will limit your loss to 10%, or a trailing stop order to buy or sell a security if it moves in an unfavorable direction, can be considered as limit orders with additional conditions, which can be checked with an additional MPC in protocol construction in Chapter 4. They are indeed also interesting for consideration (e.g. we can replace the MPC with a tailor-made hybrid protocol) but we do not consider them in this thesis and leave them to future work.

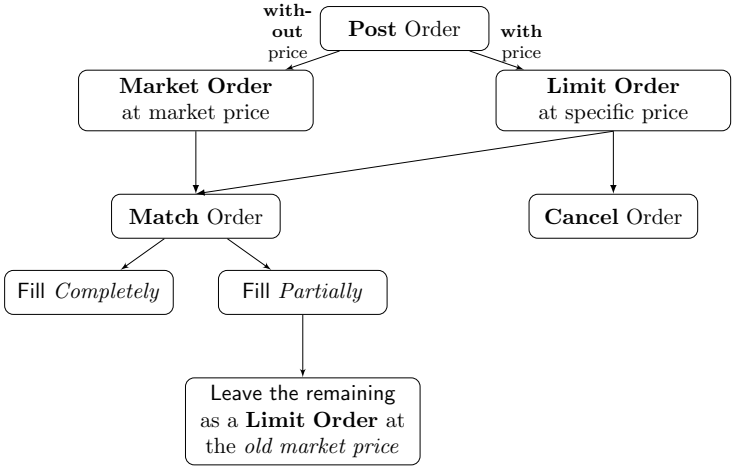


Figure 3.2: The Order Pathway

Table 3.2: Samples of Market Activity

The table shows the maximum number of active traders (#T), number of posted orders (#PO), and matching orders (#MO) for some futures contracts (Eurodollar being world's largest). Cancelled orders' number is close to that of posted orders so they are not reported. Data are obtained from the CME tapes via the Thomson Reuters Tick History database [175].

Contract	Lean Hog LHZ7			Eurodollar GEH0		
Trading Day	#T	#PO	#MO	#T	#PO	#MO
Low	15	1067	46	14	23469	85
Normal	17	3580	146	199	267089	7907
High	33	6709	536	520	376075	8402

market", which values the assets in futures contracts at the end of each trading day, i.e. all traders offset current position to 0 (the long traders who have bought some contracts must sell all of their holding volume while the short traders who have sold some contracts must buy back all their owed volume) using the current mid-price. The profit and loss are settled between long positions and short positions.

Table 3.2 illustrates the variability of the markets by comparing some days for the Eurodollar, the largest market in the world, together with Lean Hog, a less frequently traded futures.

Risk Management The exchange plays an important role as the counterparty of each transaction to minimize the credit risks and to provide transactional integrity. A key functionality of the exchange is helping traders to settle their obligations (i.e. their matched orders). To this extent, the exchange uses a margin system: traders deposit a certain amount of good faith funding into their margin account to initiate their daily trading activities (the initial margin). Furthermore, traders must maintain a minimum

Table 3.3: Informal Security Requirements

Property	Description
Availability of Order Book with Confidentiality of Trader Inventory	The exchange holds all trading information including prices, volumes, margins, and traders ownership of orders, etc. It has to protect a trader's inventory without leaking it to other traders.
Market Integrity and Loss Avoidance	The exchange implements trading (execute matching orders), and guarantee final settlements (traders' margin meets posted orders) after each event for the integrity of the marketplace.
Trader's Anonymity	The exchange must prevent the other traders from linking the orders of the same trader. This is done by managing an anonymous central limit order book where only bid and ask prices are publicly available. In this way, traders will be unable to identify and forecast others' trading strategies.
Trader's Precedence Traceability	The exchange must link the limit orders to the individual traders so that matching orders are accrued to the traders who made them in the exact order in which they where posted.

amount of guaranteed funding (the maintenance margin). If traders' margin balance falls below the maintenance margin due to volatile prices, they will receive a "margin call" asking them to deposit additional funds into the account. The margin system also provides leverage to traders as they can control a large value of futures contracts with a relatively small amount of capital.

Minimizing Price Discrimination The exchange must guarantee that traders cannot exploit the information on the cash (or lack thereof) and leverage of other traders [124]. When traders open a trading account in the exchange, they must submit personal identification documents to verify their credentials, and the exchange will provide an operator ID to each trader. Yet, *not even their IDs would be visible to the other traders.* A trader only sees the details of her own orders but not the IDs and orders of the third parties matching her own executed orders through the undisclosed exchange. Anonymous trading allows traders to execute transactions without the scrutiny and estimation of the market.

3.1.1 Informal Security Requirements

From a security perspective, an exchange is clearly an instance of a *multi-party reactive security functionality* [46]: every agent must satisfy individual constraints (monotonic), and the system as a whole must satisfy global constraints (possibly non-monotonic). The economic requirements in Table 3.1 can be directly transformed into the (informal) security requirements in Table 3.3.

3.2 Security Challenges and Design Principles

We borrow the style of Abadi and Needham [1] to enucleate the novel design principles and to illustrate our points. Some of them are just an improvement of existing constructions, others are new ones specific to financial exchanges.

3.2.1 Protect against Discrimination

While integrity is an obvious need, confidentiality and anonymity seem less critical for financial intermediation.

Indeed, a widely held belief is that people *want confidentiality and anonymity* to avoid snooping governments, or for doing dodgy transactions, but *do not need it* for the functioning of a market. After all, it is possible to see the transacted value and trace all transactions to a Bitcoin's ID using public information in the blockchain. Yet, this hardly stopped Bitcoin from thriving [164].[2] The similar broken anonymity issue might happen to Prediction Market [57], which applies to the design principles of Bitcoin to decentralize the functionality and governance of a market.

Unfortunately, this belief is wrong in our scenario. If confidentiality and anonymity fail, traders can strategically post or cancel limit orders so that other traders will be maliciously forced out of the market. A market vulnerable to those *price discrimination attacks* may collapse as traders will just flock away to avoid the risk of being "squeezed".[3]

Principle 1 (Protect against Discrimination)**.** *If the knowledge of actors' attributes can be used against them for discriminatory practices, it is prudent to protect the confidentiality of those attributes for the actors to bid anonymously (at least for what concerns those attributes).*

We illustrate such an attack scenario in Fig. 3.3. Assume Alice, Bob, Carol, and Eve are in a market. Alice starts with 1400 available cash while the other traders start with 1200. Alice accumulates 90 promises to sell (short positions) at $10 each. Each trader buys 30 contracts from Alice at this price. To estimate a trader's exposure, the Exchange assumes that all contracts are bought and sold instantaneously at the current mid-price of $10.[4] To fulfill her promise to sell 90 contracts, Alice must first buy from the current

[2] In fact, Bitcoin is considered as a more successful cryptocurrency compared with the fully privacy-preserving ZeroCash [164]. According to CoinMarketCap [64], as of May 2019, the market cap of Bitcoin was over $100 billion compared with $300 million market cap of zcash, the production version of ZeroCash.

[3] As noted by Kyle [124], "The anonymity of futures markets tends to change the nature of the market dramatically, because knowledge of who is trading what is in many cases a valuable commodity itself. Traders sometimes go to great lengths to conceal their identities while simultaneously going to equally great lengths to figure out what other traders in the market are doing.

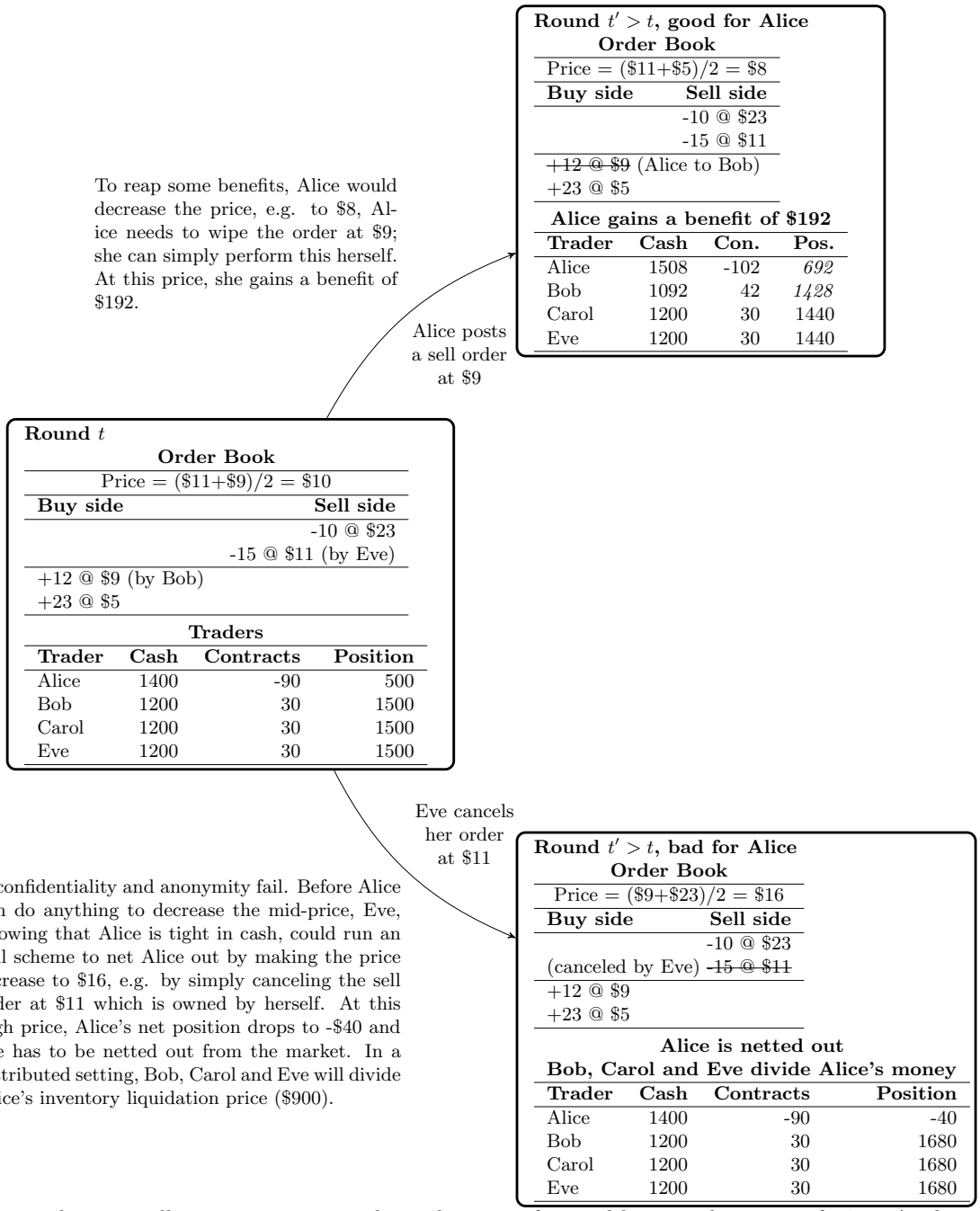
[4] We use the unweighted mid-price for simplicify of exposition.

market price. This would reduce her cash availability to $1400 - 90 \cdot 10 = 500$. This situation is shown in Fig. 3.3 (middle left). Alice would definitely want the price to shift towards her favorable direction (to decrease to \$8) for her to reap some gains, say \$192 (top right, where Alice's position increases to 692 by wiping the order at \$9).

However, if Eve knows that Alice is a small investor who is unable to pour more cash, before Alice can wipe Bob's order, she can post buy orders at slightly higher prices (or convince some other traders to cancel their buy order at a lower price). In this way, the mid-price changes and pushes the liquidation price of Alice's position higher. Alice could sell to Eve's buy order, but this pushes the contracts more deeply in red and makes the market price rises further, worsening her lack of cash. Eventually, Eve, by simply canceling her own sell order at \$11, makes the price reach \$16 (Fig. 3.3, bottom right), Alice's net position is negative below the margin call threshold, and Alice is cashed out of the Exchange, with a realized payout to the other traders.

Notice that these postings required traders to increase their risk position as their orders could have been met by Alice. If they were unaware who Alice was and her financial capacity, they would not have run the risk. Lack of confidentiality and anonymity makes the attack risk-free. Other traders can cancel their orders to decrease the price back to \$10 or even lower (when Alice's trades would have been profitable). However, Alice is unable to benefit from this price as she has already been cashed out. Other traders did not actually trade anything and still forced out Alice by adjusting their buy quotes strategically and discriminated Alice's price: their pricing strategy could only work because they *knew exactly the* amount in Alice's pocket and therefore how much needed was to nudge her out. The opposite is generated from a long position when the market is artificially deflated. Was Alice unwise? No, if Eve did not know that Alice was the cash strapped pensioner, but rather, could have possibly been a deep-pocketed pension fund, Eve would not have even tried to nudge her out.

We cannot *create* anonymity from scratch unless we assume that multiple actors are willing to join a protocol step to hide the origin of the intended action, e.g. using an MPC such as the Dining Cryptographer Network [54]. However, this violates another constraint we dub 'proportional burden' in a later discussion. To *preserve* anonymity created by the participants using an anonymous communication channel (such as Tor, like Bitcoin does) the protocol can combine such network with some privacy-preserving mechanisms such as Merkle Tree and Zero-Knowledge Proofs to achieve stronger anonymity as in Anonymous E-Cash [163], ZeroCash [164, Section I-B] or FuturesMEX (§4.4). Additionally, if the message contains the identity of a trader, e.g. in an Exchange where a limit order includes the price, volume *and the identity of the posting trader*, the communicated information must be unlinked from the posting trader's identity. To guarantee this requirement, in the



Alice accumulates 90 selling contracts currently at the price of 10 and has a cash margin of 1400. At this price her inventory liquidation price is $X_{\mathsf{Alice}} = -90 \times 10$, and her net position is $N_{\mathsf{Alice}} = 1400 + X_{\mathsf{Alice}} = 500$.

Figure 3.3: Forcing Alice out of the market

execution of a futures exchange, one can expose the public order information, i.e. price and volume, but the trader information, i.e. the trader ID, must be somehow encrypted.

3.2.2 Ensure Responsible behavior

As soon as we warrant secrets we must make sure that actors bid within their means as the scenario described in Figure 3.3 (where Alice's margin drops below the threshold) could have happened by Alice's own volition (by mistake, mischief, or just a wrong bet against the market). Loosely speaking, the Exchange would have intervened to use Alice's money to buy the 90 promises from the market and keep them ready for Bob, Carol, and Eve when they would have shown at the end of the day to claim their 30 contracts each.

In a distributed setting, the centralized Exchange is no longer present but Alice's net position is still shielded by Principle 1. If the protocol allows Alice to go deeply into the red since her position is protected, she would fail to meet her obligations against the other three players. Bob, Carol, and Eve thought to have good contracts when in reality they rely on a bad creditor whose real solvency was hidden. This situation will force the market to collapse.

Principle 2 (Ensure Responsible behavior)**.** *If an action requires the actor to fulfill some future obligations, the protocol must validate the actor's capability to ensure a responsible behavior.*

This challenge is specific to our market scenario as there is no future obligation in systems such as ZeroCash [164], Anonymous E-cash [163], or Reputation System [186] where the coins are transferred directly from the sender to the receiver or the reputation of an actor is immediately aggregated and no string attached afterward.

Implementing this principle in a futures market is almost trivial for cryptographers using the 'commit and prove' paradigm: a trader first bootstraps (by committing) the secret initial margin to make a deposit from a verifiable cash source. Such source must acknowledge the debit of the deposited amount and the credit of output reward, e.g. in ZeroCash [164, Section I-B] one can modify the POUR circuit to debit deposits and credit rewards using the output of the financial protocol execution. All participants should keep track of each other's (secret) inventory (as a commitment) as the market evolves. When a trader posts an order, the margin condition must be satisfied, e.g. Alice proves in *zero-knowledge*[5] that her position (another commitment computed from the inventory commitment using the new order book) of the new order posted by somebody is above the maintenance margin.

For a futures exchange protocol to be viable, Principle 1 (anonymous and shielded actions) and 2 (responsible and controlled actions) must be satisfied even though they at first appear to be conflicting with each other: the protocol must make sure Alice meets her

[5]Informally speaking a zero-knowledge proof is a method one party can prove to another a given statement, without conveying any additional information apart from the statement is indeed true. For further detail see §5.3.

obligations against the other three players and at the same time protect Alice's position through reverse engineering of her actions. However, a protocol can resolve this issue by using memoization to avoid the MPC when addressing the conflicting requirements of (i) providing a public trail of events, (ii) publicly verifying a constraint on a private subset of such events, (iii) showing that such private events are *all and only* applicable events. See §4.4 for further details.

3.2.3 Manage Non-Monotonic (Honest) Evolution

Again, we explore the example in Fig. 3.3 (middle left) and assume that *all parties behaved honestly*. In the beginning, Alice has proved (in zero knowledge) that her position was within her means (she proved $cash + volume \times 10 \geq 0$, where $volume = -90$ and $cash = 1400$ were secret values only known to her; however, previously committed to a public ledger). Unfortunately, Carol just wanted to buy more futures of barrels of oil for her factory to offset troubles in Venezuela. Carol goes ahead and wipes the order of Eve at \$11. The price has fluctuated for no fault of Alice nor because of any malicious strategic intent by Carol (unlike Eve in Principle 1). Price has rocketed to \$16 and *the action by the honest Carol had economically invalidated Alice's original security proof.*

Principle 3 (Manage Non-Monotonic (Honest) Evolution)**.** *If the attributes of some actors might evolve in time owing to the behavior of* other *honest participants, it is a good practice to assume that the security credentials certifying those attributes will evolve in a non-monotonic way and therefore credentials must be either revoked, or the attributes must be periodically refreshed at each point where such other parties might be acting in the protocol.*

To understand the deep design implication of this phenomenon, let us look at other *monotonic* protocols such as e-voting (casting one's vote) or simple payment protocols (transferring one's coins).

- In E2E voting [117], a voter will receive from the Election Authority a voting card with authentication code and a vote code. Vote eligibility (a correct authentication code and a well-formed vote code) cannot be changed by a vote of a *different* voter, which claims another authentication code (again unless the authentication code is used twice by the same voter but honesty reigns here) as the other votes yield no direct effect against such vote.

- To make a ZeroCash transaction [164, POUR circuit, Section I-B], a payer broadcasts the payment information and some zero-knowledge proofs. The miners check the ZK proofs: (i) the spending coin belongs to an unspent set of coins maintained as a

Merkle Tree; (ii) the payer knows a secret parameter to unlock the spending coins; and (iii) the total amount of the new coins is within the total amount of the spending coins. Another spent coin (except for double-spending in which the same coin is paid twice but this case should be ruled out as everyone is honest here) cannot invalidate any of the above proofs.

The same phenomenon happens in privacy-preserving reputation systems [186], which evaluate information quality and filter spam by providing linkage between user actions and feedback. In such systems, Alice's reputation, once gained, cannot be affected by Bob's actions to increase his own reputation. Hence, security evidence for reputation grows monotonically over honest traders actions.

The famous Danish Sugar Beet auction [38] was an example of monotonic bidding against fixed prices. A total of 400 fixed price levels were available, and everybody entered the bid to signify the amount of product they would like to buy (or sell) at each price level. Bob's bid (cryptographically represented as three secret shares) would not make Alice's bid invalid, (which were three other independent shares). The three servers (each receiving one share by each bidder) would then perform an MPC computation to add up the quantities at each price level and determine the mid-price (where supply would equal demand). Everybody who had bid at that price would have to sell/buy.

Consider the simple case of a single legitimate protocol run that comprises of multiple steps and potentially never stops.[6] Clearly, the security evidence in a step must be valid at once after the step completed. In the next step, other *honest* parties may perform some actions. If such actions do not invalidate the security evidence, *security is monotonic in the action of honest parties*.[7]

In a futures market, traders take positions (accumulating contracts in inventory) by posting buy and sell orders, which effectively changes the market price and directly affects the validity of everybody else trading inventories. Thus, the *economic constraint now involves all parties after an action made by an individual party.* In a centralized setting, the Exchange maintains the invariant. In a distributed setting, the *validity of the individual security proofs may be non-monotonic* as more honest parties join. To guarantee such non-monotonic security, each time an order arrives, all parties join forces (using an MPC) and produce a new Order Book when traders with negative positions are netted out. See §4.4 for further details.

Unfortunately, MPC, as we will discuss in the next section §3.2.4, turns out to introduce more problems than it solves.

[6]Security evidence created during a protocol run should not extend beyond that run. Several protocol failures are indeed due to protocol design errors where a credential could be used across sessions [1].

[7]We only consider the case of one protocol. For multiple protocols, monotonicity may not hold. A simple example is the revocation of credentials.

3.2.4 Account for a Large Number of Parties

A small market such as Lean Hog may includes at time 100 traders, whereas a fast and big market, e.g. Eurodollars, consists of 500+ traders (See Table 3.2). This is far more compared to most MPC empirical papers which are typically run with 2 or 3 parties.

The first largest claimed practical MPC is an auction of the Danish sugar beet where 1229 Danish farmers auctioned their production [38]. Yet, only *three* servers actually performed MPC over the secret shares generated by the 1229 bidders. Another paper also claimed to use MPC to perform financial trading over dark pools with high throughput [47]; however, the actual parties used to produce a high throughput are *two* or *three*.

Principle 4 (Account for a Large Number of Parties)**.** *If a large number of participants may join the protocol, the security mechanism must scale with such numbers. Using a handful of centralized intermediaries will defeat the very purpose of decentralized protocols.*

An MPC protocol relying on only a few trusted servers may be a reasonable security solution until one realizes that *the economic incentives are stacked against it.*

Take zcash [185], the real world deployment of ZeroCash, as an example. Since it relies on zk-SNARK [164], it requires a secure multi-party setup ceremony where the common reference string for the zero-knowledge proof (built upon what they call the toxic waste which, once known, allows one to counterfeit zcash) can be securely obtained without the toxic waste leak [20]. The first ceremony (Sprout) included only six individuals, and even though they claimed that "as long as at least one of the participants successfully deleted their private key, the toxic waste is impossible for anyone to reconstruct;" however, until now five out of six individuals' identity have been revealed (except for 'John Dobbertin'), and it clearly makes zcash vulnerable to the toxic waste reconstruction [184]. This is fixed with the second ceremony (Sapling) where the number of participants significantly increased to over 87 for the first stage and 90 for the second stage of the ceremony [183].

In a futures market, the individual traders have incentives to participate in the market as they hope to benefit from it. What about the servers? *To be trusted by a trader, a server must not financially benefit from the direction of the trade.* Yet, the server must make money to support the computational infrastructure of the trades. Some papers envisage that a server could be run by a regulator [47]. Let alone any political comment on the trust in regulators, very few regulator has the computational grit: in Europe, only the Bundesbank and the Banca d'Italia who both run TARGET2 gross settlements could perform the task. The Bank of England, the regulator of the largest exchange at the time of writing, collected bank stress data in Excel. Thus, regulators should build and pay for their infrastructure (or more likely charge the traders). Once a trader needs to pay his trusted servers, one can just pay the CME.

Here principles interact with dire consequences for security design. Monotonic security allows efficient optimizations: costly MPC with n interacting parties may be replaced by n parallelizable commitments and ZK non-interactive proofs. This replacement is possible when a party should make changes to their old secret values based on some public information and prove the correctness in zero-knowledge, e.g. ZeroCash [164]. This seems to be our case: a trader owning secret inventory and making public offers must update only the secret inventory and prove correctness in zero-knowledge (see the commit-and-prove approach §4.6).

Yet, the futures market is non-monotonic: a trader may change the market price thus invalidating (economically) all validity proofs of other traders (see §3.2.3). We require each trader to prove the satisfaction of the economic validity of its position each time a new order arrives. This conflicts with Principle 1 in the case when one party alone cannot prove the validity. Hence *some* MPC is needed.

3.2.5 Guarantee Proportional Burden

In most MPC protocols, every (honest) user does the same action: in auctions, every bidder makes *one* bid (or bids over multiple levels once [38]); or in e-voting each voter casts *one* vote.[8]

Principle 5 (Guarantee Proportional Burden)**.** *If some actors are more active than other actors, the security protocol must be designed such that the cryptographic computational effort of actors is proportional to the number of actions started by each of them.*

In the e-voting example, Alice is not expected to do more crypto work when she is casting her own vote than when Bob is casting his vote. Only at the end, all parties join the effort to avoid risks (e.g. compute election results). All those protocols implicitly impose a *proportional burden* on each actor: each computation is a burden for the party benefiting from it and the burden is essentially fair.

This works because we only need to compute the election result once. Running MPC for every trade has some practical implications when some traders only make a few operations while others make thousands or hundreds of thousands. Take the Bitcoin network as an example, some clients are far more active than others (e.g. the ones who use BTC as an investing medium and actively trade them on centralized exchanges). To address the proportional burden, a transaction from a payer to a payee must leave some (small) amount as a transaction fee. A miner in the Bitcoin network is compensated for their

[8] Obviously the auctioneers and the auditors would have had more load than a bidder or a voter; however, this only happens because they must handle multiple bidders and voters in a single run of the protocol at once.

effort with those transaction fees upon finding the Proof-of-Work to extend the longest chain [144]. As a result, the more transactions a payer makes the more fees he pays.

Using the numbers from the TSX market [130], in Feb 2012, the algorithmic traders submitted an average 250000 messages per day, but made only around 5000 trades, revealing that the algorithmic traders made 245000 vacuous bids that were never matched into orders. When running an MPC, everybody takes part, i.e. institutional investors stake computational resources for 250000 trades (just to benefit from 5000 trades).

Combining Principles at once means that a suitable protocol for such scenario must be a hybrid protocol that combines MPC and non-interactive zero-knowledge proofs on committed inputs as in FuturesMEX (Chapter 4). We replace the local constraints verification of the MPC with non-interactive proofs for efficient generation of publicly verifiable transactions and scalability w.r.t. the number of traders. Full MPC is only performed for sub-tasks capturing the non-monotonicity and anonymity requirements of the market.[9]

The challenging part of the protocol design is to identify the minimal state of the reactive security functionality implementing the futures market that would account for its non-monotonic behavior in the legitimacy of traders and assets. This is the only part where MPC needs to be applied. Fig. 4.14 (in §4.10.1) shows that using generic MPC retail traders have to participate even if they do not make an order (and they overwhelmingly *do not* made many orders [130]). They have to supply algorithmic traders with some orders of magnitude of costly computing resources. In FuturesMEX, the burden on retail traders is significantly smaller. This practical constraint of proportional burden is another reason why Tor (in Principle 1) is preferable to the Dining Cryptographers Network [54].

3.2.6 Ensure Drop-Out Tolerance

From a security perspective, the above design (commitments, zero-knowledge proof, and minimal MPC), no matter how implemented, can only be secure with abort as an adversary can abort the protocol by simply not participating in a joint MPC step. The protocol hence *fails by omission*. One might argue that this failure still makes the protocol *fail-safe* [99] so that nothing is disclosed and the parties could restart as if nothing happened. From the perspective of the other traders, the correct definition of this behavior would be *fail-useless*. Would institutional or retail investors ever join if by mistake or mischief an algorithmic trader could fail safe to 'nothing done' a day of costly MPC computation?

Principle 6 (Ensure Drop-Out Tolerance)**.** *If an actor may walk away when unhappy with the likely outcome or fail to act upon one's own honest failures, then the protocol must financially penalize such actor in proportion to its stake.*

[9] This does not violate the proportional burden requirement as each trader has the responsibility to prove the solvency if s/he still wants to be in the game (see Table 3.3.

A preliminary observation is that in practice one cannot initialize a market with a self-claimed account. The cash deposited into the market must be backed by a verifiable source where a debit is acknowledged by every market participant, e.g. zcash.[10] An approach is to penalize a faulty participant upon aborting in an MPC is to make the adversary lose some digital cash in proportion to their stakes. For instance, Kumaresan *et al.* [122] requires the adversary to make deposits and forfeit them upon dropping out. Unfortunately, not all protocols are usable in our scenario. Technically, the parties must move *in a fixed order* since *order of revelation is important* (the See-Saw mechanism, [122, p. 7]) for the aforementioned penalty mechanism to work. This fixed order may conflict with a protocol's anonymity requirement since this will reveal the identity of the trader who made a posting. Most importantly, those protocols are *not economically viable* as the baseline deposit would need to be progressively staggered in a See-Saw fashion. This mechanism is unachievable in practice owing to the anticipated variety in the financial capability of traders. In a low-frequency market, the last trader should deposit assets 67 times the stake of the trader. In large markets that increases to 1067 times larger (see Table 3.2, where a single Eurodollar contract has a notional value of 1 million dollars and margins are measured in basis points).

Hawk [119] is indeed a better solution against omission, since private deposits from the cash source can be frozen and the identified aborting parties cannot claim the deposits back in the withdraw phase. The protocol must provide security tokens of successful completion and identify evidence in case of misbehavior and in case of aborts. We refer the reader to §4.9 for additional discussion.

3.3 Summary

We have shown the example of a Futures Exchange, such as the Chicago Mercantile Exchange, where traders buy and sell contractual promises (futures) to acquire or deliver, at some future pre-specified date, assets ranging from wheat to crude oil and from bacon to cash in a desired currency. In such an exchange, the interplay between security and economic viability (as illustrated by the Price Discrimination Attack) and the exchange's essentially non-monotonic security behavior (a valid action by a trader can invalidate other traders' previously valid positions) are indeed novel challenges for security research. These challenges have deep implications for efficient designs of security protocols for financial intermediation, in particular, if we need to guarantee a *proportional burden* of computation to the various parties.

[10] Indeed, such source must be able to publicly verify the validity of the transactions from the market's operation at the end of the day to credit each the account with the corresponding amount.

In the next Chapter 4 we will show how to apply the enucleated principles to construct a concrete cryptographic and distributed implementation called FuturesMEX that securely replicates the ideal functionality of a centralized Futures Market Exchange such as the CME.

Chapter 4

FuturesMEX: A Secure, Distributed Futures Market Exchange

A futures contract is a standardized agreement between two parties to buy or sell an amount of an underlying asset, at a price agreed upon today with the settlement occurring at some future date. Such trading is carried out in a double auction market operated by a centralized clearinghouse called Futures Exchange. This chapter describes all key operations for a secure, fully distributed Futures Exchange, hereafter referred to simply as the 'Exchange'. Our distributed, asynchronous protocol simulates the centralized functionality in the assumptions of anonymity of the physical layer and availability of a distributed ledger. We consider security with abort (in absence of honest majority) and extend it to penalties. Our proof of concept implementation and its optimization (based on zk-SNARKs and SPDZ) demonstrate that the computation of actual trading days (along Thomson-Reuters Tick History DB) is feasible for low-frequency markets; however, more research is needed for high-frequency trading.

4.1 Formal Futures Market Definition

A futures market consists of N traders, in which each trader is identified via an index $i \in [N]$ and a sequence of L available prices[1] (for the limit orders) in ascending order (i.e. $p_\ell < p_{\ell'}$ if $\ell < \ell'$ for $\ell, \ell' \in [L]$). The market evolves in rounds, where T is the

[1] In the CME Globex, trading operations starts with an indicative opening price (IOP). Other prices are an integer number of upward or downward ticks from the IOP. A price is always non-zero and each underlying asset of a futures contract usually has a reasonable upper bound for the price. Hence we can map possible prices into a finite list of L available prices and refer to a price only with its index ℓ.

maximum (constant) number of rounds[2]. The data stored (and updated) for the current round $t \in [T]$ is a tuple $(\mathcal{O}, \mathcal{I})$.

- The set $\mathcal{O}$ is the *limit order book*, and consists of a sequence of tuples $o' = (t', \ell', i', v')$, where o' represents a limit order posted at round $t' \leq t$ by a trader $P_{i'}$ for a desired volume $v' \neq 0$ of price $p_{\ell'}$. A limit order is called a "sell" order if $v' < 0$, otherwise, called a "buy" order.
- $\mathcal{I}_i = (m_i, v_i)$ is the *inventory*[3] of a trader $i \in [N]$ where:
  - The value v_i is the number of contracts held by the trader (for long positions $v_i > 0$, for short ones $v_i < 0$);
  - The value m_i is the cash available to the trader.

Initially, every trader starts with no contract in the inventory and a non-negative deposit[4] (i.e. $\forall i \in [N] : v_i = 0, m_i \geq 0$), and the market is initiated with an empty order book (i.e. $\mathcal{O} = \emptyset$).

To express the constraint that a trader can meet her obligations and make orders within her means we introduce some auxiliary functions. The *instant net position* η_i is the cash she can get (or must pay) upon liquidating all her contracts:

$$\eta_i = m_i + \mathsf{cash}(v_i) \tag{4.1}$$

where $\mathsf{cash}(v_i)$ represents the *liquid* value of the inventory, i.e. the amount of cash a trader P_i can get (or must pay) upon selling (or buying) all volume holding v_i at the current buy (or sell) quotes in the order book.

[2]At CME an open cry starts at 7:20 and ends at 13:59:00, the evolution of time is accounted for with the number of rounds.

[3]For simplicity, we assume that a trader's deposited cash is all the cash available to that trader and s/he only trades a single kind of contract in a market. To support the trading of multiple types of contract, one can replicate the protocol for each type of contract trading as the markets are indeed separated. For each contract, a trader has a separate margin requirement as one may not necessarily reconcile Lean Hogs versus Eurodollars (they have different accountability limits, different liquidity, different regulatory requirements, etc. [60]). *If* one has a global trade account reconciliation mechanism for margins with some usefulness, the complexity might be too high. Indeed, the current solution of *centralized* (!) markets is to let traders deal with this at their end privately, as this is something that each trader must be able to deal with. Cash for margins could be transferred from one account to another. Technically, this is equivalent to withdraw the cash from one trading account, transfer it back on the ledger and deposit it in another trading account. This operation of withdrawal and deposit of more money in the trading accounts is not new as it is simply a payment transfer (See the previous Chapter 2). Hence it is omitted.

[4]As mentioned in §2.3, we can bootstrap the market with zcash [185], we can simply extend its POUR transaction [164] to accept one more input/output: a cash commitment. The additional output from the POUR transaction can be used as an input for the initialization of the market for a trader to deposit some cash into the market from the zcash network. The final cash commitment is an output of the finalization of the market used as the input to the POUR transaction for a trader to withdraw the market's digital cash back to the zcash network.

Table 4.1: Market Indicators for the current round t of the Futures Market

Indicator	Notation	Definition	Description
Best sell price index	ℓ_{sell}	$\min\{\ell' \mid (t', \ell', i', v' < 0) \in \mathcal{O}\}$	Index of the lowest price of all sell orders in the order book.
Best buy price index	ℓ_{buy}	$\max\{\ell' \mid (t', \ell', i', v' > 0) \in \mathcal{O}\}$	Index of the highest price of all buy orders in the order book.
Mid price	$\bar{p}$	$(p_{\ell_{\mathsf{sell}}} + p_{\ell_{\mathsf{buy}}})/2$	The average value of the best buy price $p_{\ell_{\mathsf{buy}}}$ and best sell price $p_{\ell_{\mathsf{sell}}}$
Available volume at price p_h	V_h	$\sum_{(t' \leq t, h, i', v') \in \mathcal{O}} \lvert v' \rvert$	The sum of absolute volumes over all orders at price p_h
Available sell volume up to p_h	V_h^{sell}	$\sum_{\ell=\ell_{\mathsf{sell}}}^{h} V_\ell$	Aggregation of all volumes available from the best sell price ℓ_{sell} to the final maximum acceptable price p_ℓ ($\ell \geq \ell_{\mathsf{sell}}$)
Available buy volume down to p_h	V_h^{buy}	$\sum_{\ell=h}^{\ell_{\mathsf{buy}}} V_\ell$	Aggregation of all volumes available from the final least acceptable price h to the best buy price ℓ_{buy} ($\ell \leq \ell_{\mathsf{buy}}$)

Table 4.2: Value $\mathsf{cash}(v)$ to liquidate an inventory of volume v

Cases	Definition	Description
$v > 0$ (long) and $V_1^{\mathsf{buy}} \geq v$	$\sum_{h=\ell+1}^{\ell_{\mathsf{buy}}} p_h \cdot V_h + p_\ell \cdot (v - V_{\ell+1}^{\mathsf{buy}})$	Cash a trader can get upon selling all volume v at the current buy quotes in the order book, where ℓ is the greatest index such that $V_\ell^{\mathsf{buy}} \geq v$.
$v > 0$ (long) and $V_1^{\mathsf{buy}} < v$	$\sum_{h=1}^{\ell_{\mathsf{buy}}} p_h \cdot V_h + p_1 \cdot (v - V_1^{\mathsf{buy}})$	The order book *does not have enough* supply on the buy side.
$v < 0$ (short) and $V_L^{\mathsf{sell}} \geq \lvert v \rvert$	$-\sum_{h=\ell_{\mathsf{sell}}}^{\ell-1} p_h \cdot V_h - p_\ell \cdot (V_{\ell-1}^{\mathsf{sell}} - \lvert v \rvert)$	Cost a trader must pay to buy all volume v from the current sell quotes in the order book, where ℓ is the least index such that $V_\ell^{\mathsf{sell}} \geq \lvert v \rvert$
$v < 0$ (short) and $V_L^{\mathsf{sell}} < \lvert v \rvert$	$-\sum_{h=\ell_{\mathsf{sell}}}^{L} p_h \cdot V_h - p_L \cdot (V_L^{\mathsf{sell}} - \lvert v \rvert)$	The order book *does not have enough* supply on the sell side.

