University of London
Imperial College of Science, Technology and Medicine
Department of Computing

On the Brink of a Second Financial System: Modelling and Mitigating Risk in Decentralised Finance

Lewis J. F. Gudgeon

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of the University of London and
the Diploma of Imperial College, April 2023

Abstract

This thesis focuses on risk and fragility within Decentralised Finance (DeFi). This thesis presents new evidence on the interconnected and fragile nature of DeFi protocols and develops an approach to mitigate risk in DeFi that relies upon redundancy. Within this context, our contributions are threefold.

Firstly, we focus on a subset of DeFi protocols: Protocols for Loanable Funds (PLFs). PLFs use smart contract code to facilitate the intermediation of loanable funds and, in doing so, allow agents to borrow and save programmatically. Within these protocols, interest rate mechanisms seek to equilibrate the supply and demand for funds. After reviewing methodologies used to set interest rates in PLFs and examining how these interest rate rules have changed in response to changes in liquidity, our main contribution is to model the market efficiency and inter-connectedness between protocols.

Second, we make two contributions by focusing on one particular DeFi protocol, MakerDAO's DAI. The first is to examine how governance system design weaknesses could enable an attacker to take complete control of the protocol. We present a novel strategy utilising *flash loans* that enables the execution of a governance attack in just *two transactions* without locking any assets. Second, we develop a stress-testing framework for a stylised DeFi lending protocol, focusing on the impact of a *drying-up* of liquidity on protocol solvency.

Our third contribution is to develop an approach to minimising the frequency and severity of exploits in DeFi attacks. The idea is to implement a program logic *more than once*, ideally using *different* programming languages. Then, for each implementation, the results should *match* before allowing the state of the blockchain to change. We provide a novel algorithm for implementing dissimilar redundancy for smart contracts.

Taking these contributions together, this thesis presents new methods for modelling and measuring financial risk in DeFi, and — focussing on smart contract risk alone — develops an approach to mitigating it.

Copyright

Acknowledgements

I would like to express my sincerest thanks to:

- My supervisor, Professor William Knottenbelt for his unbounded enthusiasm, optimism and guidance throughout this PhD. This PhD would not have been possible without his support. Like so many, I feel incredibly fortunate to have had such an empowering, supportive and energetic figure so closely involved in my work.
- The Engineering and Physical Sciences Research Council (EPSRC) for awarding me a studentship, making this PhD financially possible.
- My collaborators Arthur Gervais, Dominik Harz, Rami Khalil, Ariah Klages-Mundt, William Knottenbelt, Jun-You Liu, Benjamin Livshits, Toshiko Matsui, Patrick McCorry, Andreea Minca, Pedro Moreno-Sanchez, Daniel Perez, Sam Werner, Stefanie Roos, Tammy Yang and Alexei Zamyatin.
- The Department of Computing and particularly Dr Amani El-Kholy for exceptional support, not least pastoral support, throughout the PhD.
- My parents, Lisa and Jonathan, who, for my entire life and at significant personal sacrifice have sought to ensure that my brother Oli and I have had access to the best education possible.
- My partner Capucine, for far more than I will express here, and for acting as the supporter-of-last-resort for me with this thesis during moments of turbulence.

Dedication

To my grandparents, Diana and Roy, Mario and Toni for their unfailing encouragement, senses of humour, and openness to the world.

‘Le vrai courage, c’est la combinaison de la confiance, de la vulnérabilité et de l’audace.’

Nicolas Riom

Contents

Abstract **i**

Acknowledgements **v**

1 Introduction **1**

1.1 Motivation 1

1.2 Contributions 2

1.3 Statement of originality 4

1.4 Publications 4

2 DeFi: Background **6**

2.1 What is DeFi? 6

2.2 Primitives 8

2.2.1 Blockchain 8

2.2.2 Smart contracts 8

2.2.3 Oracles 9

2.2.4 Governance 11

2.3 Protocol types 12

2.3.1 Decentralized exchanges . . . 12
2.3.2 Protocols for loanable funds . . . 13
2.3.3 Stablecoins . . . 14
2.4 The emergence of DeFi . . . 14
2.5 Risk, bugs and hacks . . . 15

3 Protocols for Loanable Funds 16

3.1 Introduction . . . 16
3.2 Protocols for loanable funds . . . 18
3.2.1 Comparison to traditional lending . . . 18
3.2.2 Use cases . . . 19
3.2.3 Design space dimensions . . . 19
3.2.4 Maker . . . 21
3.3 Interest rate models . . . 22
3.3.1 Model one: linear rates . . . 23
3.3.2 Model two: non-linear rates . . . 24
3.3.3 Model three: kinked rates . . . 24
3.3.4 Making rates stable . . . 25
3.3.5 Summary . . . 26
3.4 Market liquidity . . . 27
3.4.1 Liquidity and illiquidity across PLFs . . . 27
3.4.2 Case Study: DAI on Compound . . . 32
3.4.3 Summary . . . 38

3.5 Market efficiency . . . 38
3.5.1 Uncovered interest parity . . . 39
3.5.2 UIP in a PLF . . . 40
3.5.3 Empirical approach . . . 40
3.5.4 UIP regression results — borrowing rates . . . 41
3.5.5 UIP regression results — saving rates . . . 44
3.5.6 Summary . . . 45
3.6 Market dependence . . . 46
3.6.1 Vector error correction models . . . 46
3.6.2 Results . . . 47
3.6.3 Robustness checks . . . 54
3.6.4 Summary . . . 57
3.7 Related work . . . 58
3.8 Conclusion . . . 59

4 Decentralized Financial Crisis 60

4.1 Introduction . . . 60
4.2 Governance attack on Maker . . . 63
4.2.1 Disclosure to Maker . . . 63
4.2.2 Background and threat model . . . 64
4.2.3 Crowdfunding and flash loans . . . 65
4.2.4 Practical attack viability . . . 67
4.2.5 The attacks . . . 68

4.2.6 Profitability analysis 70
4.3 DeFi lending protocols 71
4.3.1 DeFi lending protocol model 71
4.3.2 Economic security constraints 72
4.4 Stress-testing DeFi lending 75
4.4.1 Stress-testing framework 75
4.4.2 Simulation approach 76
4.4.3 Simulation results 79
4.5 Composability and contagion 85
4.6 Related work 90
4.7 Conclusions 90

5 Dissimilar Redundancy 92

5.1 Introduction 92
5.2 Methodology 94
5.2.1 Overview 94
5.2.2 Technical challenges 96
5.2.3 Implementation 99
5.3 Evaluation 100
5.3.1 Development-time testing 100
5.3.2 Real-world deployment 103
5.4 Related work 104
5.4.1 Ethereum clients 104

5.4.2 Application to bug bounties 105

5.4.3 Differential fuzzing 105

5.5 Conclusion 106

6 Conclusion **107**

6.1 Summary of thesis achievements 107

6.2 Looking toward the future 108

A A list of exploits in DeFi **110**

A.1 Exploits 110

Bibliography **116**

List of Tables

3.1 Comparison of different protocols for loanable funds. 21

3.2 Median of the total supply of loanable funds in USD for the largest markets on Compound, Aave and dYdX, since each market's inception until 7 May 2020. . . 27

3.3 Interest rate model and parameter changes for the cDAI contract since 17 December 2019 (before this date, an earlier variation of the interest rate model —'Jump Rate Model'—was in force since 23 November 2019; we omit this period for clarity of exposition.). 34

3.4 Table of UIP results for daily frequency data, using borrowing rates. Using Newey-West heteroscedasticity and autocorrelation robust standard errors (reported in parentheses.) 42

3.5 Table of UIP results for weekly frequency data, using borrowing rates. Using Newey-West heteroscedasticity and autocorrelation robust standard errors (reported in parentheses.) 43

3.6 Table of UIP results for daily frequency data, using saving rates. Using Newey-West heteroscedasticity and autocorrelation robust standard errors (reported in parentheses.) 44

3.7 Table of UIP results for weekly frequency data, using saving rates. Using Newey-West heteroscedasticity and autocorrelation robust standard errors (reported in parentheses.) 45

3.8 Vector Error Correction Model Results — DAI. 49

3.9 Vector Error Correction Model Results — USDC. 51

4.1 Parameters of DeFi lending platforms, comprising 93% of DeFi market as of April 15th, 2020. 76

4.2 Financial damage for an initial failure of the MakerDAO protocol either through a price crash or a governance attack. 87

4.3 Assuming DAI:USD peg maintained at 1:1, total USD liquidity on all DAI pairs by marketplace. Rounded to the nearest thousand. 88

5.1 Contracts deployed on Polygon mainnet. “Sol” are Solidity contracts. “Vy” are Vyper contracts. “Proxied” are contracts used by the proxy. “Standalone” are contracts interacted with directly. 103

5.2 Function cost 103

A.1 Funds lost in protocol hacks. Source: DeFiLlama, https://defillama.com/hacks . 110

List of Figures

2.1 CeFi vs DeFi. 7

2.2 Breakdown of DeFi protocol type by chain and protocol type. Note that chains smaller than 800m USD have been aggregated into the 'Smaller chains' category for readability. Protocol categories smaller than 500m have also been aggregated. Data from DeFiLama https://defillama.com/, a snapshot taken on 11 April 2023. 10

2.3 The Curve protocol governance dashboard, available at https://dao.curve.fi/dao 12

2.4 Total Value Locked (TVL) in all DeFi projects across all blockchains. Data from DeFiLama https://defillama.com/, a snapshot taken on 11 April 2023. 15

3.1 Average utilisation and borrowing interest rates for all markets on Aave, Compound and dYdX. These plots are of values stored on-chain. 28

3.2 Total funds borrowed and supplied (i.e. liquidity) for ETH markets on dYdX, Compound and Aave. 29

3.3 Total funds borrowed and supplied (i.e. liquidity) for DAI markets on dYdX, Compound and Aave. Periods where utilisation was between 80% and 90%, are highlighted in salmon, while utilisation higher than 90% is shaded in red. 29

3.4 Total funds borrowed and supplied (i.e. liquidity) for USDC markets on dYdX, Compound and Aave. Periods where utilisation was between 80% and 90%, are highlighted in salmon, while utilisation higher than 90% is shaded in red. 30

3.5 Utilisation and borrow rates for DAI on Aave (top), dYdX (middle) and Compound (bottom). Periods in which utilisation equalled or exceeded 100% are highlighted in red. 31

3.7 Three interest rate regimes in Compound. 35

3.10 Borrowing rate distribution around kink, 21 February — 13 March. 38

3.11 Daily borrowing interest rates on DAI across protocols. 48

3.12 Daily borrowing interest rates on USDC across protocols. 48

3.13 Impulse Response Function: impact of a shock (at the beginning) to Compound's DAI borrow rate on Aave and dYdX's DAI borrow rate. 52

3.14 Impulse Response Function: impact of a shock to Compound's USDC borrow rate on Aave and dYdX's USDC borrow rates. 53

3.15 DAI cointegrating equation 1. 54

3.16 DAI cointegrating equation 2. 55

3.17 USDC cointegrating equation 1 55

3.18 USDC cointegrating equation 2. 56

3.19 DAI cointegrating equations misspecification test. 56

3.20 USDC cointegrating equations misspecification test. 57

4.1 Twitter poll for of a crowdfunding attack on MKR governance. 66

4.2 Evolution of the number of MKR tokens staked on different executive (i.e., governing) candidate contracts. We observe that at times the MKR amount of the executive contract dropped below 50k MKR. 67

4.3 Daily traded volume of MKR tokens between 2020-01-30 and 2020-02-08. 68

4.4 Example flash loan attack against Maker. All steps can be executed within one transaction, assuming the flash loan pool and DEX have sufficient liquidity. To execute the attack, the adversary would not need upfront capital besides the gas fees (estimated to amount to c. 15 USD). 69

4.5 Close prices for ETH/USD over the period January 1st, 2018 to February 7th, 2020. 77

4.6 Broadly symmetric log returns for ETH/USD over the period January 1st, 2018 to February 7th, 2020. 78

4.7 Monte Carlo forecast of ETH prices over the next 100 days from February 7th, 2020. 80

4.8 Monte Carlo forecast of the reserve asset price over the next 100 days from February 7th, 2020. 80

4.9 For the simulation yielding the fastest undercollateralisation event, the co-evolution of the ETH and reserve asset prices where the asset price returns are strongly positively correlated. 81

4.10 A DeFi lending protocol is experiencing a sharp decline in its collateral and reserve assets price. Panels correspond to four different levels of system debt, with each panel showing the evolution of the collateral margin (solid lines) and the total debt outstanding (dashed lines). Each panel also shows the consequences of different liquidity parameters. The margin becomes negative in panels 3 and 4—entering the red region below zero—the situation in which the lending protocol has become undercollateralised. 82

4.11 A DeFi lending protocol is experiencing a sharp decline in the price of its collateral and reserve assets, where the assets correlate 0.1. Panels correspond to 4 different levels of system debt, with each panel showing the evolution of the collateral margin and the total debt outstanding. Each panel also shows the consequences of different liquidity parameters. 83

4.12 A DeFi lending protocol is experiencing a sharp decline in the price of its collateral asset with a negatively correlated reserve asset, where the assets correlate -0.9. Panels correspond to 4 different levels of system debt, with each panel showing the evolution of the collateral margin and the total debt outstanding. Each panel also shows the consequences of different liquidity parameters. 84

4.13 For the simulation yielding the fastest undercollateralisation event, the co-evolution of the ETH and reserve asset prices where the asset price returns are strongly negatively correlated. 84

4.14 Number of days before the collateral margin becomes negative, depending on the amount of system debt and the initial amount of ETH that can be sold within 24 hours. 85

4.15 Both the price crash and the governance attack lead to an under-collateralisation of a DeFi lending protocol. In turn, agents will try to exit the protocol by buying other assets (liquidity sweeping). Any agent in a protocol using the now under-collateralised asset as debt will face the same situation and, in turn, try to exit by buying other assets (evaporating collateral). 86

4.16 Financial losses for 30 DeFi protocols which use the debt asset as collateral. “O/C” stands for overcollateralisation. 89

5.1 Overview of our dissimilar redundancy framework 95

5.2 Encoding for registering checks. C is the number of arguments. The three types of arguments are shown. Type 0 represents a static argument, and N is the length of the static content. Type 1 represents the call data argument where O and L are the offset and length to retrieve from call data. Type 2 represents the environment argument, and T is the type of environment variable (e.g. msg.sender) to use 98

5.3 Example checks registration for an ERC-20 token transferFrom function 99

5.4 Testing code using dissimilar redundancy 101

5.5 Failing tests when fuzzing the auction contract with a correct and an incorrect implementation 102

Chapter 1

Introduction

1.1 Motivation

The 2007/08 Global Financial Crisis was the most severe financial crisis since the Great Depression of 1929 [Moh09]. While the precipitating factors were several, chief among them was a lack of transparency in the global financial system. In the US subprime mortgage market, mortgages were securitised into Mortgage Backed Securities. The securitisation process made it difficult for credit rating agencies to ascertain the actual level of financial risk and exposure by financial institutions [CGM09]. Securitised pools of loans that leveraged complex methods of pooling and tranching various underlying instruments were at the centre of the meltdown [CGM09]. In addition, parallel to complex securitisation methods was the rapid spread of an intermediation model known as 'originate to distribute', whereby originates loans were immediately resold to other investors, leading to an obfuscation of the underlying credit quality. As a result, counterparty risks accumulated with a few key players [CGM09]. Of note is financial technology's role in the Global Financial Crisis [For08]. From the advent of Collateralised Debt Obligations one year before the crisis, financial technology enabled the pooling and tranching of credit assets into complex structured products [For08]. Such products were challenging for credit rating agencies to correctly ascertain the inherent financial risk: modelling methodologies needed to be stronger, and there needed to be more due diligence regarding the quality of collateral pools underlying rated securities [For08].

While financial system opacity was one of the principal causes of the Global Financial Crisis, another was misbehaviour and due diligence failures. Investors did not sufficiently examine the assets underlying structured investments, overlooking leverage and tail risks [For08]. Incentive distortions were pervasive, with compensation schemes in financial institutions encouraging disproportionate short-term risk-taking. While technology was a precipitating factor in the Global Financial Crisis, the primary motivation of this thesis is to consider whether technology can help prevent such crises. Bitcoin was released as a direct response to the twin problems of financial opacity and misbehaviour of banks [Nak08]. Bitcoin sought to create a transparent financial mechanism that did not require trusting third parties. Fifteen years later, Bitcoin has been followed by an explosion of interest in the possibility of blockchain technology revolutionising finance. The central motivation for this thesis is to contribute to the evolution of this technology, with a particular focus on risk.