The function $\widehat{\cdot}$ represents the estimated value of a trader's inventory variables if the market accepted her new order. Auxiliary definitions for the calculation of market conditions are listed in Table 4.1 (mid-price, best sell price[5], etc.) while $\mathsf{cash}(v_i)$ is defined in Table 4.2. In case of insufficient supply, our formula assumes that arbitrarily more supply would become available at the worst possible price (p_L for buy and p_1 for sell). For the estimated value of the inventory, when a trader P_i posts an order (t, ℓ, i, v) at price p_ℓ for

[5]Technically, if the order book is empty on the sell side or the buy side, the respective best price can be set to *unidentified*. Then, the first order will set the best price. For simplicity of exposition, we can assume that the first buy (or sell) order will set the best buy (or sell price); then, the second order of the opposite type will set the best sell (buy) price. In this way, we have an operable order book after the first two rounds. In practice, the CME uses a Pre-Open phase for an operable order book before openning the market for actual trades [61]. It is possible to incorporate this phase into our protocol as an additional MPC that computes the Pre-Open phase before running FuturesMEX.

Table 4.3: Formal Security Requirements

The two requirements of traders confidentiality and anonymity imply that $\widehat{m_i}$ and $\widehat{v_i}$, as well as $\widehat{\eta_i}$ must also be confidential (otherwise one could recover the inventory by reversing the computation from orders).

Property	**Description**
Confidentiality of Trader Inventory	Only P_i knows the values of $\mathcal{I}_i = (m_i, v_i)$ as well as η_i with the exception of time T after mark-to-market when $v_i = 0$.
Market Integrity	Barring withdrawals and deposits the amount of cash available by all traders is constant[7] ($\sum_{i=1}^{N} m'_i = \sum_{i=1}^{N} m_i$) where m'_i is the margin at time $t' \leq t$), the total volume holding is zero ($\sum_{i=1}^{N} v_i = 0$), and the best buy price is less than the best sell price ($1 \leq \ell_{\mathsf{buy}} < \ell_{\mathsf{sell}} \leq L$).
Loss Avoidance	All traders have a positive instant net position ($\eta_i \geq 0$) and can afford the new limit order at posting time ($\widehat{\eta_i} \geq 0$).
Trader's Anonymity	For any order (t, ℓ, i, v) posted at time t, the order information (t, ℓ, v) must be made public before time $t+1$, while the value i is only known to P_i. It is also important that i is unlinkable to an existing order $o' = (t', \ell', i, v')$.
Trader's Precedence Traceability	Let $\mathcal{O}$ be the current order book, (t, ℓ, i, v) be an order, and t' be the smallest round $t' < t$ such that $(t', \ell, i', -v') \in \mathcal{O}$ then the order book $\mathcal{O}^*$ at time $t+1$ *respects traders precedence* given order (t, ℓ, i, v) and order book $\mathcal{O}$ iff 1. if no such t' exists for $\mathcal{O}$, then $\mathcal{O}^* = \mathcal{O} \cup \{(t, \ell, i, v)\}$, 2. if $\lvert v\rvert < \lvert v'\rvert$, then $\mathcal{O}^* = \mathcal{O} \cup \{(t', \ell, i', v - v')\} \setminus \{(t', \ell, i', -v')\}$ 3. else $\mathcal{O}^*$ respects traders precedence given order $(t, \ell, i, v - v')$ and order book $\mathcal{O} \setminus \{(t', \ell, i', -v')\}$

a volume v in round t, we have:

$$\widehat{m_i} = m_i - p_\ell \cdot v, \quad \widehat{v_i} = v_i + v, \quad \widehat{\eta_i} = \widehat{m_i} + \mathsf{cash}(\widehat{v_i}) \tag{4.2}$$

We can now formalize in Table 4.3 the properties, which must hold at every round,[6] corresponding to the security/economic requirements informally introduced in Table 3.3. We will also discuss the *proportional burden* property, a practical requirement that we introduced in Chapter 3, in the description of the futures market functionality.

4.2 The Ideal Reactive Functionality

For expository purposes, both in the functionality's and in the protocol's description, we allow an adversary to abort the computation after receiving its own intermediate

[6]Since the order book is public and contains all orders, it is obvious that any $N-1$ colluding parties could reverse engineer the trading strategy and position of the lone honest trader. This is also true for a "normal" centralized exchange such as the CME Globex platform. Since our target is to be as secure as the centralized exchange, we ignore this and other unrealistic attacks that could be carried on the centralized exchanges.

[7] Technically deposit and withdraw operations are not new [164] (see the possible solutions in Chapter 2). To support such operations in FuturesMEX one can simply add new phases to such Deposit (or Withdraw) into the protocol similar to other steps to pour money into one's account as in **Initialize** (or withdraw from one's account but still satisfy the margin condition as in **Post/Cancel Order**).

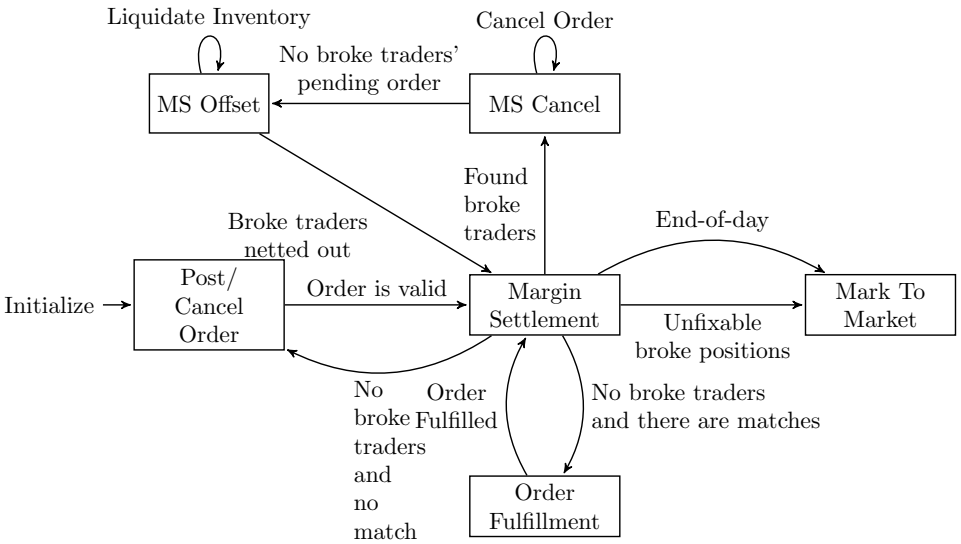


Figure 4.1: Market State Transition Diagram

outputs. This flavor of security is known as security with aborts [111]. In §4.9, we change the protocol to avoid scot-free aborts.

The futures market evolution is captured by an ideal reactive functionality $\mathcal{F}_{\mathsf{CFM}}$ where all traders send their private initial inventory to a trusted third party (during the so-called **Initialize** phase) which makes the market evolve on their behalf. A typical evolution of the market includes processing orders (**Post/Cancel Order** phases) and netting out traders with insufficient funds (we refer to these traders hereon as "broke" traders) to maintain their position (**Margin Settlement** phase), and finally offsetting all positions (**Mark to Market** phase). This evolution is summarized in Fig. 4.1. and a formal description is found in Fig. 4.2 and Fig. 4.3.

Intuitively, the matching process performed during the **Post Order** phase (see Fig. 4.2) takes the new order (t, ℓ, i, v) and matches it with all previous limit orders of the opposite side in the order book having the same price. In other words, if the limit order is a buy order, it will be matched with a sell order, and vice versa. The priority to match is given to the limit order with a smaller round index. When a match is found, the trade is reconciled, and the available cash and the volume holding of the traders are updated accordingly (i.e. on buy side: increase volume, decrease cash; on sell side: decrease volume, increase cash). The matching process stops either when the new order is fulfilled, or no past order that can fill the new one is available. In the latter case, the remaining volume is left in the order book as a new limit order.

An important feature of $\mathcal{F}_{\mathsf{CFM}}$ is to guarantee payable losses by each trader (i.e. $\eta_i \geq 0$).

Futures Exchange Ideal Functionality $\mathcal{F}_{\mathsf{CFM}}$ runs in phases with a set of traders $(P_1, \ldots, P_N)$ and a list of prices $(p_1, \ldots, p_L)$ in ascending order.

Initialization: Upon $(\mathsf{init}, P_i, m_i)$ from all traders, accept the input iff $m_i \geq 0$. Hence, store $(m_i, v_i := 0)$ as the inventory of P_i. Finally, initialize $t := 0$ and $\mathcal{O} := \emptyset$.

Post/Cancel Order: If $t < T$ (otherwise go to **Mark to Market**), upon receiving $(\mathsf{post\_order}, P_i, \ell, v)$ (resp. $(\mathsf{cancel\_order}, P_i, t')$) from P_i, let $t := t + 1$:

1. Check $\ell \geq \ell_{\mathsf{buy}}$ for $v < 0$ ($\ell \leq \ell_{\mathsf{sell}}$ for $v > 0$). In case of **Cancel Order**, retrieve (t', ℓ', j, v') from $\mathcal{O}$ and check $j = i$..
2. Let $\mathcal{I}_i^*$ be an identical copy of $\mathcal{I}_i$, check $\widehat{\eta_i} \geq 0$ w.r.t. to $\mathcal{I}_i^*$ and the order book $\mathcal{O}^* := \mathcal{O} \cup (t, \ell, i, v)$ (resp. $\mathcal{O}^* := \mathcal{O} \setminus (t', \ell', j, v')$):
3. If any check fails, send $(\mathsf{invalid\_post}, t, \ell, v)$ (resp. $(\mathsf{invalid\_cancel}, t')$) to every trader; else send $(\mathsf{post\_order}, t, \ell, v)$ (resp. $(\mathsf{cancel\_order}, t')$) to every trader and proceed to **Margin Settlement** with input $\mathcal{I}_i^*$ and $\mathcal{O}^*$ (c.f. Fig. 4.3). ; if "succeed", let $\mathcal{I}_i = \mathcal{I}_i^*$, $\mathcal{O} := \mathcal{O}^*$, otherwise proceed to **Mark to Market**.
4. In case of **Post Order**, fulfill the order starting from the earliest opposite order of the same price already in the order book, until the new order is filled or there is no past order to match it with. (c.f. **Order Fulfillment** in Fig. 4.3).

Mark To Market: offset all positions using the mid-price, i.e. $\forall P_i : m_i := m_i + v_i \cdot \bar{p}$, and $v_i := 0$.

Figure 4.2: The operations of the ideal functionality $\mathcal{F}_{\mathsf{CFM}}$ for posting, cancelling and marking to market

Hence, when the last round is reached, all traders must offset their position,[8] and the data at round T will consist of all zero volumes, non-negative balances, and an empty order book.

Since the net position might change owing to the updates of the order book, it is necessary to check the new instant net position η_i^* of each trader P_i after the update. In case of any negative net position (i.e. $\eta_i < 0$), the last update is rolled back and all traders enter the **Margin Settlement** (MS) phase (see Fig. 4.3). This requires each new broke trader to cancel all pending orders (becomes *canceled* in MS Cancel) *according to the time of submission*, and buy/sell all contracts in the inventory that the trader is short/long, at whatever price available at the moment (becomes *netted* in MS Offset). At the end of the **Margin Settlement** phase the order fulfillment is resumed, and the update will be committed.

Remark 4.1. *Since the order of netting out broke traders affects the final positions of such traders (the early netted out traders may wipe the best price) it is important that the*

[8] We will use the mid-price to offset all positions in our functionality as it has the simplest formula. Our protocol construction is unaffected even if the formula is changed since this input is public. In practice, one often uses the volume-weighted-average-price (VWAP) of the last minute trades. The mid-price is only used when within the last one minute there is no trade [59]. VWAP, compared to mid-price, is less but still dependent on the last minute single order. However, shifting the VWAP or the mid-price is not trivial as one may need sufficient cash to perform this (sometimes this can be expensive).

Margin Settlement is run with a candidate order book $\mathcal{O}^*$ and a candidate inventory $\mathcal{I}^*$ starting with a set of new broke traders $\mathcal{B} := \emptyset$.

1. Repeat the following steps until $\eta_i^* \geq 0$ for all current good traders $P_i \notin \mathcal{B}$:
   (a) Compute the new instant net position η_i^* of all good traders P_i; if $\eta_i^* < 0$ let $\mathcal{B} := \mathcal{B} \cup \{P_i\}$.
   (b) While there is a limit order $o_i := (t', \ell', i, v')$ in either $\mathcal{O}^*$ and $\mathcal{O}$ such that $P_i \in \mathcal{B}$, then poll P_i to remove o_i from both $\mathcal{O}^*$ and $\mathcal{O}$, send $(\mathsf{remove}, (t', \ell', v'))$ to each trader, on a first come first served basis.
2. The ideal functionality will poll the broke traders $P_i \in \mathcal{B}$ for market orders (with fixed input $(t, \ell_{\mathsf{sell}}, i, v)$ for short position or $(t, \ell_{\mathsf{buy}}, i, v)$ for long position and v must be at the same time within their holding or owed volume and the current available volume at the market price) to net each of them out (the chance to take the better price are left to the broke traders in a first come first served manner) until $\mathcal{B} := \emptyset$:
   (a) If the market cannot supply the margin settlement of P_i, i.e, there is no order to match, return "fail". Otherwise run the matching order steps (from step 1 to step 4) of Order Fullfilment.
   (b) If $v_i = 0$, let $\mathcal{B} := \mathcal{B} \setminus \{P_i\}$.
3. Go back to step 1 if there are new good traders P_i becomes broke, i.e. P_i was not in $\mathcal{B}$ and $\eta_i < 0$.
4. Else if there is a broke position that cannot be fixed ($m_i < 0$ even if $v_i = 0$), return "fail".
5. Else return "succeed".

Order Fulfillment for $o_i = (t, \ell, i, v)$ starts with $t' = 1$, repeat the following for each entry $o_j = (t', \ell, j, v') \in \mathcal{O}$ such that $v \cdot v' < 0$:

1. Send $(\mathsf{match}, t', \ell, v')$ to each trader;
2. Compute the matched volume $\delta := \mathsf{min}(|v|, |v'|)$, then remove δ from o_i and o_j, i.e. in case $v > 0$, $o_i^* := (t, \ell, i, v - \delta)$ and $o_j^* := (t', \ell, j, v' + \delta)$ (otherwise swap i and j).
3. Let $\mathcal{O}^*$ be an identical copy of $\mathcal{O}$, where the orders o_i and o_j are replaced, respectively, with o_i^* and o_j^*.
4. In case $v > 0$, update the inventories as follows (in case $v < 0$, swap i and j in the equations below):

$$m_i^* := m_i - p_\ell \delta \qquad v_i^* := v_i + \delta \qquad\qquad m_j^* := m_j + p_\ell \delta \qquad v_j^* := v_j - \delta;$$

5. Let $\mathcal{I}^*$ be an identical copy of $\mathcal{I}$ where the inventories of P_i and P_j are replaced, respectively, with (m_i^*, v_i^*) and (m_j^*, v_j^*).
6. Run **Margin Settlement** with input $\mathcal{O}^*$ and $\mathcal{I}^*$.
7. If **Margin Settlement** returns "fail", proceed to **Mark to Market** (Fig.4.2) otherwise, let $\mathcal{O} := \mathcal{O}^*$, $\mathcal{I} := \mathcal{I}^*$, and:

$$\text{if } v' = 0, \text{ let } \mathcal{O} := \mathcal{O} \setminus o_j; \qquad \text{if } v = 0 \text{ , let } \mathcal{O} = \mathcal{O} \setminus o_i$$

8. Define $t' := t' + 1$, and repeat the above until $t' = t$ or $v = 0$.

Figure 4.3: The operations of the ideal functionality $\mathcal{F}_{\mathsf{CFM}}$ for margin settlement and order fulfillment

ideal functionality is agnostic to a particular order of netting out broke traders *and such order is left to the broke traders' own action. A broke trader needs to look up the best available order* on the opposite side *and* posts a market order of the same price. *The new order's volume must be within the current holding volume of the trader and the available volume at the best price. This means that a broke trader may need to post multiple market orders to finally offset her position.*

For simplicity, after a trader P_i is *netted out*, the trader cannot post a normal new order in the market in the subsequent rounds. In the (unlikely) worst-case, a scenario where: (i) the market cannot supply the margin settlement of broke traders (because, e.g. they hold too many contracts comparing to the currently available volume in the order book), or (ii) even the margin settlement cannot bring a broke trader's position back to non-negative, the ideal functionality proceeds directly to **Mark to Market**.[9]

Non-monotonicity. A challenging feature of the futures market's ideal functionality is its intrinsic non-monotonic behavior, in a sense made precise below.

Remark 4.2. *The properties of private values of a honest trader P_i executing the ideal functionality of Fig. 4.2–4.3 are* non-monotonic *in the actions of other honest traders. Let P_i be a* good trader *(private value $\eta_i \geq 0$) at round t with order book $\mathcal{O}$. We further assume that at round $t+1$, the order book is updated to $\mathcal{O}^*$ owing to an offer posted by another good trader $P_j \neq P_i$. The new order book $\mathcal{O}^*$ affects the value* $\mathsf{cash}(v_i)$ *(Table 4.2), resulting in a* negative *instant net position η_i (Eq.* (4.1)*). This makes P_i a* bad trader *at round $t+1$, even if P_i was inactive during that round.*

Security properties. We illustrate how $\mathcal{F}_{\mathsf{CFM}}$ fulfills the security requirements of the futures market (described in Table 4.3). The *Trader Anonymity* property is guaranteed by broadcasting only $(\mathsf{post\_order}, \ell, v)$ upon receiving a $(\mathsf{post\_order}, P_i, \ell, v)$ from P_i. The same reasoning applies for canceling orders. The *Confidentiality of Trader Inventory* is guaranteed as $\mathcal{F}_{\mathsf{CFM}}$ keeps the trader's inventory secret and all broadcastdc post_order, cancel_order, invalid_post, invalid_cancel, match and remove contain no inventory information (m_i, v_i, η_i $\widehat{m_i}$ or $\widehat{v_i}$). Furthermore, all computations of $\mathcal{F}_{\mathsf{CFM}}$ respect the conditions of Market Integrity in Table 4.3. The *Trader's Precedence Traceability* property is also maintained owing to the followings: (i) only the owner of an order can match/cancel that order, and (ii) only a *good* trader can post/cancel in normal phase, whereas only *broke* traders can cancel, and *canceled* traders can post during the margin settlement phase.

The practical constraint, i.e. *Proportional Burden* (see Principle 5 in §3.2), is obviously satisfied by the centralized functionality (if you do nothing you do not talk to the

[9] In such cases our FuturesMEX protocol is finalized and can be restarted.

functionality). Thus, we return to its satisfaction of the actual distributed protocol in §4.6.

4.3 Assumptions and Crypto Building Blocks Overview

We use several standard crypto blocks for both protocol construction and reliability of security proofs.

Anonymous Communication Network and Secure Broadcast Channel Recall that the ideal functionality of a futures market guarantees full anonymity of the traders. To this end, we assume an underlying anonymous network that hides the traders' identities (e.g. IP addresses).[10] Typically, a trader always uses this anonymous channel unless it is a joint computation during an MPC. Numerous prior works have already used this assumption, notably ZeroCash [164]. Moreover, we assume secure broadcast channels between traders, and such channels are implemented through a consensus protocol, e.g. PBFT [48].[11] Finally, we assume that all traders are online during the protocol, and availability attacks, e.g. Denial of Service, are considered solved by using an external mechanism, e.g. Anti DDOS Protection Service such as CloudFlare [58], or buying optical fiber [83] (already used in a centralized exchange).[12]

Commitment Schemes. We rely on a non-interactive commitment scheme Com, with domain $\{0,1\}^*$. We write $[\![v]\!] := \mathsf{Com}(v; r_v)$ for a commitment to a value v using randomness $r_v \in \{0,1\}^*$. To open a given commitment $[\![v]\!]$, it suffices to reveal (v, r_v) so that a verifier can check $[\![v]\!] = \mathsf{Com}(v; r_v)$. For the proof of security, $[\![v]\!]$ need to statistically hide the committed value v. After publishing $[\![v]\!]$, it is computationally infeasible to open the commitment in two different ways. We follow Goldreich [95] for the formal definitions. We use the following standard *NP* relations: (i) R^{oc}, for validity of commitments and ownership of an opening; (ii) R^{zero^-} (resp. R^{zero^+}, R^-, R^+) for commitments to non-positive (resp. non-negative, negative, positive) values; (v) R^{ec}, for equality of two openings; (iv) R^{nec}, for commitments to values different from a constant; (v) R^{lec} (R^{gec}) for a values less than or equal to (greater than or equal to) a constant and (vi) R^{or} for a commitment to a value of opposite sign and has an absolute value greater than or equal to a constant (see Table 4.7.)

[10] In case of using Tor, a trader must build a new Tor circuit for each round to avoid linkability.

[11] The consensus protocol is unnecessarily performed by all traders. A scalable solution is to use a set of semi-trusted servers (whose size is much smaller compared with a fewer number of traders) for consensus since no secret is recorded in such a channel.

[12] In case a trader aborts intentionally, s/he will be penalized. See §4.9.

Hybrid Ideal Functionalities. To implement $\mathcal{F}_{\mathsf{CFM}}$, we use hybrid ideal functionalities with simulation-based proofs that rely on the composition theorem [46].

All our functionalities receive the values/randomnesses and the corresponding commitments, and check whether the commitment corresponds to the claimed value, otherwise return $\perp$ (as in R^{oc}). The remaining features outlined below are specific to our application, and similar to range proofs [39, 43] (see §4.5).

- The *Secure All Non-Negative Check* functionality $\mathcal{F}_{\mathsf{anncheck}}$ receives the net position of every trader η_i and guarantees solvency (i.e. $\bigwedge_i \eta_i \geq 0$).
- The *Secure Sum Comparison* functionality $\mathcal{F}_{\mathsf{compare}}$ receives from every party a pair of old and new binary flags $\langle f_i, f_i^* \rangle$. It checks whether the total number of flags has not changed (i.e. $\sum_i f_i = \sum_i f_i^*$).
- Finally, the zero-knowledge (ZK) functionality $\mathcal{F}_{\mathsf{zk}}^{R}$ is parameterized by an *NP* relation R. A trader P_i plays the role of a prover, whereas all other traders $\{P_j\}_{j \neq i}$ play the role of verifiers. As the prover sends the statement x_i and the corresponding witness w_i to the functionality, each verifier sends its own statement x_j that needs to be checked. Each verifier gets the outcome of $R(x_j, w_i)$ if $x_i = x_j$, otherwise it gets $\perp$. For simplicity, we omit the zk subscript. MPC functionalities will be identified by subscripts (e.g. $\mathcal{F}_{\mathsf{compare}}$) and zk by superscripts (e.g. $\mathcal{F}^{\mathsf{gec}}$).

The *NP* relations for the market are summarized in Table 4.5. To describe these relations, we use some auxiliary values not needed in the ideal functionality $\mathcal{F}_{\mathsf{CFM}}$ (albeit they might be present in an actual centralized exchange implementation). These auxiliary values are defined in §4.4 and Table 4.4. Some of our relations test the requirements of Table 4.3, whereas other relations validate intermediate results in our protocol, sharing similarities with the *NP* statement POUR in $\mathsf{ZeroCash}$ [164] (see §4.5.1.)

Remark 4.3. *In this chapter, all our relations (e.g. Table 4.5, various relations in §4.5, §4.8, and §4.9) must first check whether the prover knows the secret value v and the corresponding randomness r_v of a public commitment $[\![v]\!]$ (as in R^{oc}). Thus, we only specify $[\![v]\!]$ in the statement and v in the witness, leaving r_v as an implicit part of the witness unless necessary, e.g. we specify r_v in $\mathcal{F}^{\mathsf{token}}$ since it is needed to describe the double commitment $[\![\tau_i]\!]$.*

Remark 4.4. *As we employ several zero knowledge functionalities in our protocol, instantiated with zk-SNARK [21], a first setup ceremony is required for global information such as proving keys and verifying keys, achieved securely in practice with MPC [20].*

Table 4.4: FuturesMEX Notation

Nota.	Description
ρ	Root of a Merkle tree
$path$	Authentication path of a token τ_i in a Merkle tree with root ρ
$(\hat{p}_h, V_h^{\mathsf{buy}})$	Adjusted range choice for embedding the partial sum on the buy side for long position trader to use in the net position calculation, where $\hat{p}_h = \frac{\sum_{l=h}^{\ell_{\mathsf{buy}}} p_\ell \cdot V_\ell}{V_h^{\mathsf{buy}}}$
$\mathcal{O}_{\mathsf{buy}}$	Current adjusted range choices for long position trader to use in net position calculation, defined as $\{((0,0),(p_{\ell_{\mathsf{buy}}}, V_{\ell_{\mathsf{buy}}}^{\mathsf{buy}})), ((p_{\ell_{\mathsf{buy}}}, V_{\ell_{\mathsf{buy}}}^{\mathsf{buy}}),(p_{\ell_{\mathsf{buy}}-1}, V_{\ell_{\mathsf{buy}}-1}^{\mathsf{buy}})), ((\hat{p}_{\ell_{\mathsf{buy}}-1}, V_{\ell_{\mathsf{buy}}-1}^{\mathsf{buy}}),(p_{\ell_{\mathsf{buy}}-2}, V_{\ell_{\mathsf{buy}}-2}^{\mathsf{buy}})), \ldots, ((\hat{p}_2, V_2^{\mathsf{buy}}),(p_1, V_{\mathsf{max}}))\}$
$(\hat{p}_h, V_h^{\mathsf{sell}})$	Adjusted range choice for embedding the partial sum on the sell side for short position trader to use in the net position calculation, where $\hat{p}_h = \frac{\sum_{l=\ell_{\mathsf{sell}}}^{h} p_\ell \cdot V_\ell}{V_h^{\mathsf{sell}}}$
$\mathcal{O}_{\mathsf{sell}}$	Current adjusted range choices for short position trader to use in the net position calculation, defined as $\{((0,0),(p_{\ell_{\mathsf{sell}}}, V_{\ell_{\mathsf{sell}}}^{\mathsf{sell}})), \ldots, ((\hat{p}_{L-2}, V_{L-2}^{\mathsf{sell}}),(p_{L-1}, V_{L-1}^{\mathsf{sell}})), ((\hat{p}_{L-1}, V_{L-1}^{\mathsf{sell}}),(p_L, V_{\mathsf{max}}))\}$
p_{lb}	Lower bound (adjusted) price used for the net position calculation
V_{lb}	Lower bound cumulative volume used for the net position calculation
p_{ub}	Upper bound price used for the net position calculation
V_{ub}	Lower bound cumulative volume used for the net position calculation
δ_c	Incremental value for the pending order counter

4.4 Solution Overview

The first challenging part of the protocol construction is to identify a suitable form for the state of the reactive security functionality $\mathcal{F}_{\mathsf{CFM}}$ that would account for its non-monotonic behavior in the legitimacy of traders and assets. A simple (but wrong) solution is to use just the private inventory values of the individual traders. Each trader could prove in ZK that it respects the constraints stated in Table 4.3. Unfortunately, new valid orders could make the constraints of some other traders invalid, i.e. the protocol no longer considers the ZK proof valid. Moreover, we must store the private state in a way such that after an order is accepted by the market (i.e. by the ensemble of agents) it is impossible to link it to the next order by the same trader. This is not just a union of the individual (unopened) inventories. By looking at the unchanged inventories, the traders could identify the trader who initiated the order. A global MPC step to update the entire state is a solution but it would put an unnecessary burden on inactive traders. A further challenge is to keep a fully *ordered* list (i.e. orders must be executed according to arrival time).

First, we *augment* the private state of each trader using additional information besides the inventory m_i and v_i. We *memoize* the value of the estimation $\widehat{m_i}$ and $\widehat{v_i}$, and a counter c_i to track the number of pending orders of each trader P_i. Each time a trader P_i posts an order (ℓ, v), the memoized values are updated as $\widehat{m_i} = m_i - p_\ell \cdot v$, $\widehat{v_i} = v_i + v$, and $c_i = c_i + \delta_c$ where $\delta_c = 1$. For order cancellations or complete matches of pending orders, the reverse computation is performed ($\delta_c = -1$). The use of memoized values is a quick calculation to do and to verify cryptographically. Yet, the foremost reason for such a

Table 4.5: Futures Market Relations

Relation	Additional Conditions	Statement	Witness
R^{token}	The new leaf $[\![\tau_i]\!]$ is correctly constructed from the inventory values, i.e. $[\![\tau_i]\!] = \mathsf{Com}(\mathsf{Com}(m_i\|\|v_i\|\|\widehat{m_i}\|\|\widehat{v_i}\|\|c_i\|\|f_{\mathsf{bad},i}\|\|f_{\mathsf{del},i}\|\|f_{\mathsf{out},i}; r_i); r_{\tau_i})$	$[\![\tau_i]\!], [\![m_i]\!], [\![v_i]\!], [\![\widehat{m_i}]\!], [\![\widehat{v_i}]\!],$ $[\![c_i]\!], [\![f_{\mathsf{bad},i}]\!], [\![f_{\mathsf{del},i}]\!], [\![f_{\mathsf{out},i}]\!]$	$m_i, v_i, \widehat{m_i}, \widehat{v_i}, c_i,$ $f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i}, r_i,$ r_{τ_i}
R^{inv}	The new inventory values $m_i, v_i, \widehat{m_i}, \widehat{v_i}, c_i,$ $f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i}$ are correctly constructed from an old inventory (with token τ_i'), i.e. $\mathsf{Auth}(\rho, path_i, [\![\tau_i']\!]) = 1;$ $\tau_i' = \mathsf{Com}(m_i\|\|v_i\|\|\widehat{m_i}\|\|\widehat{v_i}\|\|c_i\|\|f_{\mathsf{bad},i}\|\|f_{\mathsf{del},i}\|\|f_{\mathsf{out},i}; r_i')$	$\rho, \tau_i', [\![m_i]\!], [\![v_i]\!], [\![\widehat{m_i}]\!], [\![\widehat{v_i}]\!],$ $[\![c_i]\!], [\![f_{\mathsf{bad},i}]\!], [\![f_{\mathsf{del},i}]\!], [\![f_{\mathsf{out},i}]\!]$	$path, [\![\tau_i']\!], m_i, v_i, \widehat{m_i},$ $\widehat{v_i}, c_i,$ $f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i}, r_i'$
R^{uinv}	The new inventory values $\widehat{m_i}^*, \widehat{v_i}^*, c_i^*$ are correctly updated from an old inventory (w.r.t. δ_c, ℓ, v), i.e. $\widehat{m_i}^* = \widehat{m_i} - \delta_c \cdot p_\ell \cdot v;\ \widehat{v_i}^* = \widehat{v_i} + \delta_c \cdot v;$ $c_i^* = c_i + \delta_c.$	$[\![\widehat{m_i}^*]\!], [\![\widehat{v_i}^*]\!], [\![c_i^*]\!], [\![\widehat{m_i}]\!], [\![\widehat{v_i}]\!],$ $[\![c_i]\!], \delta_c, \ell, v$	$\widehat{m_i}^*, \widehat{m_i}, \widehat{v_i}^*, \widehat{v_i}, c_i^*, c_i$
R^{rng}	The upper and lower bounds of cumulative volumes and prices $p_{\mathsf{lb}}, V_{\mathsf{lb}}, p_{\mathsf{ub}}, V_{\mathsf{ub}}$ are correctly selected from the $\mathcal{O}_{\mathsf{buy}}$ or $\mathcal{O}_{\mathsf{sell}}$, i.e. $V_{\mathsf{lb}} \le \|v\| \le V_{\mathsf{ub}}$ and one of the following holds: $v > 0 \wedge ((p_{\mathsf{lb}}, V_{\mathsf{lb}}), (p_{\mathsf{ub}}, V_{\mathsf{ub}})) \in \mathcal{O}_{\mathsf{buy}}$ or $v < 0 \wedge ((p_{\mathsf{lb}}, V_{\mathsf{lb}}), (p_{\mathsf{ub}}, V_{\mathsf{ub}})) \in \mathcal{O}_{\mathsf{sell}}$ or $v = 0 \wedge (p_{\mathsf{lb}} V_{\mathsf{lb}}) = (p_{\mathsf{ub}}, V_{\mathsf{ub}}) = (0, 0)$	$[\![v]\!], [\![p_{\mathsf{lb}}]\!], [\![V_{\mathsf{lb}}]\!], [\![p_{\mathsf{ub}}]\!], [\![V_{\mathsf{ub}}]\!],$ $\mathcal{O}_{\mathsf{buy}}, \mathcal{O}_{\mathsf{sell}}$	$v, p_{\mathsf{lb}}, V_{\mathsf{lb}}, p_{\mathsf{ub}}, V_{\mathsf{ub}}$
R^{net}	(Estimation) of an instant net position η_i (resp. $\widehat{\eta_i}$) are correctly computed, i.e. $\eta_i = m_i + p_{\mathsf{lb}} \cdot V_{\mathsf{lb}} + p_{\mathsf{ub}} \cdot (\|v_i\| - V_{\mathsf{lb}})$	$[\![m_i]\!], [\![v_i]\!], [\![\eta_i]\!], [\![p_{\mathsf{lb}}]\!], [\![V_{\mathsf{lb}}]\!],$ $[\![p_{\mathsf{ub}}]\!]$	$m_i, v_i, \eta_i, p_{\mathsf{lb}}, V_{\mathsf{lb}}, p_{\mathsf{ub}}$
R^{match}	The order fulfillment is correctly done, i.e. $m_i^* = m_i - p_\ell \cdot v;\ v_i^* = v_i + v;\ c_i^* = c_i + \delta_c.$	$[\![m_i^*]\!], [\![v_i^*]\!], [\![c_i^*]\!],$ $[\![m_i]\!], [\![v_i]\!], [\![c_i]\!], \delta_c, p_l, v$	$m_i^*, m_i, v_i^*, v_i, \widehat{v_i}, c_i^*, c_i$
R^{flags}	The transition from the flags $(f_{\mathsf{bad},i}, f_{\mathsf{del},i},$ $f_{\mathsf{out},i})$ to the flags $(f_{\mathsf{bad},i}^*, f_{\mathsf{del},i}^*, f_{\mathsf{out},i}^*)$ is consistent with the values (η_i^*, v_i^*, c_i^*), as shown in the diagram in Fig. 4.4.	$[\![f_{\mathsf{bad},i}]\!], [\![f_{\mathsf{bad},i}^*]\!], [\![f_{\mathsf{del},i}]\!], [\![f_{\mathsf{del},i}^*]\!],$ $[\![f_{\mathsf{out},i}]\!], [\![f_{\mathsf{out},i}^*]\!], [\![\eta_i^*]\!], [\![v_i^*]\!], [\![c_i^*]\!]$	$f_{\mathsf{bad},i}, f_{\mathsf{bad},i}^*, f_{\mathsf{del},i}, f_{\mathsf{del},i}^*,$ $f_{\mathsf{out},i}, f_{\mathsf{out},i}^*, \eta_i^*, v_i^*, c_i^*$
R^{mtm}	A trader P_i is correctly marked to market, i.e. $m_i^* = m_i + \bar{p} \cdot v_i$	$[\![m_i]\!], [\![v_i]\!], [\![m_i^*]\!], \bar{p}$	m_i, m_i^*, v_i

device is that the values $\widehat{m_i}, \widehat{v_i}$ of a trader are needed to *prevent the linking of limit orders during the verification procedure.* This also allows the instantaneous computation of $\widehat{\eta_i}$.

Memoization avoids the use of MPC when addressing the conflicting requirements of (i) providing a public trail of events, (ii) publicly verifying a constraint on a private subset of such events as well as (iii) showing that such private events are *all and only* applicable events. To meet (iii) Alice would have had to show which orders belonged to her to add them to her estimated net position. Since the full order book is visible (i), her full trading strategy would then be visible to the other players. In contrast, if we make sure that an order is private to trader Bob (ii), this very property does not allow Alice to prove that the order in question does not belong to her (and does not make her over budget), so failing (iii). Again a full MPC protocol would be a solution but this would force other traders to participate to the posting of any order from a third party. As mentioned, such burden would be considered unacceptable.

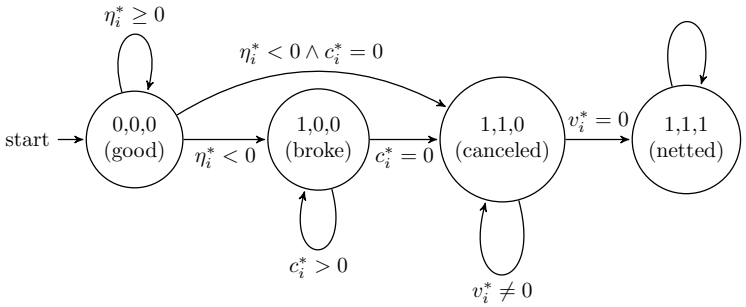


Figure 4.4: Inventory flags state transition diagram

Table 4.6: Merkle Tree's supported operations

Definition	**Description**
$\rho = \mathsf{Add}(\mathcal{T}, [\![v']\!])$	Adds a new leaf (the hash of $[\![v']\!]$) to the tree and generates a new root ρ.
$path = \mathsf{Path}(\mathcal{T}, [\![v']\!])$	Returns the authentication path from $[\![v']\!]$ to ρ.
$\{0,1\} \leftarrow \mathsf{Auth}(\rho, path, [\![v]\!])$	Authenticates $[\![v]\!]$ in $\mathcal{T}$ w.r.t. the authentication path *path* (where output 1 means the authentication succeeded).

Next, we introduce three flags to represent the status of a potential broke trader. A trader's inventory is marked with three flags $f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i}$. We call an inventory with a non-negative instant net position a *good* inventory ($f_{\mathsf{bad},i}$=0, $f_{\mathsf{del},i}$=0, $f_{\mathsf{out},i}$=0), otherwise it is a *broke* inventory ($f_{\mathsf{bad},i} = 1, f_{\mathsf{del},i} = 0, f_{\mathsf{out},i} = 0$). A good trader can do a normal post/cancel action, whereas a broke trader will cancel a previous order in the **Margin Settlement** phase. Finally, we call an inventory *canceled* if it is a *broke* inventory with no pending order (after canceling all orders in the **Margin Settlement** phase) at the time of commitment ($f_{\mathsf{bad},i}$=1, $f_{\mathsf{del},i}$=1, $f_{\mathsf{out},i}$=0); an inventory is *netted* if it has a zero volume holding (after matching an offset position during **Margin Settlement**) at the time of commitment ($f_{\mathsf{bad},i}$=1, $f_{\mathsf{del},i}$=1, $f_{\mathsf{out},i}$=1). The state transition diagram in Fig. 4.4 shows how the inventory switches from one state to another and the condition causing the transition. This status will capture the non-monotonic evolution of the validity of commitments and zk proofs once a valid order (of another trader) is accepted.

The overall state is then captured by a token τ_i, which is the commitment of all values in the inventory (with fresh randomness r_i). Such a value is only known to the trader, and each trader keeps the token secret and only broadcasts a new commitment to commit to a new inventory. Such an inventory is considered as *unspent*. At a later point, a trader can reveal the token and retrieve a previously committed inventory, in which the inventory is considered as *spent* and the corresponding token will become unusable.

The anonymity of the inventory is guaranteed by relying on Merkle trees [137] in conjunction with the zero-knowledge proofs (in Anonymous E-Cash [163]). Throughout the execution of the protocol, a Merkle tree $\mathcal{T}$ based on a collision-resistant hash function

$\mathsf{H}: \{0,1\}^* \to \{0,1\}^*$, where the leaves are commitments, is maintained and updated. ρ denotes the root of the tree, and *path* denotes the authentication path from a leaf $[\![v]\!]$ to the root ρ. The number of leaves is not fixed a-priori; one can efficiently update a Merkle tree $\mathcal{T}$ by appending a new leaf, resulting in a new tree $\mathcal{T}'$ with root ρ. This can be done in time/space proportional to tree depth. Table 4.6 summarizes the supported ops Add, Path and Auth of a Merkle tree $\mathcal{T}$.

Preserving Traders' Anonymity. The commitment (repsectively the retrieval) of trader inventories to the Merkle Tree $\mathcal{T}$ is obtained by running a sub-protocols Π_{put} (resp. Π_{get} , see Fig. 4.5) as follows:

- Executing protocol Π_{put}, the trader broadcasts a commitment to the token corresponding to the current inventory $\tau_i = \mathsf{Com}(m_i||v_i||\widehat{m_i}||\widehat{v_i}||c_i||f_{\mathsf{bad},i}||f_{\mathsf{del},i}||f_{\mathsf{out},i}; r_i)$.

  Thus, the trader proves the token is correctly constructed (using $\mathcal{F}^{\mathsf{token}}$) and appended into the Merkle tree $\mathcal{T}$ (with operation Add), before broadcasting the new root of the tree. The other traders will check that the new root is correctly computed before accepting it.

- In an execution of protocol Π_{get}, a trader can retrieve a previously committed inventory (say, at round $t' < t$), and *spend* it for posting or canceling an order (l, v), by revealing the secret *unspent* token τ_i' and proving that the newly committed values are consistent updates of the values committed at round t'. This is done using $\mathcal{F}^{\mathsf{inv}}$ (to retrieve the inventory) and then $\mathcal{F}^{\mathsf{uinv}}$ (to update the inventory), i.e. proving that $[\![\tau_i']\!]$ is a leaf of the current tree and $m_i = m_i'$, $v_i = v_i'$, $\widehat{m_i} = \widehat{m_i}' - \delta_c \cdot p_\ell \cdot v$, $\widehat{v_i} = \widehat{v_i}' + \delta_c \cdot v$, and $c_i = c_i' + \delta_c$, while all the flags $f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i}$ stay the same. Every time an inventory is retrieved, two sets of commitments are generated corresponding to the inventory values before and after the update. The token τ_i' is now marked as *spent* and will not be usable for retrieving any inventory.

The main Merkle tree $\mathcal{T}$ is forked (via sub-protocol Π_{backup}, see below) into a backup tree $\mathcal{T}_{\mathcal{U}}$ to use during the **Mark to Market** phase in case there are still traders with a negative net position after the **Margin Settlement** phase. We use this feature to challenge the non-monotonicity of security (see Margin Settlement phase in the next §4.6) and go beyond security-with-abort (see §4.9).

4.5 FuturesMEX Crypto Building Blocks

In Table 4.7, we review a few standard *NP* relations related to commitment schemes. Below we give the formal description of the auxiliary ideal functionalities that support

Sub-protocol Π_{put}: The protocol is run by $(P_1, \ldots, P_N)$ in order to let P_i commit to a new inventory by adding the commitment of the inventory token to a Merkle Tree $\mathcal{T}$ (resulting in a new root ρ^*).

1. P_i picks $r_i, r_{\tau_i} \leftarrow \{0,1\}^*$, computes $\tau_i = \mathsf{Com}(m_i||v_i||\widehat{m_i}||\widehat{v_i}||c_i||f_{\mathsf{bad},i}||f_{\mathsf{del},i}||f_{\mathsf{out},i}; r_i)$, and broadcasts $[\![\tau_i]\!] = \mathsf{Com}(\tau_i, r_{\tau_i})$.
2. P_i proves that the token is correctly constructed (using $\mathcal{F}^{\mathsf{token}}$).
3. P_i runs $\rho^* = \mathsf{Add}(\mathcal{T}, [\![\tau_i]\!])$ and broadcasts ρ^*. The other traders accept ρ^* iff the token has been correctly added to the tree.

Sub-protocol Π_{get}: The protocol is used when a trader posts a new order $(\delta_c = 1, l, v)$, cancels a pending order $(\delta_c = -1, l, v)$, or simply retrieves an **unspent** inventory that was committed at round $t' \leq t$ into a Merkle Tree $\mathcal{T}$; any **spent** token is rejected.

1. P_i recovers $path = \mathsf{path}(\mathcal{T}, [\![\tau_i']\!])$ and broadcasts τ_i'.
2. P_i proving that the newly committed values are consistent updates of the values committed at round t' (using $\mathcal{F}^{\mathsf{inv}}$ and additionally $\mathcal{F}^{\mathsf{uinv}}$ if $(\delta_c, l, v) \neq (0,0,0)$). The token τ_i' is now marked as **spent** by all traders.

Figure 4.5: Sub-protocols Π_{put} and Π_{get}

our protocol construction.

The Secure All Non-Negative Check $\mathcal{F}_{\mathsf{anncheck}}$ runs on common inputs $\{[\![\eta_i]\!]\}_{i\in[N]}$, receives inputs from a set of players $(P_1, \ldots, P_N)$, and interacts with them as follows.

1. Receive $(P_i, \eta_i, r_{\eta_i})$ from each P_i and accept the input iff $(R^{\mathsf{oc}}([\![\eta_i]\!], (\eta_i, r_{\eta_i})) = 1)$.
2. Output $\perp$ to all the players in the presence of any invalid input.
3. Else, the output of all players is defined to be 1 if $\eta_i \geq 0$ $(\forall i \in [N])$, and 0 otherwise.

The Secure Sum Comparison $\mathcal{F}_{\mathsf{compare}}$ runs on common inputs $\langle [\![f_i]\!], [\![f_i^*]\!] \rangle$, receives inputs from a set of players $(P_1, \ldots, P_N)$, and interacts with them as follows.

1. Receive $(P_i, f_i, r_{f_i}, f_i^*, r_{f_i^*})$ from each P_i and accept the input iff $(R^{\mathsf{oc}}([\![f_i]\!], (f_i, r_{f_i})) = 1) \wedge (R^{\mathsf{oc}}([\![f_i^*]\!], (f_i^*, r_{f_i^*})) = 1)$.
2. Output $\perp$ to all the players in the presence of any invalid input.
3. Else, the output of all players is defined to be 1 if $\sum f_i = \sum f_i^*$, and 0 otherwise.

The zero-knowledge functionality $\mathcal{F}^R$ is parametrized by an NP relation R, and receives inputs from a prover P_i and a set of verifiers $\{P_j\}_{j\neq i}$. The interaction is as follows.

1. Player P_i sends (P_i, x_i, w_i) to $\mathcal{F}^R$.

Table 4.7: Standard *NP* relations for commitment schemes

Definition	Description
$R^{\mathsf{oc}}([\![v]\!], (v, r_v))$	The relation of opening ownership, whose output is one iff (v, r_v) is the opening of $[\![v]\!]$, i.e., $[\![v]\!] = \mathsf{Com}(v; r_v)$.
$R^{\mathsf{zero}^-}([\![v]\!], (v, r_v))$	The relation of commitments to non-positive values, whose output is one iff $R^{\mathsf{oc}}([\![v]\!], (v, r_v)) = 1$ and $v \leq 0$.
$R^{\mathsf{zero}^+}([\![v]\!], (v, r_v))$	The relation of commitments to non-negative values, whose output is one iff $R^{\mathsf{oc}}([\![v]\!], (v, r_v)) = 1$ and $v \geq 0$.
$R^{-}([\![v]\!], (v, r_v))$	The relation of commitments to negative values, whose output is one iff $R^{\mathsf{oc}}([\![v]\!], (v, r_v)) = 1$ and $v < 0$.
$R^{+}([\![v]\!], (v, r_v))$	The relation of commitments to positive values, whose output is one iff $R^{\mathsf{oc}}([\![v]\!], (v, r_v)) = 1$ and $v > 0$.
$R^{\mathsf{ec}}(([\![v]\!], [\![v']\!]), (v, r_v, v', r_{v'}))$	The relation of commitments to equal values, whose output is one iff $R^{\mathsf{oc}}([\![v]\!], (v, r_v)) = 1$, $R^{\mathsf{oc}}([\![v']\!], (v', r_{v'})) = 1$ and $v = v'$.
$R^{\mathsf{nec}}(([\![v]\!], v'), (v, r_v))$	The relation of commitments to unequal values, whose output is one iff $R^{\mathsf{oc}}([\![v]\!], (v, r_v)) = 1$ and $v \neq v'$.
$R^{\mathsf{lec}}(([\![v]\!], c), (v, r_v))$	The relation of commitments to values less than or equal to c, whose output is one iff $R^{\mathsf{oc}}([\![v]\!], (v, r_v)) = 1$ and $v \leq c$.
$R^{\mathsf{gec}}(([\![v]\!], c), (v, r_v))$	The relation of commitments to values less than or equal to c, whose output is one iff $R^{\mathsf{oc}}([\![v]\!], (v, r_v)) = 1$ and $v \geq c$.
$R^{\mathsf{or}}(([\![v]\!], c), (v, r_v))$	The relation of commitments to values of opposite sign and greater than or equal to c, whose output is one iff $R^{\mathsf{oc}}([\![v]\!], (v, r_v)) = 1$ and $v \cdot c < 0 \wedge \lvert v\rvert \geq \lvert c\rvert$.

2. Each player P_j sends (P_j, x_j) to $\mathcal{F}^R$.
3. Output $R(x_j, w_i)$ to each trader P_j if $x_i = x_j$, otherwise return $\perp$.

If R^{type} is a relation, for simplicity we often write $\mathcal{F}^{\mathsf{type}}$ instead of $\mathcal{F}^{R^{\mathsf{type}}}$ for the corresponding zero-knowledge ideal functionality.

4.5.1 Futures Market Relations

The futures market relations used for the protocol construction in §4.6 are described in detail in this subsection.

Relation for Token Generation

The relation R^{token} takes as input the following statement and witness:

$$x_i^{\mathsf{token}} = ([\![\tau_i]\!], [\![m_i]\!], [\![v_i]\!], [\![\widehat{m_i}]\!], [\![\widehat{v_i}]\!], [\![c_i]\!], [\![f_{\mathsf{bad},i}]\!], [\![f_{\mathsf{del},i}]\!], [\![f_{\mathsf{out},i}]\!]) \tag{4.3}$$

$$w_i^{\mathsf{token}} = (m_i, v_i, \widehat{m_i}, \widehat{v_i}, c_i, f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i}, r_i, r_{\tau_i}). \tag{4.4}$$

The output of R^{token} is defined to be one iff the following conditions are met: (i) The commitments in the statement are valid commitments of the corresponding values in the witness; (ii) $[\![\tau_i]\!] = \mathsf{Com}(\mathsf{Com}(m_i||v_i||\widehat{m_i}||\widehat{v_i}||c_i||f_{\mathsf{bad},i}||f_{\mathsf{del},i}||f_{\mathsf{out},i}; r_i); r_{\tau_i})$.

Relation for Inventory Retrieval

The relation R^{inv} takes as input the following statement and witness:

$$x_i^{\mathsf{inv}} = (\rho, \tau_i', [\![m_i]\!], [\![v_i]\!], [\![\widehat{m_i}]\!], [\![\widehat{v_i}]\!], [\![c_i]\!], [\![f_{\mathsf{bad},i}]\!], [\![f_{\mathsf{del},i}]\!], [\![f_{\mathsf{out},i}]\!]) \tag{4.5}$$

$$w_i^{\mathsf{inv}} = (path_i, [\![\tau_i']\!], r_i', m_i, v_i, \widehat{m_i}, \widehat{v_i}, c_i, f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i}). \tag{4.6}$$

The output of R^{inv} is defined to be one iff the following conditions are satisfied: (i) The commitments in the statement are valid commitments of the corresponding values in the witness; (ii) $\mathsf{Auth}(\rho, path_i, [\![\tau_i']\!]) = 1$; (iii) The value τ_i' is the commitment of the inventory at a previous round, i.e. $\tau_i' = \mathsf{Com}(m_i||v_i||\widehat{m_i}||\widehat{v_i}||c_i||f_{\mathsf{bad},i}||f_{\mathsf{del},i}||f_{\mathsf{out},i}; r_i')$.

Relation for Correct Post/Cancel Update

The relation R^{uinv} takes as input the following statement and witness:

$$x_i^{\mathsf{uinv}} = ([\![\widehat{m_i}]\!], [\![\widehat{v_i}]\!], [\![c_i]\!], [\![\widehat{m_i}^*]\!], [\![\widehat{v_i}^*]\!], [\![c_i^*]\!], \delta_c, \ell, v) \tag{4.7}$$

$$w_i^{\mathsf{uinv}} = (\widehat{m_i}, \widehat{m_i}^*, \widehat{v_i}, \widehat{v_i}^*, c_i, c_i^*). \tag{4.8}$$

The output of R^{uinv} is defined to be one iff the following conditions are satisfied: (i) The commitments in the statement are valid commitments of the corresponding values in the witness; (ii) $\widehat{m_i}^* = \widehat{m_i} - \delta_c \cdot p_\ell \cdot v$, $\widehat{v_i}^* = \widehat{v_i} + \delta_c \cdot v$, and $c_i^* = c_i + \delta_c$.

Relation for Correct Range Choice

The relation R^{rng} takes as input the following statement and witness:

$$x_i^{\mathsf{rng}} = ([\![v]\!], [\![p_{\mathsf{lb}}]\!], [\![V_{\mathsf{lb}}]\!], [\![p_{\mathsf{ub}}]\!], [\![V_{\mathsf{ub}}]\!], \mathcal{O}_{\mathsf{buy}}, \mathcal{O}_{\mathsf{sell}}) \tag{4.9}$$

$$w_i^{\mathsf{rng}} = (v, p_{\mathsf{lb}}, V_{\mathsf{lb}}, p_{\mathsf{ub}}, V_{\mathsf{ub}}). \tag{4.10}$$

The output of R^{rng} is defined to be one iff the following conditions are satisfied: (i) The commitments in the statement are valid commitments of the corresponding values in the witness; (ii) $V_{\mathsf{lb}} \leq |v| \leq V_{\mathsf{ub}}$; (iii) Eq. (4.11) $\vee$ Eq. (4.12) $\vee$ Eq. (4.13) below holds:

$$v > 0 \wedge ((p_{\mathsf{lb}}, V_{\mathsf{lb}}), (p_{\mathsf{ub}}, V_{\mathsf{ub}})) \in \mathcal{O}_{\mathsf{buy}} \tag{4.11}$$

$$v < 0 \wedge ((p_{\mathsf{lb}}, V_{\mathsf{lb}}), (p_{\mathsf{ub}}, V_{\mathsf{ub}})) \in \mathcal{O}_{\mathsf{sell}} \tag{4.12}$$

$$v = 0 \wedge (p_{\mathsf{lb}} V_{\mathsf{lb}}) = (p_{\mathsf{ub}}, V_{\mathsf{ub}}) = (0, 0). \tag{4.13}$$

Relation for Correct Computation/Speculation of an Instant Net Position

The relation R^{net} takes as input the following statement and witness:

$$x_i^{\mathsf{net}} = ([\![m_i]\!], [\![v_i]\!], [\![\eta_i]\!], [\![p_{\mathsf{lb}}]\!], [\![V_{\mathsf{lb}}]\!], [\![p_{\mathsf{ub}}]\!], [\![V_{\mathsf{ub}}]\!]) \tag{4.14}$$

$$w_i^{\mathsf{net}} = (m_i, v_i, \eta_i, p_{\mathsf{lb}}, V_{\mathsf{lb}}, p_{\mathsf{ub}}, V_{\mathsf{ub}}). \tag{4.15}$$

The output of R^{net} is defined to be one iff the following conditions are satisfied: (i) The commitments in the statement are valid commitments of the corresponding values in the witness; (ii) $\eta_i = m_i + p_{\mathsf{lb}} \cdot V_{\mathsf{lb}} + p_{\mathsf{ub}} \cdot (|v_i| - V_{\mathsf{lb}})$.

Relation for Correct Flags Update

The relation R^{flags} takes as input the following statement and witness:

$$x_i^{\mathsf{flags}} = ([\![f_{\mathsf{bad},i}]\!], [\![f^*_{\mathsf{bad},i}]\!], [\![f_{\mathsf{del},i}]\!], [\![f^*_{\mathsf{del},i}]\!], [\![f_{\mathsf{out},i}]\!], [\![f^*_{\mathsf{out},i}]\!], [\![\eta_i^*]\!], [\![v_i^*]\!], [\![c_i^*]\!]) \tag{4.16}$$

$$w_i^{\mathsf{flags}} = (f_{\mathsf{bad},i}, f^*_{\mathsf{bad},i}, f_{\mathsf{del},i}, f^*_{\mathsf{del},i}, f_{\mathsf{out},i}, f^*_{\mathsf{out},i}, \eta_i^*, v_i^*, c_i^*). \tag{4.17}$$

The output of R^{flags} is defined to be one iff the following conditions are satisfied: (i) The commitments in the statement are valid commitments of the corresponding values in the witness; (ii) The transition from the flags $(f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i})$ to the flags $(f^*_{\mathsf{bad},i}, f^*_{\mathsf{del},i}, f^*_{\mathsf{out},i})$ is consistent with the values (η_i^*, v_i^*, c_i^*), as shown in the diagram in Fig. 4.4.

Relation for Correct Match

The relation R^{match} takes as input the following statement and witness:

$$x_i^{\mathsf{match}} = ([\![m_i]\!], [\![v_i]\!], [\![c_i]\!], [\![m_i^*]\!], [\![v_i^*]\!], [\![c_i^*]\!], \delta_c, p_\ell, v) \tag{4.18}$$

$$w_i^{\mathsf{match}} = (m_i, m_i^*, v_i, v_i^*, c_i, c_i^*). \tag{4.19}$$

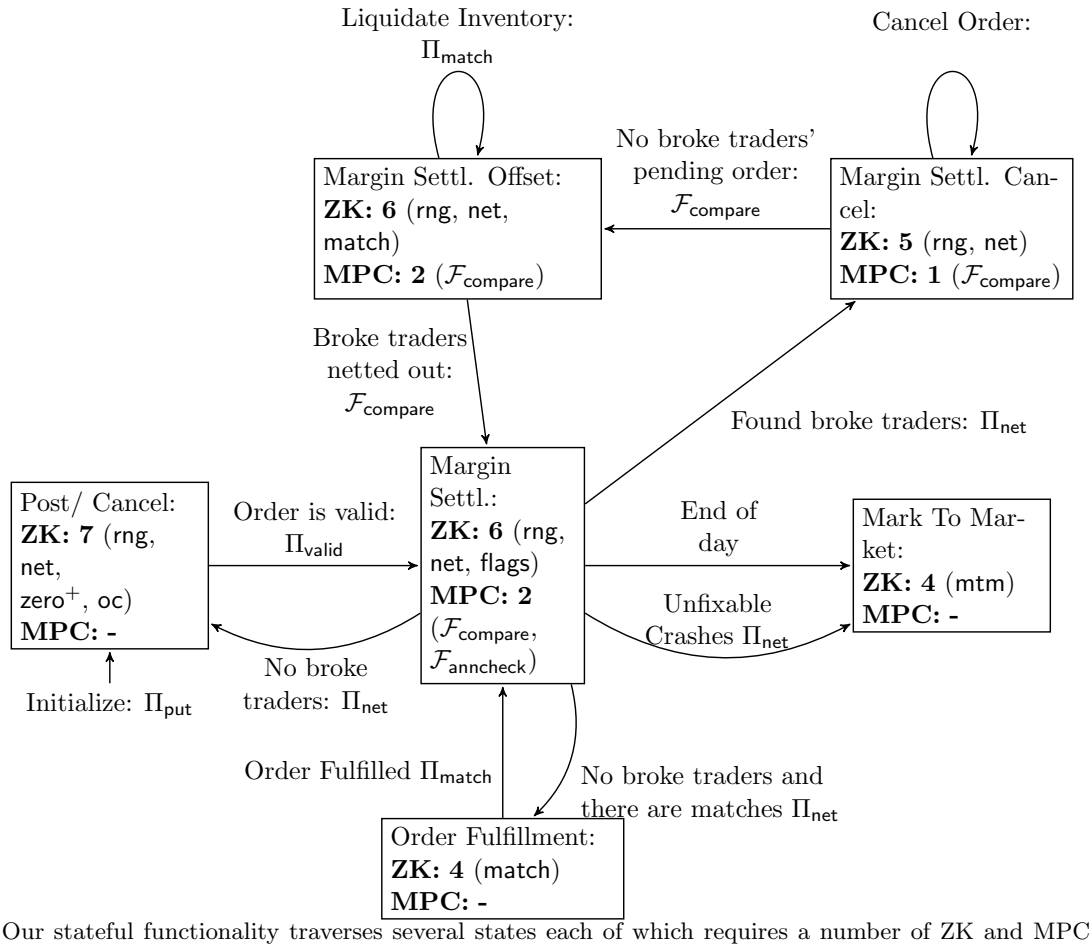
The output of R^{match} is defined to be one iff the following conditions are satisfied: (i) The commitments in the statement are valid commitments of the corresponding values in the witness; (ii) $m_i^* = m_i - p_\ell \cdot v$ and $v_i^* = v_i + v$ and $c_i^* = c_i + \delta_c$.

Relation for Correct Mark to Market

The relation R^{mtm} takes as input the following statement and witness:

$$x_i^{\mathsf{mtm}} = ([\![m_i]\!], [\![v_i]\!], [\![m_i^*]\!], \bar{p}) \tag{4.20}$$

$$w_i^{\mathsf{mtm}} = (m_i, m_i^*, v_i). \tag{4.21}$$



Our stateful functionality traverses several states each of which requires a number of ZK and MPC steps. The subprotocols Π_{get}, Π_{put} and their functionalities inv, uinv, token are needed to interact with the trader's inventory. Hence, for every state we show the remaining required ZK proof steps and the MPC steps. The numbers are of ZK and MPC steps. We distinguish Margin Settlement into three sub-states: Margin Settl. (where net positions are checked), Margin Settl. Cancel (where we remove the pending orders), and Margin Settl. Offset (where we offset the positions).

Figure 4.6: Hybrid Implementation of the Ideal Functionality

The output of R^{mtm} is defined to be one iff the following conditions are satisfied: (i) The commitments in the statement are valid commitments of the corresponding values in the witness; (ii) $m_i^* = m_i + \bar{p} \cdot v_i$.

4.6 Protocol Construction

At this point, an easy solution is to run the *entire* reactive functionality as a global MPC. As we mentioned previously, this is unacceptable from the perspective of most traders: the burden of computation should rather be shifted to parties wishing to prove something (e.g. good and bad standing). All traders should be involved only when the global consistency

of the market is at stake. We illustrate the burden of using just MPC empirically with our simulation in §4.10.1.

Fig. 4.6 summarizes how various security functionalities and sub-protocols have been used to implement each step of the global ideal functionality.

Common Sub-Protocols The protocols in Fig. 4.7 are used extensively as sub-routines in our main protocol.

We denote by superscript $\cdot^*$ the updated values (computed locally by P_i after an update of the order book, e.g. m_i^*), that are used as common inputs for the sub-protocols. In particular we use as common inputs the commitments of the inventory values, e.g. $[\![m_i]\!]$, the related order information (δ_c, ℓ, v) and the Merkle Tree $\mathcal{T}$, where P_i additionally holds the committed values and the corresponding randomnesses.

Protocol Description The overall protocol runs in four phases, described on a high-level below. The formal description is found from Fig. 4.8 to 4.11.

Initialize Phase: Every trader participating in the futures market must commit to a valid initial inventory. This is done by each trader individually during the first round, as follows:

1. P_i holds an initial non-negative secret amount of cash (i.e. $m_i \geq 0$), zero volume holding (i.e. $v_i = 0$), an initial estimation of the cost to pay for pending orders (i.e. $\widehat{m_i} = m_i$), and an zero estimation of the volume holding for pending orders (i.e. $\widehat{v_i} = 0$) and all initial zero inventory flags.
2. P_i commits to its initial inventory and proves in zero knowledge, in which such an inventory is valid (as defined above), by using the functionality $\mathcal{F}^{\mathsf{zero}^+}$ for m_i and $\mathcal{F}^{\mathsf{ec}}$ for $\widehat{m_i}$, while simply decommitting v_i and $\widehat{v_i}$ and the flags is sufficient to prove that they are all zeros.
3. The traders run protocol Π_{put} to commit the inventory of P_i; the backup tree $\mathcal{T}_{\mathcal{U}}$ is identical to $\mathcal{T}$.

Post/Cancel Order Phase: A good trader can post a new order $(\delta_c = 1, \ell, v)$ or cancel a previous order $(\delta_c = -1, \ell', v')$.

1. The traders run Π_{valid}.
2. The traders run Π_{net}; this can lead to **Mark to Market**.
3. The traders run Π_{match} for each match in the order book (only for **Post Order**).

Sub-protocol Π_{valid} is run by $(P_1, \ldots, P_N)$ in order to let P_i prove a valid **Post Order** or **Cancel Order** action. Every time a trader P_i posts or cancels an order, say (δ_c, ℓ, v), the protocol has to check for its validity.

1. In case of **Cancel Order**, P_i proves the ownership of the order using $\mathcal{F}^{\mathsf{oc}}$.
2. All traders run Π_{get} to take out the corresponding inventory then P_i proves that s/he can perform the action by decommitting the inventory flags to show:
   (a) $f_{\mathsf{bad},i} = 0$ in a normal post/cancel action;
   (b) $(f_{\mathsf{bad},i} = 1) \wedge (f_{\mathsf{del},i} = 0)$ for a cancel action during **Margin Settlement**;
   (c) $(f_{\mathsf{bad},i} = 1) \wedge (f_{\mathsf{del},i} = 1) \wedge (f_{\mathsf{out},i} = 0)$ for a post action during **Margin Settlement**. All traders check that the posted order is matchable with the best opposite order. In addition, P_i proves that s/he picks the best pending order correctly, i.e. $v_i \cdot v < 0 \wedge |v_i| \geq |v|$, using $\mathcal{F}^{\mathsf{or}}$.
3. P_i proves it has a non-negative estimation for instant net position $\widehat{\eta_i}$ (only in a normal post/cancel action) using $\mathcal{F}^{\mathsf{rng}}$, $\mathcal{F}^{\mathsf{net}}$, and $\mathcal{F}^{\mathsf{zero}^+}$.
4. All traders run Π_{put} to put back the new inventory.

Sub-protocol Π_{net} Every time the order book is updated via (i) a post/cancel action of a trader P_i, or (ii) a match of two traders P_i and $P_{i'}$, all the traders (including P_i and $P_{i'}$), need to be checked for negative instant net position.

1. Repeat the following for each trader P_i to retrieve the inventory then update and check their new inventory flags.
   (a) All traders run Π_{get}.
   (b) P_i commits to the ranges $[\![p_{\mathsf{lb}}]\!], [\![V_{\mathsf{lb}}]\!], [\![p_{\mathsf{ub}}]\!], [\![V_{\mathsf{ub}}]\!]$ and proves that those range choices are correct using $\mathcal{F}^{\mathsf{rng}}$.
   (c) P_i commits to the instant net position η_i, updates the flags accordingly and proves integrity using $\mathcal{F}^{\mathsf{net}}$ (using $[\![p_{\mathsf{lb}}]\!], [\![V_{\mathsf{lb}}]\!], [\![p_{\mathsf{ub}}]\!], [\![V_{\mathsf{ub}}]\!]$ above), and $\mathcal{F}^{\mathsf{flags}}$.
   (d) All traders run Π_{put}
2. All traders run $\mathcal{F}_{\mathsf{compare}}$.
3. During a normal phase, if $\mathcal{F}_{\mathsf{compare}}$ returns 1, run Π_{backup}.

Sub-protocol Π_{match} for updating the inventories upon a match of orders $(t, \ell, [\![i]\!], v)$ of P_i and $(t', \ell, [\![i']\!], v')$ of $P_{i'}$ $(v \cdot v' < 0)$.

1. P_i proves the ownership of the order using $\mathcal{F}^{\mathsf{oc}}$, then all traders run Π_{get} to take out the corresponding inventory.
2. P_i computes the new inventory values and broadcasts the new commitments.
3. All traders run Π_{put} to put back the inventory.
4. $P_{i'}$ performs steps 1-3.

Sub-protocol Π_{backup} is run to fork a backup tree. Additionally, the common inputs include $[\![\eta_i^*]\!]$, and P_i also holds η_i^*.

1. All traders run $\mathcal{F}_{\mathsf{anncheck}}$.
2. If $\mathcal{F}_{\mathsf{anncheck}}$ returns 1, all traders run Π_{put} to obtain $\mathcal{T}_{\mathcal{U}}{}^*$.

Figure 4.7: Sub-protocols Π_{valid}, Π_{net}, Π_{match}, and Π_{backup}

4. After each match, all traders run Π_{net} again; if it returns 1, run Π_{backup}; otherwise go to **Margin Settlement**.

Margin Settlement Phase: This phase is started by Π_{net} only when there is at least one new broke trader when the order book is updated (post, cancel or match happens) and is re-started every time there is at least one new trader with a bad standing (i.e. $f_{\mathsf{bad},i} = 0$ and $\eta_i^* < 0$) during the execution of this phase. It proceeds as described below; afterwards the protocol goes back to the previous phase (whatever it was).

1. On a first come first served basis, if there is an order $o' = (t', l, [\![i]\!], v)$ belongs to a broke trader P_i she must cancel it:
   (a) The traders run Π_{get} with parameters $(-1, \ell, v)$, to retrieve P_i's two inventories: one before the cancellation of the pending order and one after the cancellation.
   (b) The traders run Π_{valid}.
   (c) All traders forward the necessary flags $f^*_{\mathsf{del},i}$ and $f_{\mathsf{bad},i}$ to $\mathcal{F}_{\mathsf{compare}}$ to check whether $\sum f^*_{\mathsf{del},i} = \sum f_{\mathsf{bad},i}$, i.e. all pending orders of the broke traders have been canceled. If the check is successful, move to next step. Otherwise go back to Step (a).
2. The traders run Π_{net} to check and restart this phase if there are new broke traders.
3. In a first come first served manner, the broke traders offset their positions until all broke traders are netted out.
   (a) The traders run Π_{get} to retrieve their inventory.
   (b) The traders find matches on the order book at the current best price, say between P_i and $P_{i'}$.
   (c) The traders run Π_{valid} then both traders locally update their inventory, commit to the new inventory, and prove in zero knowledge that the match has been done correctly (using $\mathcal{F}^{\mathsf{match}}$).
   (d) All traders forward the necessary flags $f^*_{\mathsf{out},i}$ and $f_{\mathsf{bad},i}$ to $\mathcal{F}_{\mathsf{compare}}$ to check whether $\sum f^*_{\mathsf{out},i} = \sum f_{\mathsf{bad},i}$, i.e. all the positions have been offset, if the check is successful, go to next step.
4. The traders run Π_{net} to check and restart this phase if there are new broke traders.
5. All traders run Π_{backup} to check if a backup tree can be forked, if not, go to **Mark to Market** phase.

Mark to Market Phase: This phase is invoked at the last round $t = T$, or during **Margin Settlement**.

1. The traders run Π_{get} to retrieve their inventory.
2. The traders locally updates their inventory, commits to the new inventory, and proves in zero knowledge that the match has been done correctly (using $\mathcal{F}^{\mathsf{mtm}}$).
3. Finally the new inventory is added back to the Merkle tree $\mathcal{T}$ by running Π_{put}.

The *Proportional Burden* is fulfilled, for what is technically possible, as we require traders posting/canceling an order to prove the validity of their actions before other traders prove the validity of their inventories according to the new order book. The latter is necessary for distributed risk management. It could be optimized by having a trader proving the validity of an inventory for a range of price values rather than just the current price (e.g. up/downward ticks as appropriate).

Initialize Phase: This phase runs in the first round where each trader P_i commits to a **good** inventory with cash m_i.

1. Trader P_i commits to and broadcasts the following values:
   (a) A secret cash balance $[\![m_i]\!]$, and a zero volume holding $[\![v_i]\!]$.
   (b) A secret estimation of cash balance regarding pending orders $[\![\widehat{m_i}]\!]$.
   (c) A zero expected volume holding for regarding orders $[\![\widehat{v_i}]\!]$.
   (d) A zero counter for the number of regarding orders $[\![c_i]\!]$.
   (e) Three zero inventory flags $[\![f_{\mathsf{bad},i}]\!], [\![f_{\mathsf{del},i}]\!], [\![f_{\mathsf{out},i}]\!]$.
   (f) A token $[\![\tau_i]\!]$, where $\tau_i = \mathsf{Com}(m_i||v_i||\widehat{m_i}||\widehat{v_i}||c_i||f_{\mathsf{bad},i}||f_{\mathsf{del},i}||f_{\mathsf{out},i}; r_i)$ with $r_i \leftarrow \{0,1\}^*$.
2. P_i sends $(P_i, [\![m_i]\!], m_i)$ and each other $P_{j\neq i}$ sends $(P_j, [\![m_i]\!])$ to $\mathcal{F}^{\mathsf{zero}^+}$.
3. P_i sends $(P_i, ([\![\widehat{m_i}]\!], [\![m_i]\!]), (\widehat{m_i}, m_i))$ and each other $P_{j\neq i}$ sends $(P_j, ([\![\widehat{m_i}]\!], [\![m_i]\!]))$ to $\mathcal{F}^{\mathsf{ec}}$.
4. P_i decommits $[\![v_i]\!], [\![\widehat{v_i}]\!], [\![c_i]\!], [\![f_{\mathsf{bad},i}]\!], [\![f_{\mathsf{del},i}]\!], [\![f_{\mathsf{out},i}]\!]$ to show that the values are 0.
5. All traders run Π_{put} on common input $(\mathcal{T}, [\![m_i]\!], [\![\widehat{m_i}]\!], [\![v_i]\!], [\![\widehat{v_i}]\!], [\![c_i]\!], [\![f_{\mathsf{bad},i}]\!], [\![f_{\mathsf{del},i}]\!], [\![f_{\mathsf{out},i}]\!])$, where P_i additionally holds $(m_i, v_i, \widehat{m_i}, \widehat{v_i}, c_i, f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i})$.
6. Set $\mathcal{T}_{\mathcal{U}} := \mathcal{T}$.