The intended audience for this thesis is one with a pre-existing understanding of the fundamentals of blockchain technology who, primarily, wish to understand in detail aspects relating to financial risk in blockchain systems.

1.2 Contributions

Protocols for Loanable Funds.

Chapter 3 coins the phrase Protocols for Loanable Funds (PLFs) and provides a taxonomy of the interest rate models currently employed by PLFs, resulting in three categories: linear, non-linear and kinked rates. We collect and analyse data on interest rates, utilisation and the total funds borrowed and supplied on three of the largest PLFs, and make the dataset publicly available. We present the first liquidity study of the markets for DAI, ETH and USDC across these PLFs, finding that periods of illiquidity are common, often shared between protocols and that liquidity reserves can be very unbalanced, with in some cases as few as *three* accounts controlling c. 50% of the total liquidity. We also find that realised interest rates tend to cluster around the kink of a kinked interest rate model. Investigating the largest PLF, Compound, we find that the no-arbitrage condition of Uncovered Interest Parity typically does *not* hold, suggesting that markets associated with these protocols may be relatively inefficient and agents may not be optimally reacting to interest rate incentives. We examine the market dependence between PLFs and find that the borrowing interest rates exhibit some interdependence, with Compound appearing to influence borrowing rates on other, smaller PLFs.

Exposition of the Prospect of a Decentralised Financial Crisis.

In chapter 4, we examine the prospect of a decentralised financial crisis. We first examine one specific potential trigger of a decentralised financial crisis: a governance attack on the MakerDAO protocol. We show how, before Maker implemented a parameter change, it was feasible to successfully steal the funds locked in the protocol covertly and within two blocks or two transactions. By exploiting recent flash loan pool contracts, we show how an attacker with no capital (besides gas fees) could have executed such an attack if the flash loan pools provided sufficient liquidity (which they did not at the time of writing). We then turn to the formal modelling of DeFi lending protocols, providing definitions for economically-resilient DeFi lending protocols and introducing overcollateralisation, liquidity, and counterparty risk as formal constraints. These definitions formalise financial risk constraints for more than 93% of the funds locked in DeFi lending protocols as of April 15th, 2020 [Pul19a]. We develop a methodology to quantitatively stress-test a DeFi protocol for its financial robustness, inspired by risk assessments performed by central banks in traditional financial systems. We simulate a price crash event with our stress-test methodology to a stylised DeFi lending protocol that resembles the largest DeFi lending protocols to date by volume: Maker, Compound, Aave and dYdX. For plausible parameter ranges, we find that a DeFi lending protocol could become undercollateralised within 19 days.

Dissimilar redundancy for DeFi protocols.

In chapter 5, we focus on one aspect of risk in DeFi: smart contract risk. We provide an implementation of a protocol for dissimilar redundancy for a DeFi protocol[1]. The idea is to provide a mechanism to catch smart contract coding bugs using multiple implementations of the same smart contract program. We evaluate the protocol on a smart contract auction system implemented in both Solidity and Vyper, verify that a fuzzing approach would be able to detect

[1]https://github.com/danhper/smart-contract-dissimilar-redundancy

purposefully introduced bugs, and provide the costs in USD of using a protocol for dissimilar redundancy.

1.3 Statement of originality

I hereby declare that this thesis represents my own original work, as well as joint work with co-authors of the included publications. No work included in this thesis has been submitted for any other degree or qualification, neither by myself or my co-authors. All sources used in this research have been duly acknowledged and referenced. This thesis is the product of my independent research and represents a significant contribution to the field of distributed ledgers security.

1.4 Publications

The following published papers form the foundation for this thesis.

- Lewis Gudgeon, Sam M. Werner, Daniel Perez, and William J. Knottenbelt. DeFi Protocols for Loanable Funds: Interest Rates, Liquidity and Market Efficiency. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, page 92–112, 2020 (Chapter 3)

- Lewis Gudgeon, Daniel Perez, Dominik Harz, Benjamin Livshits, and Arthur Gervais. The Decentralized Financial Crisis. In *2020 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 1–15. IEEE, 2020 (Chapter 4)

- Daniel Perez and Lewis Gudgeon. Dissimilar redundancy in defi. In *Mathematical Research for Blockchain Economy: 3rd International Conference MARBLE 2022, Vilamoura, Portugal*, pages 109–125. Springer, 2023 (Chapter 5)

In addition, the following papers include work by the author and are incorporated into the thesis where explicitly indicated.

- Dominik Harz, Lewis Gudgeon, Arthur Gervais, and William J Knottenbelt. Balance: Dynamic adjustment of cryptocurrency deposits. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1485–1502, 2019
- Dominik Harz, Lewis Gudgeon, Rami Khalil, and Alexei Zamyatin. Promise: Leveraging future gains for collateral reduction. In *Mathematical Research for Blockchain Economy*, pages 143–160. Springer, Cham, 2020
- Toshiko Matsui and Lewis Gudgeon. The speculative (in) efficiency of the cme bitcoin futures market. In *Mathematical Research for Blockchain Economy*, pages 91–103. Springer, Cham, 2020
- Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 201–226. Springer, Cham, 2020
- Ariah Klages-Mundt, Dominik Harz, Lewis Gudgeon, Jun-You Liu, and Andreea Minca. Stablecoins 2.0: Economic foundations and risk-based models. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 59–79, 2020
- Sam M Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J Knottenbelt. Sok: Decentralized finance (defi). *arXiv preprint arXiv:2101.08778*, 2021

Chapter 2

DeFi: Background

2.1 What is DeFi?

To understand Decentralised Finance (DeFi), we must start by stating what a financial system is. A financial system is fundamentally concerned with apparatus that enables the exchange of funds between financial market participants, such as investors, lenders and borrowers [OS03]. Financial systems can operate at various scales, from the regional such as the Brixton Pound [Pou23] to the global. The core elements of a financial system are as follows:

Financial institutions: provide financial services to individuals and businesses. Examples include banks, credit unions, investment firms and insurance companies.

Financial instruments: assets that can be traded on financial markets. In traditional finance, these are commonly stocks, bonds and currencies.

Financial markets: places where financial instruments are bought and sold. In traditional finance, examples include stock exchanges, bond markets, commodity markets and foreign-exchange markets.

Payment and Settlement systems: used to facilitate funds transfer between parties (individuals and businesses). Traditional finance methods include wire transfers and card payment networks such as VISA and Mastercard.

Regulatory bodies: responsible for overseeing and enforcing laws and regulations that govern financial institutions and markets. In the UK, the financial regulators are the Financial Conduct Authority (FCA) and the Competition and Markets Authority (CMA).

Central Bank: typically national entities responsible for managing national monetary policy. In the UK, this is the Bank of England.

DeFi is an emerging, blockchain-based financial system that is an evolution of each of these components of a financial system. The following will be used as a working definition of DeFi.

Definition 2.1 (Decentralized Finance)**.** *A financial system of smart contract protocols supervenient on decentralised blockchains that is:*

1. *Non-custodial: participants always have full control over their funds*
2. *Permissionless: access to financial services cannot be blocked by third parties*
3. *Transparent: the system should be openly-auditable by anyone*
4. *Composable: financial services should be able to be arbitrarily composed, that is, connected like "Lego".*

The project of DeFi is to take the core elements of a financial system and construct them per Definition 2.1. Table 2.1 shows how DeFi implements a financial system's functionality.

The following sections in this chapter provide a concise summary of essential background concepts relied upon in later chapters. For a complete overview of DeFi, we refer the reader to [WPG$^+$21].

Component	**Traditional Finance (examples)**	**DeFi**
Financial institutions	Banks, insurance companies	Smart contracts
Financial instruments	Stocks, bonds and currencies	Smart contracts
Financial markets	Stock exchanges, bond markets	Smart contracts
Payment and Settlement systems	Wire transfers, card payment networks	On-chain transactions
Regulatory bodies	Financial Conduct Authority, Securities and Exchange Commission	On-chain/off-chain hybrid
Central Bank	Bank of England, Federal Reserve	Smart contracts

Figure 2.1: CeFi vs DeFi.

2.2 Primitives

2.2.1 Blockchain

A DeFi protocol supervenes on a layer-one blockchain, which we assume provides standard ledger functionality [BMTZ17, BGK+18, DGKR18, PSS17]. It is outside the scope of this thesis to provide a complete exposition of a generic layer-one blockchain. Instead, we provide the following list of properties we assume the blockchain underlying a DeFi protocol to have.

- We assume that there exists a transaction-based state machine [W+14]
- The underlying blockchain can provide finality [CL99, MXC+16], construed as a guarantee that once committed to the blockchain, a transaction cannot be modified or reversed
- The blockchain offers sufficient protection from selfish mining attacks and that fewer than $\frac{1}{3}$ of the miners are malicious [ES14]
- The blockchain allows its users to run psuedo-Turing complete programs on its distributed infrastructure. These programs must be strictly deterministic: given the same inputs, the same blockchain state should always obtain.
- Blockchain programs should be able to communicate with each other and affect each other's state
- Blockchain state changes should be atomic: they either occur in their entirety or fail in their entirety.

2.2.2 Smart contracts

At the heart of DeFi are computer programs intended to execute automatically without reliance on any particular third party to ensure that they do so. These are called smart contracts, and many layer-one blockchains support their operation [W+14, Bin23, Car23, Pro23b, Ava23, Tro23, Pro23a]. These programs are typically written in a Turing complete programming language such as Solidity [Fou20] (i.e., roughly, can implement any arbitrarily complex program logic).

The Ethereum Virtual Machine (EVM) is the runtime environment [Fou20] that enables smart contracts to run in Ethereum. Central to this is the world state, which is a mapping between 160-bit addresses and account states [W+14]. In Ethereum, this mapping from addresses is not stored on the blockchain but instead maintained in a modified Merkle Patricia tree, stored in a backend that maintains a mapping of byte arrays to byte arrays. The account state is comprised of a nonce (scalar balance equal to the number of transactions sent from the address), balance, storage root (a 256-bit hash of the root node of a Merkle Patricia tree that encodes the storage contents of the account) and a code hash (the hash of the EVM code of the account). The latter field is particularly pertinent to smart contracts as this code gets executed if the address receives a message call.

Smart contracts communicate with each other via transactions that are constructed and initiated by an actor external to Ethereum [W+14]. There are two types of transaction: those which result in a message call and those which cause a new account to be created. It is the message-call transaction type that is necessary for smart contract communication. A message-call transaction consists of an array of unlimited size that specifies the input data of a message call.

To provide a suitable basis for DeFi, smart contracts must [WPG+21]:

- Be expressive enough programs to encode the rules of a financial application.
- Provide for conditional operation and iteration (with bounds).

Coupled with the requirements on a layer-one blockchain above, these conditions jointly provide a suitable medium on which a DeFi protocol can execute.

2.2.3 Oracles

A crucial property of the programming environment offered by the EVM is that the virtual machine is wholly sandboxed and isolated: the EVM has no access to network, filesystem or other computer processes [Fou20]. This means a smart contract program deployed in the EVM has no intrinsic way of communicating with the outside world. To communicate with the outside world, a mechanism is needed to import off-chain information into the EVM execution context.

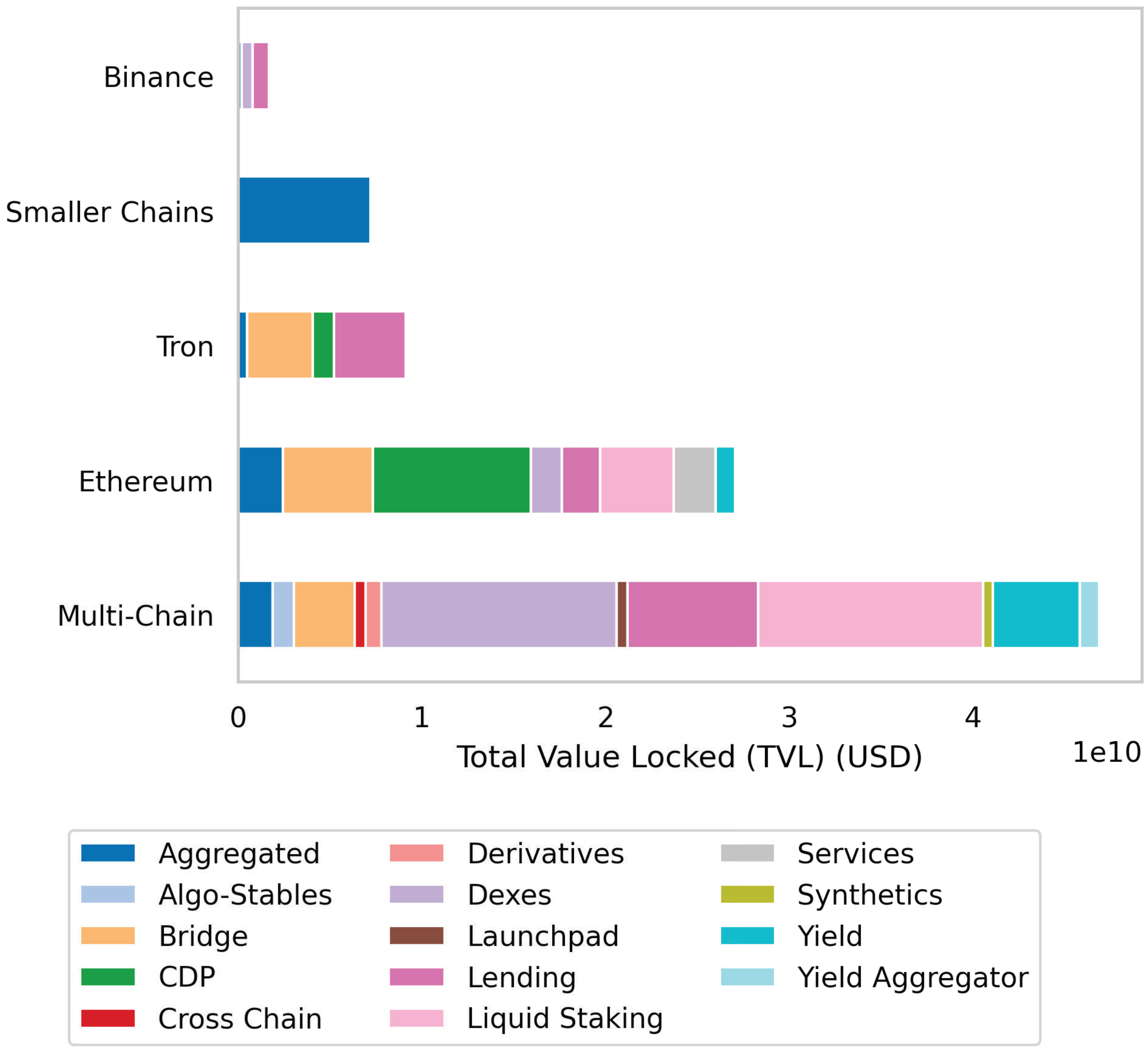


Figure 2.2: Breakdown of DeFi protocol type by chain and protocol type. Note that chains smaller than 800m USD have been aggregated into the 'Smaller chains' category for readability. Protocol categories smaller than 500m have also been aggregated. Data from DeFiLama https://defillama.com/, a snapshot taken on 11 April 2023.

By construction, the correctness of the imported information is not verifiable on-chain, so DeFi protocols need to rely on incentive mechanisms to ensure that oracles accurately and honestly report prices.

We can distinguish between three different types of information in the context of a smart contract powered DeFi [WPG+21]:

- Off-chain ground truth (e.g. the temperature in Hawaii today)
- On-chain ground truth (e.g. the balance of an on-chain address)
- On-chain estimates of off-chain ground truth (e.g., an important example of this is an oracle reported price, e.g. the price of ETH in USD in an off-chain market)

Given their financial nature, DeFi protocols often rely on on-chain estimates of off-chain prices to function. Thus, oracles are a crucial building block of DeFi in the context of sandboxed execution environments. In the next section, we provide a landscape of the major categories of extant DeFi protocol.