Figure 4.8: The **Initialize** phase of protocol Π_{DFM}

Post Order Phase: This phase is run in order to post an order $o = (t, \ell, i, v)$ at round t (where $\ell \geq \ell_{\mathsf{buy}}$ for $v < 0$, and $\ell \leq \ell_{\mathsf{sell}}$ for $v > 0$).

1. P_i broadcasts the values $(1, \ell, v)$ and a commitment $[\![i]\!]$.
2. All traders run Π_{valid};
3. All traders run Π_{net}.
4. Starting from $t' = 1$, repeat the following steps for each matching entry $(t', \ell, [\![i']\!], v') \in \mathcal{O}$ such that $v \cdot v' < 0$, until $v = 0$ or $t' = t$:
   (a) All traders run Π_{match};
   (b) All traders run Π_{net}.

Cancel Order Phase: This phase is run in order to cancel an order $o = (t', \ell, i, v)$.

1. P_i broadcasts the values t' and $(-1, \ell, v)$.
2. All traders run Π_{valid};
3. All traders run Π_{net}.

Figure 4.9: The **Post Order** and **Cancel Order** phase of protocol Π_{DFM}

Margin Settlement Phase: This phase is run whenever at least one inventory was added to the Merkle Tree $\mathcal{T}$ during the sub-protocol Π_{net}. Thereafter, the protocol goes back to the invoking phase.

1. While there is an order $o' = (t', l, [\![i]\!], v)$ belongs to a broke trader P_i she has to cancel it:
   (a) P_i broadcasts the values t' and $(-1, \ell, v)$.
   (b) All traders run Π_{valid} on common inputs $(\mathcal{T}, [\![m_i]\!], [\![\widehat{m_i}]\!], [\![v_i]\!], [\![\widehat{v_i}]\!], [\![c_i]\!], [\![f_{\mathsf{bad},i}]\!], [\![f_{\mathsf{del},i}]\!], [\![f_{\mathsf{out},i}]\!], [\![m_i^*]\!], [\![v_i^*]\!], [\![\widehat{m_i}^*]\!], [\![\widehat{v_i}^*]\!], [\![c_i^*]\!], [\![f_{\mathsf{bad},i}^*]\!], [\![f_{\mathsf{del},i}^*]\!], [\![f_{\mathsf{out},i}^*]\!], \delta_c, l, v)$, where P_i additionally holds $([\![\tau_i']\!], \tau_i', r_i', m_i, m_i^*, v_i, v_i^*, \widehat{m_i}, \widehat{m_i}^*, \widehat{v_i}, \widehat{v_i}^*, c_i, c_i^*, f_{\mathsf{bad},i}, f_{\mathsf{bad},i}^*, f_{\mathsf{del},i}, f_{\mathsf{del},i}^*, f_{\mathsf{out},i}, f_{\mathsf{out},i}^*)$.
   (c) P_i forwards $f_{\mathsf{del},i}^*$ and $f_{\mathsf{bad},i}$ to $\mathcal{F}_{\mathsf{compare}}$; if $\mathcal{F}_{\mathsf{compare}}$ returns 1, proceed to next step, else go back to step (a).
2. The traders run Π_{net}, and return to step 1 if there is any new broke inventory.
3. Offset the volume holding v_i of every broke trader P_i until $v_i = 0$. A broke trader P_i locally looks up the order book from $t' = 1$ to $t' = t$ and posts an opposite order for an order $o' = (t', l, [\![i']\!], v')$ (where $l = \ell_{\mathsf{sell}}$ for $v_i < 0$, and $l = \ell_{\mathsf{buy}}$ if $v_i > 0$), then the following steps are performed for each o':
   (a) The traders run Π_{valid}.
   (b) The traders run Π_{match}.
   (c) P_i forwards $f_{\mathsf{out},i}^*$ and $f_{\mathsf{bad},i}$ to $\mathcal{F}_{\mathsf{compare}}$. If $\mathcal{F}_{\mathsf{compare}}$ returns 1, proceed to the next step, else, go back to step (a).
4. The traders run Π_{net}, and return to step 1 if there is any new broke inventory.
5. All traders run Π_{backup} to check if a backup tree can be forked, if not, the traders proceed to **Mark to Market** using the tree $\mathcal{T}_{\mathcal{U}}$.

Figure 4.10: The **Margin Settlement** phase of protocol Π_{DFM}

Mark to Market Phase: This phase is called either during the **Margin Settlement**, or in the last round, where every trader P_i retrieves and commits to a **good** inventory with new marked-to-market values.

1. All traders run Π_{get} on common input $(\mathcal{T}, [\![m_i]\!], [\![\widehat{m_i}]\!], [\![v_i]\!], [\![\widehat{v_i}]\!], [\![c_i]\!], [\![f_{\mathsf{bad},i}]\!], [\![f_{\mathsf{del},i}]\!], [\![f_{\mathsf{out},i}]\!], [\![m_i^*]\!], [\![v_i^*]\!], [\![\widehat{m_i}^*]\!], [\![\widehat{v_i}^*]\!], [\![c_i^*]\!], [\![f_{\mathsf{bad},i}^*]\!], [\![f_{\mathsf{del},i}^*]\!], [\![f_{\mathsf{out},i}^*]\!])$, where P_i additionally holds $([\![\tau_i']\!], \tau_i', r_i', m_i, m_i^*, v_i, v_i^*, \widehat{m_i}, \widehat{m_i}^*, \widehat{v_i}, \widehat{v_i}^*, c_i, c_i^*, f_{\mathsf{bad},i}, f_{\mathsf{bad},i}^*, f_{\mathsf{del},i}, f_{\mathsf{del},i}^*, f_{\mathsf{out},i}, f_{\mathsf{out},i}^*)$.
2. P_i computes the following:

   $$m_i^* := \widehat{m_i}^* := m_i + \bar{p} \cdot v_i \qquad \text{and} \qquad v_i^* := \widehat{v_i}^* := c_i^* := f_{\mathsf{bad},i}^* := f_{\mathsf{del},i}^* := f_{\mathsf{out},i}^* := 0.$$

3. P_i broadcasts $[\![m_i^*]\!]$, $[\![v_i^*]\!]$, $[\![\widehat{m_i}^*]\!]$, $[\![\widehat{v_i}^*]\!]$, $[\![c_i^*]\!]$ and $[\![f_{\mathsf{bad},i}^*]\!]$, $[\![f_{\mathsf{del},i}^*]\!]$, $[\![f_{\mathsf{out},i}^*]\!]$.
4. P_i decommits $[\![v_i^*]\!]$, $[\![\widehat{v_i}^*]\!]$, $[\![c_i^*]\!]$ and $[\![f_{\mathsf{bad},i}^*]\!]$, $[\![f_{\mathsf{del},i}^*]\!]$, $[\![f_{\mathsf{out},i}^*]\!]$. to show the values are 0.
5. P_i sends $(P_i, x_i^{\mathsf{mtm}}, w_i^{\mathsf{mtm}})$ to $\mathcal{F}^{\mathsf{mtm}}$, while $P_{j\neq i}$ sends $(P_j, x_j^{\mathsf{mtm}})$—cf. Eq. (4.20)-(4.21).
6. P_i sends $(P_i, ([\![\widehat{m_i}^*]\!], [\![m_i^*]\!]), (\widehat{m_i}^*, m_i^*))$ and each other $P_{j\neq i}$ sends $(P_j, ([\![\widehat{m_i}^*]\!], [\![m_i^*]\!]))$ to $\mathcal{F}^{\mathsf{ec}}$.
7. All traders run Π_{put} on common input $(\mathcal{T}, [\![m_i^*]\!], [\![\widehat{m_i}^*]\!], [\![v_i^*]\!], [\![\widehat{v_i}^*]\!], [\![c_i^*]\!], [\![f_{\mathsf{bad},i}^*]\!], [\![f_{\mathsf{del},i}^*]\!], [\![f_{\mathsf{out},i}^*]\!])$, where P_i additionally holds $(m_i^*, v_i^*, \widehat{m_i}^*, \widehat{v_i}^*, c_i^*, f_{\mathsf{bad},i}^*, f_{\mathsf{del},i}^*, f_{\mathsf{out},i}^*)$.

Figure 4.11: The **Mark To Market** phase of protocol Π_{DFM}

4.7 Security Analysis

The theorem below states the security of our protocol Π_{DFM} from §4.6.

Theorem 4.1. *Let* Com *be a statistically hiding (and computationally binding) commitment scheme. Protocol* Π_{DFM} *from §4.6 securely realizes the ideal functionality* $\mathcal{F}_{\mathsf{CFM}}$ *in the* $(\mathcal{F}_{\mathsf{zk}}, \mathcal{F}_{\mathsf{compare}}, \mathcal{F}_{\mathsf{anncheck}})$*-hybrid model, where the zero-knowledge functionality* $\mathcal{F}_{\mathsf{zk}}$ *supports all the NP relations defined in §4.3, and §4.5.*

4.7.1 Proof sketch

Let us first describe the intuitive approach to our security proof. As in standard simulation-based security proofs, we exhibit an efficient simulator interacting with the ideal functionality $\mathcal{F}_{\mathsf{CFM}}$ that is able to fake the view of any efficient adversary corrupting a subset $I \subseteq [N]$ of the traders in an execution of protocol Π_{DFM}[13].

Our protocol is designed in a "hybrid world" with several auxiliary ideal functionalities (mainly for zero-knowledge proofs and for running secure comparisons). Importantly, in such a world, there is no security issue when using these functionalities: a composition theorem ensures that our protocol is still secure when we replace the auxiliary ideal functionalities with sub-protocols securely realizing them. An advantage of working in the hybrid model is that the simulator gets to see the inputs that corrupted traders forward to the auxiliary ideal functionalities in the clear. The simulator has to play the role of the trusted party that computes each auxiliary functionality in the hybrid model.

On a very high level, our simulator $\mathcal{S}$ works as follows. During the **Initialize** phase, it commits to zero values for each commitment forwarded by a honest trader in the real protocol; the commitments to the token of each inventory are added to a simulated Merkle Tree that is maintained internally by the simulator. During a **Post/Cancel Order** action, it relies on the ideal functionality $\mathcal{F}_{\mathsf{CFM}}$ to post/cancel the corresponding orders; afterwards, in the **Margin Settlement** phase, for each match notification received from the ideal functionality $\mathcal{F}_{\mathsf{CFM}}$, the simulator commits to zero for each commitment forwarded by a honest trader in the real protocol execution. During the **Mark to Market** phase, it commits to zero values for each commitment forwarded by a honest trader in the real protocol.

The hiding property of the commitment scheme implies that the above simulation is indistinguishable to the view generated in a mental experiment where the simulator $\mathcal{S}$ is given the real inputs corresponding to each honest trader. The only difference between this mental experiment and a real protocol execution is that in the former experiment

[13] We assume the set I is fixed before the protocol execution starts.

the market evolves using the inventories held at the beginning by each corrupted trader, whereas in the latter experiment the adversary can try to cheat and fake the inventory of a corrupted trader (e.g. by claiming an order pertaining to a honest trader). However, the binding property of the commitment scheme and the collision resistance of the Merkle Tree, ensure that such cheating attempts only succeed with a negligible probability.

This allows us to conclude that the view simulated in the ideal world (with the functionality $\mathcal{F}_{\mathsf{CFM}}$) is computationally indistinguishable from the view in a real execution of the protocol, thus establishing the security of Π_{DFM}.

4.7.2 Security Proof

Proof. It is clear that Π_{DFM} computes $\mathcal{F}_{\mathsf{CFM}}$. We proceed to prove the security of Π_{DFM}. Let $\mathcal{A}$ be a non-uniform deterministic PPT adversary. The simulator $\mathcal{S}$ is given access to the ideal functionality $\mathcal{F}_{\mathsf{CFM}}$, and can also read the stored/updated values of the corrupted traders (that $\mathcal{A}$ controls) from $\mathcal{F}_{\mathsf{CFM}}$; recall that, since we prove only static security, the set of corrupted traders I is fixed before the protocol execution starts.

Sub-routines. To simplify the simulator's description, we introduce sub-routines $\mathcal{S}_{\mathsf{put}}$ and $\mathcal{S}_{\mathsf{get}}$ as well as $\mathcal{S}_{\mathsf{valid}}$, $\mathcal{S}_{\mathsf{backup}}$, $\mathcal{S}_{\mathsf{net}}$, and $\mathcal{S}_{\mathsf{match}}$ that will call $\mathcal{S}_{\mathsf{put}}$ and $\mathcal{S}_{\mathsf{get}}$ when it is related to committing and retrieving inventories. The sub-routines will later be invoked by $\mathcal{S}$; while reading them, think of the simulator's behaviour as a simulation strategy for the corresponding protocols Π_{put}, Π_{get}, Π_{valid}, Π_{backup}, Π_{net}, and Π_{match}. Each sub-routine invokes $\mathcal{A}$ and receives messages from it. The sub-routines use the aforementioned $\mathcal{S}_{\mathsf{get}}$ and $\mathcal{S}_{\mathsf{put}}$. Since we are working in the hybrid model, whenever $\mathcal{A}$ interacts with an ideal functionality the simulator receives $\mathcal{A}$'s inputs to the functionality in the clear, and thus it can perfectly emulate the output of the hybrid functionality.

$\mathcal{S}_{\mathsf{put}}$: When a trader P_i commits to an inventory, it acts as a prover while the other traders act as verifiers. If P_i is corrupted, $\mathcal{S}$ needs to simulate the views of both the prover P_i and the corrupted verifiers P_j, by receiving the inputs from P_i and forwarding them to each corrupted verifier $P_{j\neq i}$. Otherwise, $\mathcal{S}$ only needs to simulate the view of the corrupted verifiers P_j, by forwarding $[\![\mathbf{0}]\!]$ and $\rho = \mathsf{Add}(\mathcal{T}, [\![\mathbf{0}]\!])$ to each corrupted verifier P_j. In both cases $\mathcal{S}$ simulates the output of $\mathcal{F}^{\mathsf{token}}$ for each corrupted players, abort the simulation if any check fails.

$\mathcal{S}_{\mathsf{get}}$: Similar to $\mathcal{S}_{\mathsf{put}}$ but using $\mathcal{F}^{\mathsf{inv}}$ and $\mathcal{F}^{\mathsf{uinv}}$.

$\mathcal{S}_{\mathsf{valid}}$: Similar to $\mathcal{S}_{\mathsf{put}}$ but using $\mathcal{F}^{\mathsf{oc}}$, $\mathcal{F}^{\mathsf{rng}}$, $\mathcal{F}^{\mathsf{net}}$, $\mathcal{F}^{\mathsf{zero}^+}$ and $\mathcal{F}^{\mathsf{or}}$.

$\mathcal{S}_{\mathsf{backup}}$**:** Similar to $\mathcal{S}_{\mathsf{put}}$ but with $\mathcal{F}^{\mathsf{rng}}$, $\mathcal{F}^{\mathsf{net}}$ and $\mathcal{F}_{\mathsf{anncheck}}$ (i.e. output 1 if the ideal functionality $\mathcal{F}_{\mathsf{CFM}}$ does not trigger **Mark To Market**).

$\mathcal{S}_{\mathsf{net}}$**:** When a trader P_i needs to be checked for a non-negative instant net position, it acts as a prover while the other traders act as verifiers. The steps are also similar to $\mathcal{S}_{\mathsf{put}}$ but with $\mathcal{F}^{\mathsf{rng}}$, $\mathcal{F}^{\mathsf{net}}$, $\mathcal{F}^{\mathsf{flags}}$ and $\mathcal{F}_{\mathsf{compare}}$.

Remark 4.5. *To simulate the output of $\mathcal{F}_{\mathsf{compare}}$ the simulator simply uses the public output of $\mathcal{F}_{\mathsf{CFM}}$ during the net position check that may trigger the* ***Margin Settlement*** *phase, i.e. whenever $\mathcal{F}_{\mathsf{CFM}}$ outputs a list of pending orders to be removed or an order to be matched (to offset a broke inventory), the simulator outputs 0 for $\mathcal{F}_{\mathsf{compare}}$ and $\mathcal{F}_{\mathsf{anncheck}}$.*

$\mathcal{S}_{\mathsf{match}}$**:** When a trader P_i posts an order, it acts as a prover together with some other trader $P_{i'}$ with a matching order, while the other traders act as verifiers. We distinguish four cases for the honesty of P_i and $P_{i'}$. The steps are also similar to $\mathcal{S}_{\mathsf{put}}$ but with $\mathcal{F}^{\mathsf{oc}}$ and $\mathcal{F}^{\mathsf{match}}$.

Simulator description. We are now ready to describe the simulator. In each round $t \leq T$, the simulator $\mathcal{S}$ runs as follows depending on the current phase the protocol.
Initialization: Let P_i be the trader committing to a **good** inventory. If P_i **is corrupted**:

1. Receive the commitments of inventory values and $[\![\tau_i]\!]$ from P_i; obtain the inputs that $\mathcal{A}$ sends to $\mathcal{F}^{\mathsf{zero}^+}$ and $\mathcal{F}^{\mathsf{ec}}$, and simulate the output of such ideal functionalities for each corrupted trader in I.
2. Forward $(\mathsf{init}, P_i, m_i)$ to $\mathcal{F}_{\mathsf{CFM}}$; if the ideal functionality returns 0 simulate an abort of the protocol.
3. Receive the decommitments corresponding to $[\![v_i]\!]$, $[\![\widehat{v_i}]\!]$, $[\![c_i]\!]$, $[\![f_{\mathsf{bad},i}]\!]$, $[\![f_{\mathsf{del},i}]\!]$, $[\![f_{\mathsf{out},i}]\!]$, and simulate an abort of the protocol if such values are not valid openings.
4. Run $\mathcal{S}_{\mathsf{put}}$ (for the case of corrupted P_i).

If P_i **is honest** we proceed as follows.

1. Forward commitments to zero for each of the values broadcast by P_i in the first step of the initialize phase; obtain the inputs that $\mathcal{A}$ sends to $\mathcal{F}^{\mathsf{zero}^+}$ and $\mathcal{F}^{\mathsf{ec}}$, and simulate the output of such ideal functionalities for each corrupted trader in I.
2. Open the commitments to zero corresponding to $[\![v_i]\!]$, $[\![\widehat{v_i}]\!]$, $[\![c_i]\!]$, $[\![f_{\mathsf{bad},i}]\!]$, $[\![f_{\mathsf{del},i}]\!]$, $[\![f_{\mathsf{out},i}]\!]$.
3. Run $\mathcal{S}_{\mathsf{put}}$ (for the case of honest P_i).

Post/Cancel Order: Let P_i be the trader posting an order or canceling a previous order. We distinguish two cases for Cancel Order and four cases for Post Order. If P_i **is corrupted:**

1. For Post Order, receive the values $(1, l, v)$ and $[\![i]\!]$ from P_i and forward $(\mathsf{post\_order}, P_i, t, l, v)$ to $\mathcal{F}_{\mathsf{CFM}}$. Otherwise, receive the values t' and $(-1, l, v)$ from P_i and forward $(\mathsf{cancel\_order}, P_i, t')$ to $\mathcal{F}_{\mathsf{CFM}}$.
2. Run $\mathcal{S}_{\mathsf{valid}}$ and $\mathcal{S}_{\mathsf{net}}$ (for the case of corrupted P_i).
3. For Post Order, for each command $(\mathsf{match}, t', l, v')$ received from $\mathcal{F}_{\mathsf{CFM}}$ run $\mathcal{S}_{\mathsf{match}}$ and then $\mathcal{S}_{\mathsf{net}}$ (for the case of corrupted P_i).

If P_i **is honest:**

1. For Post Order, receive $(\mathsf{post\_order}, t, l, v)$ from $\mathcal{F}_{\mathsf{CFM}}$. Otherwise receive $(\mathsf{cancel\_order}, t')$ from $\mathcal{F}_{\mathsf{CFM}}$.
2. Run $\mathcal{S}_{\mathsf{valid}}$ and $\mathcal{S}_{\mathsf{net}}$ (for the case of honest P_i).
3. For Post Order, for each command $(\mathsf{match}, t', l, v')$ received from $\mathcal{F}_{\mathsf{CFM}}$ run $\mathcal{S}_{\mathsf{match}}$ and then $\mathcal{S}_{\mathsf{net}}$ (for the case of honest P_i).

If P_i **is honest and** $P_{i'}$ **is corrupted (or viceversa):** proceed as above, depending on who is honest/corrupted.

Margin Settlement: During this phase the simulator $\mathcal{S}$ first obtains the list of pending orders that needs to be canceled directly from the broke corrupted traders and indirectly from the broke honest traders (through a cancellation received from the ideal functionality). Similarly the simulator $\mathcal{S}$ obtains the new orders to be matched directly from the broke corrupted traders and indirectly from the broke honest traders (through a new order received from the ideal functionality). If P_i **is corrupted** we proceed as follows.

1. For each order to be canceled:
   (a) Receive the values t' and $(-1, l, v)$ from P_i; obtain the inputs that $\mathcal{A}$ sends to $\mathcal{F}^{\mathsf{oc}}$, and simulate the output of such ideal functionality for each corrupted trader in I.
   (b) Run $\mathcal{S}_{\mathsf{valid}}$ (for the case of corrupted P_i).
   (c) Obtain the inputs that $\mathcal{A}$ sends to $\mathcal{F}_{\mathsf{compare}}$, and simulate the output of such ideal functionality, i.e. output 0 all the time until the last order to be canceled, for each corrupted trader in I.

2. Run $\mathcal{S}_{\mathsf{net}}$ (for the case of corrupted P_i).
3. For each order to be matched:
   (a) Receive the values $(1, l, v)$ and $[\![i]\!]$ from P_i; forward $(\mathsf{post\_order}, P_i, t, l, v)$ to $\mathcal{F}_{\mathsf{CFM}}$.
   (b) Run $\mathcal{S}_{\mathsf{valid}}$ and $\mathcal{S}_{\mathsf{match}}$ (for the case of corrupted P_i).
   (c) Obtain the inputs that $\mathcal{A}$ sends to $\mathcal{F}_{\mathsf{compare}}$, and simulate the output of such ideal functionality, i.e. output 0 all the time until the last order to be matched, for each corrupted trader in I.
4. Run $\mathcal{S}_{\mathsf{net}}$ (for the case of corrupted P_i).
5. Run $\mathcal{S}_{\mathsf{backup}}$ (for the case of corrupted P_i).

If P_i **is honest:** Same as mentioned above, except that the values (l, v') for each order to be canceled and matched are obtained from $\mathcal{F}_{\mathsf{CFM}}$.

Mark To Market: Let P_i be the trader committing to a good inventory with marked-to-market values. If P_i **is corrupted:**

1. Run $\mathcal{S}_{\mathsf{get}}$ (for the case of corrupted P_i).
2. Receive the commitments and obtain the inputs that $\mathcal{A}$ sends to $\mathcal{F}^{\mathsf{mtm}}$, and simulate the output of such ideal functionality for each corrupted trader in I.
3. Run $\mathcal{S}_{\mathsf{put}}$ (for the case of corrupted P_i).

If P_i **is honest:**

1. Run $\mathcal{S}_{\mathsf{get}}$ (for the case of honest P_i).
2. Forward commitments to zero for each of the values broadcast by P_i in the second step of the **Mark to Market** phase; obtain the inputs that $\mathcal{A}$ sends to $\mathcal{F}^{\mathsf{mtm}}$, and simulate the output of such ideal functionality for each corrupted trader in I.
3. Run $\mathcal{S}_{\mathsf{put}}$ (for the case of honest P_i).

Indistinguishability of the simulation. We need to show that for all PPT adversaries $\mathcal{A}$, all $I \subseteq [N]$, and every auxiliary input $z \in \{0,1\}^*$, the following holds:

$$\mathtt{REAL}_{\Pi_{\mathsf{DFM}},\mathcal{A}(z),I} \approx_c \mathtt{IDEAL}_{\mathcal{F}_{\mathsf{CFM}},\mathcal{S}(z),I}.$$

We start by considering a hybrid experiment $\mathtt{HYBRID}^1_{\mathcal{A}(z),\mathcal{S}_1,I}$ with a simulator $\mathcal{S}_1$ that runs exactly the same as $\mathcal{S}$, except that $\mathcal{S}_1$ also plays the role of the ideal functionality

$\mathcal{F}_{\mathsf{CFM}}$ on its own. This means that $\mathcal{S}_1$ directly receives the inputs of other honest traders that are not under control of $\mathcal{A}$. Clearly, for all adversaries $\mathcal{A}$, all subsets I, and every auxiliary in put $z \in \{0,1\}^*$, we have that $\mathtt{HYBRID}^1_{\mathcal{A}(z),\mathcal{S}_1,I} \equiv \mathtt{IDEAL}_{\mathcal{F}_{\mathsf{CFM}},\mathcal{S}(z),I}$, as there is no difference in generating the view of $\mathcal{A}$ in the two experiments.

Next, we consider another hybrid experiment $\mathtt{HYBRID}^2_{\mathcal{A}(z),\mathcal{S}_2,I}$ with a simulator $\mathcal{S}_2$ that runs exactly the same as $\mathcal{S}_1$, except that whenever $\mathcal{S}_1$ committs to zero values when dealing with dishonest verifiers, $\mathcal{S}_2$ commits to the real values received from the honest provers. The lemma below shows that the two experiments are statistically close.

Lemma 4.1. *For all (unbounded) adversaries $\mathcal{A}$, all $I \subseteq [N]$, and every $z \in \{0,1\}^*$:* $\mathtt{HYBRID}^1_{\mathcal{A}(z),\mathcal{S}_1,I} \approx_s \mathtt{HYBRID}^2_{\mathcal{A}(z),\mathcal{S}_2,I}$

Proof. The proof is down to the statistical hiding property of the non-interactive commitment Com.

We consider a variant of the statistical hiding property where a distinguisher $\mathcal{D}$ is given access to a left-or-right oracle $O_{\mathsf{lr}}(b,\cdot)$, parametrized by a bit $b \in \{0,1\}$, that upon input $v \in \{0,1\}^*$ returns $[\![v]\!]$ (if $b = 0$) or $[\![\mathbf{0}]\!]$ (if $b = 1$), where $|\mathbf{0}| = |v|$; hence, we have Com is statistically hiding if for all computationally unbounded $\mathcal{D}$,

$$\left|\Pr\left[\mathcal{D}^{O_{\mathsf{lr}}(0,\cdot)}(1^\lambda) = 1\right] - \Pr\left[\mathcal{D}^{O_{\mathsf{lr}}(1,\cdot)}(1^\lambda) = 1\right]\right| \leq \nu(\lambda),$$

for a negligible function $\nu : \mathbb{N} \to [0,1]$. By a standard hybrid argument, as long as $\mathcal{D}$ makes a polynomial (in λ) number of oracle queries, the above flavor of statistical hiding is equivalent to that of Com. Assume there exists a distinguisher $\mathcal{D}'$ and a polynomial $p(\lambda)$, such that, for some $I \subseteq [N]$ and $z \in \{0,1\}^*$, and for infinitely many values of $\lambda \in \mathbb{N}$, we have that

$$\Bigg|\Pr\left[\mathcal{D}'(\mathtt{HYBRID}^1_{\mathcal{A}(z),\mathcal{S}_1,I}) = 1\right] - \Pr\left[\mathcal{D}'(\mathtt{HYBRID}^2_{\mathcal{A}(z),\mathcal{S}_2,I}) = 1\right]\Bigg| \geq 1/p(\lambda).$$

We can construct a distinguisher $\mathcal{D}$ breaking the statistical hiding property of Com as follows. $\mathcal{D}$ runs $\mathcal{A}$ and simulates an execution of protocol Π_{DFM} exactly as $\mathcal{S}_1$ does, except that whenever $\mathcal{S}_1$ forwards a commitment to zero, $\mathcal{D}$ asks a query to the left-or-right oracle and sends the output of the oracle to $\mathcal{A}$; the value v for each oracle query is equal to the value $\mathcal{S}_2$ would commit to (instead of committing to zero).

In case $\mathcal{D}$ receives always commitments to zero, the view of $\mathcal{A}$ when run by $\mathcal{D}$ is identical to the view in the first hybrid experiment; on the other hand, in case $\mathcal{D}$ receives

always commitments to the values queries to the left-or-right oracle, the view of $\mathcal{A}$ when run by $\mathcal{D}$ is identical to the view in the second hybrid experiment.

Thus, $\mathcal{D}$ retains the same advantage of $\mathcal{D}'$. This concludes the proof. □

The lemma below says that the view of the adversary in the last hybrid experiment is computationally indistinguishable from the view in the real experiment.

Lemma 4.2. *For all PPT adversaries $\mathcal{A}$, all $I \subseteq [N]$, and every $z \in \{0,1\}^*$, it is* $\mathtt{HYBRID}^2_{\mathcal{A}(z),\mathcal{S}_2,I} \approx_c \mathtt{REAL}_{\Pi_{\mathsf{DFM}},\mathcal{A}(z),I}$.

Proof. Fix $I \subseteq [N]$, and $z \in \{0,1\}^*$. Consider the following events, defined over the probability space of the last hybrid experiment.

Event $\mathtt{Bad}_{\mathtt{inv}}$: The event becomes true whenever $\mathcal{A}$ can modify the inventory of a corrupted trader P_i, by finding two distinct valid openings for a token τ_i. The computational binding property of Com implies that $\Pr[\mathtt{Bad}_{\mathtt{inv}}]$ is negligible.

Event $\mathtt{Bad}_{\mathtt{spend}}$: The event becomes true whenever $\mathcal{A}$ can double spend the inventory of a corrupted trader P_i, by finding two distinct valid openings for $[\![\tau_i]\!]$. The computational binding property of Com implies that $\Pr[\mathtt{Bad}_{\mathtt{inv}}]$ is negligible.

Event $\mathtt{Bad}_{\mathtt{forge}}$: The event becomes true whenever $\mathcal{A}$ forges an inventory of a trader P_i, by finding two distinct valid authentication paths for a leaf $[\![\tau_i]\!]$ of the Merkle Tree. The computational binding property of the Merkle Tree, which follows from the collision resistance of the underlying hash function, implies that $\Pr[\mathtt{Bad}_{\mathtt{forge}}]$ is negligible.

Event $\mathtt{Bad}_{\mathtt{swap}}$: The event becomes true whenever $\mathcal{A}$ claim a pending order of an honest trader $P_{i'}$, by finding two valid openings for the commitment $[\![i']\!]$. The computational binding property of Com implies that $\Pr[\mathtt{Bad}_{\mathtt{swap}}]$ is negligible.

Define $\mathtt{Bad} := \mathtt{Bad}_{\mathtt{inv}} \vee \mathtt{Bad}_{\mathtt{spend}} \vee \mathtt{Bad}_{\mathtt{forge}} \vee \mathtt{Bad}_{\mathtt{swap}}$. It is not hard to see that conditioning on $\mathtt{Bad}$ not happening, the view of $\mathcal{A}$ is identical in the two experiments. This is because the only difference between the last hybrid and the real experiment is that in the former experiment the values $m_i, v_i, \widehat{m_i}, \widehat{v_i}, c_i, f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i}$ are read from the internal storage of $\mathcal{S}_2$ (playing the role of $\mathcal{F}_{\mathsf{CFM}}$), whereas in the latter experiment these values are specified by the attacker. Hence, by a standard argument, for all PPT distinguishers $\mathcal{D}$:

$$\Big|\Pr\left[\mathcal{D}(\mathtt{HYBRID}^2_{\mathcal{A}(z),\mathcal{S}_2,I}) = 1\right] - \Pr\left[\mathcal{D}(\mathtt{REAL}_{\Pi_{\mathsf{DFM}},\mathcal{A}(z),I}) = 1\right]\Big| \leq \Pr[\mathtt{Bad}].$$

The proof of the lemma now follows by a union bound. □

Combining the above two lemmas, we obtain that the real and ideal experiment are computationally close, as desired. □

4.8 Protocol Optimization

Several optimizations to the protocol improve the practical performance of FuturesMEX and reduce the computational cost to 30%.

We streamline the number of the validations of the commitments by packing $f_{\mathsf{bad},i}$, $f_{\mathsf{del},i}$, $f_{\mathsf{out},i}$ into a single integer f_i. This improves the circuit $\mathcal{F}^{\mathsf{token}}$, $\mathcal{F}^{\mathsf{invt}}$ as well as $\mathcal{F}^{\mathsf{flags}}$. Then, we combine the circuits $\mathcal{F}^{\mathsf{invt}}$, $\mathcal{F}^{\mathsf{uinv}}$, $\mathcal{F}^{\mathsf{rng}}$, and $\mathcal{F}^{\mathsf{net}}$. Proof generations can also be parallelized in a protocol step, e.g. in Π_{valid}, $\mathcal{F}^{\mathsf{invt}}$, $\mathcal{F}^{\mathsf{uinv}}$, $\mathcal{F}^{\mathsf{rng}}$, $\mathcal{F}^{\mathsf{net}}$, $\mathcal{F}^{\mathsf{zero}^+}$ and $\mathcal{F}^{\mathsf{token}}$ are independent of each other.

Furthermore, since functionalities $\mathcal{F}_{\mathsf{compare}}$ and $\mathcal{F}_{\mathsf{anncheck}}$ are used extensively in our protocol, their consistency check of the commitments slows down the entire protocol. We can replace the full functionality $\mathcal{F}_{\mathsf{compare}}$ with a lighter functionality $\mathcal{F}_{\mathsf{dtc}}$ to detect the flag in an unwanted state without validating the consistency of the commitments and to randomly select a P_y in the unwanted state to open the flag to check. This is not a problem as P_y is not linked to any traders from the previous steps or the subsequent ones due to the anonymity mechanism (Merkle Tree and ZK Functionalities). She is just an anonymous volunteer for this round. The full positive check functionality $\mathcal{F}_{\mathsf{anncheck}}$ can be similarly replaced.

The *Secure Detection* $\mathcal{F}_{\mathsf{dtc}}$ runs with common input (f) (the unwanted state such as the broke state) and interacts with a set of players $\langle P_1, \ldots, P_N \rangle$ and receive (P_i, f_i, r_i) from each P_i. Upon receiving all inputs, let $\mathcal{B}^* = \langle P_y \rangle$ be the set of traders such that $(f_y = f)$, and let c_f be the size of $\mathcal{B}^*$, if $c_f > 0$, compute $j = \sum r_i \bmod c_f$ and output the index y of the j-th broke trader P_y in $\mathcal{B}^*$ to all players and $\perp$ otherwise. In the protocol, each trader samples a random r_i and forwards (P_i, f_i, r_i) to $\mathcal{F}_{\mathsf{dtc}}$ with a common input f to obtain y or $\perp$. If the outcome is $\perp$, each trader P_i proves that her flag is different from the f (using $\mathcal{F}^{\mathsf{nec}}$). Otherwise, the trader P_y proves that the inventory flag is the same as the common input by decommitting the flags. Any trader unable to prove either properties is considered aborting.

The *Secure All Non-Negative Check* $\mathcal{F}_{\mathsf{anncheck}}$ is modified similarly to $\mathcal{F}_{\mathsf{dtc}}$ to interact with a set of players $\langle P_1, \ldots, P_N \rangle$ as follows: For all inputs (P_i, η_i, r_i) from each P_i, let $\mathcal{B}^* = \langle P_y \rangle$ be the set of traders such that $(\eta_i < 0)$, and let c_f be the size of $\mathcal{B}^*$, if $c_f > 0$, compute $j = \sum r_i \bmod c_f$ and output the index y of the j-th broke trader P_y in $\mathcal{B}^*$ to all players and $\perp$ otherwise. In the protocol, if $\mathcal{F}_{\mathsf{anncheck}}$ outputs $\perp$, all traders need to prove a non-negative net position using $\mathcal{F}^{\mathsf{zero}^+}$. Otherwise, the trader P_y proves the negative position with $\mathcal{F}^{-}$. Any trader unable to prove either properties is considered aborting.

Remark 4.6. *In the security proof (§4.7) when packing the three flags into one single integer or combining several circuits into one for optimization, the simulator changes*

accordingly. In the simulator description, we just replace the three flags ($f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i}$) *with one* f_i*. It is similar for optimized functionalities be replaced with the new one. In* $\mathcal{S}_{\mathsf{get}}$*, we replace* $\mathcal{F}^{\mathsf{inv}}$ *and* $\mathcal{F}^{\mathsf{uinv}}$ *with* $\mathcal{F}^{\mathsf{inv+uinv}}$ *or in* $\mathcal{S}_{\mathsf{net}}$ *we replace* $\mathcal{F}_{\mathsf{compare}}$ *with* $\mathcal{F}_{\mathsf{dtc}}$*.*

4.8.1 Optimized Building Blocks

For the flag packing optimization, the relations simply combine $f_{\mathsf{bad},i}$, $f_{\mathsf{del},i}$, $f_{\mathsf{out},i}$ into f_i. Possible values are 000, 100, 110, and 111 for good, broke, canceled and netted. The state transition function is similar to Fig. 4.4. Below is the combination of relations.