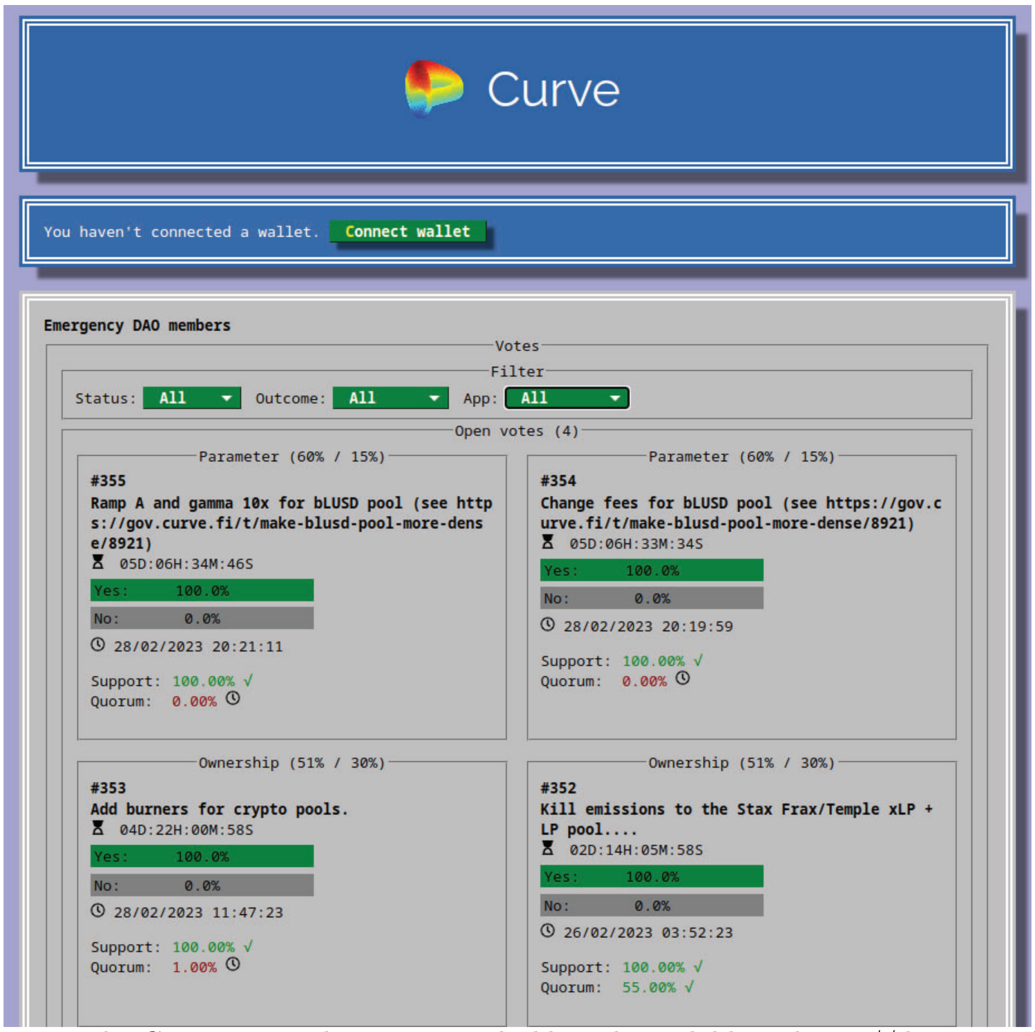
2.2.4 Governance

In traditional finance, governance is typically centralised, with several entities, such as regulators and banks, possessing the power to govern financial institutions, instruments and markets. In DeFi, governance is typically decentralised, with the power to change a DeFi protocol resting with on-chain actors. Governance decisions are commonly achieved via a consensus mechanism, with protocol stakeholders voting on proposals using tokens.

At least in theory, DeFi governance can be more transparent than traditional finance, with all information regarding the governance process being publicly accessible. Typically, a governance proposal is made and voted on for a change in system parameters to be effected.

Figure 2.3 demonstrates a typical on-chain governance dashboard.

The governing community of a DeFi protocol is commonly called a Decentralized Autonomous Organisation (DAO). A DAO comprises a set of individuals and is then governed by rules


Curve

You haven't connected a wallet. Connect wallet

Emergency DAO members

Votes

Filter

Status: All Outcome: All App: All

Open votes (4)

Parameter (60% / 15%)

#355
Ramp A and gamma 10x for bLUSD pool (see https://gov.curve.fi/t/make-blusd-pool-more-dense/8921)
05D:06H:34M:46S
Yes: 100.0%
No: 0.0%
28/02/2023 20:21:11
Support: 100.00% ✓
Quorum: 0.00%

Parameter (60% / 15%)

#354
Change fees for bLUSD pool (see https://gov.curve.fi/t/make-blusd-pool-more-dense/8921)
05D:06H:33M:34S
Yes: 100.0%
No: 0.0%
28/02/2023 20:19:59
Support: 100.00% ✓
Quorum: 0.00%

Ownership (51% / 30%)

#353
Add burners for crypto pools.
04D:22H:00M:58S
Yes: 100.0%
No: 0.0%
28/02/2023 11:47:23
Support: 100.00% ✓
Quorum: 1.00%

Ownership (51% / 30%)

#352
Kill emissions to the Stax Frax/Temple xLP + LP pool....
02D:14H:05M:58S
Yes: 100.0%
No: 0.0%
26/02/2023 03:52:23
Support: 100.00% ✓
Quorum: 55.00% ✓

Figure 2.3: The Curve protocol governance dashboard, available at https://dao.curve.fi/dao

encoded in smart contracts. DAO members then participate in decision-making by voting on proposals using tokens or other means.

2.3 Protocol types

2.3.1 Decentralized exchanges

In traditional finance, asset exchange typically requires a third party to facilitate it. Decentralised exchanges (DEXs) are on-chain protocols that enable on-chain assets to be swapped natively on-chain without the involvement of a third party [Ind, LBC+19]. All trades are settled on-chain and, as such, are openly auditable.

There are two principal types of DEX. The first is an orderbook-DEX, which emulates the order-book means of asset exchange in traditional finance: users submit a bid and ask quotes for a particular asset, and these are then matched on-chain. The second construct is new: that of an Automated Market Maker (AMM). An AMM defines the price of one asset in terms of another in accordance with a rule, using a peer-to-pool method [XPCF23]. For instance, the rule might be that the sum of two asset amounts must remain equal (Constant Sum Market Maker): $k = x + y$, where k is held constant and x and y refer to the asset amounts. Alternatively, another common AMM rule is that in a given asset pool, the product of the two asset amounts must remain constant, such as $k = xy$. In such a construction, a trade is not allowed to change the value of the product of the tokens x and y. For more information about AMMs, we refer the reader to [XPCF23].

2.3.2 Protocols for loanable funds

Protocols for Loanable Funds (PLFs) intermediate markets for loanable funds, with suppliers of funds earning interest. The term PLF was coined in [GWPK20] by the author. As mentioned above, protocols must protect against borrowers defaulting on debt obligations. Where loans need to be valid for more than a single transaction, this protection is currently achieved by requiring borrowers to *over-collateralise* their loans, allowing the lender to redeem the pledged collateral should a borrower default on a position[1]. Where the loan must be valid only for a single transaction, *flash loans* enable agents to borrow without collateral, whereby the atomicity afforded by smart contracts protects the loaned amount. The transaction is reversed if the loan is not repaid with interest.

In the context of lending protocols, a borrower defaults on a loan when the value of the locked collateral drops below some fixed liquidation threshold. The liquidation thresholds vary between asset markets across different protocols. In the event of default, the lending protocol seizes and liquidates the locked collateral at a discount to cover the underlying debt. Additionally, a penalty fee is charged against the debt before paying out the remaining collateral to the borrower.

[1]Therefore loans of this type on DeFi lending protocols are instances of secured loans, where an agent can only borrow against collateral they already own; they cannot enter into 'net debt'. We address this further in Section 3.2.

2.3.3 Stablecoins

For a crypto asset to be a viable medium of exchange and store of value, price stability needs to be guaranteed. *Stablecoins* are crypto assets with a price stabilisation mechanism to maintain some target peg. Here we briefly outline two of the most widely used stabilisation mechanisms [MSS20]:

Fiat-collateralized. Each unit of stablecoin is pegged to some fixed amount of fiat currency (typically USD). This is generally realised via a network of banks maintaining the fiat collateral and is not decentralised. Stablecoins such as USDT [Lim16] and USDC [Cir20] belong in this category.

Cryptoasset-collateralized. An amount of some other crypto asset backs each unit of the stablecoin. A stabilisation mechanism is needed to protect against the volatility of the collateral. The most prominent of such stablecoins is DAI [Mak]. To borrow newly minted units of DAI, where one DAI is pegged to 1 USD, a user has to pledge an over-collateralised amount of cryptocurrency (e.g. ETH), which becomes locked up in a smart contract. In case the price of DAI deviates from its peg, arbitrageurs are incentivised to buy or sell DAI should the price drop below or rise above 1 USD, respectively. A borrower of DAI must keep the associated collateralisation ratio above some liquidation threshold. Otherwise, the borrow position will be liquidated at a discount, and a penalty fee will be charged against the debt.

2.4 The emergence of DeFi

At the commencement of this thesis, DeFi was in its infancy. In October 2018, there was less than 1m USD locked in DeFi protocols; as of 3 January 2023, there is 39bn USD locked in DeFi protocols. In the intervening period, there was a spike to more than 180bn USD. See Figure 2.4.

Many extensive surveys exist of DeFi protocol types (e.g. [WPG$^+$21]). Figure 2.2 shows the main protocol types across the major blockchains. This thesis only presents background mate-

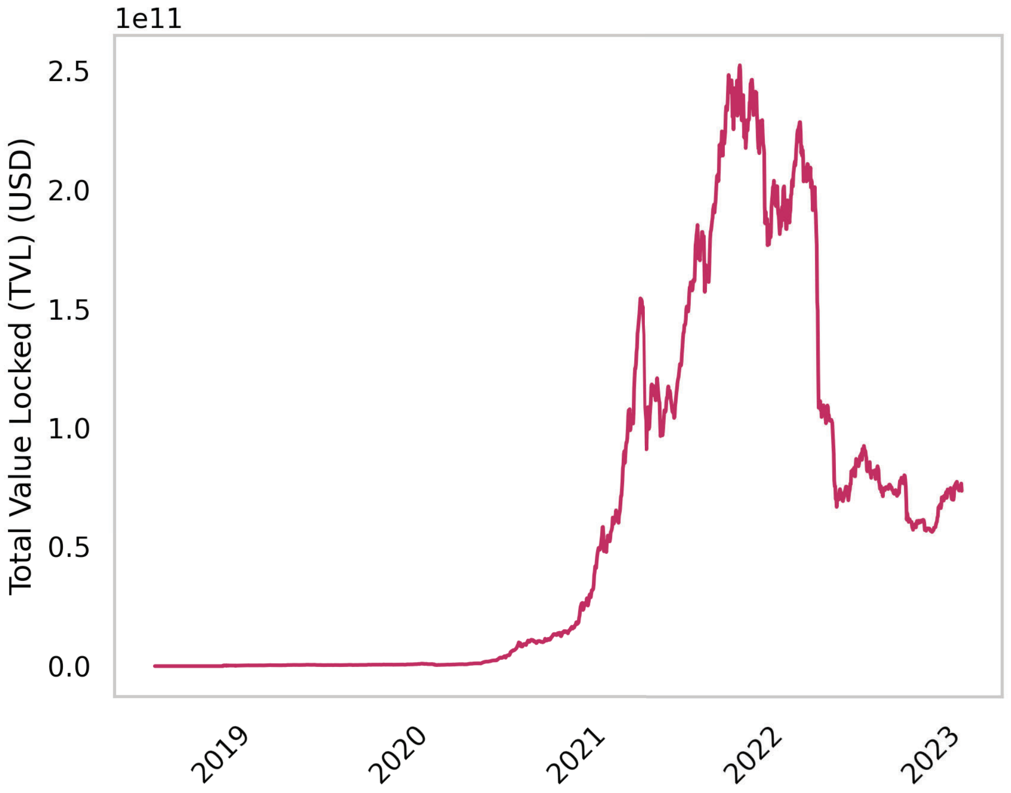


Figure 2.4: Total Value Locked (TVL) in all DeFi projects across all blockchains. Data from DeFiLama https://defillama.com/, a snapshot taken on 11 April 2023.

rial on a subset of these protocol types relevant to the later chapters. We provide an overview of all types in the figure for completeness.

2.5 Risk, bugs and hacks

A notable feature of the DeFi protocol ecosystem is the large number of software exploits that have occurred, often leading to significant financial losses [PL21, ABC17].

Table A.1 in the appendix lists most known hacks to protocols since the industry's inception. It makes clear that the frequency and severity of exploits are immense, with an average loss of 2,531,366 USD per day since mid-2017. There are many different ways in which smart contracts can be exploited, as indicated under the 'Technique' column.

Chapter 3

Protocols for Loanable Funds

3.1 Introduction

The first contribution of this thesis is to examine the development within DeFi of protocols which facilitate programmatic borrowing and saving. Such protocols represent a significant advancement for DeFi due to the importance of these operations to an economy. Markets for loanable funds, a matching market for savers and would-be borrowers, in principle, enable agents to engage in inter-temporal consumption smoothing, whereby agents choose their present and future consumption to maximize their overall welfare [Fis30]. That is, access to loans enables a borrower to consume more today than their income would permit, paying back the loan when their income is higher. On the other hand, savers, for whom income is higher than their present consumption, can deposit their funds and earn interest on them [Rob34, Ohl37].

Here, we term protocols that intermediate funds between users as Protocols for Loanable Funds (PLFs). In doing so, we note such protocols are *not* directly acting as a fully-fledged replacement for banks, not least because traditional banks are not intermediaries of loanable funds. Instead, they provide financing through money creation [JK15] (see Section 3.2). Further, at present PLFs only offer *secured* lending, where agents can only borrow an amount provided they can front at least this amount as collateral. This reflects the trustless setting within which PLFs operate: absent the typical repercussions of reneging on debt commitments in traditional finance, in DeFi, agents could default on their loans without recourse.[1] Therefore, the extent to which PLFs facilitate 'true' borrowing—where an agent gets into a position of net debt—is limited.

In PLFs, interest rates reflect the prevailing *price* of funds resulting from supply and demand. Therefore, the mechanism used to set these rates is a crucial aspect of protocol design: it provides the pre-conditions under which the process of *tatonnement*—or reaching the equilibrium—occurs [Wal87]. In traditional finance, interest rates are primarily set by central banks—via a base rate—and function as a key lever in the management of credit in economies [oE20, Boa20]. Lowering the base rate makes it relatively cheaper to borrow while discouraging saving. In the context of PLFs, the interest rate setting mechanism is decided upon at the protocol level, commonly via a governance process.

In this chapter, we seek to gain insights into how currently-deployed PLFs operate, setting out the interest rate models they employ. Moreover, we seek to characterize the periods of illiquidity—roughly, where most of the funds within a PLF are loaned out and unavailable for withdrawal by their depositors—that these protocols have experienced. We then seek to understand how efficient these protocols are at present, investigating whether the no-arbitrage condition of Uncovered Interest Parity (UIP) holds *within* a particular protocol. The efficiency of the markets indicates the level of financial maturity and the responsiveness of agents to economic incentives. Finally, we look at the interrelation of interest rate markets across protocols, developing a Vector Error Correction Model (VECM) for the dynamics between Compound [Com19], dYdX [dYd20] and Aave [AAV20a] in the markets for the stablecoins DAI and USDC.

Contributions

This chapter makes the following contributions:

- We provide a taxonomy of the interest rate models currently employed by PLFs, resulting in three categories: linear, non-linear and kinked rates.

[1]The enforcement of strong identities, mapping on-chain to real-world identities, would plausibly alter this trade-off.

- We collect and analyze data on interest rates, utilization and the total funds borrowed and supplied on three of the largest PLFs. We have made the dataset publicly available. This dataset is composed of data stored on the Ethereum blockchain.
- We present the first liquidity study of the markets for DAI, ETH and USDC across these PLFs, finding that periods of illiquidity are common, often shared between protocols and that liquidity reserves can be very unbalanced, with in some cases as few as *three* accounts controlling c. 50% of the total liquidity. We also find that realized interest rates tend to cluster around the kink of a kinked interest rate model.
- Investigating the largest PLF, Compound, we find that the no-arbitrage condition of Uncovered Interest Parity typically does *not* hold, suggesting that markets associated with these protocols may be relatively inefficient and agents may not be optimally reacting to interest rate incentives.
- We examine the market dependence between PLFs and find that the borrowing interest rates exhibit some interdependence, with Compound appearing to influence borrowing rates on other, smaller PLFs.