Optimized Relation for Inventory Retrieval + Correct Post/Cancel Update

The relation $R^{\mathsf{inv+uinv}}$ takes as input the following statement and witness:

$$x_i^{\mathsf{inv+uinv}} = (\rho, \tau_i', [\![m_i]\!], [\![v_i]\!], [\![\widehat{m_i}^*]\!], [\![\widehat{v_i}^*]\!], [\![c_i^*]\!], [\![f_i]\!], \delta_c, l, v) \tag{4.22}$$

$$w_i^{\mathsf{inv+uinv}} = (path_i, [\![\tau_i']\!], r_i', m_i, v_i, \widehat{m_i}, \widehat{v_i}, c_i, f_i). \tag{4.23}$$

The output of $R^{\mathsf{inv+uinv}}$ is defined to be one iff the following conditions are met: (i) The commitments in the statement are consistent with the corresponding values in the witness; (ii) $\mathsf{Auth}(\rho, path_i, [\![\tau_i']\!]) = 1$; (iii) The value τ_i' is the commitment of the inventory at a previous round, i.e. $\tau_i' = \mathsf{Com}(m_i||v_i||\widehat{m_i}||\widehat{v_i}||c_i||f_i; r_i')$. (iv) $\widehat{m_i}^* = \widehat{m_i} - \delta_c \cdot p_\ell \cdot v$, $\widehat{v_i}^* = \widehat{v_i} + \delta_c \cdot v$, and $c_i^* = c_i + \delta_c$.

Optimized Relation for Correct Computation/Speculation of an Instant Net Position

The relation $R^{\mathsf{rng+net}}$ takes as input the following statement and witness:

$$x_i^{\mathsf{rng+net}} = ([\![m_i]\!], [\![v_i]\!], [\![\eta_i]\!], \mathcal{O}_{\mathsf{buy}}, \mathcal{O}_{\mathsf{sell}}) \tag{4.24}$$

$$w_i^{\mathsf{rng+net}} = (m_i, v_i, \eta_i). \tag{4.25}$$

The output of $R^{\mathsf{rng+net}}$ is defined to be one iff the following conditions are satisfied: (i) The commitments in the statement are consistent with the corresponding values in the witness; (ii) $\eta_i = m_i + p_{\mathsf{lb}} \cdot V_{\mathsf{lb}} + p_{\mathsf{ub}} \cdot (|v_i| - V_{\mathsf{lb}})$ where (iii) $V_{\mathsf{lb}} \leq |v| \leq V_{\mathsf{ub}}$; (iv) Eq. (4.26) $\vee$ Eq. (4.27) $\vee$ Eq. (4.28) below holds:

$$v > 0 \wedge ((p_{\mathsf{lb}}, V_{\mathsf{lb}}), (p_{\mathsf{ub}}, V_{\mathsf{ub}})) \in \mathcal{O}_{\mathsf{buy}} \tag{4.26}$$

$$v < 0 \wedge ((p_{\mathsf{lb}}, V_{\mathsf{lb}}), (p_{\mathsf{ub}}, V_{\mathsf{ub}})) \in \mathcal{O}_{\mathsf{sell}} \tag{4.27}$$

$$v = 0 \wedge (p_{\mathsf{lb}} V_{\mathsf{lb}}) = (p_{\mathsf{ub}}, V_{\mathsf{ub}}) = (0, 0). \tag{4.28}$$

4.9 Beyond Security-With-Abort

We opt towards the mechanism of Hawk [119, Appendix G, §B] in which private deposits are frozen and the identified aborting parties cannot claim back the deposits in the withdrawal phase. This fits precisely our scenario as the deposit can be defined to match the initial margin, (which is the largest amount a trader can lose when being netted out[14]). First we show that honest participants can eventually move to the **Mark To Market** phase to cash their inventory. Let us denote by Adv the set of adversaries who abort between time t and time $t+1$ given a backup tree $\mathcal{T}_{\mathcal{U}}$ with a solvable inventory of all traders (m_i, v_i) and the corresponding mid price $\bar{p}$. Since $\mathcal{T}_{\mathcal{U}}$ is a valid tree[15] it satisfies the constraints in Table 4.3 ($\eta_i \geq 0$). Since $\eta_i = m_i + \mathsf{cash}(v_i) \leq m_i + \bar{p} \cdot v_i$, we have

$$0 \leq \sum\nolimits_{i \notin Adv} (m_i + \bar{p} \cdot v_i) \leq \sum\nolimits_i m_i^0$$

This implies that there will be no unexpected loss to cover ($0 \leq \ldots$) nor additional money would be created ($\ldots \leq \sum_i m_i^0$). Then honest traders can proceed to the **Mark To Market** phase to split their own trade proceedings. Possibly this will include the money that they have obtained from the traders controlled by the adversary which abandoned the computation. If enough traders accept the move so that it ends into the public ledger this would be considered an acceptable solution. From an economics perspective, the adversary would be penalized with at least its initial cash margin which could be substantial.

Now, we just need to extend our protocol to identify the aborting parties in various protocols and prevent them from claiming the deposit in **Mark To Market** phase by requiring each trader to present a proof of participation in the round where the abort happens. A further step is to divide the money of the adversary if at the end of the **Initialize** phase, the total sum of money is computed (by an MPC protocol, i.e. $\mathcal{F}_{\mathsf{sum}}$ that receives m_i from each trader and computes $\sum_i m_i$). In the **Mark To Market** phase, by computing the sum of money of the honest traders after the updates of the inventories, we can find the difference corresponding to the money of the adversary and share it by updating the inventories with the shares.

Formally, in an abort, every honest trader maintains a set of *spent* tokens τ_i' of the participants. The **Mark To Market** phase now runs exactly the same as before except

[14] In practice traders can deposit additional funds when receiving a margin call. These incremental deposits could be easily incorporated into our setting by adding a step similar to **Initialize** with an additional ZK for the increment.

[15] The latest backup tree is the only point of restoration where every trader's margin is non-negative. Ideally, a recovery procedure should exist, we leave this for future work. However, the trader who responsible for the abort is still penalized.

that a trader must prove in zero knowledge he knows the opening of a token τ_i' in the last step of the relation R^{oinv}, taking as input the statement $x_i^{\mathsf{oinv}} = (\tau_i')$ and witness $w_i^{\mathsf{oinv}} = (r_i', m_i, v_i, \widehat{m_i}, \widehat{v_i}, c_i, f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i})$. The output of R^{oinv} is defined to be one iff $\tau_i' = \mathsf{Com}(m_i||v_i||\widehat{m_i}||\widehat{v_i}||c_i||f_{\mathsf{bad},i}||f_{\mathsf{del},i}||f_{\mathsf{out},i}; r_i')$. In the simplest settings, as in a joint step, the set of tokens from participants can be trivially constructed. We discuss the disqualification of an adversary in a more complex case where he refuses to match an order o':

1. All traders P_i cancel all orders o with $o \neq o'$.
2. All traders P_i prove they do not own o' by decommitting $[\![c_i]\!]$ and showing that $c_i = 0$.

Similarly, if the **Margin Settlement** phase is aborted,

1. In cancellation phase, all traders retrieve their inventory with a token τ_i', and prove that the inventory flags is not *broke*, i.e. $(f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i}) \neq (1, 0, 0)$.
2. In liquidation phase, all traders retrieve their inventory with a token τ_i', and prove that the inventory flags is not *canceled*, i.e. $(f_{\mathsf{bad},i}, f_{\mathsf{del},i}, f_{\mathsf{out},i}) \neq (1, 1, 0)$.

For checking that the inventory flag is not in a specified state an additional *NP* relation is necessary. We define $R^{\mathsf{3nec}}(([\![v_1]\!], [\![v_2]\!], [\![v_3]\!], v_1', v_2', v_3'), (v_1, r_1, v_2, r_2, v_3, r_3))]$ as the relation of commitments to equal values, whose output is one iff $R^{\mathsf{oc}}([\![v_i]\!], v_i) = 1$ and $v_i \neq v_i'$ for $i = \{1, 2, 3\}$.

4.10 Implementation

We have implemented all phases of our protocol both the direct and the optimized versions. For the anonymous network and the distributed consensus protocol, off-the-shelf implementations will do.[16]

Implementation of Components. We follow ZeroCash [164] in the instantiation of our building blocks, at a security level of 128 bits. Let $\mathsf{H}(\cdot)$ be a collision-resistant compression hash function that maps l bits input ($l \geq 512$) into 256 bits output (e.g. SHA256[17]). We

[16] We use the distributed ledger HyperLedger [109] in PBFT mode as a Byzantine fault tolerant storage for each protocol step. Each broadcast is replaced with a write into the distributed ledger. To communicate anonymously, the traders hide behind a Tor network [172] While we mention Tor, zcash, and HyperLedger in our implementation, we can replace any sub-protocol with other protocols for the same task, without affecting the security. See [6,179] for a comparison of different solutions.

[17] We have chosen SHA-256 as it is possible to use zcash in FuturesMEX, and we can leverage its builtin implementation in our development framework libsnark.

use $\mathsf{H}(\cdot)$ to instantiate the commitment scheme Com and the hash function for the binary Merkle Tree $\mathcal{T}$. The zero-knowledge functionality $\mathcal{F}^R$ is instantiated with zk-SNARKs for arithmetic circuit satisfiability [21], while generic MPC is used for the hybrid ideal functionalities $\mathcal{F}_{\mathsf{compare}}$ and $\mathcal{F}_{\mathsf{anncheck}}$.

Our zk code is based on the libsnark library [166]. We split $\mathcal{F}^{\mathsf{inv}}$ into $\mathcal{F}^{\mathsf{invm}}$ to check whether the token is one of the leaves of the Merkle Tree and $\mathcal{F}^{\mathsf{invt}}$ to check the consistency of new commitments and old tokens. Our prototype supports 32 bits signed integers (see footnote 1 on prices in §4.2), a Merkle Tree of depth 10 and up to 10 range choices[18] (i.e. $\mathcal{O}_{\mathsf{sell}}$ and $\mathcal{O}_{\mathsf{buy}}$ used in $\mathcal{F}^{\mathsf{rng}}$). Our MPC uses the SPDZ library [23, 69, 145, 168].[19]

4.10.1 Evaluation

We evaluate our protocol in three steps. First, we evaluate the performance of the cryptographic primitives used in our protocol, i.e. the zk-SNARK circuits and the MPC functionalities, both in the offline and online phase on a concrete implementation. Then, we use the obtained values to estimate the performance of each phase in our protocol. Finally, we evaluate the full protocol's performance by matching the above estimates with the public data available from the order books. while network latencies are critical for high-speed trading, we ignore them here since this issue is well understood by traders either using known optimizations [125, 142] or buying dark fiber to cut delays between exchanges [83].

All our experiments are run on an Amazon EC2 r4.4xlarge instance (Intel Xeon E5-2686 v4 @ 2.3Ghz, 16 cores, 122 GB RAM). We exclude communication cost since each operation requires less than 20 commitments and 10 zk-SNARKs proof. Per operation, the data are less than 4KB (see Table 4.8). The MPC offline phase protocol is pipelined with zk-SNARKs proof generation; thus, we exclude the MPC offline phase as well.

zk-SNARK Circuits Performance In Table 4.8, we report the performance metrics for the pre-processing steps: the key generation time (KeyGen), the size of the proving keys (PK), and the size of the verifying (VK) keys. We also report the time to generate a proof (Proving Time) and to verify it (Verify time), as well as the size of the proof during the actual trading execution. Proving key size and proof generation time scale is linear with the a number of commitments of the relation.

[18] The set of net position range choices is a subset of the price ranges. The size of this set only affects the performance in the direct implementation where the net position choices were committed. In the optimized version, we have removed such an intermediate step; hence, the size of such a set does not affect the protocol performance.

[19] While libsnark is efficient and scalable, SPDZ hits the limit of 10 parties owing to the complexity in implementing SHA256 with the library; i.e. right-shift is not natively supported for 32-bits word. Thus, we must implement it using left-shift and other bitwise operations.

Table 4.8: zk-SNARK Simple and Opt. Circuits Performance

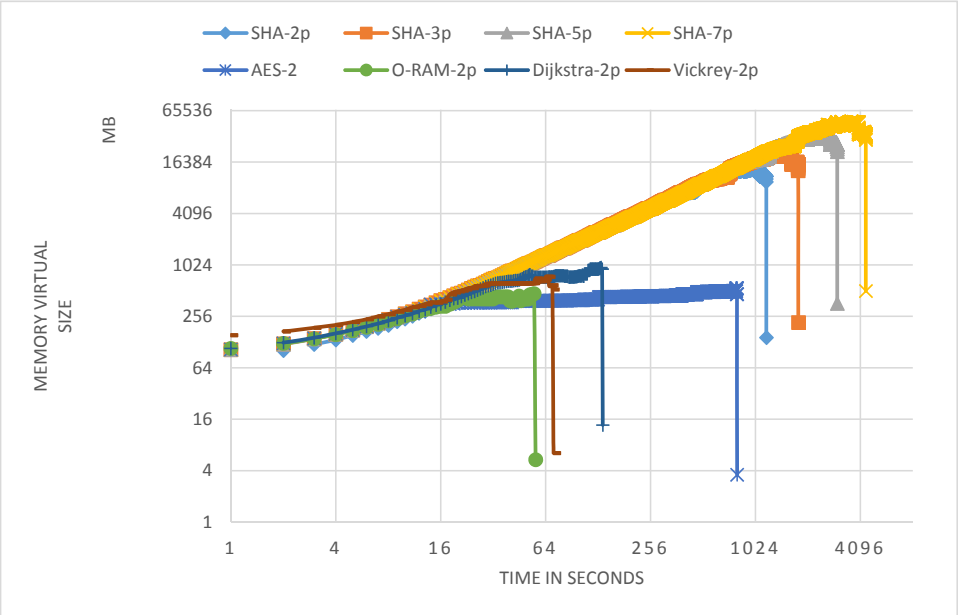
	Pre-processing			On-Line Trading		
Circuit	KeyGen (ms)	| PK | (MB)	| VK | (KB)	Prove (ms)	| Proof | (B)	Verify (ms)
$\mathcal{F}^{\mathsf{rng}}$	8759	119	9	4752	287	31
$\mathcal{F}^{\mathsf{invm}}$	16778	210	2	8447	..	29
$\mathcal{F}^{\mathsf{token}}$	15925	189	..	7642	..	27
$\mathcal{F}^{\mathsf{flags}}$	12943	171	..	6115	..	..
$\mathcal{F}^{\mathsf{invt}}$	12954	171	..	6111	..	..
$\mathcal{F}^{\mathsf{uinv}}$	9650	116	..	4748	..	..
$\mathcal{F}^{\mathsf{match}}$	9644	116	..	4748	..	..
$\mathcal{F}^{\mathsf{net}}$	9638	115	..	4691	..	..
$\mathcal{F}^{\mathsf{mtm}}$	5456	57	..	2365	..	..
$\mathcal{F}^{\mathsf{ec}}$	3343	38	..	1429	..	..
$\mathcal{F}^{\mathsf{zero}^+}$	1639	19	..	739	..	26
$\mathcal{F}^{\mathsf{max}}$	1635	19	..	739	..	..
$\mathcal{F}^{\mathsf{oc}}$	1635	19	..	729	..	..
Optimized						
$\mathcal{F}^{\mathsf{token}}$	..	157	..	5691	..	..
$\mathcal{F}^{\mathsf{flags}}$	..	97	..	3908	..	..
$\mathcal{F}^{\mathsf{invt}}$+$\mathcal{F}^{\mathsf{uinv}}$	..	137	..	5193	..	..
$\mathcal{F}^{\mathsf{rng}}$+$\mathcal{F}^{\mathsf{net}}$	..	84	..	3509	..	..

Table 4.9: MPC Performance

MPC Funct	Bytecode Size			On-Line Time		
#Traders	3	5	10	3	5	10
$\mathcal{F}_{\mathsf{compare}}$	425 MB	709 MB	1.4 GB	14s	24s	67s
$\mathcal{F}_{\mathsf{anncheck}}$	212 MB	354 MB	708 MB	7s	13s	36s

Performance of the MPC functionalities To gauge the effectiveness of our MPC components, we evaluate the performance for each functionality separately. Table 4.9 reports the size of the bytecode and the corresponding running times for 3, 5, and 10 traders. We expect this compilation phase to be run at each trader's computer as for security reason one would want to compile the circuit himself to make sure she runs the correct computation. The memory requirement for the compilation of the MPC functionalities using SHA-256 commitments crashed after 10 traders by exceeding 120 GB. We found that the dynamic memory requirement is typically 100x the final bytecode size (Fig. 4.12). This was not reported before (e.g. [69]), and it is an important insight on the limit of the technology.

Overall Evaluation In our experiment, we employed the futures trades of the first quarter of 2017 for the Lean Hog futures market (see Table 3.2) collected from the Thomson Reuters Tick History database [175]. For each day, we have five level limit orders (buy and sell, which we chose for the $\mathcal{F}^{\mathsf{rng}}$) and have transaction data at ticks level with mil-



The built-in examples of SPDZ are evaluated: (1) AES-2 is a 2-party AES encryption where one party inputs the message; whereas the other parties input the key; (2) O-RAM-2 is an Oblivous RAM retrieval while Dijisktra-2 is implemented based on O-RAM-2; (3) Vickrey-2 is a 2-party Vickrey Auction. We implemented the SHA-256 functionality and evaluated for 2, 3, 5, and 7 parties (SHA-2p, etc.). For all cases, the memory requirements are 100 times the final byte code size.

Figure 4.12: An evaluation of the memory requirements of different MPC functionalities over time

lisecond timestamps. At a low end, we must support a minimum of 10 traders and this is the limit we chose for illustrating our prototype. From the dataset, we are unable to determine the status of each trader (trader anonymity!), so we assume they have a large margin and never enter a broke state (i.e. we still check all the net positions to capture non-monotonicity; however, we exclude the worst case in Margin Settlement where we need to net out traders as a consequence of non-monotonicity). We can combine the number of posts, cancel, and matched orders from the market data (e.g. Table 3.2, 4.10 and 4.11) to estimate the corresponding execution overhead for a day trading. The final results for a Merkle Tree of depth 10 are reported in Table 4.10. The total runtime for each main operations, i.e. Post, Match and Cancel, are 187s, 188s and 177s respectively in the plain implementation. In the optimized implementation the total runtime are 51s, 52s and 53s respectively. In Table 4.11 we give some examples for the estimation of the protocol's main operations runtime in case we need a Merkle Tree of higher depth. The actual timing of the protocol is still slow compared with the millisecond delay required

Table 4.10: Runtime of Individual Market Operations with Merkle Tree of depth 10

Each individual operation is done in a few seconds for a market of 10 traders. With simple optimizations, we boost the performance and reduce the role of other traders in the computation of the critical "ecological" constraint.

	Plain Prot. Runtime		Opt. Prot. Runtime	
Protocol	Trader	Others (%)	Trader	Others (%)
Initialize	11s	-	9s	-
Post Order	39s	148s (79%)	24s	27s (53%)
Cancel Order	40s	148s (79%)	25s	27s (52%)
Match Order	29s	148s (84%)	26s	27s (51%)
MarkToMarket	28s	-	25s	-

Table 4.11: Runtime of Individual Market Operations with Merkle Tree of depth higher than 10

We use the Merkle Tree of depth 10 as a baseline measurement (see Table 4.8). As the performance of the $\mathcal{F}^{\mathsf{invm}}$ functionality (which only checks that a token is committed into the Merkle Tree as in the Auth operation) grows linearly with the depth of the Merkle Tree, we can estimate the performance in case of a Merkle Tree with higher depth d by multiplying with the scaling factor $\frac{d}{10}$. We simply replace the base measurement with the adjusted performance when we aggregate the runtime of each protocol operation.

Merkle Tree Depth	$d = 13$		$d = 15$		$d = 17$	
Protocol	Plain	Optimized	Plain	Optimized	Plain	Optimized
Post Order	192s	56s	196s	60s	199s	63s
Cancel Order	193s	57s	197s	61s	200s	64s
Match Order	182s	58s	186s	62s	189s	65s

by the CME.

To estimate the cost of a naïve MPC implementation, we use as building block the simplest of our stateless MPC functionalities $\mathcal{F}_{\mathsf{dtc}}$ to detect negative inputs and open one index. It costs only 0.2s for 10 traders. Then, we estimate the cost of a naïve MPC implementation of our stateful ideal functionality by accumulating the steps in Fig. 4.2 and Fig. 4.6 under the favorable assumption (for MPC), in which the execution times accrue linearly with the number of steps.

To estimate the burden of computation, we observe that the summary of each data point from the THTR is described by a tuple $\langle d, n_p, n_c, n_m, n_t \rangle$ where:

- d is the trading date,
- n_p is the number of post orders (# increases),
- n_c is the number of cancelled orders (# decreases),
- n_m is the number of matched orders (# actual trades),
- and n_t is the number of traders.

As the plain implementation cannot go beyond 10 traders we have assumed that only 10 traders could actually participate (so we cap n_t at 10). With this cap made, we estimate the required computation in a naïve MPC implementation according to Fig. 4.2 as follows.

- For **Post/Cancel Order**, an order requires 3 sub-steps per trader which yields $3n_t$ sub-steps to process an order.
- Similarly, for **Match Order**, an order requires 2 sub-steps per trader hence $2n_t$ sub-steps to match a trade.
- In each sub-step, one phase must walk through $\frac{n_p}{2}$ orders in average (These operations contribute to most of the generic MPC overhead).

They are then multiplied by the time $\tau^{mpc}(n_t)$ required by the elementary MPC operation $\mathcal{F}_{\mathsf{dtc}}$.

Differently, while generic MPC requires *all traders to compute for one trader*, the hybrid protocol allows traders to produce and verify the proof *on their own simultaneously* so the cost is not affected by n_t. The proof generation time $\tau^{hyb}_{i,gen}$ and the proof verification time $\tau^{hyb}_{i,ver}$ are actually performed by different traders. Furthermore we have limited the MPC time to only $\tau^{hyb}_{i,mpc}(n_t)$ per trader during the check of non-negative net positions. This is not important for calculating the overall crypto-overhead of operating the market in a distributed fashion (before moving to the next order both operations must be done) but will be important for the calculation of the proportional burden.

Therefore the total time to process a trading day d reported in Fig. 4.13 of the single trader follows the equations given below:

$$\begin{aligned} T^{mpc}_d &= \frac{n_p}{2}\left(\sum_{i=p,c} n_i 3n_t + n_m 2n_t\right)\tau^{mpc}(n_t) \\ T^{hyb}_d &= \sum_{i=p,c,m} n_i\left(\tau^{hyb}_{i,gen} + \tau^{hyb}_{i,ver} + \tau^{hyb}_{i,mpc}(n_t)\right) \end{aligned}$$

and similarly for the optimized version where the costs are estimated according to Table 4.10.

To estimate the fraction of retail and institutional traders ρ_t we use the data from [130] as well as the fraction of orders performed by retail traders ρ_o ($\rho_t = 0.71$, $\rho_o = 0.18$). Albeit TSX and CME are different exchanges, the skeweness against retail traders might be even more pronounced when the market is more active. For the MPC naïve computation a party has to participate in the computation regardless of whether the

party actually made any order. Therefore, the overall burden of computation by retail traders for naïve MPC (Fig. 4.14) is as follows:

$$R_d^{mpc} = \rho_t n_t T_d^{mpc}$$

In the hybrid approach, a retail trader only needs to verify proofs when an institutional trader has to generate proof. Hence, the computation by retail traders (Fig. 4.14) is determined by the following equation:

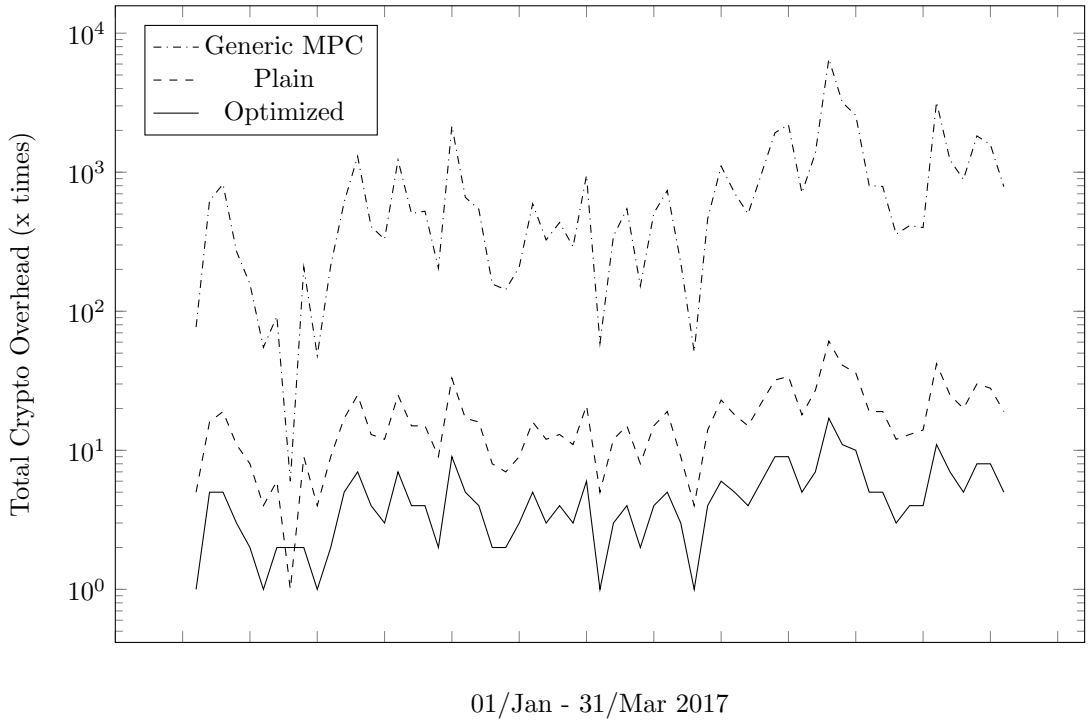
$$R_d^{hyb} = \rho_t n_t \sum_{i=p,c,m} n_i \left(\rho_o \tau_{i,gen}^{hyb} + (1-\rho_o)\tau_{i,ver}^{hyb} + \tau_{i,mpc}^{hyb}(n_t) \right)$$

Fig. 4.13 shows the overall record for the plain and optimized version as well as an estimation of a naive MPC implementation of the ideal functionality. For most of the entire quarter, our distributed protocol is optimized to have less than 5x overhead. These overheads can be further offset by parallelizing the traders' ZK-proofs (which can yield an improvement factor by 5x or 6x). This performance compares well with the estimates for the generic MPC algorithm whose smallest factor is 6x and is usually on a 10x to 1000x overhead.

Pure MPC imposes a significant burden on retail traders[20] (the overwhelming majority of the market) during peak times when algorithmic traders frequently post and almost immediately cancel practically all orders (see [130]). Our hybrid approach shifts the burden on computation on algorithmic traders. As shown in Fig. 4.14, retail traders would devote significant computational resources in a pure MPC implementation for allowing speculators to speculate.

The optimized implementation can break the barrier of 10 traders and do the full 66 traders of the peak day. Parallelization can further reduce the runtime of the sub-protocol to just 8s (compared with 24s of sequential proof generation). Furthermore, additional practical design decisions can increase the protocol throughput, e.g. if traders can prove their inventory is valid for a range of prices, they only need prove validity when the price fluctuates out of that range, or by allowing multiple traders to post/cancel in one round. We leave this for future research.

[20] Retail traders are typically hedgers. Of course they could also be speculators. This is a different classification based on the volume and quantity of orders. High frequency traders are overwhelmingly speculators. Retail traders make much less orders, mostly trades.



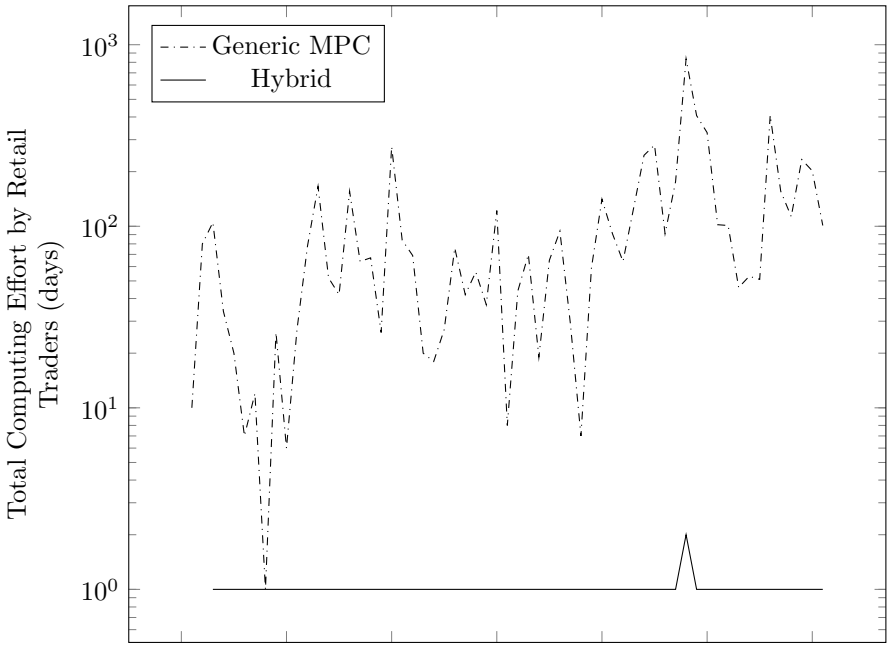
Performance of execution overhead regarding the expected processing time (1 day). For the optimized version, only 3 days of trading exhibits overheads greater than 10x, and only 16 days greater than 5x. These overheads are already offset by parallelizing the traders' ZK-proofs (each trader has to do several of these) to achieve an improvement factor of 6x.

Figure 4.13: Crypto Protocol Evaluation on Q1 of Lean-Hog

4.11 Related Work

Distributed Ledgers are ledgers maintained by a network of nodes. The most important property for distributed payment networks, being fully decentralized, is consensus among the nodes. The most prominent example of a distributed payment network is Bitcoin [144], whose core components are the Proofs-of-Work and the Blockchain. The current bottleneck of Bitcoin is its low throughput regarding its transactions-per-second (TPS) (roughly 10 TPS compared to 2000 TPS of Visa.) Several variants/extensions of Bitcoin appeared recently, including ZeroCoin [139], ZeroCash [164], and Ethereum [77].

Secure Multiparty Computation Seminal feasibility results in the theory of MPC established that any functionality is securely realizable via a distributed protocol in the computational (resp. information-theoretic) setting, assuming an honest minority (resp. majority) [18,55,97,156,182]. The recent progress on efficient implementations of general-purpose MPC protocols [9,68,69] opened up the way to advanced applications of a privacy-preserving data mining [129]. See Orlandi [147] for an overview of applications of MPC.



With MPC retail traders have to always participate whether they make an order or not (and they overwhelmingly do not [130]). They would be supplying to algorithmic traders some orders of magnitude of costly computing resources. With our approach the burden on retail traders is significantly smaller.

Figure 4.14: Total Burden of Computation by Retail Traders

SNARKs. The influential work of Micali on computationally sound proofs [138] offered the first zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK) for all of *NP*, relying on the random oracle heuristic [15]. Later work [27, 69] showed that zk-SNARKs existed in the standard model, based on the knowledge-of-exponent assumptions. These type of assumptions seem to be inherent for constructing zk-SNARKs [93]. An overview of these results is found in Walfish and Blumberg [180]. In terms of implementation, the most efficient ones include Ben-Sasson *et al.* [19,21] and Parno *et al.* [149].

4.12 Summary

FuturesMEX is the first practical secure, distributed financial intermediation without a trusted third party. Our chosen example has been a landmark institution of financial intermediation: a Futures Market Exchange.

Besides its practical application, such a realization is interesting from a security perspective as it provides a rich security functionality with varied and potentially conflicting requirements. One should support a *public availability of information* about all actions performed by traders in the market, (such as post or cancel orders) as well as *public ver-*

ifiability of the integrity of private information from the traders. Furthermore, we must provide *participants anonymity* and *public unlinkability* with *global integrity guarantees and private linkability.*

Our hybrid protocol offers an efficient solution to the requirement of a *proportional burden of computation.* Traders infrequently making bids only participate in the protocol to control market risk with a significant saving of the computational resources they would need to stake if general purpose MPC was used to implement the ideal functionality.

We have shown that our security protocol can be actually implemented through libsnark [21], for the zero-knowledge proofs, and the SPDZ protocol [69], for securely realizing a few auxiliary sub-tasks that are used in our main protocol.

For our analysis of the actual trading days, we use the Thomson-Reuters database, including a complete trading history at the level of milliseconds, and show that the computation behind our protocol is within engineering reach. We simulated a *low frequency market* of a secure protocol using a normal server (as opposed to traders' typical supercomputers), executed within a day with a handful of exceptions.

For practical implementation, we found the libsnark library to be pretty scalable, while the SPDZ library of our direct implementation hit a hard limit of 10 traders owing to the dynamic memory requirements (with the natural encoding of SHA256 in Python). This was surprising as the final size of SPDZ bytecode was acceptable and consistent with the results of the literature [69].

Chapter 5

Dark Financial Intermediation with Witness Key Agreement

Dark pools (private markets) are increasingly popular mechanisms for trading financial instruments. In such pools, investors wish to disclose information only to trading partners whose transaction conditions and asset holdings satisfy a suitable arithmetic relation. The investor must securely authenticate traders against their committed information and the desired relation while traders should be eligible to keep their financial information secret. For example, in over-the-counter (OTC) markets, traders broadcast a price but not the quantity of their holding contracts. In this chapter, we describe a new cryptographic primitive called Witness Key Agreement (WKA) that enables secure, distributed dark financial intermediation. We then present a practical construction of WKA based on designated-verifier succinct zero-knowledge non-interactive argument of knowledge (zk-SNARK) for quadratic arithmetic programs (QAPs) and analyze the obtained scheme's security. We also provide both theoretical and practical performance evaluation for our WKA scheme.

5.1 Possible Solutions for Dark Financial Intermediation

Financial intermediation is traditionally based on trusted third-party solutions, such as exchanges (e.g. NASDAQ or the CME), or clearing mechanisms (e.g. EU's TARGET2-Securities and US's Depository Trust & Clearing Corporation). In the last years, new technologies have been proposed to replace centralized intermediaries with purely distributed protocols, e.g. the privacy-preserving cryptocurrency ZeroCash [164], or the crypto-based distributed futures exchange FuturesMEX (Chapter 4).

In those distributed systems, the users commit their financial information (e.g. accounts, or bids and quotes) to a distributed ledger and use zero-knowledge proofs to prove that their committed information satisfies a suitable relation to preserve the integrity of the market and the solvency of the participants. For example to make a buy bid in a Futures Market, one must prove (privately to the central Exchange [169] or to all parties using a suitable crypto protocol as FuturesMEX in Chapter 4) to have enough money to buy the desired number of futures contract at the current price.

Different from public markets (where the limit order books are published while information on individual traders positions and quotes is private [169]), private markets (dark pools [107], often used by institutional investors and market makers), have tighter confidentiality requirements as minimal legitimate information leakage can be very costly [141].

In such a scenario, the investors may only want to eventually disclose data to a partner who meets some constraints for solvency and liquidity. For example she might want to sell at least v shares at price p and want to disclose v and p only to traders who committed to have cash $c \geq c'$ where $c' \geq pv$. Alternatively she might be willing to buy from somebody who has at least v' shares or accept a price pegged within an interval, etc. For the very same reasons, the trader might not just want to make his information fully public, but just to reassure the investor that he meets the constraints.

From a security perspective, these constraints are easily captured by an NP-relation R which we illustrate in Table 5.1. In this, the instance ϕ is the public information (including the trader's commitments and the investor's constraints), and the witness ω is the private information (the trader's committed information). An investor may look for traders with enough capacity and use the Sufficient Capacity (SC) relation in Table 5.1. A trader may ask the investor to show interest in some price ranges, e.g. from p'_- to p'_+ using the Price Range (PR) relation and additionally check the consistency of the challenged threshold using Matchable Bid (MB) (Table 5.1, if the investor has previously committed to the desired bid price p and bid volume v where $c' \geq pv$).

To authenticate such traders, a *trivial (but wrong)* solution is to ask each trader to couple a public key pk with a zk-SNARK [27] proof $\boldsymbol{\pi}$ for the satisfaction of the desired constraints on the financial instruments represented as an arithmetic relation R. The investor will encrypt the private offer with pk after verifying the proof $\boldsymbol{\pi}$. Only the trader with the corresponding private key sk can decrypt. Since the decryption condition above says nothing about the validity of $\boldsymbol{\pi}$, one cannot guarantee that pk is actually from the trader that produced $\boldsymbol{\pi}$. It may come from a man in the middle attacker (MITM) (who does not produce the proof).

Authenticated key exchanges (AKE) such as Password-AKE [17] or Credential-AKE [42] only support relations on credentials. Here we have other relations among values not re-

Table 5.1: Dark Pool Example Relations

We describe some of the dark pool example relations below. In each relation, let us denote $[\![x]\!] = \mathsf{SHA256}(x; r_x)$ as the public SHA256 commitment of the secret business variable x using randomness r_x. For a dark pool transaction, we denote by c the cash capacity of a trader, c' the threshold given by the investor. For a bid, we denote (p, v) as the bid price and the bid volume.