The remainder of this chapter is organized as follows. Section 3.2 outlines the general design of PLFs. Section 3.3 presents a taxonomy of different interest rate models. Sections 3.4, 3.5 and 3.6 provide an analysis of market liquidity, efficiency and dependence, respectively. Section 3.7 discusses related work, before Section 3.8 concludes.

3.2 Protocols for loanable funds

3.2.1 Comparison to traditional lending

PLFs facilitate matching would-be borrowers and lenders with the interest rate set programmatically. Importantly, unlike peer-to-peer lending, funds are pooled such that a lender may lend to many borrowers and vice versa. In so doing, an open lending protocol provides a market for loanable funds. A set of smart contracts has replaced an intermediary's role in traditional finance.

By creating markets for loanable funds—as protocols for loanable funds—such protocols are not functionally equivalent to banks. The construal of banks as primarily intermediaries of loanable funds (ILFs), as in some economic theory, has been debunked (see, e.g. [JK15]). Rather than accepting deposits of pre-existing funds from savers and lending these funds to borrowers, banks primarily provide financing through *money creation*, creating new money when making a loan and constrained by their profitability and solvency requirements [JK15]. Therefore since banks are not primarily ILFs, PLFs are not functional replacements.

3.2.2 Use cases

The introduction of PLFs significantly extends the existing trading capabilities in DeFi, offering several use cases for DeFi actors. Predominantly, PLFs empower decentralised *margin trading* by facilitating *short sells* and *leveraged longs*. Margin is defined as the collateral that an agent gives to a counterparty to cover the credit risk that the agent poses for the counterparty. In a short sell, a trader sells the borrowed funds, seeking to profit by repurchasing the borrowed position at a lower price. Similarly, in a leveraged long, a trader buys another asset with the borrowed funds and profits if the purchased asset appreciates. Because of margin trading, suppliers of loanable funds can earn interest.

A further use case of PLFs lies in borrowers being able to leverage their funds as collateral while maintaining the right to repurchase the collateralised token, thereby not giving up direct ownership.

3.2.3 Design space dimensions

Interest rate model

Suppliers of loanable funds receive interest over time, while borrowers have to pay interest. A key differentiating factor across lending protocols is the chosen interest rate model, which is generally some linear or non-linear function of the available liquidity in a market. As loans on protocols for loanable funds have unlimited maturities, variable interest rates may fluctuate from the opening of a borrowing position. Using variable rate models, lending protocols can dynamically adjust the interest rate depending on the ratio of funds borrowed to supplied. This can prove particularly useful during periods of low liquidity by incentivising borrowers to repay their loans.

Reserve factor

Additionally, lending protocols employ a *reserve factor*, specifying the amount of a borrower's accrued interest to be deducted and set aside for periods of illiquidity. Hence, the interest earned by lenders is a function of the interest paid by borrowers less the reserve factor.

Interest disbursement mechanism

Interest is typically accrued per second and paid out on a per-block basis. Since the repeated payment to lenders of the accrued interest (denoted in the supplied token) would incur undesired transaction costs, accrued interest is often paid out through the use of *interest-bearing derivative tokens*, which are ERC-20 tokens that are minted upon the deposit of funds and burned when redeemed. Each market has such an associated derivative token, which appreciates with respect to the underlying asset at the same rate as interest is compounded, thereby accruing interest for the token holder. Even though loans are made with indefinite maturity, a loan is liquidated should the value of the borrowed asset's underlying collateral fall below a fixed liquidation threshold. In the case of an undercollateralised borrow position, so-called liquidators can purchase the collateral at a discount, and a penalty fee is imposed upon the borrower.

Governance mechanism

A critical component of lending protocols is decentralised governance. Lending protocols achieve decentralised governance through ERC-20 governance tokens specific to the lending protocol, whereby token holders' votes are weighted proportionally to their stake. Token holders are empowered to propose new features and changes to the existing protocol.

Protocol	Interest Rate Model	Stable Interest Rate	Variable Interest Rate	Governance Token	Interest-bearing Derivative Token	Additional Functionalities
Compound	Kinked	✗	✓	✓	✓	–
Aave	Kinked	✓	✓	✓	✓	Swap rates, flash loans
dYdX	Non-linear	✗	✓	✗	✗	Decentralized exchange, flash loans

Table 3.1: Comparison of different protocols for loanable funds.

3.2.4 Maker

Protocol overview. The Maker protocol is an Ethereum-based smart contract platform which backs and stabilises the value of the DAI stablecoin. Maker is fundamentally different to other lending protocols as Maker does not serve as an intermediary of loanable funds by matching borrowers with lenders; instead, DAI can be generated, or "borrowed", at all times. On Maker, DAI[2] can be minted by depositing some Ethereum-based asset as collateral in vaults[3], which are collateral-locking smart contracts specific to the collateral type that track the total debt of a position. As with other lending protocols, borrow positions are indefinite as long as the collateralisation ratio exceeds the liquidation threshold, where the latter depends on the risk profile of a collateral asset and may thus vary across vaults.

Interest rate model. The interest to be paid for borrowing DAI is the *stability fee* and varies between different collateral types. DAI holders can also earn interest from their tokens via the *DAI savings rate* (DSR), which denotes the interest one receives from depositing DAI into a specific savings contract. Unlike the stability fee, which differs between collateral asset types, there is only one DSR. The stability fee and the DSR are temporarily fixed and readjusted through voting via the Maker governance structure over time (see below). In theory, the DSR should be funded by the pool of stability fees paid by borrowers of collateralised DAI. However, the DSR yield is paid out in newly minted DAI, offering a counterparty risk-free investment to DAI holders. Furthermore, not only are the stability fees and DSR set independently from one another, but the pool of stability fees is also used to cover losses in the case of a collateral auction's yield not covering the associated debt. Hence, there could be periods during which the sum of stability fees is insufficient to cover the DSR returns. MKR tokens are minted and sold for DAI to compensate for this imposed system debt. This suggests that MKR token holders are the predominant Maker participants exposed to DSR risk through the depreciation

[2]Note that when we speak of DAI, we are strictly referring to multi-collateral DAI unless stated otherwise.
[3]Before multi-collateral DAI, these were termed collateralised debt positions (CDPs)

of their tokens. Ultimately, Maker uses both the stability fee and the DSR as its levers for price stabilisation, balancing the supply and demand of DAI.

Governance. Maker uses the MKR governance token with which token holders can vote on protocol changes (e.g. stability fee of an asset, collateral asset types, DSR) and thereby maintain the financial stability of DAI. There are two general voting mechanisms on maker: governance polls and executive votes, where the latter is specific to voting on changes to the DAI credit system (e.g. risk parameters). Voting takes place through Maker participants staking their MKR tokens and can take place at any point in time, where any participant can make a proposal. For instance, if the value of DAI drops below 1 USD, the DSR could be raised via a so-called executive vote to increase demand for DAI.

3.3 Interest rate models

This section outlines the main classes of interest rate models employed by PLFs. The interest rate model can differ across PLFs and by market within a particular PLF. We also describe an approach that has been taken to enable these variable rate models to offer more interest rate stability.

Definitions. For a market m, total loans L and gross deposits A, we define the utilization of deposited funds in that market as

$$U_m = \frac{L}{A} \tag{3.1}$$

Assuming both L and A are positive and $L < A$, $0 \leq U_m \leq 1$. The Interest Rate Index, I for block k, is calculated each time an interest rate changes, i.e. as users mint, redeem, borrow, repay or liquidate assets. It is given by:

$$I_{k,m} = I_{k-1,m}(1 + rt) \tag{3.2}$$

where r denotes the per-block interest rate and t denotes the difference in block height. Therefore, starting with some debt level in an initial block D_0, debt D in a market is given by

$$D_{k,m} = D_{k-1,m}(1 + rt) \tag{3.3}$$

where a portion of the interest is kept as a reserve (Π), set by reserve factor λ:

$$\Pi_m = \Pi_{k-1,m} + D_{k-1,m}(rt\lambda) \tag{3.4}$$

We now turn to classifying the actual interest rates into three main models.

3.3.1 Model one: linear rates

The first model we present is one in which interest rates are set as a linear function of utilization. With a linear interest rate model, interest rates are determined algorithmically as the equilibrium value in a loanable market m, where the borrowing interest rates i_b are given by:

$$i_{b,m} = \alpha + \beta U_m \tag{3.5}$$

where α is some constant and β is a slope coefficient on the responsiveness of the borrowing interest rate to the utilization rate. Saving interest rates i_s are given by:

$$i_{s,m} = (\alpha + \beta U_m)U_m \tag{3.6}$$

where in essence, the interest rate $i_{b,m}$ is scaled by the utilization to arrive at an interest rate for saving that is lower than that of the rate paid by borrowers. This serves to ensure that the interest rate spread $(i_{b,m} - i_{s,m})$ is positive. Some portion of this spread can be kept for reserves.

3.3.2 Model two: non-linear rates

Interest rates may also be set non-linearly, and here we present the non-linear model employed by dYdX [dYd19]. For a loanable funds market m, the borrowing interest rates i_b follow a non-linear model and are computed as:

$$i_{b,m} = (\alpha \cdot U_m) + (\beta \cdot U_m^{32}) + (\gamma \cdot U_m^{64}) \tag{3.7}$$

The saving interest rates i_s with reserve factor λ are given by:

$$i_{s,m} = (1 - \lambda) \cdot i_{b,m} \cdot U_m \tag{3.8}$$

In comparison to the linear rate model, a non-linear model allows for the interest rate to increase at an increasing rate as the protocol becomes more heavily utilized, creating a non-linearly increasing incentive for suppliers to supply to the protocol and for borrowers to repay their borrows.

3.3.3 Model three: kinked rates

In the final interest rate model, interest rates exhibit some form of kink: they sharply change at some defined threshold. Such interest rates are employed by several protocols, including [Com19, Com20, Aav20d, Aav20e].

Mathematically, kinked interest rates can be characterized as follows.

$$i_b = \begin{cases} \alpha + \beta U & \text{if } U \leq U^* \\ \alpha + \beta U^* + \gamma(U - U^*) & \text{if } U > U^* \end{cases} \tag{3.9}$$

where α denotes a per-block base rate, β denotes a per-block multiplier, U denotes the utilization ratio (with U^* denoting the optimal utilization ratio) and γ denotes a 'jump' multiplier.

In the case of Compound, the associated saving rates are given by equation (3.10).

$$i_s = U(i_b(1 - \lambda)) \tag{3.10}$$

where λ is a reserve factor.

Such models share the property of sharply changing the incentives for borrowers and savers beyond some utilization threshold, as with the non-linear model. However, they also introduce a point of sharp change in the interest rate, beyond which the interest rates start to rise sharply, in contrast to non-linear models with no kink. Therefore it might be expected that this kink would become a Schelling point[4] of convergence among agents [Sch58].

3.3.4 Making rates stable

Some platforms, such as Aave, allow the borrower to *choose* between a variable and a stable interest rate. However, it is essential to note that the "stable" interest rate is not *entirely* stable, as it can be revised if it significantly deviates from the market average. Examining Aave's implementation in detail, we first present their instantiation of a kinked interest rate model before showing how the stable rate is derived[5].

The variable interest rate is based on several parameters defined by the system. Given a particular asset's utilization rate U, the parameter U_{optimal} is the optimal utilization. In practice, this value was set to 0.8 and updated to 0.9 in May 2020 [Fra20a]. Two interest rate slopes, parameters of the system, are used to compute the variable interest rate: R_{slope1} is used when $U < U_{\text{optimal}}$ and R_{slope2} when $U \geq U_{\text{optimal}}$. Finally, given a base variable borrow rate i_{b,m,v_0}, the variable borrow interest rate i_b for market m is computed as follows:

$$i_{b,m,v} = \begin{cases} i_{b,m,v_0} + \frac{U}{U_{\text{optimal}}} \cdot R_{\text{slope1}} & \text{if } U < U_{\text{optimal}} \\ i_{b,m,v_0} + R_{\text{slope1}} + \frac{U - U_{\text{optimal}}}{1 - U_{\text{optimal}}} \cdot R_{\text{slope2}} & \text{if } U \geq U_{\text{optimal}} \end{cases} \tag{3.11}$$

To compute the stable rate, Aave computes the lending protocol-wide market rate m_r as the

[4]Informally, a solution of a coordination game that agents tend to arrive at in the absence of communication, such as two strangers who wish to meet but cannot communicate deciding to meet at noon at the Grand Central Terminal in New York City, since this somehow seems a *natural* choice [Sch58].

[5]These formulae are an adapted version of those that appear in the Aave whitepaper [AAV20a]

arithmetic mean of the total borrowed funds weighted by the borrowing rate $i_{b,m}$ for given platform p as follows:

$$m_r = \frac{\sum_{p=1}^{n} i_{b,m,p} \cdot B_{m,p}}{\sum_{p=1}^{n} B_{m,p}} \tag{3.12}$$

where $B_{m,p}$ denotes the total amount of borrowed funds for market m on lending protocol p. Hence, using the m_r as the base rate, the stable borrowing rate $i_{b,s}$ for a market m is given by:

$$i_{b,m,s} = \begin{cases} m_r + \frac{U}{U_{\text{optimal}}} \cdot R_{\text{slope1}} & \text{if } U < U_{\text{optimal}} \\ m_r + R_{\text{slope1}} + \frac{U - U_{\text{optimal}}}{1 - U_{\text{optimal}}} \cdot R_{\text{slope2}} & \text{if } U \geq U_{\text{optimal}} \end{cases} \tag{3.13}$$

It will be revised if the stable rate deviates too much from the market rate. The stable borrow rate $i_{b,m,s}$ for user z is revised upwards to the most recent stable borrow rate for the respective market when

$$i_{b,m,s,z} < \frac{B_{m,v} \cdot i_{b,m,v} + B_{m,s} \cdot i_{b,m,s}}{B_{m,v} + B_{m,s}} \tag{3.14}$$

If (3.14) holds, a borrower of funds could earn interest from a borrow position. On the contrary, should the stable rate of a borrow position exceed the latest stable rate, it would be adjusted downwards should

$$i_{b,m,s,z} > i_{b,m,s} \cdot (1 + \Delta i_{b,m,s,t}) \tag{3.15}$$

where $\Delta i_{b,m,s,t}$ denotes the change in the stable rate for a specified adjustment window t. Unlike variable interest rate denominated loans, stable rate loans have a definite maturity.

3.3.5 Summary

We have reviewed the three main interest rate models for variable interest rates and explained a mechanism which seeks to bring stability to these rates. An emergent key feature of these models is the incentive they provide to borrowers and savers at times of high utilization. In the next section, this behaviour at high utilization becomes a central object of concern.

3.4 Market liquidity

This section analyses liquidity and interest rates for loanable funds markets on Compound, dYdX and Aave.

3.4.1 Liquidity and illiquidity across PLFs

The total amount of locked loanable funds for the largest markets across Compound, Aave and dYdX are given in Table 3.2.

Currency	Total Amount Locked (median in millions of USD)		
	Compound	Aave	dYdX
(W)ETH	76.58	4.80	19.41
USDC	31.54	4.12	6.58
DAI	24.82	0.95	4.64
SAI	36.94	-	-
USDT	-	3.92	-
BAT	0.95	0.08	-
LEND	-	3.60	-
LINK	-	12.21	-

Table 3.2: Median of the total supply of loanable funds in USD for the largest markets on Compound, Aave and dYdX, since each market's inception until 7 May 2020.

It can be seen that ETH, USDC and DAI account for the majority of loanable funds on all three PLFs.[6] Hence we focus on these markets for an in-depth analysis. From Figure 3.1, it

[6]As single-collateral DAI (SAI) has been replaced by multi-collateral DAI (DAI), we solely focus on the latter for this analysis.

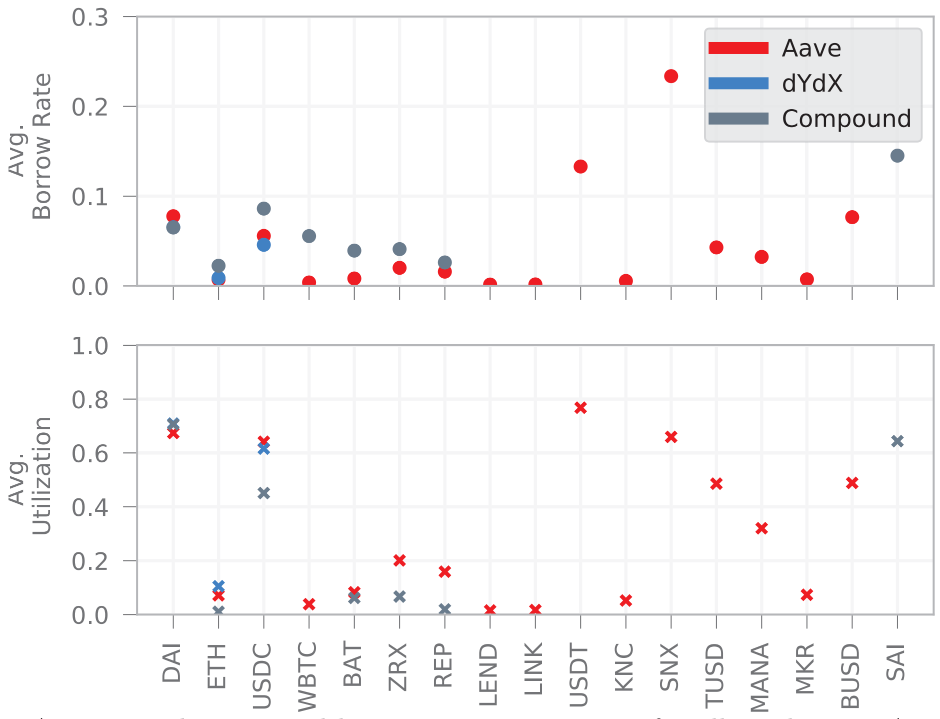


Figure 3.1: Average utilisation and borrowing interest rates for all markets on Aave, Compound and dYdX. These plots are of values stored on-chain.

becomes apparent that these three markets are very similar regarding their average borrow and utilisation rates, particularly for DAI and ETH.

Liquidity

The difference between the total supply and total borrows in the respective market gives the available liquidity for loanable funds for an asset. High liquidity allows actors to borrow funds at lower rates while guaranteeing suppliers of funds that funds can be withdrawn at any point in time. On the one hand, regarding the liquidity for ETH (see Figure 3.2), all three PLFs maintain high liquidity over time, mainly due to the total borrows remaining relatively stable. On the other hand, the markets for DAI and USDC (see Figures 3.3 and 3.4) frequently exhibit periods of much lower liquidity, with utilisation exceeding 80% and 90% respectively. Moreover, such periods of low liquidity are somewhat shared across protocols, particularly for the smaller PLFs dYdX and Aave from January to mid-March 2020.

On Thursday, 12 March 2020—*Black Thursday* [Reu20]—the total amount of locked funds

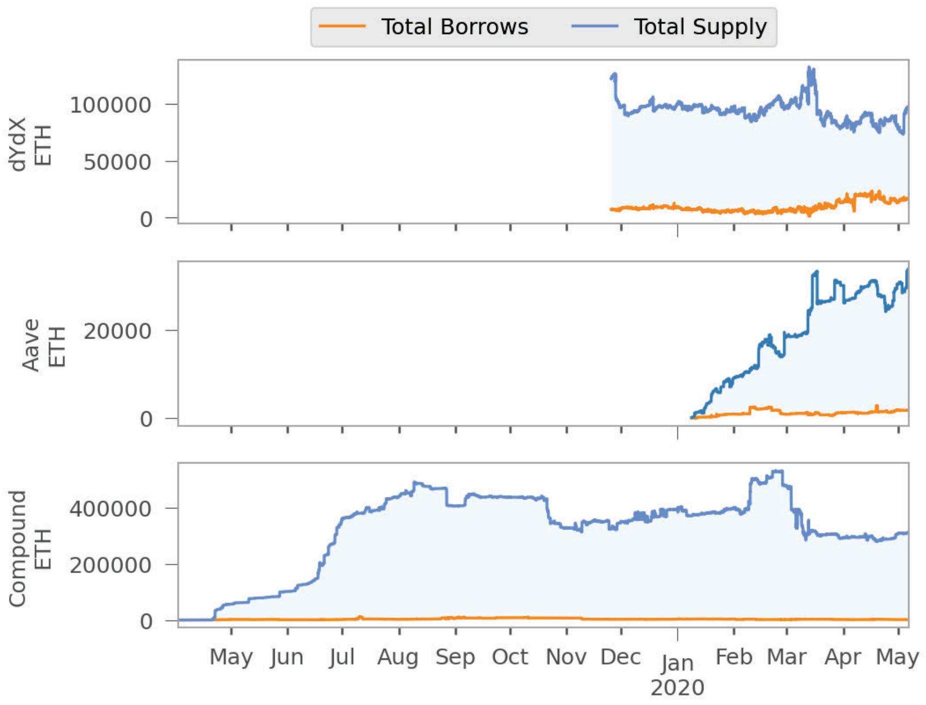


Figure 3.2: Total funds borrowed and supplied (i.e. liquidity) for ETH markets on dYdX, Compound and Aave.

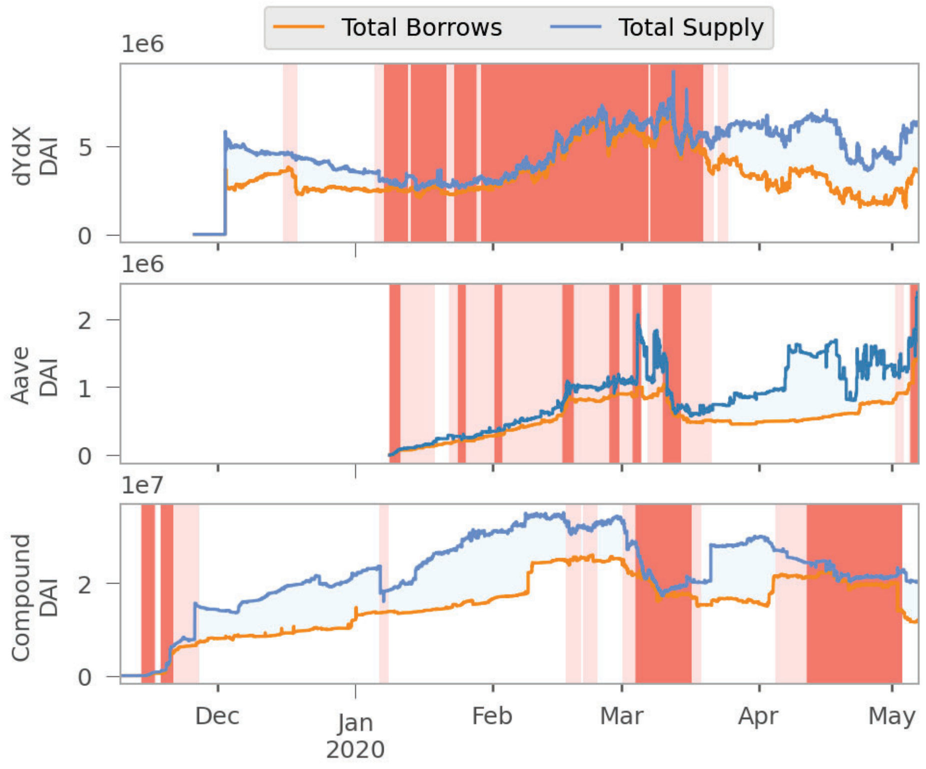


Figure 3.3: Total funds borrowed and supplied (i.e. liquidity) for DAI markets on dYdX, Compound and Aave. Periods where utilisation was between 80% and 90%, are highlighted in salmon, while utilisation higher than 90% is shaded in red.

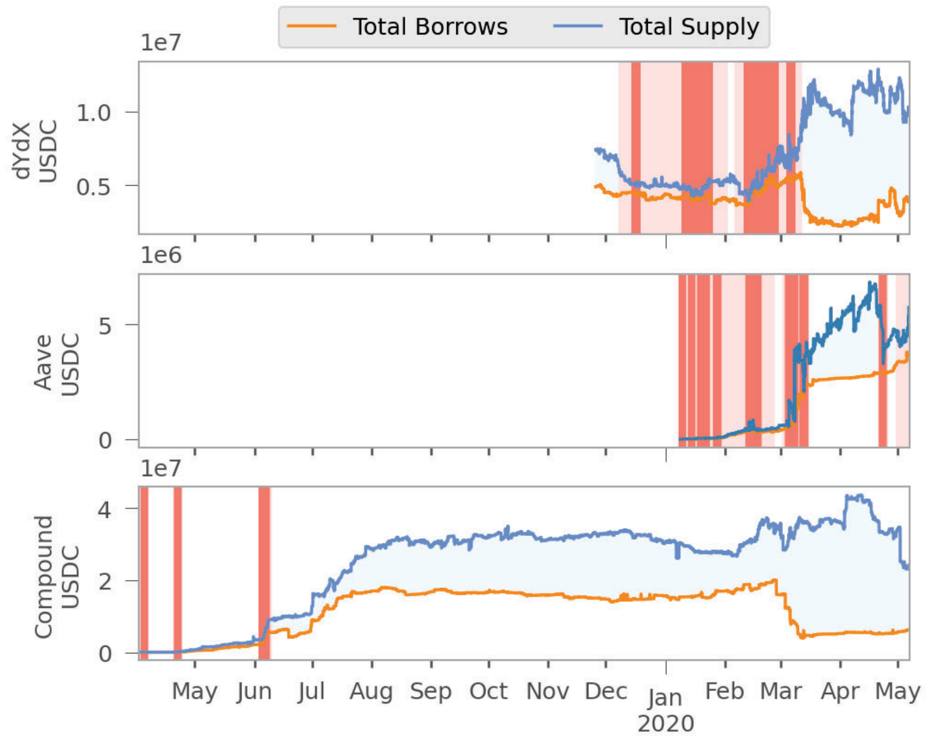


Figure 3.4: Total funds borrowed and supplied (i.e. liquidity) for USDC markets on dYdX, Compound and Aave. Periods where utilisation was between 80% and 90%, are highlighted in salmon, while utilisation higher than 90% is shaded in red.

across all DeFi protocols dropped from 897.2m USD to 559.42m USD.[7] For DAI, it can be seen how on Black Thursday, even the largest PLF, Compound, was exposed to prolonged periods of low liquidity before attracting increased liquidity again at the same time as dYdX and Aave. However, after mid-April, the market for DAI on Compound re-experienced low liquidity.

Illiquidity

On PLFs, agents are incentivised to provide liquidity via the employed interest rate model, as high interest rates would make borrowing more cost prohibitive in periods of low liquidity. However, borrowers need to be incentivised to repay their loans by sufficiently high interest rates at times of full utilisation to maintain sufficient liquidity. In the event of such illiquidity materialising, suppliers of funds would be unable to withdraw them, being forced to hold on to and continue to earn interest through their cTokens.

Out of the three PLFs, only Aave enforces a utilisation ceiling at 100%, while Compound and

[7]Source: https://defipulse.com. Accessed: 05-06-2020.

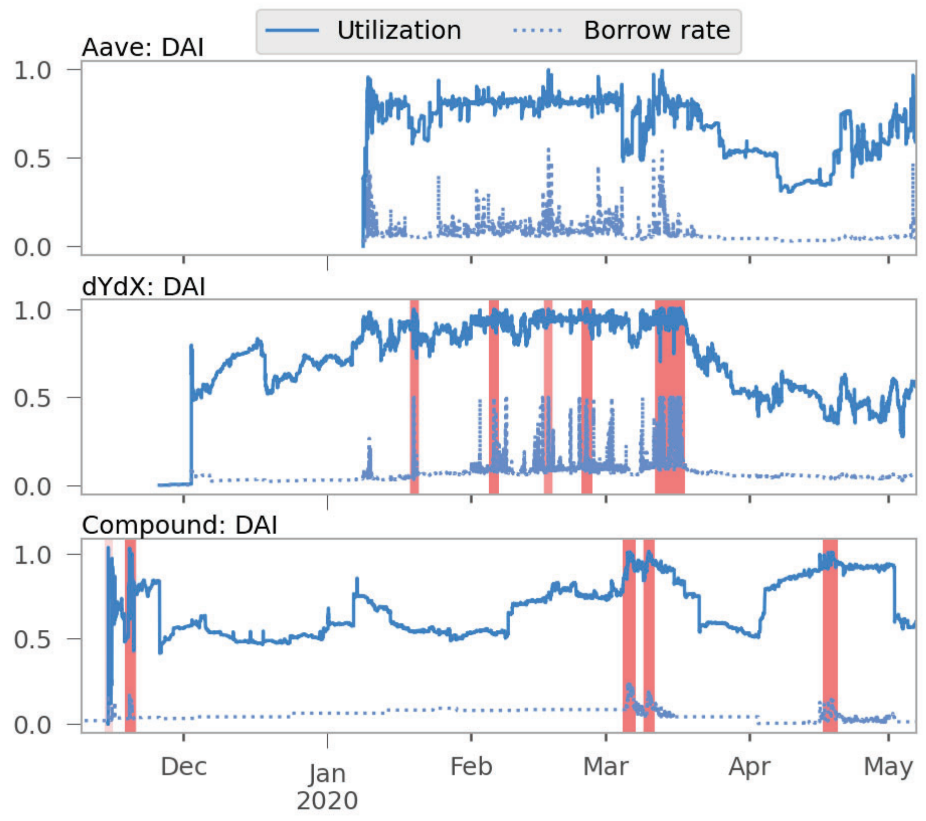


Figure 3.5: Utilisation and borrow rates for DAI on Aave (top), dYdX (middle) and Compound (bottom). Periods in which utilisation equalled or exceeded 100% are highlighted in red.

dYdX permit borrows even *beyond* full utilisation. When examining the market for DAI in Figure 3.5, it can be seen how utilisation of funds has been multiple times at and even above 100% on Compound and dYdX.

It can be seen that Aave has experienced periods of near-illiquidity. In contrast, Compound and dYdX have experienced periods of full illiquidity for DAI, i.e. all supplied funds were loaned out. When comparing the DAI borrow rates during periods of full utilisation (red) in Figure 3.5, notable differences can be made between the different interest rate regimes. On dYdX, the borrowing rate hits the model-imposed interest rate ceiling of 50%. In contrast, on Compound, the rate does not exceed 25% even at full utilisation, which can be explained by the linear nature of Compound's interest rates. Despite Aave never reaching full utilisation for DAI, due to an optimal utilisation target of 80% during the measurement period, borrow rates on Aave exceed rates on Compound during periods of high utilisation. This suggests that holding on to loans during periods of illiquidity is cheaper on Compound than on dYdX or Aave.

Fund distribution

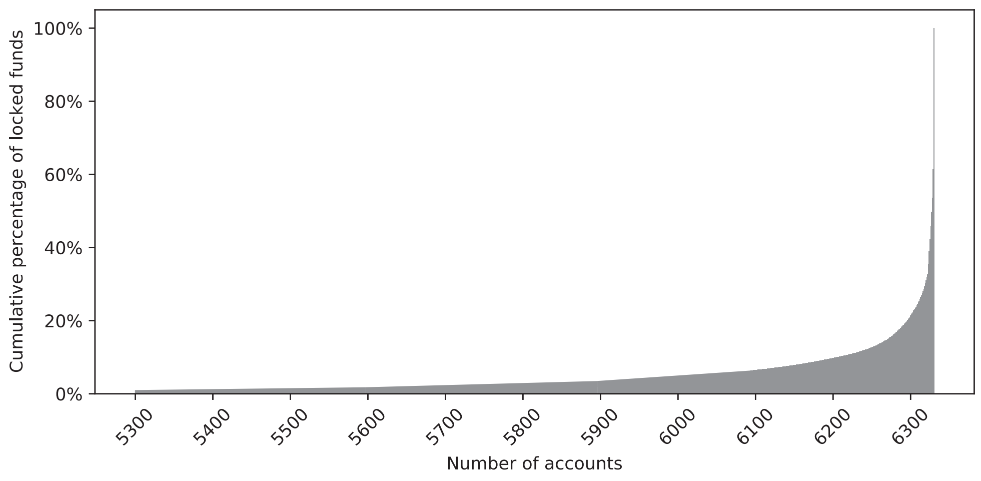
Periods of low liquidity have several implications for market participants. On one side, high utilisation implies lucrative interest rates for suppliers of funds, thereby attracting new liquidity. On the other hand, suppliers are faced with the risk of being unable to redeem their funds, for example, in the case of a 'bank run'.