Relation	**Public** ϕ	**Secret** ω	**Conditions of** R
Sufficient Capacity (SC)	$[\![c]\!], c'$	c, r_c	$[\![c]\!] = \mathsf{SHA256}(c; r_c)$, $c \geq c'$
Price Range (PR)	$[\![p]\!]$, p'_+, p'_-	p, r_p,	$[\![p]\!] = \mathsf{SHA256}(p; r_p)$, $p'_- \leq p \leq p'_+$
Matchable Bid (MB)	$[\![p]\!]$, $[\![v]\!]$, p'_+, p'_-, c'	p, v, r_p, r_v	$[\![p]\!] = \mathsf{SHA256}(p; r_p)$, $p'_- \leq p \leq p'_+$ $[\![v]\!] = \mathsf{SHA256}(v; r_v)$, $c' \geq pv$

lated to credentials as they can change dynamically. Language-AKE [104] is more flexible but it does not support non-algebraic relations such as SHA-256 employed by FuturesMEX and ZeroCash [164].

Signature of Knowledge [103] (SoK) can certainly be used to sign the public key pk to prevent the mentioned MITM attack. However, SoK delivers only the public key pk of the trader hence only allow one-way communication from the investor to the trader unless we assume a public key infrastructure (PKI) for investors.[1] Besides, the trader is unsure the upcoming message encrypted with pk is from the investor: as pk is public, anyone can see it and send a message to the trader using pk.

An interested investor could also use Witness Encryption [90] (WE) with the desired constraints on the financial instruments represented as an arithmetic relation R, and only the traders who possess the witness ω for that instance ϕ such that $R(\phi, \omega) = 1$ could decrypt. However, general WE constructions [4, 88, 89, 92] are impractical as they rely on multilinear maps or Indistinguishabiliy Obfuscation while practical WE under a GS proof [71] cannot support arithmetic relation of depth greater than 1, e.g. SHA-256 as employed by ZeroCash [164] and FuturesMEX (Chapter 4).

One can also ask the investors and traders to join an MPC protocol through an exchange. In this respect, FuturesMEX (Chapter 4) provides support for an open Order Book. Cartlidge *et al.* [47] proposed a MPC for implementing dark pools. In contrast, we aim to support communication over a distributed ledger where trader information (e.g. the cash of a trader) is bound. The presence of a distributed ledger was not considered by Cartlidge *et al.* [47]. Since Cartlidge *et al.* [47] only evaluated the 2 or 3 party cases (one of them being a regulator that nowadays never participate in such exchanges and mostly have not the technical means to do so), it is unclear whether MPC will scale to

[1] However, this is not always feasible or desirable: an appropriate PKI may be unavailable, or the parties may want to remain anonymous, and not reveal their identities as anonymity can be critical (see the Price Discrimination Attack in Chapter 3).

the full number of traders and investors (see Fig. 4.13 and 4.14 in Chapter 4).

5.2 Dark Financial Intermediation with Witness Key Agreement

To support the dark pool scenario, we propose a new cryptographic primitive called the Witness Key Agreement (WKA) scheme.

In a WKA scheme defined for an NP-relation $R(\phi, \omega)$, the investor will first run the key challenge algorithm taking as input the relation R and outputs a public challenge parameter p_c and a secret challenge parameter s_c. Then, the investor broadcasts the public challenge. The interested trader runs the key response algorithm with input the relation R, the public challenge parameter p_c, the instance ϕ (her committed information), and the corresponding witness ω (his corresponding secret information). Then, the trader outputs a public response parameter p_r and a secret key k_r. Upon receiving the public response p_r and the corresponding instance ϕ, the investor runs the key derivation algorithm taking as inputs the relation R, the secret challenge parameter s_c, the instance ϕ, and the public response p_r and outputs a secret key k_c.

The four required security properties of a WKA scheme, informally speaking, are (i) *Correctness:* the key agreement is successful, i.e. $\mathsf{k}_r = \mathsf{k}_c$ if $R(\phi, \omega) = 1$; (ii) *Adaptive Knowledge Soundness:* the trader, without ω, can produce a valid pair $(\mathsf{p}_r, \mathsf{k}_r)$ with only negligible probability; (iii) *Zero-Knowledge:* the public response p_r leaks nothing about ω (even to the investor who knows s_c); (iv) *Response Indistinguishability:* the public response p_r cannot be distinguished from a random string (except to the investor who knows s_c).

These four security properties collectively capture our security requirements. The Correctness property guarantees a successful key agreement in a normal case. The other three properties prevent misfeasance from a cheating prover (trader), verifier (investor); and a third party, respectively. The Adaptive Knowledge Soundness prevents the trader from cheating. The Zero-Knowledge property targets the designated verifier, i.e. the investor, to prevent her from learning the witness, whereas the Response Indistinguishability targets a third party (other than the prover/trader and the designated-verifier/investor) who can read the public response with access to the communication channel, e.g. the public ledger in our scenario.

Remark 5.1 (Main goal)**.** *This main goal of this chapter is to define the WKA scheme and offer an efficient construction of the WKA scheme. We aim to provide the basic functionalities of our dark pool scenario where the investor/trader must authenticate each other based on the committed information on a public ledger.*

By the wording 'efficient construction', we also mean that a performance simulation for such functionality in a realistic setting using actual trading data (in §5.7.2) is at least comparable on a day by day basis: if we were to run a day of trading messages, we would expect it to not take more than a day to actually exchange those messages.

5.2.1 Limitations of our WKA construction

Our WKA construction is based on the designated-verifier succinct zero-knowledge non-interactive argument of knowledge (zk-SNARK) compiled from a non-interactive linear proof (NILP) for Quadratic Arithmetic Programs (QAPs) by Groth [102]. Therefore, our WKA scheme inherits the limitations of those building blocks.

First, zk-SNARKs are not known to satisfy composability and therefore cannot be run out of the box in parallel within the design of larger protocols [120]. In a basic dark pool scenario, we only consider sequential composition where each execution of WKA concludes before the next execution begins [45]. For an extended scenario, one might use other instruments to identify parallel runs as described in Principle 10 for security protocol design by Abadi and Needham [1].

Second, our WKA scheme uses QAPs [91]; hence, it is only as efficient as the arithmetic circuit expressing the constraints.

Finally, we opted for simplicity rather than making the WKA scheme subversion-resistant as this requires the zero-knowledge property to be maintained even when the CRS is maliciously generated (see Bellare *et al.* [14]). Abdolmaleki *et al.* [2] and Fuchsbauer [85] constructed subversion-resistant NIZK based on Groth's zk-SNARK construction [102]. However, both works are only limited to the publicly verifiable zk-SNARK construction based on bilinear groups. Our WKA construction requires the designated-verifier zk-SNARK; therefore, those constructions are not applicable to our scheme. Hence, we consider only honest setups in our settings.

5.3 Recap of Technical Background

To make this chapter self-contained, we recap here the key technical terminology and definitions.

5.3.1 Summary of NIZK, SoK, AKE, WE and MPC

Non-interactive zero-knowledge (NIZK) proof system is a cryptographic primitive such that, given an instance ϕ and an NP-relation R, a party (the Prover) can convince another party (the Verifier) that there exists a witness ω of the instance ϕ such that

$R(\phi, \omega) = 1$, without leaking any information about ω [96, 98]. A zero-knowledge (ZK) proof system satisfies (1) *perfect completeness* (given any true statement, an honest prover should be able to convince an honest verifier); (2) *perfect soundness* (a malicious prover cannot convince a verifier of a false statement); and (3) *perfect zero-knowledge* (the proof leaks nothing more than the correctness of instance ϕ).

Non-interactivity allows the proof $\boldsymbol{\pi}$ to consist of only a single message. This property requires a one-time initial setup phase that yields a common reference string $\boldsymbol{\sigma}$ (CRS). This phase is run by a trusted third party or an MPC protocol [20]. A stronger notion of NIZK proof is *NIZK proof of knowledge* where the prover can, at the same time, convince the verifier that there exists a witness ω and the prover knows such ω (*Knowledge Soundness* [15]).

Perfect soundness can be relaxed to *computational* soundness using the notion of NIZK *argument* (of knowlegde). This relaxation allows the proof system to be *succinct*, i.e. polylogarithmic in communication complexity [41,138]. The notions of *succinct zero-knowledge non-interactive argument* (zk-SNARG) and *succinct zero-knowledge non-interactive argument of knowledge* (zk-SNARK) are those that follow such relaxation [93]. Further improvements of *online* procedures are tailored by *pre-processing* zk-SNARK to achieve that (1) the runtime of the verification algorithm is polylogarithmic to the instance size, and (2) the proof size is polynomial in λ where λ is the security parameter (for which the algorithms take the unary form 1^λ as input). Additionally, a zk-SNARK is fully-succinct if the CRS size is also polynomial in λ. Bitansky *et al.* showed that preprocessing zk-SNARKs can be compiled into fully-succinct zk-SNARKs [27].

Signature of Knowledge (SoK) [52] is a generalization of digital signature [73] aiming to fix the malleability of zk-SNARK by replacing the public (verifying) key pk with an instance ϕ in an NP-relation. SoK requires the signer to know the witness ω of ϕ to sign. Similar to NIZK, SoK hides the witness ω to prevent others from signing with the same witness. Recently, Groth *et al.* exploited the link between SoK and NIZK to construct the first succinct SoK [103].

Authenticated Key Exchange (AKE) allows two parties to share a secret key over an insecure network using various authentication means. For example *Password-Authenticated Key Exchange* (PAKE) [17] allows two parties to agree on strong keys (in different sessions) if they both know a weak shared password. *Credential-Authenticated Key Exchange* (CAKE) [42] allows two parties to generate a common secret key if a specific relation is satisfied between credentials held by the two players. CAKE indeed can also be used to instantiate PAKE. However the concrete instantiation of CAKE only supports limited

relations such as vectored unions of product relations, equality testing or product relations [42, Section 6, 7 and 8]. *Language-Authenticated Key Exchange* (LAKE) is closely related to CAKE. It allows two parties to share a secret key if they hold credentials that belong to a specific algebraic language [104]. Compared with PAKE, LAKE is more practical-oriented as it allows two members of the same group to secretly and privately authenticate each other without revealing this group beforehand. However, LAKE only supports languages defined by linear pairing product equations on committed values. Therefore it is not usable in our scenario.

Witness Encryption (WE) was introduced by Garg *et al.* [90] and refined by Bellare and Hoang [16]. In a WE scheme defined for a NP language L with witness relation $R(\phi, \omega)$, i.e. $L = \{\phi \mid \exists\, \omega \ : R(\phi, \omega) = 1\}$, the encryption algorithm takes as input a message M, an instance ϕ, and produces a ciphertext C. A party with a witness ω, such that $R(\phi, \omega) = 1$ can decrypt C (correctness); and if $\phi \notin L$ the message M is computationally hidden (soundness). Existing WE for arbitrary NP languages are currently considered impractical as they require multilinear maps [88, 92] or Indistinguishability Obfuscation [89]. The improvement proposed by Abusalah *et al.* [4] moved the computational hard part to an *offline* setup phase so that *online* encryption and decryption can be efficiently done but still relies on Indistinguishability Obfuscation. For some *particular* NP languages, WE is efficient[2]. For example, Derler *et al.* [71] proposed an offline WE construction under a Groth-Sahai (GS) proof for algebraic languages defined over bilinear groups, which can be employed for group encryption [116] and language-authenticated key exchange [104].

Smooth Projective Hash Function System (SPHF) was introduced by Cramer and Shoup [66]. Such system is defined on a language L. SPHF defines two keys: (1) the secret hashing key hk for the language L; and (2) the public projection key hp for an instance $x \in L$. The correctness property of SPHF requires that the hash value using hk and x the hash value is the same as using pk and w where w is the witness of x. The smoothness property requires that the hash value using hk and x is random without the knowledge of w. SPHF is such a strong primitive that it can be used to construct WE [71]. Therefore SPHF is also useful in constructing WKA. However, existing SPHF [71, 104] only focus on languages defined over bilinear groups hence not applicable in our scenario.

Secure Multiparty Computation (MPC) It is known that any functionality can be securely realized by a distributed protocol assuming honest minority (in the computational

[2] An example is public key encryption: ϕ is the public key pk and ω is the private key sk. Similarly for identity- and attribute-based encryption [161].

setting) [55] and honest majority (in the information-theoretic setting) [156]. Recent advances in the implementations of generic MPC protocols (see Archer *et al.* [9]) allow efficient MPC applications, e.g. privacy-preserving data mining [128] and exchanges [47]. See Orlandi [147] for an overview of MPC applications.

5.3.2 Formal Definitions for NILP and QAP

Notations. A multivariate polynomial $t : \mathbb{F}^m \to \mathbb{F}$ over a finite filed $\mathbb{F}$ has a degree d if the degree of each monomial in t is at most d and at least one monomial has degree d. A multivalued multivariate polynomial $\mathbf{t} : \mathbb{F}^m \to \mathbb{F}^\mu$ is a vector of polynomials $(t_1, \ldots, t_\mu)$, where each $t_i : \mathbb{F}^m \to \mathbb{F}$ is a multivariate polynomial.

We denote a scalar by x and a vector by $\mathbf{x}$. We write $x \leftarrow \mathbb{X}$ when picking an element x uniformly from a finite set $\mathbb{X}$. For a probabilistic algorithm, we write $y \leftarrow \mathsf{A}(x)$ when picking the randomness r and returning $y = \mathsf{A}(x; r)$. $\Pr[\epsilon|\Omega]$ denotes the probability of an event ϵ over the probability space Ω.

We denote the security parameter using 1^λ in the unary form for the polynomial runtime. We denote the negligible function $\mathsf{negl}(\cdot)$. Given two probability functions $f, g : \mathbb{N} \to [0, 1]$, we write $f(\lambda) \approx g(\lambda)$ when $|f(\lambda) - g(\lambda)| = O(\lambda^{-c})$ for every constant $c > 0$. We say that f is *negligible* when $f(\lambda) \approx 0$.

Remark 5.2 (Generation of the relation R)**.** *For convenience, we follow the notation of Groth [102], a relation generator $\mathcal{R}$ receives a security parameter 1^λ and returns a polynomial-time decidable binary relation R, i.e. $R \leftarrow \mathcal{R}(1^\lambda)$. For notational simplicity, we assume 1^λ be deduced from R.*

Linear interactive proof (LIP) was introduced by Bitansky *et al.* [27] as a natural extension of the definition of interactive proof by Goldwasser *et al.* [98]. The extension requires that each prover's message is an *affine combination* of the previous messages sent by the verifier.

LIP considers only *adversaries using affine prover strategies*, i.e. a strategy described by a tuple $(\mathbf{\Pi}, \boldsymbol{\pi}_0)$ where $\mathbf{\Pi} \in \mathbb{F}^{k \times y}$ represents a linear function, and $\boldsymbol{\pi}_0 \in \mathbb{F}^k$ represents an affine shift. Then, on input a query vector $\boldsymbol{\sigma} \in \mathbb{F}^y$, the response vector $\boldsymbol{\pi} \in \mathbb{F}^k$ is constructed by evaluating the affine relation $\boldsymbol{\pi} = \mathbf{\Pi}\boldsymbol{\sigma} + \boldsymbol{\pi}_0$. With the goal of non-interactive succinct verification, Bitansky *et al.,* focused on only input-oblivious[3] two-message LIPs for boolean circuit satisfiability problems [27].

Remark 5.3 (Enforcing Affine Prover Strategy)**.** *Intuitively speaking, a prover restricted to such affine strategy $(\mathbf{\Pi}, \boldsymbol{\pi}_0)$ can perform additions and multiplications by a constant but*

[3]The LIP verifier's messages do not depend on the instance ϕ.

not products for elements in $\boldsymbol{\sigma}$. For example, given $\boldsymbol{\sigma} = (x_1, x_2)$, such prover can compute $x_1 + x_2$ and $z_1 x_1$ (or $z_2 x_2$, where z_1, z_2 are known constants) but not $x_1 x_2$. Bitansky et al. *[27] showed that such restrictions on the prover could be obtained by executing the linear operations in the discrete logarithms or by encrypting the elements of $\boldsymbol{\sigma}$ with an additive homomorphic encryption scheme, such as the Paillier cryptosystem [148].*

Non-interactive linear proof (NILP), defined as a tuple of polynomial-time algorithms (Setup, Prove, Verify, Simulate) as in Fig. 5.1, is a useful characterization of SNARK constructions. Groth renamed the input-oblivious two-message LIPs for boolean circuit satisfiability by Bitansky *et al.* [27] into NILP [102] to clarify the the connection between LIP and NIZK proof. NILP satisfies perfect completeness and statistical knowledge soundness against *affine prover strategies*, i.e. there exists a probabilistic polynomial time (PPT) extractor ϵ such that for every PPT adversary $\hat{\mathcal{A}}$:

$$\Pr\left[\begin{array}{c} \boldsymbol{\Pi} \in \mathbb{F}^{k \times y} \\ R(\phi, \omega) \neq 1 \\ \mathsf{Verify}(R, \boldsymbol{\sigma}, \phi, \boldsymbol{\Pi}\boldsymbol{\sigma}) = 1 \end{array} \middle| \begin{array}{c} R \leftarrow \mathcal{R}(1^\lambda) \\ (\boldsymbol{\sigma}, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R) \\ (\phi, \boldsymbol{\Pi}) \leftarrow \hat{\mathcal{A}}(R) \\ \omega \leftarrow \epsilon(R, \phi, \boldsymbol{\Pi}) \end{array} \right] < \mathsf{negl}(\lambda) \tag{5.1}$$

where $\boldsymbol{\Pi}$ is a proof matrix that constitutes the proof $\boldsymbol{\pi}$.

zk-SNARK (zk-SNARG) from LIP (and NILP) The notion of zero-knowledge also applies to LIP and NILP. Bitansky *et al.* [27] shows that LIP (and NILP) can be compiled into both publicly verifiable (verifier degree 2, using bilinear maps) and designated-verifier (using linear-only encryption scheme) zk-SNARK (zk-SNARG).

Intuitively, the prover computes the proof $\boldsymbol{\pi}$ as linear combinations of the CRS $\boldsymbol{\sigma}$ and the verifier checks the argument by checking the quadratic equations corresponding to the relation R. We base our WKA scheme on the concrete efficient construction of zk-SNARK from NILP for quadratic arithmetic programs given by Groth [102]. We use linear encryption to compile such NILP to a WKA scheme. Later in this section, we summarize the definition of a Linear Encryption scheme given by Bitansky *et al.* [27].

Quadratic Arithmetic Programs (QAPs) were introduced by Gennaro *et al.* [91] and comprehensively summarized by Groth [102]. A QAP considers an arithmetic circuit C with only addition and multiplication gates over a finite field $\mathbb{F}$. Let us focus on binary relations R whose instance ϕ consists of all the public input and output wires of C whereas the witness consists of the rest (the private inputs and intermediate outputs) of C's wires. We recall the formal definition of QAP and a concrete construction of an NILP for QAP.

We assume that 1^λ is deduced from R (see Remark 5.2). Setup is run by a trusted third party or an MPC protocol [20], if the NILP is compiled into a publicly verifiable zk-SNARK. Otherwise Setup can be run by the designated-verifier. Prove is run by the Prover, and Verify is run by the Verifier (except for the publicly verifiable zk-SNARK case where Verify can be run by any party). Simulate can be run by any party with the simulation trapdoor $\boldsymbol{\tau}$ (e.g. the trusted third party or the designated-verifier).

NILP is defined as a tuple of polynomial-time algorithms (Setup, Prove, Verify, Simulate):

$(\boldsymbol{\sigma}, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R)$**:** output $\boldsymbol{\sigma} \in \mathbb{F}^y$ and $\boldsymbol{\tau} \in \mathbb{F}^x$.

$\boldsymbol{\pi} \leftarrow \mathsf{Prove}(R, \boldsymbol{\sigma}, \phi, \omega)$**:** obtain the proof matrix $\mathbf{\Pi} \leftarrow \mathsf{ProofMatrix}(R, \phi, \omega)$ (where ProofMatrix is a probabilistic polynomial time algorithm that generates $\mathbf{\Pi} \in \mathbb{F}^{k \times y}$) and output $\boldsymbol{\pi} = \mathbf{\Pi}\boldsymbol{\sigma}$.

$\{0, 1\} \leftarrow \mathsf{Verify}(R, \boldsymbol{\sigma}, \phi, \boldsymbol{\pi})$**:** obtain $\mathbf{t} \leftarrow \mathsf{Test}(R, \phi)$ where Test is a polynomial time algorithm that generates $\mathbf{t} : \mathbb{F}^{y+k} \rightarrow \mathbb{F}^\eta$ which is an arithmetic circuit corresponding to the evaluation of multivariate polynomials, such that $\mathbf{t}(\boldsymbol{\sigma}, \boldsymbol{\pi}) = 0$ if $\boldsymbol{\pi}$ is valid.

$\boldsymbol{\pi} \leftarrow \mathsf{Simulate}(R, \boldsymbol{\tau}, \phi)$**:** obtain $\mathbf{t} \leftarrow \mathsf{Test}(R, \phi)$ and solve $\mathbf{t}(\boldsymbol{\tau}, \boldsymbol{\pi}) = 0$ for the output $(\boldsymbol{\pi})$.

where y, x, k, η and d are constants or polynomials in the security parameter 1^λ.

Figure 5.1: Non-interactive linear proof [102]

Definition 5.1 (QAP)**.** *A quadratic arithmetic program $\mathbb{Q}$ over a field $\mathbb{F}$ for a relation $R(\phi, \omega)$ consists of three sequences of polynomials $\langle u_i(X), v_i(X), w_i(X)\rangle_{i=0}^m$ and a target polynomial $t(X) = \Pi_{q=1}^n (X - r_q)$ such that with $a_0 = 1$, $\phi = \{a_i\}_{i=1}^l$, and $\omega = \{a_i\}_{i=l+1}^m$, the following Eq. (5.2) holds.*

$$\sum_{i=0}^{m} a_i u_i(X) \sum_{i=0}^{m} a_i v_i(X) = \sum_{i=0}^{m} a_i w_i(X) + h(X)t(X) \tag{5.2}$$

where $u_i(X), v_i(X), w_i(X)$ are of at most degree $n-1$ and $h(X)$ is of degree $n-2$.

Remark 5.4 (QAP description)**.** *For convenience, we follow the QAP description of Groth [102], we consider the QAP for R, i.e.*

$$(\mathbb{F}, aux, l, \langle u_i(X), v_i(X), w_i(X)\rangle_{i=0}^m, t(X))$$

where $\mathbb{F}$ is a finite field; aux is some auxiliary information; $1 \leq l \leq m$; $u_i(X), v_i(X), w_i(X), t(X) \in \mathbb{F}[X]$, $u_i(X), v_i(X), w_i(X)$ are of at most degree $n-1$. Such QAP defines a binary relation

$$R = \left\{ (\phi, \omega) \middle| \begin{array}{c} a_0 = 1, \phi = \{a_i\}_{i=1}^l, \omega = \{a_i\}_{i=l+1}^m \\ \sum_{i=0}^m a_i u_i(X) \sum_{i=0}^m a_i v_i(X) = \sum_{i=0}^m a_i w_i(X) + h(X)t(X) \end{array} \right\}$$

Example 5.3.1. *A QAP can be efficiently built from an arithmetic circuit [19]. Let us take as an example a simple arithmetic circuit C that takes as input a variable _in and outputs _out = _in $* 3 + 5$.*

Such circuit consists of 3 variables: the input _in, the intermediate variable _inter, and the output variable _out; and 2 constraints: _inter = _in$*3$ and _out = _inter$+5$.

For such circuit, the obtained QAP will have $m = 3$ (the number of variables), $l = 1$ (the number of output variables) and $n = 2$ (the number of constraints).

To generate a QAP from a circuit, we need to convert it into a rank-1 constraint system (R1CS) [19]. For each constraint j, we need a group of three vectors $(\boldsymbol{z}_{a,j}, \boldsymbol{z}_{b,j}, \boldsymbol{z}_{c,j})$, whereas the common solution of all the vector groups is a vector $\boldsymbol{z}_s$ such that $(\boldsymbol{z}_s \cdot \boldsymbol{z}_{a,j})(\boldsymbol{z}_s \cdot \boldsymbol{z}_{b,j}) = (\boldsymbol{z}_s \cdot \boldsymbol{z}_{c,j})$ for all j.

For our example circuit C each vector $(\boldsymbol{z}_{a,j}, \boldsymbol{z}_{b,j}, \boldsymbol{z}_{c,j})$ and $\boldsymbol{z}_s$ must be of length 4 to capture the required constraints for 4 variables: an additional dummy variable _one at the first index representing the number 1 followed by all the circuit's input, output, and intermediate variables (_in, _out, and _inter).

Let us fix $\boldsymbol{z}_s = (1, 3, 14, 9)$ (_in $= 3$, _out $= 14$ and _inter $= 9$), we can now fill the corresponding elements for the vectors of each constraint j that satisfies

$$(\boldsymbol{z}_s \cdot \boldsymbol{z}_{a,j})(\boldsymbol{z}_s \cdot \boldsymbol{z}_{b,j}) = (\boldsymbol{z}_s \cdot \boldsymbol{z}_{c,j})$$

1. *for the first constraint _inter = _in $* 3$, we can have $\boldsymbol{z}_{a,1} = (0, 1, 0, 0)$, $\boldsymbol{z}_{b,1} = (3, 0, 0, 0)$, $\boldsymbol{z}_{c,1} = (0, 0, 0, 1)$;*
2. *for the second constraint _out = _inter $+ 5$, we can have $\boldsymbol{z}_{a,2} = (5, 0, 0, 1)$, $\boldsymbol{z}_{b,2} = (1, 0, 0, 0)$, $\boldsymbol{z}_{c,2} = (0, 0, 1, 0)$.*

Now; we convert the R1CS into its QAP form to enforce the exact same constraints using polynomials (as in Eq. (5.2)) instead of some vector products. For C we must transform the 2 groups of 3 vectors length-4 ($(\boldsymbol{z}_{a,1}, \boldsymbol{z}_{b,1}, \boldsymbol{z}_{c,1})$ and $(\boldsymbol{z}_{a,2}, \boldsymbol{z}_{b,2}, \boldsymbol{z}_{c,2})$) into 4 groups of 3 polynomials degree-1 ($u_i(X), v_i(X), w_i(X)$ for $i \in [4]$, here $n = 2$ for Eq. (5.2)) whose evaluations at each $X = x$ represents one of the constraints.

To perform this task, we take the first (second, and so on) elements of each vector $(\boldsymbol{z}_{a,j}, \boldsymbol{z}_{b,j}, \boldsymbol{z}_{c,j})$ and use Lagrangian interpolation for generating the polynomials coefficients at $X = 1$ ($X = 2$ and so on):

- $u_0(X) = (5X - 5)$, $u_1(X) = (-X + 2)$, $u_2(X) = 0$ *and* $u_3(X) = (X - 1)$
- $v_0(X) = (-2X + 5)$, $v_1(X) = 0$, $v_2(X) = 0$ *and* $v_3(X) = 0$

- $w_0(X) = 0$, $w_1(X) = 0$, $w_2(X) = (X-1)$ *and* $w_3(X) = (-X+2)$

For $t(X) = \Pi_{q=1}^{n}(X - r_q)$, *we can simply fix* $t(X) = \Pi_{q=1}^{2}(X - q) = (X-1)(X-2)$. *To obtain* $h(X)$, *we use Eq. 5.2, and we must divide*

$$\sum_{i=0}^{m} a_i u_i(X) \sum_{i=0}^{m} a_i v_i(X) - \sum_{i=0}^{m} a_i w_i(X) = -22X^2 + 66X - 44$$

(as $a_1 = 3$, $a_2 = 14$ *and* $a_3 = 9$*) by*

$$t(X) = (X-1)(X-2) = X^2 - 3X + 2$$

Hence, we have $h(X) = -22$.

The obtained QAP equation has the following form:

$$((5X-5) + a_1(-X+2) + a_3(X-1))((-2X+5)) = a_2(X-1) + a_3(-X+2) + (-22)(X-1)(X-2)$$

The NILP construction for QAP (by Groth [102], as shown in Fig. 5.2) yields a tuple of polynomial-time algorithms (Setup, Prove, Verify, Simulate). The obtained NILP has perfect completeness, perfect zero-knowledge, and statistical soundness against affine prover strategies. As shown by Groth [102], the NILP construction in Fig. 5.2 (of degree 2) is compiled into publicly verifiable zk-SNARK by executing the linear operations (Eq. (5.4), (5.5) and (5.6)) in the discrete logarithms, using pairings for the product operations in the verification equation (Eq. (5.7)). To compile an NILP for QAP into designated-verifier SNARK, one requires a linear-only encryption scheme [27].

Remark 5.5 (Soundness of zk-SNARK from NILP for QAP)**.** *Let us take as an example the NILP for QAP in Fig. 5.2. zk-SNARK from NILP for QAP is only sound if the prover uses only affine strategies, i.e. the prover cannot compute the non-affine term* $\frac{\alpha\beta}{\delta}$ *hence s/he cannot run the proof forging equation Eq. 5.8 in* Simulate. *We can enforce such restriction using bilinear maps and linear only encryption.*

In the public verifier case, as the operations are executed in the discrete logarithms the prover can only compute Eq. (5.4), (5.5) *and* (5.6). *Using bilinear maps, the public verifier can compute Eq.* (5.7). *However, it is necessary that* Setup *that yields the CRS* $\boldsymbol{\sigma}$ *is run by a trusted party or an MPC protocol [20] to prevent the* Simulate *from being able to run by the prover.*

The designated verifier case is more relaxed. The Setup *step that yields* $(\boldsymbol{\sigma}, \boldsymbol{\tau})$ *is run by the verifier. However, it is more restricted for the prover. Due to the Linear Only*

We consider the QAP that defines a binary relation R as described in Remark 5.4. NILP for such QAP is defined as a tuple of polynomial-time algorithms (Setup, Prove, Verify, Simulate):

$(\boldsymbol{\sigma}, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R)$**:** Pick $\alpha, \beta, \gamma, \delta, x \leftarrow \mathbb{F}^*$. Set the secret $\boldsymbol{\tau} = (\alpha, \beta, \gamma, \delta, x)$ and the public $\boldsymbol{\sigma}$:

$$\begin{aligned} \boldsymbol{\sigma} = \alpha, \beta, \gamma, \delta, \{x^i\}_{i=0}^{n-1}, \left\{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\right\}_{i=0}^{l}, \\ \left\{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}\right\}_{i=l+1}^{m}, \left\{\frac{x^i t(x)}{\delta}\right\}_{i=0}^{n-2} \end{aligned} \tag{5.3}$$

$\boldsymbol{\pi} \leftarrow \mathsf{Prove}(R, \boldsymbol{\sigma}, a_1, \ldots, a_m)$**:** Pick $r, s \leftarrow \mathbb{F}$ and compute $\boldsymbol{\pi} = (A, B, C)$ where:

$$A = \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta \tag{5.4}$$

$$B = \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta \tag{5.5}$$

$$C = \sum_{i=l+1}^{m} a_i \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} + \frac{h(x)t(x)}{\delta} + sA + rB - rs\delta \tag{5.6}$$

$\{0, 1\} \leftarrow \mathsf{Verify}(R, \boldsymbol{\sigma}, a_1, \ldots, a_l, \pi)$**:** Output 1 iff:

$$AB = \alpha\beta + \sum_{i=0}^{l} a_i \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\gamma + C\delta \tag{5.7}$$

$\boldsymbol{\pi} \leftarrow \mathsf{Simulate}(\boldsymbol{\tau}|R, a_1, \ldots, a_l)$**:** Pick $A, B \leftarrow \mathbb{F}$, and output $\boldsymbol{\pi} = (A, B, C)$ where:

$$C = \frac{AB}{\delta} - \frac{\alpha\beta}{\delta} - \frac{\sum_{i=0}^{l} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\delta} \tag{5.8}$$

Figure 5.2: NILP for QAP [102])

Encryption scheme (e.g. a two-ciphertext variants of the Paillier cryptosystem [148]), the prover can compute Eq. (5.4), (5.5) *and* (5.6) *(using the public key in* $\boldsymbol{\sigma}$*). However, the only party that can execute the verification equation Eq.* (5.7) *is the verifier with the corresponding decryption key* sk *(in* $\boldsymbol{\tau}$*). In such a case, the* Simulate *algorithm can be run by the verifier.*

Linear-only encryption (LE) scheme Σ (Bitansky *et al.* [27]), e.g. a two-ciphertexts variant of Paillier [148], is a tuple of polynomial-time algorithms (KeyGen, Enc, ImgVer, Dec, Add) where the ImgVer (image verification) prevents oblivious ciphertext samplings in the image of Enc using pk. This property prevents the adversary from encrypting plaintexts from scratch (see Remark 5.9 in §5.6 for further details). Add is for evaluating linear combinations of valid ciphertexts. An LE scheme satisfies *correctness*, *additive homomorphism*, *indistinguishability under chosen plaintext attack* (IND-CPA), and in addition, *linear-only homomorphism* which essentially states that it is infeasible to generate a new valid ciphertext except by evaluating an affine combination of valid ciphertexts (via Add). This property formally guarantees that given a valid ciphertext $\boldsymbol{\pi}$ by an adversary it is possible to *efficiently* extract the corresponding affine function $(\boldsymbol{\Pi}, \boldsymbol{\pi}_0)$ (see Remark 5.3) that explains $\boldsymbol{\pi}$. Such LE scheme can be instantiated using existing encryption schemes. See Bitansky *et al.* [27, §5.3] for further details.

Remark 5.6 (Security assumptions of LE)**.** *The security of an LE scheme relies on the assumptions of q-power Diffie-Hellman, q-power Knowledge of Exponent, and q-power Knowledge of Equality [27].*

5.4 Witness Key Agreement

First, we give a formal definition of the witness key agreement scheme Ω. Then we further define our variant of split designated-verifier NILP which is useful for the WKA scheme construction. Next, we show how to construct our WKA scheme Ω using NILP and non-deterministic LE. Finally, we provide a proof sketch for the security of our WKA scheme.

5.4.1 Witness Key Agreement Definition

Formally, we define witness key agreement as follows:

Definition 5.2 (Witness key agreement)**.** *Let L be an NP-language with the witness relation $R(\phi, \omega)$. We call ϕ an instance of L and ω a witness for ϕ. A Witness Key Agreement scheme Ω for L is a tuple of polynomial-time algorithms (*KChallenge, KResponse, KDerive*) as defined in Fig. 5.3.*

A witness key agreement scheme Ω satisfies *Correctness*, *Adaptive Knowledge Soundness*, *Zero-Knowledge* and *Response Indistinguishability*, as shown in Fig. 5.4.

For notational simplicity we assume that λ is deduced from the description of R (See Remark 5.2).

$(\mathsf{p}_c, \mathsf{s}_c) \leftarrow \mathsf{KChallenge}(R)$ is run by the investor and takes as input the relation R (from which the corresponding security parameter 1^λ is deduced). It outputs a pair of public and secret challenge parameter $(\mathsf{p}_c, \mathsf{s}_c)$.

$(\mathsf{p}_r, \mathsf{k}_r) \leftarrow \mathsf{KResponse}(R, \mathsf{p}_c, \phi, \omega)$ is run by the trader and takes as input the relation R, the public challenge parameter p_c, the instance ϕ, and the corresponding witness ω. It outputs a public response parameter p_r and a secret key k_r.

$\{\mathsf{k}_c, \perp\} \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r)$ is run by the investor and takes as input the relation R, the secret challenge parameter s_c, the instance ϕ and the public response parameter p_r. It outputs a key k_c or $\perp$.

Figure 5.3: Witness key agreement definition

Main intuition for WKA construction. Our observation is that the proof $\boldsymbol{\pi}$ obtained with NILP for QAP consists of k elements (by evaluating k linear functions[4] corresponding to the proof matrix $\boldsymbol{\Pi}$), in which the k-th element can be obtained in the two following ways given the first $k-1$ elements [102].

1. On the prover's side, if $\boldsymbol{\pi}$ is valid then the first $k-1$ elements fully determine the last one. For instance, in Fig. 5.2, A and B (Eq. (5.4) and (5.5)) uniquely define C (Eq. (5.6)).
2. On the verifier's side, the first $k-1$ elements (A and B) can be fed into a proof forging formula (Eq. (5.8)) to get the same k-th element (C).

Hence, after the CRS generation, by the prover computing $\boldsymbol{\pi}$ and publishing the first $k-1$ elements of $\boldsymbol{\pi}$, both parties can agree on the last secret element and use it as a shared secret key for secure communication.

5.4.2 Split Designated-Verifier Non-Interactive Linear Proof

We describe our notion of *split designated verifier NILP* based on Groth's definition [102].

Groth also defined *split NILP* for the *publicly verifiable* case when working with Type III pairings [102]. In the asymmetric pairings settings, the CRS $\boldsymbol{\sigma}$ and the proof $\boldsymbol{\pi}$ are split into two corresponding to two groups of a bilinear map ($\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$, $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$) where each part is computed from the respective part of the CRS using linear combinations of two proof matrices $\boldsymbol{\Pi}_{1,2}$ ($\boldsymbol{\Pi}_1 \in \mathbb{F}^{k_1 \times y_1}$ and $\boldsymbol{\Pi}_2 \in \mathbb{F}^{k_2 \times y_2}$ where $k_1 + k_2 = k$ and $y_1 + y_2 = y$).