To better assess the risk of a market becoming fully illiquid, we examine the cumulative percentage of locked funds for the number of Ethereum accounts on Compound in Figure 3.6. Note that we decided to focus solely on Compound because a similar pattern was found for Aave and dYdX. The distribution of funds across accounts is very similar for DAI, ETH and USDC in that a very small set of accounts controls most of the supplied funds. For instance, 50.3% of total locked DAI is controlled by only three accounts. Similarly, for ETH and USDC, the same number of accounts control 60.0% and 47.3%, respectively. Hence, for all three markets, even in times of high liquidity, a small number of suppliers of funds are in a position to drastically reduce liquidity or cause full illiquidity.

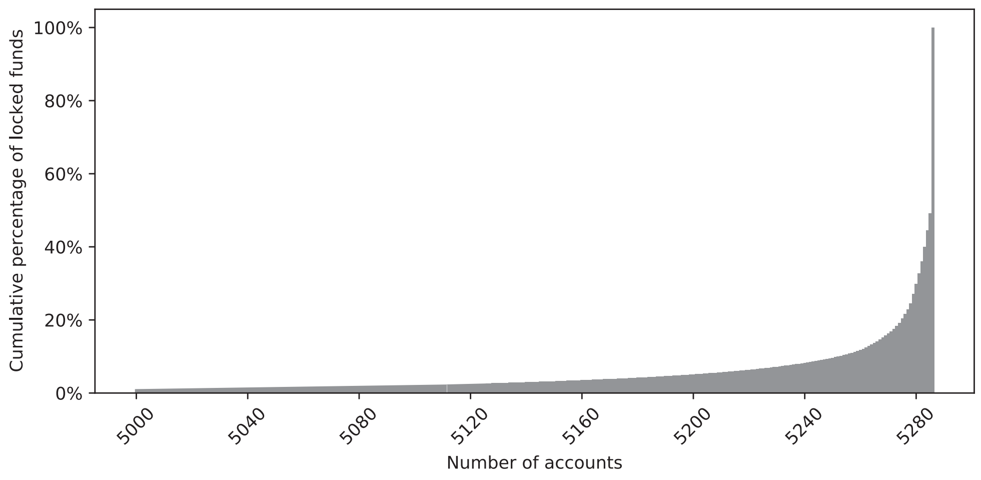
3.4.2 Case Study: DAI on Compound

In the context of liquidity, we present a case study of interest rate behaviour in the market for DAI on Compound, focusing on the period of 21 February to 21 April 2020 and its interest-bearing token cDAI. It could be seen in Figure 3.5 that for the period mentioned above, this market was exposed to a range of different utilisation levels, experiencing periods of relatively high liquidity but also illiquidity. Hence, we investigate market participants' behaviour—given by the interest rates that are observed—for different interest rate regimes during the period of interest.

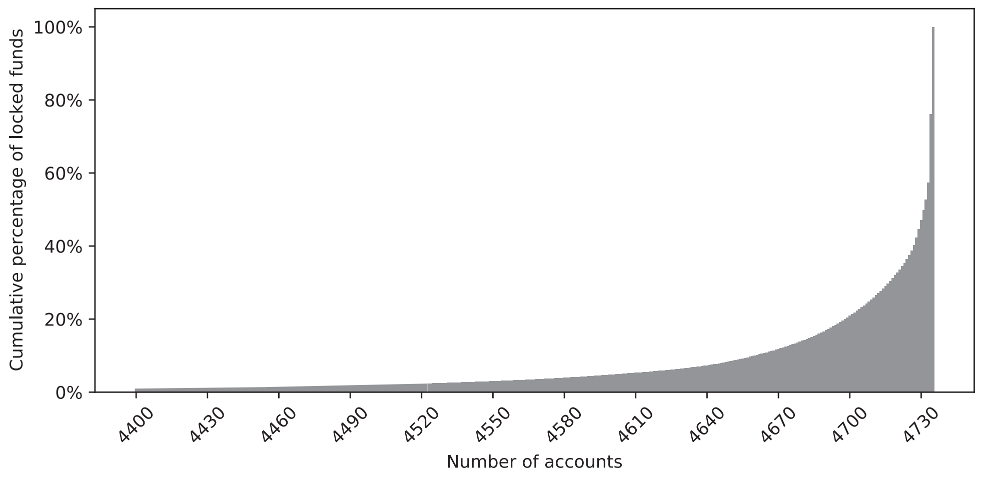
Interest rate models for the cDAI contract To illustrate kinked rates, we present the case of the DAI interest rate in Compound Finance. The cDAI token is an example of an interest-bearing derivative token based on a linear kinked interest rate model. Since 17 December 2019, the borrowing rates (i_b) have operated with equation (3.9). However, the precise parameter val-



(a) DAI



(b) ETH



(c) USDC

Figure 3.6: Cumulative percentage of locked funds on Compound for DAI, ETH and USDC on 2020-06-04.

ues used by the model have been revised multiple times. We include a list of these modifications in Table 3.3.

	Parameters			
Date	α	β	γ	U^*
17 Dec '19	19637062989	264248265	570776255707	9e17
8 Jan '20	29174130900	264248265	570776255707	9e17
26 Jan '20	37372598273	264248265	570776255707	9e17
4 Feb '20	41997859121	264248265	570776255707	9e17
9 Feb '20	36209575847	705029680	570776255707	9e17
21 Feb '20	38532925389	264248265	570776255707	9e17
14 Mar '20	19637062989	264248265	570776255707	9e17
6 Apr '20	0	2900146648	570776255707	9e17
21 Apr '20	0	264248265	570776255707	9e17
27 Apr '20	0	10569930661	570776255707	9e17

Table 3.3: Interest rate model and parameter changes for the cDAI contract since 17 December 2019 (before this date, an earlier variation of the interest rate model —'Jump Rate Model'—was in force since 23 November 2019; we omit this period for clarity of exposition.).

Interest rate behavior We consider in detail how, since 17 December 2019, agents have optimised their selection of borrowing and saving amounts given an interest rate schedule. Here we focus on a subset of three periods, namely:

- 21 February — 13 March 2020
- 14 March — 5 April 2020
- 6 April — 21 April 2020

These regimes are plotted in Figure 3.7. At the start of each period, the interest rate parameters were changed to values as specified in Table 3.3. Here we plot the behaviour of the borrowing rates, but the behaviour of the supply rates is broadly similar.

Figure 3.8a and the corresponding Figure 3.10 plot the interest rate model (the blue surface) as well as the realised interest rate (black crosses). The two points to note are that (i) there appears to be a clustering of the realised interest rates at the kink of the interest rate function and (ii) otherwise, interest rates are typically higher than the kink, corresponding to utilisation of above 90%.

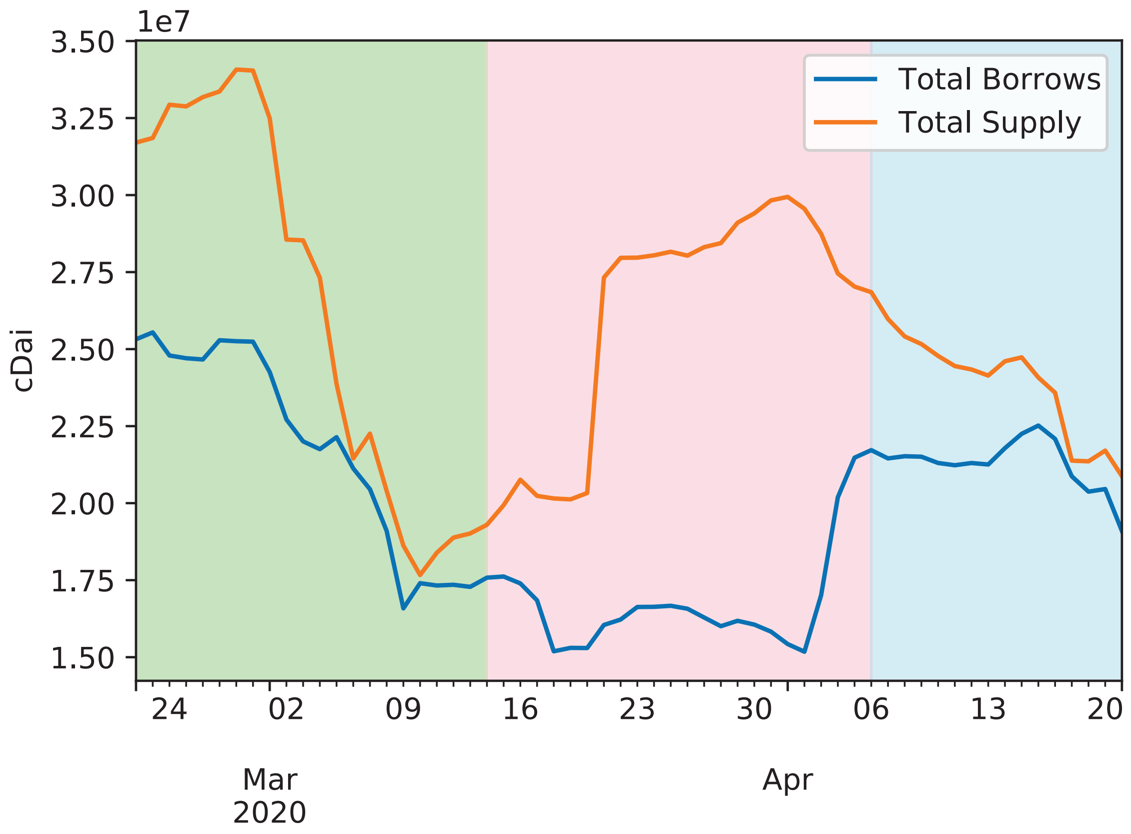
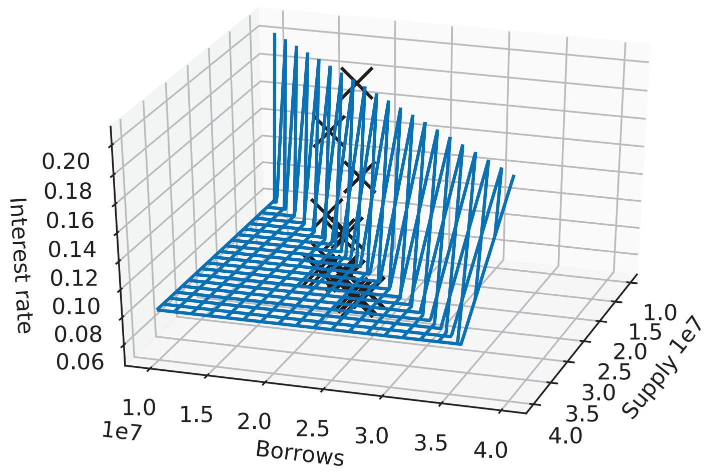


Figure 3.7: Three interest rate regimes in Compound.

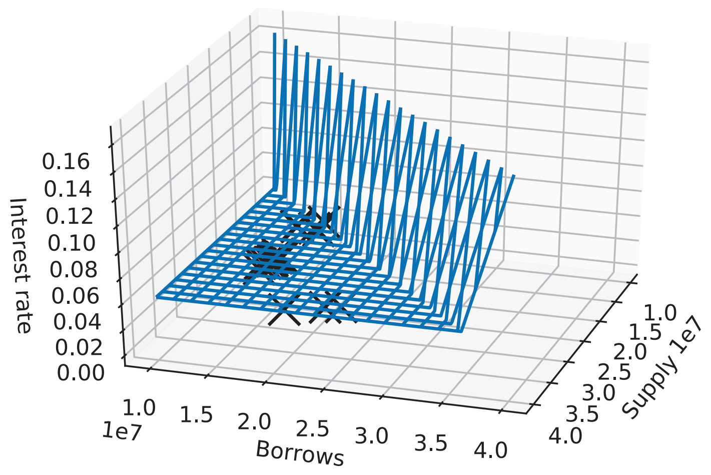
Figure 3.8b shows the interest rate model and the realised interest rates in the next period after the base rate α is reduced via a parameter change by 49.04%. Despite this change, we continue to observe a clustering of the realised interest rates at the kink. However, there does appear to be some effect of reducing the typical utilisation ratio to below the kink.

Figure 3.8c shows how the system behaves once the base rate α is set to zero, while the multiplier β is increased by nearly 1000%. Again, we observe a similar pattern: most realised interest rates appear at the kink. However, if not at the kink, now typically utilisation is above 90%.

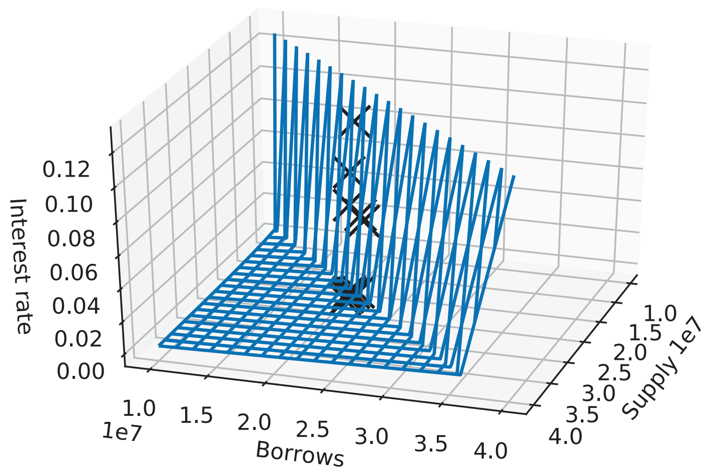
Figure 3.9 plots the evolution of the borrowing rate as a time series, and Figure 3.10 provides a histogram focussing on the distribution of the borrowing rate for the period 21 February to 13 March.



(a) 21 February — 13 March

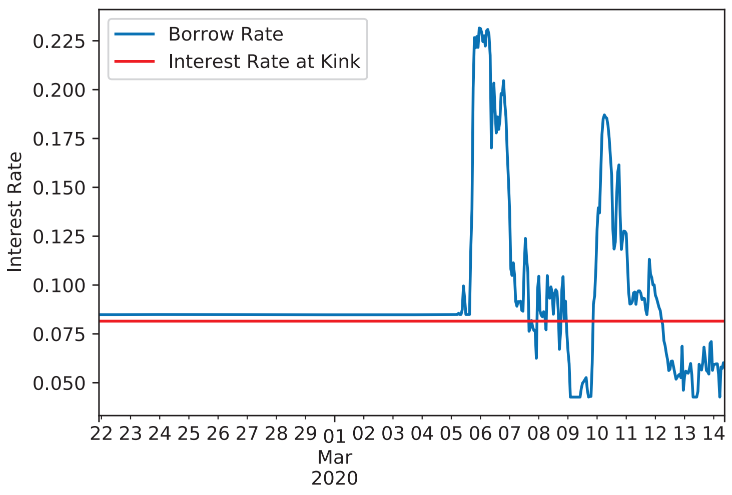


(b) 14 March — 5 April

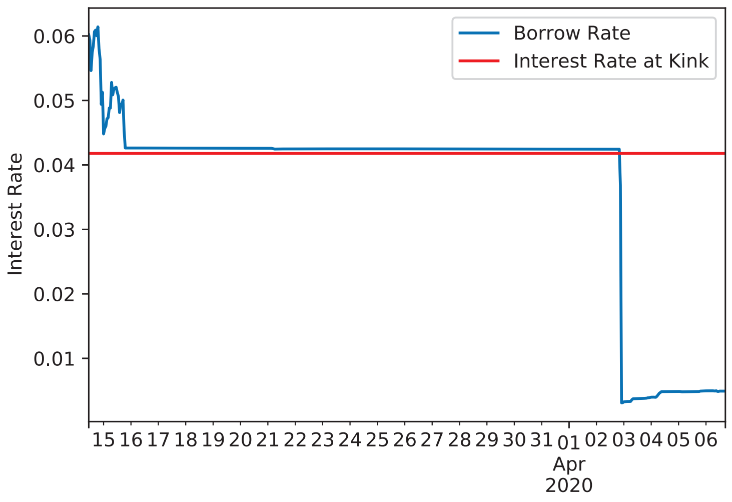


(c) 6 April — 21 April

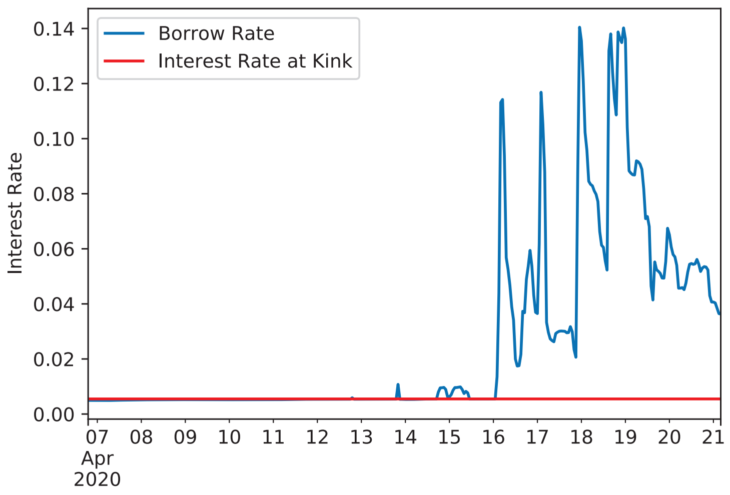
Figure 3.8: Borrowing rates surface for DAI.



(a) 21 February — 13 March



(b) 14 March — 5 April



(c) 6 April — 21 April

Figure 3.9: Borrowing rate (hourly mean) distribution around kink for DAI.

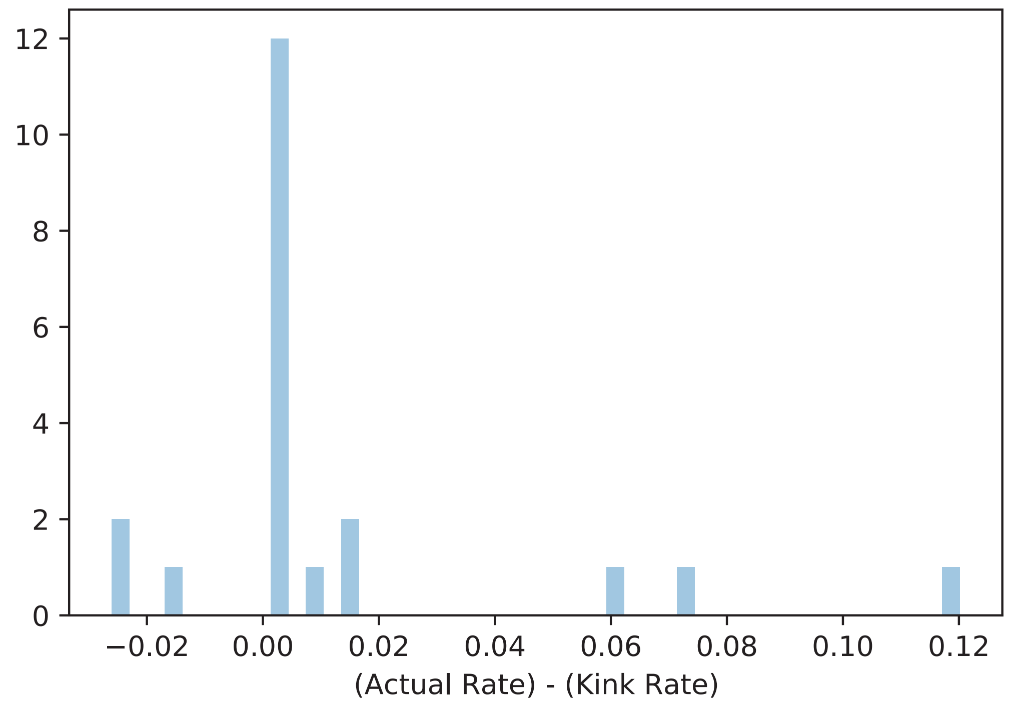


Figure 3.10: Borrowing rate distribution around kink, 21 February — 13 March.

3.4.3 Summary

We saw that, especially for DAI, there were several periods of illiquidity and that they were often shared across the three protocols. We also showed that the locked funds were very concentrated and in Figure 3.6 that a very small number of accounts had the potential to make the markets illiquid. Finally, we analysed the interest rate behaviour of DAI on Compound. We showed that the interest rates appeared clustered around the kink of the interest rate function during all the observation periods.

3.5 Market efficiency

In this section, we consider the capital market efficiency of DeFi lending protocols. Loosely, a capital market is said to be efficient if, in determining prices, it fully and correctly reflects all relevant information [Mal89]. More precisely:

Definition 3.1 (Market efficiency)**.** *A market is efficient with respect to some information set* ϕ

if prices would be unaffected by revealing that information to all market participants [Mal89].

A notable consequence of Definition 3.1 is that such efficiency implies the impossibility of making economic profits based on the information set ϕ. The market efficiency of PLFs is a question of central interest because it provides a mechanism to assess the markets' maturity and understand the agents' responsiveness to changes in the information set ϕ. Moreover, since a core mechanism common to many PLFs is the use of high interest rates at times of high utilisation —to encourage saving and discourage borrowing, incentivising agents to behave in a certain way—the extent to which PLFs are capital efficient will inform how reliable this mechanism is, at present, in incentivising agents to act in the intended way. If agents do not respond to high interest rates by reducing their borrowing requirements and increasing their supply of funds to a PLF, illiquidity resulting from high utilisation rates on a given protocol may be expected to result. Such illiquidity events, where agents cannot withdraw their funds, can be expected to cause panic in financial markets. Therefore from the point of view of financial risk, the efficiency of markets is of central interest.

Thus in this section, we consider whether PLFs are efficient within a given protocol, considering Compound [Com19] within a standard framework in assessing the efficiency of markets in the context of foreign exchange: Uncovered Interest Parity.

3.5.1 Uncovered interest parity

First, we set out Uncovered Interest Parity (UIP) as it would typically appear in the context of foreign exchange between two countries: *domestic* and *foreign*. An investor has the choice of whether to hold domestic or foreign assets. UIP is a theoretical no-arbitrage condition, which states that in equilibrium if the condition holds, a risk-neutral investor should be indifferent between holding the domestic or foreign assets because the exchange rate is expected to adjust such that returns are equivalent.

For example, consider UIP holding between GBP and USD. An investor starting with 1m GBP at $t = 0$ could either:

- receive an annual interest rate of $i_{\text{GBP}} = 3\%$, resulting in 1.03m GBP at $t = 1$

- or, immediately buy 1.23m USD at an exchange rate $S_{\text{GBP/USD}} = 0.8130$, receiving an annual interest rate of $i_{\text{USD}} = 5\%$, resulting in 1.2915m USD at $t = 1$. Once converted to GBP at the new exchange rate at $t = 1$, $S_{\text{GBP/USD}} = 0.7974$, identically yields 1.03m GBP.

If UIP holds, despite the differences in interest rates, adjustments in the exchange rate between the currencies offset any potential gain such that arbitrage is impossible. Mathematically, UIP is stated as follows.

$$1 + \iota_i = (1 + \iota_j)\frac{\mathrm{E}_t[S_{t+k}]}{S_t} \tag{3.16}$$

where $\mathrm{E}_t[S_{t+k}]$ denotes the expectation in period t of the exchange rate $S_{i/j}$ between assets i and j at time $t + k$, k is an arbitrary number of periods into the future, S_t is the current spot exchange rate between assets i and j, ι_i is the interest rate payable on asset i and ι_j is the interest rate payable on asset j. If equation (3.16) holds, then investors cannot make risk-free profit through simply exploiting the real difference in interest rates after the impact of the exchange rate is accounted for.

3.5.2 UIP in a PLF

Here, analogously, we perform a pairwise analysis of all possible pairs of tokens available within a protocol, seeking to establish whether UIP holds for that pair. For UIP to hold, it must be the case that a risk-neutral investor would be indifferent between saving (or borrowing) either of the tokens within the pair because the exchange rate between any token pair adjusts such that no risk-free profit can be made. As it is the largest PLF [Pul19a] at the time of writing, we consider to what extent the condition holds within Compound [Com19].

3.5.3 Empirical approach

To develop our empirical specification, we assume that agents have rational expectations:

$$S_{t+k} = \mathrm{E}_t[S_{t+k}] + \epsilon_{t+k} \tag{3.17}$$

where ϵ denotes a random error. Taking logs of equation 3.16 and approximating $\log(1+\iota_i) \approx \iota_i$, we test whether UIP obtains with the following empirical specification:

$$s_{t+1} - s_t = \alpha + \beta(\iota_i - \iota_j) + \epsilon \tag{3.18}$$

where $\log S_{t+1} = s_{t+1}$ and $\log S_t = s_t$.

$$\mathbf{H_0} \textbf{ Strict form UIP:} \quad \alpha = 0 \text{ and } \beta = 1 \tag{3.19}$$

Alternatively, we could impose no restriction on α perhaps reflecting a risk premium [Ale01].

$$\mathbf{H'_0} \textbf{ Weak form UIP:} \quad \beta = 1 \tag{3.20}$$

A risk premium reflects the extra return in the form of interest payment required for investors to receive the same risk-adjusted return as on a less risky token. We test both hypotheses, considering all possible token pairings on Compound and separately reporting borrowing and saving interest rates.

3.5.4 UIP regression results — borrowing rates

We use heteroskedasticity and autocorrelation robust standard errors for both borrowing and saving rate regressions, which we report in brackets in the results[8].

Considering first data at the daily frequency (Table 3.4), we find that for 20 market pairs, we reject both $\mathbf{H_0}$ and $\mathbf{H'_0}$ at the 1% level, suggesting UIP does not hold in either its strong or weak form for daily data.

The evidence is more mixed at the weekly frequency in Table 3.5. We reject $\mathbf{H_0}$ at the 1% level for 12/20 pairs but find evidence consistent with $\mathbf{H_0}$ for eight pairs. Regarding $\mathbf{H'_0}$, we cannot reject it at the 1% level in 11 cases. However, the standard errors are typically large,

[8]WBTC was excluded from the analysis due to data quality issues.

Pair	N.obs	α	β	R-squared	α p-value	β p-value	Strict form (3.19) p-value	Weak form (3.20) p-value
eth_bat	392	0.01 (0.01)	-0.482483 (0.30)	0.02	0.05	0.11	0.00	0.00
eth_zrx	389	0.00 (0.00)	-0.194958 (0.14)	0.00	0.30	0.17	0.00	0.00
eth_usdc	393	0.00 (0.01)	-0.0357526 (0.12)	0.00	0.74	0.76	0.00	0.00
eth_dai	175	0.01 (0.01)	-0.252845 (0.20)	0.01	0.10	0.22	0.00	0.00
eth_sai	397	0.00 (0.01)	-0.0315418 (0.06)	0.00	0.61	0.58	0.00	0.00
eth_rep	392	0.00 (0.00)	0.0512982 (0.11)	0.00	0.58	0.65	0.00	0.00
bat_zrx	387	-0.00 (0.00)	-0.478467 (0.25)	0.03	0.64	0.05	0.00	0.00
bat_usdc	392	0.00 (0.01)	-0.0913661 (0.11)	0.00	0.60	0.40	0.00	0.00
bat_dai	175	0.00 (0.00)	-0.328409 (0.20)	0.02	0.37	0.11	0.00	0.00
bat_sai	393	0.01 (0.01)	-0.134668 (0.08)	0.01	0.20	0.11	0.00	0.00
bat_rep	388	0.00 (0.00)	0.0854052 (0.19)	0.00	0.64	0.65	0.00	0.00
zrx_usdc	388	0.00 (0.01)	-0.0676933 (0.11)	0.00	0.62	0.54	0.00	0.00
zrx_dai	175	0.01 (0.01)	-0.514228 (0.25)	0.03	0.09	0.04	0.00	0.00
zrx_sai	389	0.01 (0.01)	-0.0759909 (0.07)	0.00	0.38	0.27	0.00	0.00
zrx_rep	387	-0.00 (0.00)	-0.23005 (0.13)	0.01	0.60	0.08	0.00	0.00
usdc_dai	175	-0.00 (0.00)	-0.0111104 (0.02)	0.00	0.77	0.53	0.00	0.00
usdc_sai	394	0.00 (0.00)	-0.00474335 (0.01)	0.00	0.74	0.52	0.00	0.00
usdc_rep	390	-0.00 (0.01)	-0.0510679 (0.11)	0.00	0.77	0.64	0.00	0.00
dai_sai	175	0.01 (0.01)	-0.267694 (0.15)	0.01	0.24	0.08	0.00	0.00
dai_rep	175	-0.01 (0.00)	-0.158339 (0.18)	0.00	0.20	0.38	0.00	0.00

Table 3.4: Table of UIP results for daily frequency data, using borrowing rates. Using Newey-West heteroscedasticity and autocorrelation robust standard errors (reported in parentheses.)

Pair	N.obs	α	β	R-squared	α p-value	β p-value	Strict form (3.19) p-value	Weak form (3.20) p-value
eth_bat	56	0.05	-2.27281	0.02	0.47	0.56	0.64	0.40
		(0.07)	(3.86)					
eth_zrx	56	0.01	-0.652686	0.00	0.71	0.24	0.00	0.00
		(0.02)	(0.56)					
eth_usdc	56	0.02	-0.254169	0.00	0.72	0.69	0.00	0.05
		(0.06)	(0.64)					
eth_dai	25	0.07	-1.86555	0.12	0.18	0.30	0.30	0.12
		(0.05)	(1.80)					
eth_sai	56	0.02	-0.263629	0.01	0.52	0.47	0.00	0.00
		(0.03)	(0.36)					
eth_rep	56	0.01	0.718818	0.01	0.68	0.48	0.91	0.78
		(0.02)	(1.02)					
bat_zrx	56	-0.01	-1.03856	0.01	0.44	0.46	0.10	0.15
		(0.02)	(1.41)					
bat_usdc	56	0.03	-0.630377	0.03	0.46	0.15	0.00	0.00
		(0.04)	(0.44)					
bat_dai	25	0.02	-2.11941	0.17	0.49	0.12	0.10	0.03
		(0.03)	(1.37)					
bat_sai	56	0.07	-0.920485	0.10	0.12	0.04	0.00	0.00
		(0.05)	(0.45)					
bat_rep	56	0.03	2.65716	0.09	0.07	0.01	0.17	0.13
		(0.02)	(1.09)					
zrx_usdc	56	0.04	-0.5622	0.01	0.59	0.53	0.00	0.09
		(0.07)	(0.90)					
zrx_dai	25	0.09	-3.48611	0.17	0.18	0.08	0.06	0.03
		(0.07)	(1.98)					
zrx_sai	56	0.05	-0.544193	0.02	0.17	0.21	0.00	0.00
		(0.04)	(0.44)					
zrx_rep	56	0.01	-0.702555	0.01	0.75	0.08	0.00	0.00
		(0.02)	(0.40)					
usdc_dai	25	-0.00	-0.0976848	0.10	0.55	0.23	0.00	0.00
		(0.00)	(0.08)					
usdc_sai	56	0.00	-0.0525398	0.09	0.08	0.02	0.00	0.00
		(0.00)	(0.02)					
usdc_rep	56	-0.03	-0.593887	0.03	0.29	0.08	0.00	0.00
		(0.03)	(0.34)					
dai_sai	25	0.07	-1.84099	0.12	0.39	0.28	0.00	0.11
		(0.09)	(1.69)					
dai_rep	25	-0.07	-1.9174	0.16	0.10	0.14	0.10	0.03
		(0.04)	(1.29)					

Table 3.5: Table of UIP results for weekly frequency data, using borrowing rates. Using Newey-West heteroscedasticity and autocorrelation robust standard errors (reported in parentheses.)