[4] $k = 3$ and the proof matrix $\boldsymbol{\Pi}$ is represented as the coefficients used in the Eq. (5.4), (5.5) and Eq. (5.6) in the concrete construction in Fig. 5.2 by Groth [102].

A WKA scheme Ω satisfies *Correctness*, *Adaptive Knowledge Soundness*, *Zero-Knowledge* and *Response Indistinguishability*:

Correctness Given a true instance, the key agreement is successful, i.e.

$$\Pr\left[\mathsf{k}_c = \mathsf{k}_r \;\middle|\; \begin{array}{c} R \leftarrow \mathcal{R}(1^\lambda) \\ R(\phi,\omega) = 1 \end{array} , \begin{array}{cc} (\mathsf{p}_c, \mathsf{s}_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\mathsf{p}_r, \mathsf{k}_r) & \leftarrow \mathsf{KResponse}(R, \mathsf{p}_c, \phi, \omega) \\ \mathsf{k}_c & \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r) \end{array}\right] = 1 \tag{5.9}$$

Adaptive Knowledge Soundness The key agreement is successful only with negligible probability if the responder knows no witness for the instance, i.e.for any PPT $\hat{\mathcal{A}}$, there exists a poly-time extractor $\epsilon_{\hat{\mathcal{A}}}$ such that

$$\Pr\left[R(\phi,\omega) \neq 1 \;\middle|\; \begin{array}{cc} R & \leftarrow \mathcal{R}(1^\lambda) \\ (\mathsf{s}_c, \mathsf{p}_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\phi, \mathsf{p}_r, \mathsf{k}_r) & \leftarrow \hat{\mathcal{A}}(R, \mathsf{p}_c) \\ \mathsf{k}_c & \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r) \\ & \mathsf{k}_c = \mathsf{k}_r \\ \omega & \leftarrow \epsilon_{\hat{\mathcal{A}}}(R, \phi, \mathsf{p}_r, \mathsf{k}_c) \end{array}\right] < \mathsf{negl}(\lambda) \tag{5.10}$$

Honest Verifier Zero-knowledge The response leaks nothing about the witness in the honest setup, i.e.there is a simulator $\mathcal{S}_{ZK}$ that outputs a simulated response $(\mathsf{p}_r, \mathsf{k}_r)$ and key k_c. Formally, for all $\lambda \in \mathbb{N}$, $R \leftarrow \mathcal{R}(1^\lambda)$, $R(\phi,\omega) = 1$ and any PPT $\hat{\mathcal{A}}$:

$$\begin{aligned} &\Pr\left[\hat{\mathcal{A}}(R, \mathsf{p}_c, \mathsf{s}_c, \phi, \mathsf{p}_r, \mathsf{k}_c) = 1 \;\middle|\; \begin{array}{cc} (\mathsf{p}_c, \mathsf{s}_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\mathsf{p}_r, \mathsf{k}_r) & \leftarrow \mathsf{KResponse}(R, \mathsf{p}_c, \phi, \omega) \\ \mathsf{k}_c & \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r) \end{array}\right] \\ &= \Pr\left[\hat{\mathcal{A}}(\perp | R, \mathsf{p}_c, \mathsf{s}_c, \phi, \mathsf{p}_r, \mathsf{k}_c) = 1 \;\middle|\; \begin{array}{cc} (\mathsf{p}_c, \mathsf{s}_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\mathsf{p}_r, \mathsf{k}_r, \mathsf{k}_c) & \leftarrow \mathcal{S}_{ZK}(R, \mathsf{p}_c, \mathsf{s}_c, \phi) \end{array}\right] \end{aligned} \tag{5.11}$$

Response Indistinguishability The public response is indistinguishable from a random string, i.e. for all $\lambda \in \mathbb{N}$, $R \leftarrow \mathcal{R}(1^\lambda)$, $R(\phi,\omega) = 1$ there exists a simulator $\mathcal{S}_{RI}$ such that for any PPT $\hat{\mathcal{A}}$:

$$\begin{aligned} &\Pr\left[\hat{\mathcal{A}}(R, \mathsf{p}_c, \phi, \mathsf{p}_r) = 1 \;\middle|\; \begin{array}{cc} (\mathsf{p}_c, \mathsf{s}_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\mathsf{p}_r, \mathsf{k}_r) & \leftarrow \mathsf{KResponse}(R, \mathsf{p}_c, \phi, \omega) \end{array}\right] \\ &= \Pr\left[\hat{\mathcal{A}}(R, \mathsf{p}_c, \phi, \mathsf{p}_r) = 1 \;\middle|\; \begin{array}{cc} (\mathsf{p}_c, \mathsf{s}_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\mathsf{p}_r) & \leftarrow \mathcal{S}_{RI}(R, \mathsf{p}_c, \phi) \end{array}\right] \end{aligned} \tag{5.12}$$

Figure 5.4: Security of Witness Key Agreement Scheme

For the designated verifier's case, we write $\boldsymbol{\sigma}$ as $\boldsymbol{\sigma}_P$ and $\boldsymbol{\tau}$ as $\boldsymbol{\sigma}_V$ to emphasize that $\boldsymbol{\sigma}_P$ is for the prover and $\boldsymbol{\sigma}_V$ is for the verifier. In proof computation, we further split the proof matrix $\mathbf{\Pi} \in \mathbb{F}^{k \times y}$ (see Fig. 5.1) into two: $\mathbf{\Pi}_1 \in \mathbb{F}^{k-1 \times y}$ and $\mathbf{\Pi}_2 \in \mathbb{F}^{1 \times y}$. The proof

Following Groth [102], we assume that λ is deduced from R.

$(\boldsymbol{\sigma}_P, \boldsymbol{\sigma}_V) \leftarrow \mathsf{Setup}(R)$**:** output $\boldsymbol{\sigma}_P \in \mathbb{F}^y$ and $\boldsymbol{\sigma}_V \in \mathbb{F}^x$.

$(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) \leftarrow \mathsf{Prove}(R, \boldsymbol{\sigma}_P, \phi, \omega)$**:** obtain the proof matrices

$$(\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \leftarrow \mathsf{ProofMatrix}(\phi, \omega | R)$$

where $\boldsymbol{\Pi}_1 \in \mathbb{F}^{k-1 \times y}$ and $\boldsymbol{\Pi}_2 \in \mathbb{F}^{1 \times y}$ and output

$$\boldsymbol{\pi}_1 = \boldsymbol{\Pi}_1 \boldsymbol{\sigma}_P \text{ and } \boldsymbol{\pi}_2 = \boldsymbol{\Pi}_2 \boldsymbol{\sigma}_P$$

$\{0,1\} \leftarrow \mathsf{Verify}(R, \boldsymbol{\sigma}_V, \phi, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$**:** obtain $\mathbf{t} \leftarrow \mathsf{Test}(R, \phi)$ where $\mathbf{t} : \mathbb{F}^{y+k} \rightarrow \mathbb{F}^{\eta}$ is an arithmetic circuit corresponding to the evaluation of multivariate polynomials such that $\mathbf{t}(\boldsymbol{\sigma}_V, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2) = 0$ if $\boldsymbol{\pi}$ is valid.

$(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) \leftarrow \mathsf{Simulate}(R, \boldsymbol{\sigma}_V, \phi)$**:** obtain $\mathbf{t} \leftarrow \mathsf{Test}(R, \phi)$ and solve $\mathbf{t}(\boldsymbol{\sigma}_V, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2) = 0$ for the output $(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$.

where y, x, k, η and d are constant or polynomials in λ.

Figure 5.5: Split designated-verifier NILP

$\boldsymbol{\pi}$ is also split into $\boldsymbol{\pi}_1 = \boldsymbol{\Pi}_1 \boldsymbol{\sigma}_P$ consists of $k-1$ elements, and $\boldsymbol{\pi}_2 = \boldsymbol{\Pi}_2 \boldsymbol{\sigma}_P$ consists of the last element. This split of $\boldsymbol{\Pi}$ and $\boldsymbol{\pi}$ is unnecessary in a designated-verifier zk-SNARK proof system. However, it is essential in our WKA scheme as we need to split the proof into two (see our main intuition in §5.4.1).

Formally a split DV NILP is of the following form.

Definition 5.3 (Split designated-verifier NILP)**.** *Let L be an NP-language with the witness relation $R(\phi, \omega)$. We call ϕ an instance of L and ω a witness for ϕ. A split designated-verifier (split DV) NILP for L consists of the tuple of polynomial-time algorithms (*Setup*,* Prove*,* Verify*,* Simulate*) as in Fig. 5.5.*

*A tuple of polynomial-time algorithms (*Setup*,* Prove*,* Verify*,* Simulate*) is a split DV NILP if it has perfect completeness, perfect zero-knowledge, and statistical soundness against affine prover strategies.*

Remark 5.7 (Split DV NILP for QAP)**.** *A split DV NILP for QAP is directly reformulated from the NILP in Fig. 5.2 as shown in Fig. 5.6. We simply split the proof matrices into two $\boldsymbol{\Pi}_1$ and $\boldsymbol{\Pi}_2$ where $\boldsymbol{\Pi}_1 \in \mathbb{F}^{2 \times y}$ corresponds to the matrix used in Eq.* (5.4) *and* (5.5)*, whereas $\boldsymbol{\Pi}_2 \in \mathbb{F}^{1 \times y}$ corresponds to the matrix used in Eq.* (5.6)*. Since the NILP in Fig. 5.2 is secure, our obtained split DV NILP is also secure (see Remark 5.5 for the*

A Split DV NILP for QAP is reformulated from the NILP for QAP in Fig. 5.2 as follows.

$(\boldsymbol{\sigma}, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R)$**:** Pick $\alpha, \beta, \gamma, \delta, x \leftarrow \mathbb{F}^*$. Set the secret $\boldsymbol{\tau} = (\alpha, \beta, \gamma, \delta, x)$ and the public $\boldsymbol{\sigma}$:

$$\begin{aligned}\boldsymbol{\sigma} = \alpha, \beta, \gamma, \delta, \{x^i\}_{i=0}^{n-1}, \left\{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\right\}_{i=0}^{l}, \\ \left\{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}\right\}_{i=l+1}^{m}, \left\{\frac{x^i t(x)}{\delta}\right\}_{i=0}^{n-2}\end{aligned} \tag{5.13}$$

$(\boldsymbol{\pi}_1 \boldsymbol{\pi}_2) \leftarrow \mathsf{Prove}(R, \boldsymbol{\sigma}, a_1, \ldots, a_m)$**:** Pick $r, s \leftarrow \mathbb{F}$ and compute $\boldsymbol{\pi}_1 = (A, B)$ and $\boldsymbol{\pi}_2 = (C)$ where:

$$A = \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta \tag{5.14}$$

$$B = \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta \tag{5.15}$$

$$C = \sum_{i=l+1}^{m} a_i \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} + \frac{h(x)t(x)}{\delta} + sA + rB - rs\delta \tag{5.16}$$

$\{0, 1\} \leftarrow \mathsf{Verify}(R, \boldsymbol{\sigma}, a_1, \ldots, a_l, \boldsymbol{\pi}_1 = (A, B), \boldsymbol{\pi}_2 = (C))$**:** Output 1 iff:

$$AB = \alpha\beta + \sum_{i=0}^{l} a_i \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\gamma + C\delta \tag{5.17}$$

$(\boldsymbol{\pi}_1 \boldsymbol{\pi}_2) \leftarrow \mathsf{Simulate}(\boldsymbol{\tau} | R, a_1, \ldots, a_l)$**:** Pick $A, B \leftarrow \mathbb{F}$, and output $\boldsymbol{\pi}_1 = (A, B)$ and $\boldsymbol{\pi}_2 = (C)$ where:

$$C = \frac{AB}{\delta} - \frac{\alpha\beta}{\delta} - \frac{\sum_{i=0}^{l} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\delta} \tag{5.18}$$

Figure 5.6: Split DV NILP for QAP

intuition on the soundness of such NILP and Groth's security proof [102, Theorem 1] for further details).

5.4.3 Construction of WKA

We construct a WKA scheme Ω from a split DV NILP as shown in Fig. 5.7. Below is the construction at a high level.

We modify the LE scheme's encryption algorithm interface for denoting used random-

Following Groth [102], we assume 1^λ is deduced from R.

$\underline{(\mathsf{p}_c, \mathsf{s}_c) \leftarrow \mathsf{KChallenge}(R)}$ runs as follows:

1. Fix a linear-only encryption scheme Σ;
2. Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$ where 1^λ is the security parameter deduced from R (see Remark 5.2); and $(\boldsymbol{\sigma}_P, \boldsymbol{\sigma}_V) \leftarrow \mathsf{Setup}(R)$;
3. Encrypt $[\sigma_{P,i}, r_{P,i}] \leftarrow \Sigma.\mathsf{Enc}(\mathsf{pk}, \sigma_{P,i})$ for each $\sigma_{P,i} \in \boldsymbol{\sigma}_P$;
4. Encrypt $[r_{P,i}] \leftarrow \Sigma.\mathsf{Enc}(\mathsf{pk}, r_{P,i})$ for each $r_{P,i}$ above;
5. Return $\mathsf{p}_c = (\mathsf{pk}, \{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y}, \{[r_{P,i}]\}_{i=1}^{y})$ and $\mathsf{s}_c = (\mathsf{sk}, \boldsymbol{\sigma}_V)$.

$\underline{(\mathsf{p}_r, \mathsf{k}_r) \leftarrow \mathsf{KResponse}(R, \mathsf{p}_c, \phi, \omega)}$**:** Upon receiving the challenge p_c,

1. Run $(\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \leftarrow \mathsf{ProofMatrix}(\phi, \omega | R)$;
2. Compute $\{[\pi_{1,j}, r_{1,j}]\}_{j=1}^{k-1} = \boldsymbol{\Pi}_1(\{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y})$ (with $\Sigma.\mathsf{Add}$);
3. Compute $[\pi_2, r_2] = \boldsymbol{\Pi}_2(\{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y})$ (with $\Sigma.\mathsf{Add}$);
4. Compute $[r_2] = \boldsymbol{\Pi}_2(\{[r_{P,i}]\}_{i=1}^{y})$ (with $\Sigma.\mathsf{Add}$);
5. Return $\mathsf{p}_r = (\{[\pi_{1,j}, r_{1,j}]\}_{j=1}^{k}, [r_2])$ and $\mathsf{k}_r = [\pi_2, r_2]$.

$\underline{\{\mathsf{k}_c, \perp\} \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r)}$ Output $\perp$ if any verification fails:

1. Verify $\mathsf{ImgVer}(\mathsf{sk}, [\pi_{1,j}, r_{1,j}]) = 1$ for $1 \leq j \leq k-1$; and $\mathsf{ImgVer}(\mathsf{sk}, [r_2]) = 1$;
2. Decrypt $\pi_{1,j} = \Sigma.\mathsf{Dec}(\mathsf{sk}, [\pi_{1,j}, r_{1,j}])$ for $1 \leq j \leq k-1$;
3. Obtain $\mathbf{t} \leftarrow \mathsf{Test}(R, \phi)$; solve $\mathbf{t}(\boldsymbol{\sigma}_V, \{\pi_{1,j}\}_{j=1}^{k}, \pi_2) = 0$ using $\{\pi_{1,j}\}_{j=1}^{k-1}$ to get π_2;
4. Decrypt $r_2 = \Sigma.\mathsf{Dec}(\mathsf{sk}, [r_2])$
5. Return $\mathsf{k}_r = \Sigma.\mathsf{Enc}(\mathsf{pk}, \pi_2, r_2)$ (r_2 as randomness).

Figure 5.7: Construction of Witness Key Agreement

ness explicitly. We omit the randomness r and write only $[m] \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ in case r is unnecessary in subsequent computation. We write $[m] = \mathsf{Enc}(\mathsf{pk}, m, r)$ in case we input directly the randomness into the encryption algorithm. We also require that the additive homomorphism of LE applies to both the message and the used randomness i.e. $\mathsf{Add}(\mathsf{pk}, \langle [m_i, r_i] \rangle, \langle \alpha_i \rangle)$ evaluates $[\sum \alpha_i m_i, \sum \alpha_i r_i]$.

The challenge phase In KChallenge, the investor first generates a CRS ($\boldsymbol{\sigma}_P$, $\boldsymbol{\sigma}_V$) from R (using a split DV NILP). Then, the investor encrypts each element $\{\sigma_{P,i}\}_{i=1}^{y}$ of the $\boldsymbol{\sigma}_P$ with an LE scheme (with key pair pk, sk). Additionally, we require the investor to encrypt the randomnesses $\{r_{P,i}\}_{i=1}^{y}$ used for the encryption of the CRS $\{\sigma_{P,i}\}_{i=1}^{y}$ in KChallenge into $\{[r_{P,i}]\}_{i=1}^{y}$. Finally, the investor publishes a challenge that consists of pk and the encrypted elements $(\{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y}, \{[r_{P,i}]\}_{i=1}^{y})$. The investor keeps

private sk and the plain CRS $\boldsymbol{\sigma}_V$.

The response phase Upon seeing the challenge, in $\mathsf{KResponse}$, the trader computes a response by generating a valid proof $\boldsymbol{\pi}$ for the desired tuple (ϕ, ω) (using the proof matrix of the split DV NILP and the additive homomorphic operation Add of the LE scheme). When the trader evaluates the last encrypted element $[\pi_2, r_2]$ using the proof matrix $\boldsymbol{\Pi}_2$ and the encrypted CRS $\{[\sigma_{P,i}]\}_{i=1}^{y}$, by the additively homomorphic property of the LE scheme, the trader can also evaluate the ciphertext $[r_2]$ of the randomness r_2 of the encrypted $[\pi_2, r_2]$ using the same $\boldsymbol{\Pi}_2$ and $\{[r_{P,i}]\}$. The trader publishes the first encrypted $k-1$ elements $\{[\pi_{1,j}, r_{1,j}]\}_{j=1}^{k-1}$ and the encrypted randomness $[r_2]$ as a public response and keeps secret the last encrypted element $[\pi_2, r_2]$.

Key derive phase When the investor sees the instance ϕ and the corresponding response, in $\mathsf{KDerive}$, he can decrypt the encrypted elements using sk to get $\{\pi_{1,j}\}_{j=1}^{k-1}$ and forge the last element π_2 using the plain CRS $\boldsymbol{\sigma}_V$. The investor then uses the evaluated $[r_2]$ to reconstruct the correct ciphertext $[\pi_2, r_2]$ of the last element, i.e. the investor decrypts $[r_2]$ to get r_2 to use as the randomness in the final encryption of π_2 to get $[\pi_2]$. After that, both parties agree on the same ciphertext $[\pi_2, r_2]$ of the last element.

5.4.4 Security Analysis

Theorem 5.1 (Security of the WKA scheme Ω)**.** *If the LE scheme Σ satisfies correctness, additive homomorphism, IND-CPA, and linear-only homomorphism, and the underlying split DV NILP satisfies perfect completeness, perfect zero-knowledge, and statistical knowledge soundness against affine prover strategies, then, the WKA scheme Ω satisfies correctness, adaptive knowledge soundness, honest verifier zero-knowledge, and response indistinguishability.*

As *correctness* follows the algorithm's description, we focus on *adaptive knowledge soundness*, *honest verifier zero-knowledge*, and *response indistinguishability* of Ω. For simplicity, we only sketch the proofs as follows:

Adaptive Knowledge Soundness. In the soundness game the adversary $\hat{\mathcal{A}}$ comes up with some proof $(\{[\pi_{1,j}]\}_{j=1}^{k-1}, [\pi_2]) = (\mathsf{p}_r, \mathsf{k}_r)$ for the instance ϕ.

1. Given $\mathsf{k}_c \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r)$, the assumption $\mathsf{k}_c = \mathsf{k}_r$ implies that the sampled proof $(\{[\pi_{1,j}]\}_{j=1}^{k-1}, [\pi_2])$ passes the image verifications. If $(\{[\pi_{1,j}]\}_{j=1}^{k-1}, [\pi_2])$ are not affine combinations of $\{[\sigma_{P,i}]\}_{i=1}^{y}$; $\hat{\mathcal{A}}$ has broken the IND-CPA or the linear-only homomorphism property of Σ.

2. Otherwise if $(\{[\pi_{1,j}]\}_{j=1}^{k-1}, [\pi_2])$ are affine combinations of $\{[\sigma_{P,i}]\}_{i=1}^{y}$. There exists an extractor for some matrices $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$ from $(\{[\pi_{1,j}]\}_{j=1}^{k-1}, [\pi_2])$ and $\{[\sigma_{P,i}]\}_{i=1}^{y}$ such that $\{[\pi_{1,j}, r_{1,j}]\}_{j=1}^{k-1} = \mathbf{\Pi}_1(\{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y})$ and $[\pi_2, r_2] = \mathbf{\Pi}_2(\{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y})$ unless $\hat{\mathcal{A}}$ has broken the IND-CPA or the linear-only homomorphism of Σ.

3. Consequently, from the extractable matrices $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$, as $\mathsf{k}_c = \mathsf{k}_r$ implies that $(\{\pi_{1,j}\}_{j=1}^{k-1}, \pi_2)$ is a valid proof for R (as $(\{\pi_{1,j}\}_{j=1}^{k-1}, \pi_2)$ has passed the test $\mathbf{t}(\boldsymbol{\sigma}_V, \{\pi_{1,j}\}_{j=1}^{k}, \pi_2) = 0$, see Fig. 5.1), there must exists an extractor for the witness ω such that $R(\phi, \omega) = 1$. Otherwise, $\hat{\mathcal{A}}$ has broken the statistical soundness property of the underlying split DV NILP (see Eq. 5.1).

Thus, we conclude that Ω is adaptively knowledge sound. □

Honest Verifier Zero-knowledge. To prove the honest verifier zero-knowledge property of Ω, we show how to construct $\mathcal{S}_{ZK}$ from the underlying split DV NILP (Setup,Prove,Verify,Simulate):

1. Fix an LE scheme Σ;
2. Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$; and $(\boldsymbol{\sigma}_P, \boldsymbol{\sigma}_V) \leftarrow \mathsf{Setup}(R)$;
3. Encrypt $[\sigma_{P,i}, r_{P,i}] = \Sigma.\mathsf{Enc}(\mathsf{pk}, \sigma_{P,i})$ for each $\sigma_{P,i} \in \boldsymbol{\sigma}_P$;
4. Encrypt $[r_{P,i}] = \Sigma.\mathsf{Enc}(\mathsf{pk}, r_{P,i})$ for each $r_{P,i}$ used above;
5. Set $\mathsf{p}_c = (\mathsf{pk}, \{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y}, \{[r_{P,i}]\}_{i=1}^{y})$ and $\mathsf{s}_c = (\mathsf{sk}, \boldsymbol{\sigma}_V)$.
6. Run $(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) \leftarrow \mathsf{Simulate}(R, \boldsymbol{\sigma}_V, \phi)$ where $\boldsymbol{\pi}_1 = \{\pi_{1,j}\}_{j=1}^{k-1}$ and $\boldsymbol{\pi}_2 = \{\pi_2\}$;
7. Encrypt $[\pi_{1,j}] \leftarrow \Sigma.\mathsf{Enc}(\mathsf{pk}, \pi_1^j)$ for each $\pi_{1,j} \in \boldsymbol{\pi}_1$;
8. Encrypt $[\pi_2, r_2] \leftarrow \Sigma.\mathsf{Enc}(\mathsf{pk}, \pi_2)$ and $[r_2] \leftarrow \Sigma.\mathsf{Enc}(\mathsf{pk}, r_2)$;
9. Set $\mathsf{p}_r = (\{[\pi_{1,j}]\}_{j=1}^{k}, [r_2])$ and $\mathsf{k}_r = [\pi_2]$.
10. Return $(\mathsf{s}_c, \mathsf{p}_c, \mathsf{p}_r, \mathsf{k}_r = [\pi_2], \mathsf{k}_c = [\pi_2])$.

The simulation and the real protocol only differ in Step 6 where the simulated proof $(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$ is obtained. Owing to the zero-knowledge property of the underlying split DV NILP, the simulated proof and the real proof are statistically indistinguishable. Hence, the views of the adversary $\hat{\mathcal{A}}$ in the simulation and in the real protocol are statistically indistinguishable. We conclude that Ω is honest verifier zero-knowledge. □

Response Indistinguishability. To prove the response indistinguishability property of Ω, we show how to construct $\mathcal{S}_{RI}$:

1. Randomly pick $(\mathbf{\Pi}_1, \mathbf{\Pi}_2) \leftarrow (\mathbb{F}^{k-1\times y}, \mathbb{F}^{1\times y})$;
2. Compute $\{[\pi_{1,j}, r_{1,j}]\}_{j=1}^{k-1} = \mathbf{\Pi}_1(\{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y})$ (with $\Sigma.\mathsf{Add}$);
3. Compute $[\pi_2, r_2] = \mathbf{\Pi}_2(\{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y})$ (with $\Sigma.\mathsf{Add}$);
4. Compute $[r_2] = \mathbf{\Pi}_2(\{[r_{P,i}]\}_{i=1}^{y})$ (with $\Sigma.\mathsf{Add}$);
5. Return $\mathsf{p}_r = (\{[\pi_{1,j}, r_{1,j}]\}_{j=1}^{k}, [r_2])$.

The simulation and the real protocol is only different in Step 1 where instead of the valid proof matrices (as in the real protocol), $\mathcal{S}$ obtains the completely random matrices $(\mathbf{\Pi}_1, \mathbf{\Pi}_2)$ (in the simulation). Since the adversary $\hat{\mathcal{A}}$ can see the IND-CPA secure ciphertexts (p_c), the views of the adversary $\hat{\mathcal{A}}$ (without s_c) in the simulation and in the real protocol are computationally indistinguishable unless $\hat{\mathcal{A}}$ has broken the IND-CPA property of Σ. Thus, we conclude that Ω satisfies Response Indistinguishability. □

5.5 Witness Key Agreement for Quadratic Arithmetic Program

Finally, we show in Fig. 5.8 how to construct Ω using a split DV NILP obtained from the NILP in Fig. 5.2.

Theorem 5.2. *If the LE scheme Σ satisfies correctness, additive homomorphism, IND-CPA, and linear-only homomorphism, then the construction in Fig. 5.8 yields a WKA scheme Ω that satisfies correctness, adaptive knowledge soundness, honest verifier zero-knowledge, and response indistinguishability.*

Proof. Since our NILP is reformulated from Groth's NILP in Fig. 5.2 (see Remark 5.7), it satisfies perfect completeness, perfect zero-knowledge, and statistical knowledge soundness against affine prover strategies (see the full security proof of the NILP by Groth [102, Theorem 1].

The correctness, additive homomorphism, IND-CPA, and linear-only homomorphism of Σ *and* the perfect completeness, perfect zero-knowledge and statistical knowledge soundness against affine prover strategies of the underlying split DV NILP implies that Ω satisfies correctness, adaptive knowledge soundness, honest verifier zero-knowledge, and response indistinguishability (Theorem 5.1). □

5.6 Instantiation

Similarly to Gennaro *et al.* [91] and Bitansky *et al.* [27], we choose to instantiate the linear-only encryption scheme Σ with a variant of the Paillier cryptosystem [148].

We assume 1^λ can be deduced from R. Compared with the original NILP in Fig. 5.2, our NILP neglects γ since we only need Eq (5.4), (5.5), (5.6) and (5.8) that do not contain γ (γ is only needed in the verification equation Eq. (5.7)).

$(\mathsf{p}_c, \mathsf{s}_c) \leftarrow \mathsf{KChallenge}(R)$**:** Fix an LE scheme Σ (with key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$), run $(\boldsymbol{\sigma}_P, \boldsymbol{\sigma}_V) \leftarrow \mathsf{Setup}(R)$ to obtain $\boldsymbol{\sigma}_V = (\alpha, \beta, \delta, x)$ and generate the ciphertexts $\{[\sigma_{P,i}, r_{P,i}] \leftarrow \Sigma.\mathsf{Enc}(\mathsf{pk}, \sigma_{P,i})\}$ and $[r_{P,i}] \leftarrow \Sigma.\mathsf{Enc}(\mathsf{pk}, r_{P,i})$ for each $\sigma_{P,i} \in \boldsymbol{\sigma}_P$ where

$$\boldsymbol{\sigma}_P = \left\{ \alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \left\{ \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} \right\}_{i=l+1}^{m}, \left\{ \frac{x^i t(x)}{\delta} \right\}_{i=0}^{n-2} \right\}; \tag{5.19}$$

Return $\mathsf{p}_c = (\mathsf{pk}, \{[\sigma_{P,i}, r_{P,i}]_{i=1}^{y}\}, \{[r_{P,i}]\}_{i=1}^{y})$ and $\mathsf{s}_c = (\mathsf{sk}, \boldsymbol{\sigma}_V)$.

$(\mathsf{p}_r, \mathsf{k}_r) \leftarrow \mathsf{KResponse}(\phi = \{a_i\}_{i=0}^{l}, \omega = \{a_i\}_{i=l+1}^{m}, R, \mathsf{p}_c)$**:** Upon receiving the challenge p_c,

1. Pick $r, s \leftarrow \mathbb{F}$;
2. Compute $[A]$, $[B]$, and $[C]$ (as well as $[r_2]$) using the affine functions in Fig. 5.2 (Eq. (5.4), (5.5) and (5.6)) on $\{[\sigma_{P,i}, r_{P,i}]_{i=1}^{y}\}$ (and $\{[r_{P,i}]\}_{i=1}^{y}$) with $\Sigma.\mathsf{Add}$;
3. Set $[\pi_{1,1}] = [A]$, $[\pi_{1,2}] = [B]$ and $[\pi_2, r_2] = [C]$;
4. Return $\mathsf{p}_r = ([\pi_{1,1}], [\pi_{1,2}], [r_2])$ and $\mathsf{k}_r = [\pi_2, r_2]$.

$\{\mathsf{k}_c, \perp\} \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r)$ outputs $\perp$ if any verification fails:

1. Verify $\mathsf{ImgVer}(\mathsf{pk}, [\pi_{1,j}]) = 1$ for $j = \{1, 2\}$;
2. Verify $\mathsf{ImgVer}(\mathsf{pk}, [r_2]) = 1$;
3. Decrypt $A = \Sigma.\mathsf{Dec}(\mathsf{sk}, [\pi_{1,1}])$; and $B = \Sigma.\mathsf{Dec}(\mathsf{sk}, [\pi_{1,2}])$;
4. Decrypt $r_2 = \Sigma.\mathsf{Dec}(\mathsf{sk}, [r_2])$;
5. Compute C as in Eq. (5.8) with A and B;
6. Return $\mathsf{k}_r = \Sigma.\mathsf{Enc}(\mathsf{pk}, C, r_2)$ (using r_2 as randomness).

Figure 5.8: Witness key agreement for QAP

Remark 5.8 (Extension of NILP over a ring)**.** *The Paillier encryption scheme (in Fig. 5.9) technically requires a* ring*, not a field (as applicable to the WKA scheme Ω). This is not an issue since it is unlikely to encounter encodings of nontrivial zero divisors in $\mathbb{Z}_N^*$ unless one can factor N.*

Remark 5.9 (On oblivious ciphertext sampling)**.** *Additional ciphertexts are required to adapt the Paillier encryption scheme into linear-only encryption: the encryption of a message m will output a pair of ciphertexts $c = \mathsf{Enc}(\mathsf{pk}, m)$ and $c' = \mathsf{Enc}(\mathsf{pk}, \theta m)$ (for some pre-defined secret parameter θ). Hence, the decryption of a ciphertext $[m]$ requires decrypting two ciphertexts $c = [m]$ and $c' = [\theta m]$.* ImgVer *must check this additional*

The original *Scheme 3* of the Paillier encryption scheme requires a multiplicative group $\mathbb{Z}^*_{N^2}$, for $N = pq$, where p and q are two prime numbers:

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$**:** runs as follows.

1. Select random primes p and q ($|p|, |q| \leq \frac{\lambda}{2}$),
2. Compute $N = pq$ and $\gamma = \mathsf{lcm}(p-1, q-1)$,
3. Randomly select g where $g \in \mathbb{Z}^*_{N^2}$ and the order of g is γN.
   As shown by Paillier [148], g is selected efficiently by checking whether $gcd(\mathsf{L}(g^\gamma \mod N^2), N) = 1$ where $\mathsf{L}(u) = \frac{u-1}{N}$ (for $u \equiv 1 \mod N$).

Output public $\mathsf{pk} = (N, g)$ and keep secret $\mathsf{sk} = (p, q, \gamma)$.

$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$**:** Pick a random $r \in \mathbb{Z}_N$ and output $c = g^{m+rN} \mod N^2$. As the order of g is γN, there could be bias in the output distribution of Enc if one picks $r \in \mathbb{Z}_N$ (this bias was present in the original Paillier's paper). To remove this bias one could pick $r \in \mathbb{Z}_\gamma$. However this bias should be negligible as the attacker should not able to distinguish between sampling in N or sampling in $\phi(N)$. Furthermore, even though γ, the secret key, is not usually available to the party that runs Enc, in our case, the investor knows γ and is the only party supposed to run Enc, so she could pick $r \in \mathbb{Z}_\gamma$ and avoid the bias.

$\{0, 1\} \leftarrow \mathsf{ImgVer}(\mathsf{sk}, \mathsf{pk}, c)$**:** Output 1 iff $c \in \mathbb{Z}^*_{N^2} \wedge \gcd(c, N) = 1$.

$m = \mathsf{Dec}(\mathsf{sk}, c)$**:** Output $m = \frac{\mathsf{L}(c^\gamma \mod N^2)}{\mathsf{L}(g^\gamma \mod N^2)} \mod N$.

$\hat{c} = \mathsf{Add}(\langle \alpha_i \rangle_{i=0}^n | \mathsf{pk}, \langle c_i \rangle_{i=0}^n)$**:** Output $\hat{c} = \prod_{i=0}^n c_i^{\alpha_i} \mod N^2$.

The additive homomorphism is straight forward to verify as:

$$\prod_{i=0}^{n} (c_n^i)^{\alpha_i} = g^{\sum_{i=0}^n \alpha_i m_i + (\sum_{i=0}^n \alpha_i r_i) N} \mod N^2$$

Figure 5.9: The Scheme 3 of the Paillier cryptosystem [148]

*linear relation (*ImgVer *decrypts c, c', and check their consistency with θ, i.e. $c' = \theta c$). This variant is also considered by Gennaro* et al. *[91] and Bitansky* et al. *[27] when instantiating the linear-only encryption scheme.*

5.7 Performance Evaluation

The Paillier cryptosystem is the main ingredient of our construction, and its performance is well-studied in the literature. Paillier mentioned several optimization techniques in the original paper [148], and Jost *et al.* [114] took a step to improve the performance

(including parallelization) by orders of magnitude faster as compared with a naïve implementation.

We use the baseline performance reported by Jost *et al.* [114]. In their implementation, to generate the randomness g^{rN} they precomputed 2^w exponentiations of g^{rN} for different random r and randomly combined z of them to obtain the final randomness. As the precomputed values may be repeated, the actual number of samples is $\binom{2^w+z-1}{z}$. For $z = 10$ and $w = 16$, this gives approximately 2^{138} samples.

Jost *et al.* [114] reported a number of experiments with different values of the parameters: $w = (16, 20)$ and $z = (4, 5, 8, 9, 10, 16)$. For the key length of 2048-bit (whose security is 112-bit), we choose the pair (w, z) with the least storage ($w = 16$) and an acceptable level of security ($z = 10$) which gives 138-bit security for guessing the randomness (must be at least 112-bit). For the timing of the Paillier encryption scheme we use the data from Table 4 by Jost *et al.* [114] for the encryption time. The numbers were obtained on an Intel i7-4600U CPU at 2.10GHz with 4 cores running Ubuntu 64-bit v14.04. In particular, the reported result shows that, at 2048-bit key length, the encryption rate for 32-bit messages can reach 56K/s at the cost of 5.7s pre-computation time.

We estimate the theoretical and practical performance of our WKA scheme Ω based on the number of encryptions, decryptions, and scalar multiplications for computing $\mathbf{\Pi}_1(\{[\sigma_{P,i}]\})$ and $\mathbf{\Pi}_2(\{[\sigma_{P,i}]\})$ (Table. 5.2). We ignore the cost of additions as they are fast. For the Paillier cryptosystem, while encryption requires exponentiation, addition is only performed as the multiple-precision modular multiplication on the ciphertexts. The number of decryptions is also small in our concrete construction, i.e. $k = 3$. Most computational costs are for encryptions and scalar multiplications since the scalar multiplications are performed as exponentiations in the Paillier scheme.

5.7.1 Theoretical Performance Evaluation

Let us recall that m is the number of variables of a QAP, l is the number of instance variables, and $n - 1$ is the degree of polynomials of the QAP. The KChallenge algorithm requires the generation of $\{[\sigma_{P,i}]\}$; hence, it requires $m-l+2n$ encryptions on the investor's side. The KResponse algorithm requires only the proof computation on the trader's side, yielding $m - l + 3n$ scalar multiplications. The above numbers are doubled to fix the malleability of the encryption scheme (see Remark 5.9). Then, it is doubled again for computing the ciphertexts of the randomnesses. Finally, the KDerive algorithm requires k decryptions and 1 encryption on the investor's side.

Table 5.2: Theoretical Performance evaluation (non-deterministic case)

Algorithm	**#Encryption**	**#Decryption**	**#Multiplication**
KChallenge	$4(m-l+2n)$	-	-
KResponse	-	-	$4(m-l+3n)$
KDerive	1	k	-

Note: m is the number of variables in a QAP, l is the number of instance variables, and $n-1$ is the degree of polynomials in the QAP. The number of decryption k is construction dependent. In our case, we have $k=3$.

Table 5.3: Specific circuit evaluation

Relation R	m	$n-1$	$\mathcal{T}_C$ (s)	$\mathcal{T}_R$ (s)
SC	25821	28312	5.8	7.8
PR	26080	28572	5.8	7.9

Note: The encryption scheme supports 2048-bit key length and provides 112-bit security. Recall m is the number of variables and $n-1$ is the degree of polynomials of the QAP.

5.7.2 Practical Performance Evaluation

We implement the relations SC and PR in Table 5.1 as arithmetic circuits using the libsnark library [166] to measure the number of required variables m and the corresponding degree of the polynomials $(n-1)$. The runtime of KChallenge and KResponse, the most costly algorithms, are *estimated with* m *and* n[5], for 112-bit security at 2048-bit key length, using the 32-bit messages and the encryption rate as in Scheme 3 from Jost *et al.* [114].

As shown in Table 5.3, the performance of SC and PR are almost the same as their circuit complexities are similar. KChallenge ($\mathcal{T}_C$) requires only 5.8s for the SC relation and 5.9s for the PR relation. After the KChallenge, the key-agreement with KResponse ($\mathcal{T}_R$) takes only 7.8s for SC and 7.9s for PR.

For our simulation we make use of the aggressive Bloomberg Tradebook [37] for the period from Mar 13th to May 1st 2019 (35 days of trading in total).

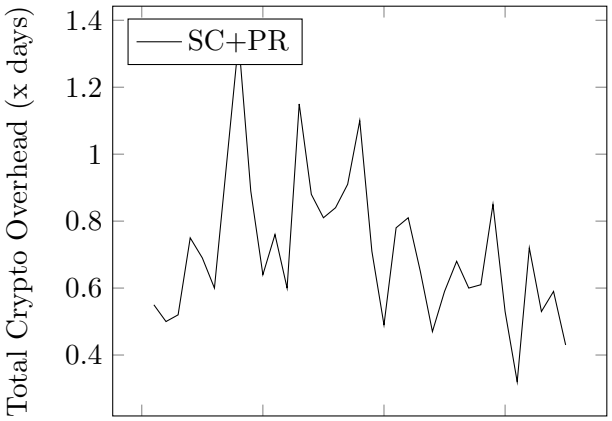
Bloomberg Tradebook is an agency broker that serves financial services providers in the US. The brokerage pool is a darkpool (quoting is restricted to the market participants) and only trades (market orders) are reported to the market. There is no central limit order book, but an electronic blind matching algorithm that utilizes a direct market access tool for individual positions. The traders then can match against each other's price-demand schedules (hence a very similar set up to our WKA approach). So there are no brokers or market makers in this set up.

The data available through Bloomberg is price data that contains the underlying number of quote-message-queries needed prior to the trade being executed. Hence, there

[5] l is small, $l < 100$ in all examples.

Table 5.4: Market data samples for low and high days of Bloomberg Tradebook

Day	Low	High
# Messages	4514	14103
# Trades	53	55



Even with the aggressive Bloomberg Tradebook, only 3 out of 35 days of trading exhibits overheads greater than 1x in our simulation.

Figure 5.10: WKA Evaluation on Bloomberg Tradebook

is an establishment of price, quantity and demand prior to each trade being executed. The data here are for a specific date matched set of US Dollar denominated corporate securities traded on Bloomberg Tradebook. Each data point in the tradebook is the number of messages (which will be communications of quotes within the dark pool) prior to an individual time stamped trade being executed. It should be noted that while each trade is atomic (i.e.a transaction between two specific counter parties) the messages between trades are not necessarily directly connected to the specific trade. Given that the trade is a globally observable variable, it is reasonable to presume that most messages within the trade time interval correspond to a particular transaction being explored and validated.

The number of trades within this specific market are low, around 55 per day. The number of messages is quite high, at around 6500 messages per day at an average of around 130 messages quoted per trade executed. This is quite typical. Futures markets, such as the Eurodollar, Crude Oil and agricultural commodities, have very similar limit-order (public quotes) to traded securities ratios of between 50 to 300 quotes per trade. Hence, this benchmark provides a realistic test case for our algorithm for a single security.

We combine the number of messages and trades from the extracted market data (exam-

ples shown in Table 5.4) and Table 5.3 to estimate the corresponding execution overhead throughout a day of trading. The final results are reported in Fig. 5.10. Performance is evaluated in terms of execution overhead to the expected processing time (1 day). We combine the relations SC with PR and we consider the execution time of a message as the running time of SC's KChallenge (5.8s). For trades execution time we consider the sequential execution of KResponse from SC and the whole challenge and response time of PR (21.6s). As shown in Fig. 5.10 we can fit 32 out of 35 days of trading within the day in our simulation.

5.8 Summary

We introduced the notion of witness-key-agreement. Specifically we defined split designated-verifier non-interactive linear proof following Groth's definition of NILP [102]. We then compiled the obtained split DV NILP into a Witness Key Agreement scheme using Linear-Only Encryption.

Our obtained construction is efficient. After a one-time setup that yields a common challenge for a relation R of interest, a party can agree on a secret key with another party given that the latter knows a witness of a committed instance. Finally, our concrete WKA scheme for quadratic arithmetic programs yields both succinct communication complexity, i.e.the response to the common challenge consists of only 3 encrypted elements, and efficient response computation and key derivation, i.e.only linear to the QAP size.

Our scheme is particularly suitable for private auctions in financial intermediation in which one party wants to privately communicate with another party about committed financial information which satisfies a relation R of interest.

Chapter 6

Conclusion

6.1 Research Contribution

We have addressed **RQ1**, whose main goal is the understanding of simple payment networks, with a high-level review of the traditional and crypto-based Payment Transaction Networks (PTNs). We have identified and categorized the security challenges posed by such systems and discussed the possible solutions to those challenges.

However, addressing only such challenges is not enough to build more advanced financial intermediation, such as a futures exchange, owing to (i) the interaction between security and economic viability; (ii) the essentially non-monotonic behavior (Alice's good standing is invalidated by an honest Bob), and (iii) the proportional burden requirement in which the computational effort must be on par with the activities. To address **RQ2**, we have identified and enucleated design principles to six non-trivial challenges (noticeably the Price Discrimination Attack being the most interesting one) to decentralize a futures exchange.

The distilled principles are then used to address **RQ3** as we have shown how to build FuturesMEX, a secure, distributed futures exchange:

1. We have put forward a cryptographic ideal functionality for a distributed futures market, that captures all of the key security requirements. This is an ideal realization of a distributed futures market, where the market is run by a trusted third party which knows the secret inputs of all participating traders, and lets the market evolve on their behalf. Such a functionality, by construction, embodies features described in **RQ2**.

2. We have designed a cryptographic protocol securely realizing our ideal functionality. Our protocol combines multiparty computation (MPC) and non-interactive zero-knowledge proofs on committed inputs, only relying on the basic assumptions

of secure broadcast channels between traders and an anonymous network. These assumptions already appeared in several prior works, most notably ZeroCash [164]. We replace the local constraints verification of the MPC with non-interactive proofs for efficient generation of publicly verifiable transactions and scalability w.r.t. the number of traders. Full MPC is only performed for sub-tasks capturing the non-monotonicity and anonymity requirements of the market. We prove the security of our protocol with security-with-abort—where we allow an adversary to abort the computation after receiving its own intermediate outputs [111]—and extend it so that an aborting adversary is penalized by forfeiting its hard won stake in the market, the ultimate discouragement in our setting.

3. We have shown that our approach is feasible. We have done so, by providing a proof of concept implementation using the libsnark library [166] for the zero-knowledge proofs, and the SPDZ protocol [69] for securely realizing the required MPC sub-tasks. We have further optimized our protocol in order to yield a 70% efficiency gain. Our results show that our solution is *feasible for low frequency markets* at CME (e.g.trading in Lean Hog commodities): a trading day can be executed in a day by an Amazon's EC2 large VM. Further optimizations are needed for high frequency trading in the largest markets (Eurodollar, Foreign Exchange and Crude Oil futures), for instance by parallelizing proof generation as most of them are independent, improvements in the zk-SNARK implementation; or different commitment functions or batch proofs for good standing (e.g. proving the validity of a trader's inventory for a range of prices).

For **RQ4**, we have presented the *first practical Witness Key Agreement scheme under designated-verifier zk-SNARK proof for Quadratic Arithmetic Programs* (QAP) [27, 91, 102] and shown how to instantiate the scheme with the Paillier cryptosystem [148]. In our WKA scheme, a designated investor firsts broadcast a common reference string (CRS) as a challenge for the relation R of interest. A trader then publishes a partial zk-SNARK proof as a response for the committed instance that satisfies R. Using the partial zk-SNARK proof, the investor derives a shared secret key with the trader. The CRS for R is only computed once and is reusable for as many instances as desired. Partial proof size is also succinct (low communication complexity), as it has at most three elements regardless of R. Response computation and key derivation are efficient, i.e.only linear in the QAP size. We have also provided the security analysis and performance evaluation (both theoretical and practical of our WKA scheme using the Bloomberg tradebook. From the results obtained, for the period from Mar 13 to May 1st 2019, our solution is suitable for 32 out of 35 days of dark trading.

6.2 Future Work

The first interesting avenue of future research is the choice of majority for the distributed consensus protocol: majority of traders or majority weighted by volumes. These solutions need further validation by financial economists. Secondly, anonymous communication networks often yield higher latency, an interesting research direction is to develop a secure distributed trading platform without such networks. Drop-out and abort are also the issues; even though they are also applicable to all secure multiparty computation protocols that do not rely on the honest majority. Another direction is to analyze liveliness and robustness against collusion and price discrimination. For simplicity, our protocol by default goes to mark-to-market upon failure. Alternatives are possible, e.g. margin-call for additional funds. We leave this line for future work.

Bibliography

[1] Martín Abadi and Roger Needham. Prudent Engineering Practice for Cryptographic Protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, 1996.

[2] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A Subversion-Resistant SNARK. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 3–33. Springer, 2017.

[3] Dennis Abrazhevich. Classification and Characteristics of Electronic Payment Systems. In *Electronic Commerce and Web Technologies*, pages 81–90. Springer, 2001.

[4] Hamza Abusalah, Georg Fuchsbauer, and Krzysztof Pietrzak. Offline Witness Encryption. In *International Conference on Applied Cryptography and Network Security*, pages 285–303. Springer, 2016.

[5] Franklin Allen and Anthony M Santomero. The Theory of Financial Intermediation. *Journal of Banking & Finance*, 21(11-12):1461–1485, 1997.

[6] Mashael AlSabah and Ian Goldberg. Performance and Security Improvements for Tor: A Survey. *ACM Computing Surveys*, 49(2):32, 2016.

[7] Milton Anderson. The Electronic Check Architecture, 1998. `http://echeck.org/files/ArchitectualOverview.pdf`. Accessed: 2019-05-01.

[8] Apple. Apple Pay. `http://www.apple.com/apple-pay/`. Accessed: 2019-05-01.

[9] David W Archer, Dan Bogdanov, Benny Pinkas, and Pille Pullonen. Maturity and Performance of Programmable Secure Computation. *IEEE security & privacy*, 14(5):48–56, 2016.

[10] BACS. BACS. `http://www.bacs.co.uk`. Accessed: 2019-05-01.

[11] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous Credentials Light. In *2013 ACM SIGSAC conference on Computer & Communications Security*, pages 1087–1098. ACM, 2013.

[12] Bank of England. Payment and Settlement. `https://www.bankofengland.co.uk/payment-and-settlement`. Accessed: 2019-05-01.

[13] Bank of England. One Bank Research Agenda. `https://www.bankofengland.co.uk/-/media/boe/files/research/one-bank-research-agenda---summary.pdf`. Accessed: 2019-05-01.

[14] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an Untrusted CRS: Security in the face of Parameter Subversion. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 777–804. Springer, 2016.

[15] Mihir Bellare and Oded Goldreich. On Defining Proofs of Knowledge. In *International Cryptology Conference*, pages 390–420. Springer, 1992.

[16] Mihir Bellare and Viet Tung Hoang. Adaptive Witness Encryption and Asymmetric Password-Based Cryptography. In *IACR International Workshop on Public Key Cryptography*, pages 308–331. Springer, 2015.

[17] Steven M Bellovin and Michael Merritt. Encrypted Key Exchange: Password-based Protocols Secure Against Dictionary Attacks. In *1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 72–84. IEEE, 1992.

[18] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *20th ACM symposium on Theory of Computing*, pages 1–10. ACM, 1988.

[19] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge. In *Advances in Cryptology*, pages 90–108. Springer, 2013.

[20] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs. In *2015 IEEE Symposium on Security and Privacy*, pages 287–304. IEEE, 2015.

[21] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. In *23rd USENIX Security Symposium*, pages 781–796, 2014.

[22] Josh Benaloh and Michael De Mare. One-Way Accumulators: A Decentralized Alternative to Digital Signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 274–285. Springer, 1993.

[23] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-Homomorphic Encryption and Multiparty Computation. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 169–188. Springer, 2011.

[24] David Bernhard and Bogdan Warinschi. Cryptographic VotingA Gentle Introduction. In *Foundations of Security Analysis and Design VII*, pages 167–211. Springer, 2014.

[25] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of Clients in Bitcoin P2P Network. In *2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29. ACM, 2014.

[26] BIS. Real-Time-Gross-Settlement System. `https://www.bis.org/cpmi/publ/d22.pdf`. Accessed: 2019-05-01.

[27] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Omer Paneth, and Rafail Ostrovsky. Succinct Non-Interactive Arguments via Linear Interactive Proofs. In *Theory of Cryptography Conference*, pages 315–333. Springer, 2013.

[28] BitcoinWiki. Bitcoin Address. `https://en.bitcoin.it/wiki/Address`. Accessed: 2019-05-01.

[29] BitcoinWiki. Bitcoin Coinbase Transaction. `https://en.bitcoin.it/wiki/Coinbase`. Accessed: 2019-05-01.

[30] BitcoinWiki. Bitcoin Mining Introduction. `https://en.bitcoin.it/wiki/Mining#Introduction`. Accessed: 2019-05-01.

[31] BitcoinWiki. Bitcoin Proof of Stake. `https://en.bitcoin.it/wiki/Proof_of_Stake`. Accessed: 2019-05-01.

[32] BitcoinWiki. Bitcoin Proof of Work. `https://en.bitcoin.it/wiki/Proof_of_work`. Accessed: 2019-05-01.

[33] BitcoinWiki. Bitcoin Transaction. `https://en.bitcoin.it/wiki/Transaction`. Accessed: 2019-05-01.

[34] BitcoinWiki. Bitcoin Transaction Fees. `https://en.bitcoin.it/wiki/Transaction_fees`. Accessed: 2019-05-01.

[35] BitcoinWiki. Bitcoin Transaction Input. `https://en.bitcoin.it/wiki/Transaction#Input`. Accessed: 2019-05-01.

[36] BitcoinWiki. Bitcoin Transaction Output. `https://en.bitcoin.it/wiki/Transaction#Output`. Accessed: 2019-05-01.

[37] Bloomberg. Tradebook Bloomberg Professional Services. `https://www.bloomberg.com/professional/solution/tradebook/`, 2019. Accessed: 2019-05-01.

[38] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, et al. Secure Multiparty Computation Goes Live. In *International Conference on Financial Cryptography and Data Security*, pages 325–343. Springer, 2009.

[39] Fabrice Boudot. Efficient Proofs that a Committed Number lies in an Interval. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 431–444. Springer, 2000.

[40] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable Multi-party Computation for zk-SNARK Parameters in the Random Beacon Model. Cryptology ePrint Archive, Report 2017/1050, 2017. `https://eprint.iacr.org/2017/1050`. Accessed: 2019-05-01.

[41] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[42] Jan Camenisch, Nathalie Casati, Thomas Groß, and Victor Shoup. Credential Authenticated Identification and Key Exchange. In *International Cryptology Conference*, pages 255–276. Springer, 2010.

[43] Jan Camenisch, Rafik Chaabouni, et al. Efficient Protocols for Set Membership and Range Proofs. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 234–252. Springer, 2008.

[44] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact E-Cash. In *Advances in Cryptology*, pages 302–321. Springer, 2005.

[45] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *2001 IEEE International Conference on Cluster Computing*, pages 136–145. IEEE, 2001.

[46] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *34th ACM Symposium on Theory of Computing*, pages 494–503. ACM, 2002.

[47] John Cartlidge, Nigel P. Smart, and Younes Talibi Alaoui. MPC Joins the Dark Side. Cryptology ePrint Archive, Report 2018/1045, 2018. `https://eprint.iacr.org/2018/1045`. Accessed: 2019-05-01.

[48] Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Transactions on Computer Systems*, 20(4):398–461, 2002.

[49] Miguel Castro, Barbara Liskov, et al. Practical Byzantine Fault Tolerance. In *3rd USENIX Symposium on Operating Systems Design & Implementation*, volume 99, pages 173–186, 1999.

[50] Tushar Deepak Chandra and Sam Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the ACM*, 43(2):225–267, 1996.

[51] CHAPS. CHAPS. `http://www.chapsco.co.uk`. Accessed: 2019-05-01.

[52] Melissa Chase and Anna Lysyanskaya. On Signatures of Knowledge. In *International Cryptology Conference*, pages 78–96. Springer, 2006.

[53] David Chaum. Blind Signatures for Untraceable Payments. In *Advances in Cryptology*, pages 199–203. Springer, 1982.

[54] David Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[55] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty Unconditionally Secure Protocols. In *the twentieth ACM symposium on Theory of Computing*, pages 11–19. ACM, 1988.

[56] David Chaum, Amos Fiat, and Moni Naor. Untraceable Electronic Cash. In *Advances in Cryptology*, pages 319–327. Springer-Verlag New York, Inc., 1990.

[57] Jeremy Clark, Joseph Bonneau, Edward W Felten, Joshua A Kroll, Andrew Miller, and Arvind Narayanan. On Decentralizing Prediction Markets and Order Books. In *Workshop on the Economics of Information Security*, 2014.

[58] CloudFlare. CloudFlare. `https://www.cloudflare.com`. Accessed: 2019-05-01.

[59] CME. Eurodollar. `https://www.cmegroup.com/confluence/display/EPICSANDBOX/Eurodollar#Eurodollar-NormalDailySettlement`. Accessed: 2019-05-01.

[60] CME. Margin: Know What's Needed. `https://www.cmegroup.com/education/courses/introduction-to-futures/margin-know-what-is-needed.html`. Accessed: 2019-05-01.

[61] CME. Market and Instrument States. `https://www.cmegroup.com/confluence/display/EPICSANDBOX/Market+and+Instrument+States`. Accessed: 2019-05-01.

[62] CME Group. Eurodollar. `http://www.cmegroup.com/confluence/display/EPICSANDBOX/Eurodollar`. Accessed: 2019-05-01.

[63] CoinDesk. Understanding The DAO Attack. `http://www.coindesk.com/understanding-dao-hack-journalists/`. Accessed: 2019-05-01.

[64] CoinMarketCap. CoinMarketCap. `https://coinmarketcap.com/`. Accessed: 2019-05-01.

[65] Rama Cont, Arseniy Kukanov, and Sasha Stoikov. The Price Impact of Order Book Events. *Journal of Financial Econometrics*, 12(1):47–88, 2014.

[66] Ronald Cramer and Victor Shoup. Universal Hash Proofs and A Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 45–64. Springer, 2002.

[67] Wei Dai. b-money. `http://www.weidai.com/bmoney.txt`, 1998. Accessed: 2019-05-01.

[68] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P Smart. Practical Covertly Secure MPC for Dishonest Majority–or: Breaking the SPDZ Limits. In *European Symposium on Research in Computer Security*, pages 1–18. Springer, 2013.

[69] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. In *Advances in Cryptology*, pages 643–662. Springer, 2012.

[70] Hans Degryse, Geoffrey Tombeur, Mark Van Achter, and Gunther Wuyts. *Dark Trading*, chapter 12, pages 213–230. Wiley-Blackwell, 2013.

[71] David Derler and Daniel Slamanig. Practical Witness Encryption for Algebraic Languages or How to Encrypt under Groth–Sahai Proofs. *Designs, Codes and Cryptography*, 86(11):2525–2547, 2018.

[72] Tim Dierks and Christopher Allen. The TLS Protocol version 1.0, 1998. `https://www.ietf.org/rfc/rfc2246.txt`. Accessed: 2019-05-01.

[73] Whitfield Diffie and Martin E Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[74] John R Douceur. The Sybil Attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[75] Evan Duffield and Daniel Diaz. Dash: A PrivacyCentric CryptoCurrency, 2015. `http://blockchainlab.com/pdf/Dash-WhitepaperV1.pdf`. Accessed: 2019-05-01.

[76] Cynthia Dwork, Nancy Ann Lynch, and Larry Stockmeyer. Consensus in the Presence of Partial Synchrony. *Journal of the ACM*, 35(2):288–323, 1988.

[77] Ethereum. A Next-Generation Smart Contract and Decentralized Application Platform. `https://github.com/ethereum/wiki/wiki/White-Paper`. Accessed: 2019-05-01.

[78] Eurosystem. Target2. `https://target2.ecb.europa.eu/`. Accessed: 2019-05-01.

[79] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-NG: A Scalable Blockchain Protocol. In *13th USENIX Symposium on Networked Systems Design & Implementation*, pages 45–59, 2016.

[80] Ittay Eyal and Emin Gün Sirer. Majority Is Not Enough: Bitcoin Mining Is Vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[81] Hal Finney. RPOW - Reusable Proofs of Work. `https://cryptome.org/rpow.htm`, 2004. Accessed: 2019-05-01.

[82] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM*, 32(2):374–382, 1985.

[83] Forbes. Wall Street's Speed War. `http://www.forbes.com/forbes/2010/0927/outfront-netscape-jim-barksdale-daniel-spivey-wall-street-speed-war.html`. Accessed: 2019-05-01.

[84] FRBServices. Fedwire® Funds Service. `https://www.frbservices.org/assets/financial-services/wires/funds.pdf`. Accessed: 2019-05-01.

[85] Georg Fuchsbauer. Subversion-Zero-Knowledge SNARKs. In *IACR International Workshop on Public Key Cryptography*, pages 315–347. Springer, 2018.

[86] Eiichiro Fujisaki and Koutarou Suzuki. Traceable Ring Signature. In *International Workshop on Public Key Cryptography*, pages 181–200. Springer, 2007.

[87] Futures Industry Association. Largest Derivatives Exchanges Worldwide in 2015, by Number of Contracts Traded (in Millions). `https://www.statista.com/statistics/272832/largest-international-futures-exchanges-by-number-of-contracts-traded/`. Accessed: 2016-02-01.

[88] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate Multilinear Maps from Ideal Lattices. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–17. Springer, 2013.

[89] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016.

[90] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness Encryption and Its Applications. In *45th ACM symposium on Theory of Computing*, pages 467–476. ACM, 2013.

[91] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic Span Programs and Succinct NIZKs without PCPs. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 626–645. Springer, 2013.

[92] Craig Gentry, Allison Lewko, and Brent Waters. Witness Encryption from Instance Independent Assumptions. In *International Cryptology Conference*, pages 426–443. Springer, 2014.

[93] Craig Gentry and Daniel Wichs. Separating Succinct Non-Interactive Arguments from all Falsifiable Assumptions. In *43rd ACM symposium on Theory of computing*, pages 99–108. ACM, 2011.

[94] Danezis George and Sarah Meiklejohn. Centrally Banked Cryptocurrencies. In *Network and Distributed System Security Symposium 2016*, pages 1–14, 2016.

[95] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2009.

[96] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that Yield Nothing but Their Validity or All Languages In NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, 38(3):690–728, 1991.

[97] Shafi Goldwasser. How to Play any Mental Game, or A Completeness Theorem for Protocols with an Honest Majority. In *19th ACM symposium on Theory of Computing*, pages 218–229, 1987.

[98] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[99] Li Gong. *Fail-stop Protocols: An Approach to Designing Secure Protocols*. SRI International,Computer Science Laboratory, 1994.

[100] Google. Google Wallet. `https://www.google.com/wallet/`. Accessed: 2019-05-01.

[101] Martin D Gould, Mason A Porter, Stacy Williams, Mark McDonald, Daniel J Fenn, and Sam D Howison. Limit Order Books. *Quantitative Finance*, 13(11):1709–1742, 2013.

[102] Jens Groth. On the size of Pairing-Based Non-Interactive Arguments. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 305–326. Springer, 2016.

[103] Jens Groth and Mary Maller. Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs. In *International Cryptology Conference*, pages 581–612. Springer, 2017.

[104] Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages. In *International Workshop on Public Key Cryptography*, pages 272–291. Springer, 2013.

[105] Dan Harkins and Dave Carrel. The Internet Key Exchange (IKE), 1998. `https://tools.ietf.org/html/rfc2409`. Accessed: 2019-05-01.

[106] Larry Harris. *Trading and Exchanges: Market Microstructure for Practitioners*. Oxford University Press, USA, 2003.

[107] Robert Hatch. Reforming the Murky Depths of Wall Street: Putting the Spotlight on the Security and Exchange Commission's Regulatory Proposal Concerning Dark Pools of Liquidity. *The George Washington Law Review*, 78:1032, 2009.

[108] John Hull, Sirimon Treepongkaruna, David Colwell, Richard Heaney, and David Pitt. *Fundamentals of Futures and Options Markets*. Pearson Higher Education AU, 2013.

[109] HyperLedger. HyperLedger. `https://www.hyperledger.org`. Accessed: 2019-05-01.

[110] IOTA. IOTA. `https://www.iota.org/`. Accessed: 2019-05-01.

[111] Yuval Ishai, Jonathan Katz, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On Achieving the "Best Of Both Worlds" in Secure Multiparty Computation. *SIAM Journal on Computing*, 40(1):122–141, 2011.

[112] Markus Jakobsson and Ari Juels. Proofs of Work and Bread Pudding Protocols. In *Secure Information Networks*, pages 258–272. Springer, 1999.

[113] S Peyton Jones, Jean-Marc Eber, and Julian Seward. Composing Contracts: An Adventure in Financial Engineering. *ACM SIG-PLAN Notices*, 35(9):280–292, 2000.

[114] Christine Jost, Ha Lam, Alexander Maximov, and Ben JM Smeets. Encryption Performance Improvements of the Paillier Cryptosystem. *IACR Cryptology ePrint Archive*, 2015:864, 2015.

[115] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC Press, 2014.

[116] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Group Encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 181–199. Springer, 2007.

[117] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. An Efficient E2E Verifiable E-Voting System without Setup Assumptions. *IEEE Security & Privacy*, 2017.

[118] Sunny King. Primecoin: Cryptocurrency with Prime Number Proof-of-Work, 2013. `http://primecoin.io/bin/primecoin-paper.pdf`. Accessed: 2019-05-01.

[119] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. In *2016 IEEE symposium on security and privacy*, pages 839–858. IEEE, 2016.

[120] Ahmed E Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, T-H Hubert Chan, Charalampos Papamanthou, Rafael Pass, Abhi Shelat, and Elaine Shi. How to Use SNARKs in Universally Composable Protocols. *IACR Cryptology ePrint Archive*, 2015:1093, 2015.

[121] Ranjit Kumaresan and Iddo Bentov. How to use Bitcoin to Incentivize Correct Computations. In *the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 30–41. ACM, 2014.

[122] Ranjit Kumaresan, Tal Moran, and Iddo Bentov. How to Use Bitcoin to Play Decentralized Poker. In *the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 195–206. ACM, 2015.

[123] Ranjit Kumaresan, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Improvements to Secure Computation With Penalties. In *the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 406–417. ACM, 2016.

[124] Albert Sidney Kyle. *A Theory of Futures Market Manipulations*. Number 64. Center for the Study of Futures Markets, Columbia Business School, Columbia, 1983.

[125] John Labuszewski, John Nyhoff, James Boudreault, et al. Disseminating Floor Quotes from Qpen Outcry Markets, April 23 2015. US Patent App. 14/061,286.

[126] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[127] Charles Lee. Litecoin. `https://github.com/litecoin-project/litecoin`, 2011. Accessed: 2019-05-01.

[128] Yehida Lindell. Secure Multiparty Computation for Privacy Preserving Data Mining. In *Encyclopedia of Data Warehousing and Mining*, pages 1005–1009. IGI Global, 2005.

[129] Yehuda Lindell and Benny Pinkas. Secure Multiparty Computation for Privacy-Preserving Data Mining. *Journal of Privacy and Confidentiality*, 1(1):5, 2009.

[130] Katya Malinova, Andreas Park, and Ryan Riordan. Do Retail Traders suffer from High Frequency Traders?, 2013.

[131] Jerry W Markham. Manipulation of Commodity Futures Prices-the Unprosecutable Crime. *Yale Journal on Regulation*, 8:281, 1991.

[132] Billy Markus and Jackson Palmer. Dogecoin. `https://github.com/dogecoin/dogecoin`, 2013. Accessed: 2019-05-01.

[133] Fabio Massacci, Chan Nam Ngo, Jing Nie, Daniele Venturi, and Julian Williams. The Seconomics (Security-Economics) Vulnerabilities of Decentralized Autonomous Organizations. In *Cambridge International Workshop on Security Protocols*, pages 171–179. Springer, 2017.

[134] Henri Massias, X Serret Avila, and J-J Quisquater. Design of a Secure Timestamping Service with Minimal Trust Requirement. In *20th Symposium on Information Theory in the Benelux*, 1999.

[135] MasterCard. MasterCard. `https://www.mastercard.us/en-us.html`. Accessed: 2019-05-01.

[136] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. A Fistful of Bitcoins: Characterizing Payments among Men With No Names. In *2013 Internet Measurement Conference*, pages 127–140. ACM, 2013.

[137] Ralph C Merkle. A Digital Signature based on a Conventional Encryption Function. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 369–378. Springer, 1987.

[138] Silvio Micali. Computationally Sound Proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[139] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. ZeroCoin: Anonymous Distributed E-Cash from Bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411. IEEE, 2013.

[140] Steven P Miller, B Clifford Neuman, Jeffrey I Schiller, and Jermoe H Saltzer. Kerberos Authentication and Authorization System. Project Athena Technical Plan Section E. 2.1, 1987. `http://web.mit.edu/Saltzer/www/publications/athenaplan/e.2.1.pdf`. Accessed: 2019-05-01.

[141] Hitesh Mittal. Are You Playing in a Toxic Dark Pool?: A Guide to Preventing Information Leakage. *The Journal of Trading*, 3(3):20–33, 2008.

[142] Matt Morano, Ian Wall, Samuel Gaer, and Kai Neumann. Distributed trading bus architecture, February 15 2011. US Patent 7,890,412.

[143] MTS. MTS Market. `https://www.mtsmarkets.com`. Accessed: 2019-05-01.

[144] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. `https://bitcoin.org/bitcoin.pdf`. Accessed: 2019-05-01.

[145] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A New Approach to Practical Active-Secure Two-Party Computation. In *Advances in Cryptology*, pages 681–700. Springer, 2012.

[146] Nxt.org. Decentralized Financial Ecosystem. `http://nxt.org`, 2013. Accessed: 2019-05-01.

[147] Claudio Orlandi. Is Multiparty Computation Any Good In Practice? In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5848–5851. IEEE, 2011.

[148] Pascal Paillier. Public-Key Cryptosystems based on Composite Degree Residuosity Classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.

[149] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly Practical Verifiable Computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE, 2013.

[150] PayPal. PayPal. `https://www.paypal.com/us/webapps/mpp/home`. Accessed: 2019-05-01.

[151] Torben Pryds Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *International Cryptology Conference*, pages 129–140. Springer, 1991.

[152] Colin Percival. Stronger Key Derivation via Sequential Memory-Hard Functions, 2009. `https://www.tarsnap.com/scrypt/scrypt.pdf`. Accessed: 2019-05-01.

[153] Craig Pirrong. The Economics of Clearing in Derivatives Markets: Netting, Asymmetric Information, and the Sharing of Default Risks through a Central Counterparty. *Asymmetric Information, and the Sharing of Default Risks Through a Central Counterparty (January 8, 2009)*, 2009.

[154] PotCoin. PotCoin. `https://github.com/potcoin/potcoin`, 2014. Accessed: 2019-05-01.

[155] Michael O Rabin. Randomized Byzantine Generals. In *24th Symposium on Foundations of Computer Science*, pages 403–409. IEEE, 1983.

[156] Tal Rabin and Michael Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In *the twenty-first ACM symposium on Theory of Computing*, pages 73–85. ACM, 1989.

[157] Ripple Labs. Executive Summary for Financial Institutions. `https://ripple.com/solutions/executive-summary-for-financial-institutions/`. Accessed: 2019-05-01.

[158] Ripple Labs . Gateway Guide. `https://ripple.com/build/gateway-guide/`. Accessed: 2019-05-01.

[159] Ioanid Roşu. A Dynamic Model of the Limit Order Book. *The Review of Financial Studies*, 22(11):4601–4641, 2009.

[160] Nicolas van Saberhagen. CryptoNote v 1.0. `https://cryptonote.org/whitepaper_v1.pdf`, 2012. Accessed: 2019-05-01.

[161] Amit Sahai and Brent Waters. Fuzzy Identity-based Encryption. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473. Springer, 2005.

[162] Kazue Sako. An Auction Protocol which Hides Bids of Losers. In *International Workshop on Public Key Cryptography*, pages 422–432. Springer, 2000.

[163] Tomas Sander and Amnon Ta-Shma. Auditable, Anonymous Electronic Cash. In *International Cryptology Conference*, pages 555–572. Springer, 1999.

[164] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. ZeroCash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

[165] Claus-Peter Schnorr. Efficient Signature Generation by Smart Cards. *Journal of cryptology*, 4(3):161–174, 1991.

[166] SCIPR Lab. libsnark: a C++ library for zkSNARK proofs. `https://github.com/scipr-lab/libsnark`. Accessed: 2019-05-01.

[167] Yonatan Sompolinsky and Aviv Zohar. Accelerating Bitcoin's Transaction Processing. Fast Money Grows on Trees, Not Chains. *IACR Cryptology ePrint Archive*, 2013(881), 2013.

[168] SPDZ-2. SPDZ-2. `https://github.com/bristolcrypto/SPDZ-2`. Accessed: 2019-05-01.

[169] Daniel F Spulber. Market Microstructure and Intermediation. *Journal of Economic Perspectives*, 10(3):135–152, 1996.

[170] SWIFT. Discover SWIFT. `https://www.swift.com/about-us/discover-swift`. Accessed: 2019-05-01.

[171] Nick Szabo. Bit gold. `http://unenumerated.blogspot.it/2005/12/bit-gold.html`, 2005. Accessed: 2019-05-01.

[172] The Tor Project. Tor. `https://www.torproject.org/download/`. Accessed: 2019-05-01.

[173] TheClearingHouse. CHIPS. `https://www.theclearinghouse.org/payment-systems/chips`. Accessed: 2019-05-01.

[174] TheVerge. Data glitch sets tech company stock prices at USD 123.47. `https://www.theverge.com/2017/7/3/15917950/nasdaq-nyse-stock-market-data-error`. Accessed: 2019-05-01.

[175] ThomsonReuters. Thomson Reuters Tick History. `http://financial-risk-solutions.thomsonreuters.info/TickHistory`. Accessed: 2019-05-01.

[176] US CFTC. Mission & Responsibilities, 2016. `http://www.cftc.gov/About/MissionResponsibilities/index.htm`. Accessed: 2019-05-01.

[177] U.S. Securities and Exchange Commission. Concept Release on Equity Market Structure, 2010. `https://www.sec.gov/rules/concept/2010/34-61358.pdf`. Accessed: 2019-05-01.

[178] Visa. VISA. `https://usa.visa.com`. Accessed: 2019-05-01.

[179] Marko Vukolić. The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.

[180] Michael Walfish and Andrew J Blumberg. Verifying Computations without ReExecuting Them. *Communications of the ACM*, 58(2):74–84, 2015.

[181] Wikipedia. List of Futures Exchanges. `https://en.wikipedia.org/wiki/List_of_futures_exchanges`. Accessed: 2019-05-01.

[182] Andrew Chi-Chih Yao. Protocols for Secure Computations. In *23rd Symposium on Foundations of Computer Science*, volume 82, pages 160–164, 1982.

[183] ZCash. Parameter Generation. `https://z.cash/technology/paramgen`. Accessed: 2019-05-01.

[184] ZCash. The Design of the Ceremony. `https://z.cash/blog/the-design-of-the-ceremony/`. Accessed: 2019-05-01.

[185] ZCash. ZCash. `https://z.cash/`. Accessed: 2019-05-01.

[186] Ennan Zhai, David Isaac Wolinsky, Ruichuan Chen, Ewa Syta, Chao Teng, and Bryan Ford. AnonRep: Towards Tracking-Resistant Anonymous Reputation. In *13th USENIX Symposium on Networked Systems Design & Implementation*, pages 583–596, 2016.