Pair	N.obs	α	β	R-squared	α p-value	β p-value	Strict form (3.19) p-value	Weak form (3.20) p-value
eth_bat	392	0.00	-1.10434	0.02	0.02	0.00	0.00	0.00
		(0.00)	(0.28)					
eth_zrx	389	0.00	-0.61295	0.00	0.51	0.09	0.00	0.00
		(0.00)	(0.36)					
eth_usdc	393	0.00	-0.0164582	0.00	0.86	0.91	0.00	0.00
		(0.01)	(0.15)					
eth_dai	175	0.01	-0.21499	0.01	0.13	0.24	0.00	0.00
		(0.01)	(0.18)					
eth_sai	397	-0.00	-0.00335582	0.00	0.89	0.95	0.00	0.00
		(0.00)	(0.05)					
eth_rep	392	0.00	-0.0695602	0.00	0.44	0.28	0.00	0.00
		(0.00)	(0.06)					
bat_zrx	387	-0.00	-1.37843	0.05	0.76	0.00	0.00	0.00
		(0.00)	(0.48)					
bat_usdc	392	0.00	-0.100931	0.00	0.70	0.52	0.00	0.00
		(0.01)	(0.16)					
bat_dai	175	0.01	-0.263267	0.01	0.11	0.13	0.00	0.00
		(0.01)	(0.17)					
bat_sai	393	0.00	-0.095	0.01	0.35	0.14	0.00	0.00
		(0.00)	(0.06)					
bat_rep	388	0.00	0.0467639	0.00	0.85	0.85	0.00	0.00
		(0.00)	(0.25)					
zrx_usdc	388	0.00	-0.0559146	0.00	0.74	0.70	0.00	0.00
		(0.01)	(0.14)					
zrx_dai	175	0.02	-0.341124	0.02	0.09	0.08	0.00	0.00
		(0.01)	(0.19)					
zrx_sai	389	0.00	-0.0586814	0.00	0.47	0.31	0.00	0.00
		(0.01)	(0.06)					
zrx_rep	387	0.00	-0.692352	0.01	0.96	0.01	0.00	0.00
		(0.00)	(0.28)					
usdc_dai	175	0.00	-0.0174017	0.00	0.31	0.44	0.00	0.00
		(0.00)	(0.02)					
usdc_sai	394	0.00	-0.00493231	0.00	0.77	0.51	0.00	0.00
		(0.00)	(0.01)					
usdc_rep	390	-0.00	-0.0931906	0.00	0.67	0.50	0.00	0.00
		(0.01)	(0.14)					
dai_sai	175	-0.00	-0.216161	0.01	0.24	0.09	0.00	0.00
		(0.00)	(0.13)					
dai_rep	175	-0.01	-0.153443	0.00	0.23	0.39	0.00	0.00
		(0.01)	(0.18)					

Table 3.6: Table of UIP results for daily frequency data, using saving rates. Using Newey-West heteroscedasticity and autocorrelation robust standard errors (reported in parentheses.)

making it difficult to reject any null hypothesis. Overall, we find no evidence of UIP holding for daily data, while for weekly data, we find some supporting evidence.

3.5.5 UIP regression results — saving rates

Looking at saving rates at the daily frequency, Table 3.6, similarly to borrowing rates for all 20 pairs, we reject both $\mathbf{H_0}$ and $\mathbf{H'_0}$ at the 1% level, suggesting UIP does not hold in either its strong or weak form for daily data.

The evidence is more mixed at the weekly frequency, in Table 3.7. We reject $\mathbf{H_0}$ at the 1% level for 12/20 pairs but find evidence consistent with $\mathbf{H_0}$ for eight pairs. Regarding $\mathbf{H'_0}$, we

Pair	N.obs	α	β	R-squared	α p-value	β p-value	Strict form (3.19) p-value	Weak form (3.20) p-value
eth_bat	56	0.04	-11.4299	0.21	0.01	0.00	0.00	0.00
		(0.02)	(1.51)					
eth_zrx	56	0.00	-2.21453	0.01	0.86	0.04	0.00	0.00
		(0.02)	(1.06)					
eth_usdc	56	0.01	-0.174619	0.00	0.84	0.83	0.01	0.16
		(0.05)	(0.83)					
eth_dai	25	0.08	-1.62795	0.10	0.25	0.35	0.28	0.14
		(0.07)	(1.73)					
eth_sai	56	0.00	-0.154556	0.01	0.79	0.65	0.00	0.00
		(0.02)	(0.34)					
eth_rep	56	0.01	-0.079075	0.00	0.48	0.84	0.03	0.01
		(0.02)	(0.40)					
bat_zrx	56	-0.01	-5.48743	0.07	0.41	0.21	0.20	0.15
		(0.01)	(4.39)					
bat_usdc	56	0.03	-0.841115	0.02	0.48	0.18	0.00	0.00
		(0.04)	(0.63)					
bat_dai	25	0.07	-1.73374	0.12	0.31	0.26	0.10	0.09
		(0.07)	(1.54)					
bat_sai	56	0.04	-0.817567	0.10	0.17	0.03	0.00	0.00
		(0.03)	(0.38)					
bat_rep	56	0.00	5.60551	0.14	0.77	0.00	0.00	0.00
		(0.01)	(0.39)					
zrx_usdc	56	0.04	-0.819943	0.01	0.53	0.46	0.00	0.10
		(0.06)	(1.10)					
zrx_dai	25	0.13	-2.35098	0.11	0.18	0.15	0.07	0.05
		(0.10)	(1.62)					
zrx_sai	56	0.04	-0.543762	0.03	0.08	0.09	0.00	0.00
		(0.02)	(0.32)					
zrx_rep	56	0.01	-1.1295	0.01	0.50	0.00	0.00	0.00
		(0.02)	(0.38)					
usdc_dai	25	0.00	-0.166452	0.18	0.10	0.10	0.00	0.00
		(0.00)	(0.10)					
usdc_sai	56	0.00	-0.0475424	0.09	0.15	0.02	0.00	0.00
		(0.00)	(0.02)					
usdc_rep	56	-0.03	-0.866689	0.03	0.28	0.03	0.00	0.00
		(0.03)	(0.40)					
dai_sai	25	-0.02	-1.31716	0.07	0.22	0.36	0.24	0.12
		(0.02)	(1.43)					
dai_rep	25	-0.09	-1.72587	0.14	0.13	0.18	0.09	0.05
		(0.06)	(1.29)					

Table 3.7: Table of UIP results for weekly frequency data, using saving rates. Using Newey-West heteroscedasticity and autocorrelation robust standard errors (reported in parentheses.)

cannot reject it at the 1% level in 9 cases. However, the standard errors are typically large, so it would be difficult to reject any hypothesis. Overall, we again find no evidence of UIP holding for daily data, while for weekly data, we find some supportive evidence.

3.5.6 Summary

Looking at daily and weekly frequency data for borrowing and saving, we find weak evidence that UIP holds as the time horizon increases. This parallels empirical results in traditional foreign exchange markets [Isa06]. This suggests that overall the markets within the Compound PLF may not be fully capital efficient. It seems plausible that these results are not only idiosyncratically valid for Compound. The finding that this PLF is not capital efficient at the daily frequency is not surprising — there is considerable evidence that UIP does not hold even in traditional foreign exchange markets [CM05]. In addition, this suggests that the *currency carry trade*—where an investor borrows a low yield currency to obtain a high yield currency—is likely to be profitable since, in such inefficient markets, differences in yield are not offset by corresponding changes in the exchange rate between the currencies. Moreover, we submit that in the context of a PLF, to the extent that there is market inefficiency, agents may not be fully responding to these incentives.

3.6 Market dependence

We now consider the extent of inter-connectedness *between* protocols by considering how changes in an interest rate for a given token on one PLF are related to changes in the interest rate for the token on another PLF.

For example, consider the borrowing rate for DAI, $i_{b,DAI}$. A priori, we expect that if $i_{b,DAI}$ is higher on one PLF than others, agents would be incentivized to borrow from those PLFs with a lower borrowing rate, deleveraging on one PLF and leveraging on others. But this influx of borrowers for the token on other PLFs would, in turn, increase the borrowing rates on those protocols.

In this section, taking the stablecoins DAI and USDC, we investigate whether there is evidence of such dynamics and find that such behaviour is indeed observable. Moreover, we quantify the speed of adjustment to new equilibria values and, in so doing, measure in one way the responsiveness of agents to their incentives in PLFs.

3.6.1 Vector error correction models

We model both the short and long-run dynamics between borrowing rates for DAI and USDC by using a Vector Error Correction Model (VECM).

Where time series are non-stationary (e.g. a random walk), the required criteria for a regression to produce the Best Linear Unbiased Estimator (BLUE) are not satisfied because the variables are not covariance stationary.[9] However, if a linear combination of non-stationary time series exists, where this combination is stationary, the series is said to be *cointegrated*. VECMs permit the modelling of the stationary relationships between such time series and allow estimation of both the long-run and short-run adjustment dynamics. This is appropriate for interest rates in PLFs which a priori might be expected to display both short and long-run dynamics. A VECM model is as follows.

$$\mathbf{\Delta y_t} = \mathbf{v} + \mathbf{\Pi y_{t-1}} + \sum_{i=1}^{p-1} \mathbf{\Gamma_i \Delta y_{t-i}} + \epsilon_\mathbf{i} \tag{3.21}$$

where Δ denotes a single time step, $\mathbf{y_t}$ is a vector of K variables, $\mathbf{v}$ is a vector of $K \times 1$ parameters, $\mathbf{\Pi} = \sum_{j=1}^{j=p} \mathbf{A_j} - \mathbf{I_k}$ (I_k denotes an indicator vector), where $\mathbf{A_j}$ is a matrix of $K \times K$ parameters from a vector autoregression (VAR)[10], $\mathbf{\Gamma_i} = -\sum_{j=i+1}^{j=p} \mathbf{A_j}$ and ϵ is a $K \times 1$ vector of disturbances. Assuming that Π has reduced rank $0 < r < K$ it can further be expressed as $\mathbf{\Pi} = \alpha\beta'$ [Sta13]. In terms of interpretation, α provides the adjustment coefficients, and β provides the parameters of the cointegrating (i.e. long-run) equations.

3.6.2 Results

Separately, we focus on the borrowing rates for DAI and USDC, considering Compound, Aave and dYdX. We present the borrowing rates for DAI in Figure 3.11 and for USDC in Figure 3.12.

DAI results First, we consider the markets for DAI. Testing for the number of cointegrating relationships between the three series using Johansen's multiple trace test method [Sta13], we find evidence of at most two cointegrating relationships. After iteratively tuning the model with post-estimation results, we find the optimum lag length to be 5. The results are presented in full in Table 3.8.

In terms of short adjustment coefficients, we find a statistically significant coefficient on Aave DAI of 0.38. When the borrowing rate on Compound is high, Aave's borrowing rate quickly

[9]Covariance stationary means that the mean and autocovariance are finite and time-invariant.

[10]A VAR(p) can be expressed as $\mathbf{y_t} = \mathbf{v} + \mathbf{A_1 y_{t-1}} + \mathbf{A_2 y_{t-2}} + ... + \mathbf{A_p y_{t-p}} + \epsilon$

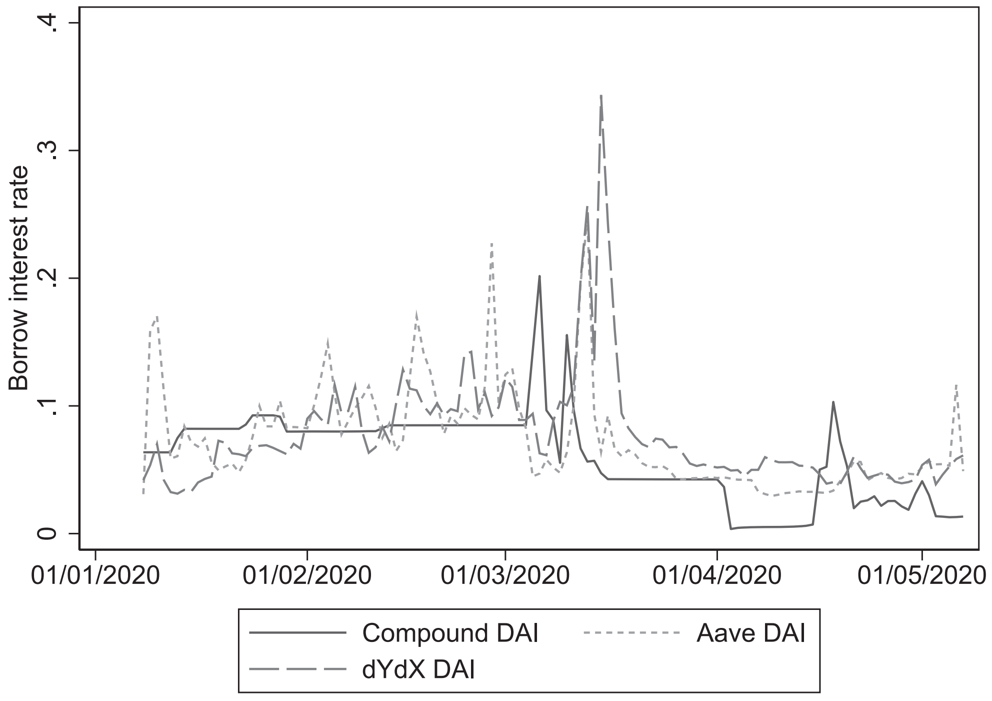


Figure 3.11: Daily borrowing interest rates on DAI across protocols.

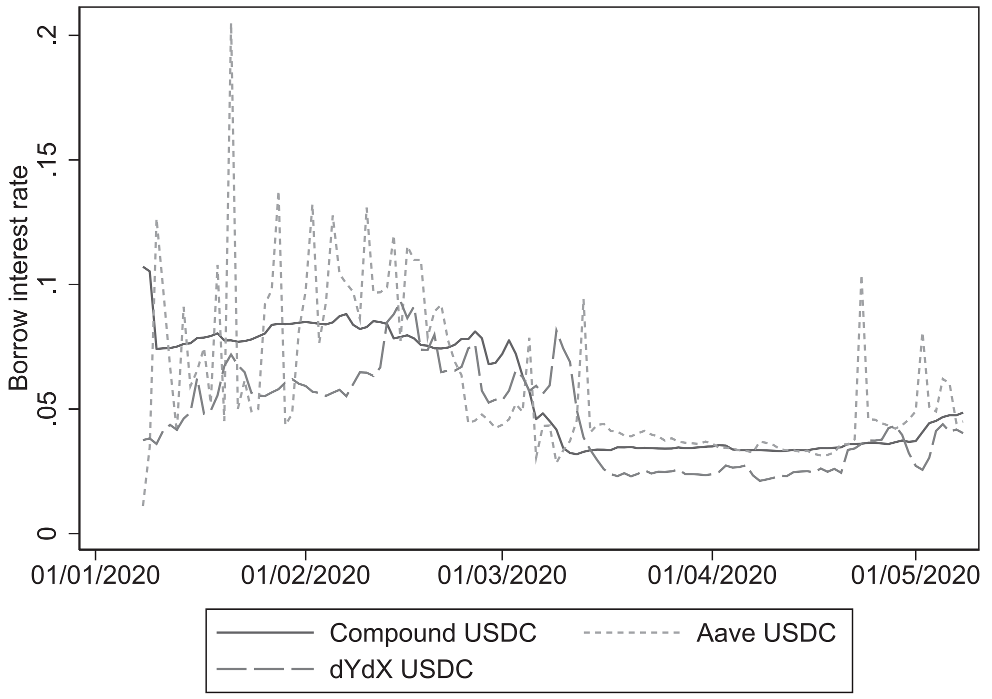


Figure 3.12: Daily borrowing interest rates on USDC across protocols.

	(1)
D_c_dai	
L._ce1	-0.0695
	(-1.20)
L._ce2	0.143
	(1.60)
D_a_dai	
L._ce1	0.381^{***}
	(4.66)
L._ce2	-0.533^{***}
	(-4.23)
D_d_dai	
L._ce1	0.284^{***}
	(4.07)
L._ce2	-0.0387
	(-0.36)
Long-run (_ce1)	
Compound DAI	1
Aave DAI	0
	(omitted)
DyDx DAI	-1.151^{***}
	(-5.75)
Constant	0.0296
Long-run (_ce2)	
Compound DAI	0
	(omitted)
Aave DAI	1
DyDx DAI	-0.9906^{***}
	(-5.94)
Constant	0.0051
Observations	116

t statistics in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Table 3.8: Vector Error Correction Model Results — DAI.

increases to match it. Similarly, we find a similar effect for dYdX DAI, with a slightly slower adjustment speed of 0.28. Interestingly, we do not find evidence of the Compound DAI rate adjusting to changes in the Aave or dYdX DAI rates in the short run, suggesting that Compound's interest rate changes drive changes in both Aave and dYdX's borrowing rates, which may indicate that Compound has market power. This is expected: as shown in Figure 3.3, Compound has the largest borrow and supply volumes for DAI compared to the other two PLFs and thus will plausibly shape interest rates across protocols. We obtain the following long-run cointegrating relationships:

$$DAI_{Compound} = -1.151 DAI_{dYdX} - 0.030 \tag{3.22}$$

and

$$DAI_{Aave} = -0.991 DAI_{dYdX} - 0.005 \tag{3.23}$$

such that for DAI, dYdX has a long-run cointegrating relationship with Compound and Aave.

We present the impact of a shock to Compound's DAI borrow rate on Aave and dYdX's in Figure 3.13. It can be seen that a positive shock to the borrowing rate results in a permanent increase in the borrowing rate on Aave and dYdX.

USDC results For USDC, we find that between the series, there are two cointegrating relationships [Sta13]. Again, testing for the number of cointegrating relationships between the three series using Johansen's multiple trace test method [Sta13], we find evidence of at most two cointegrating relationships. After iteratively tuning the model with post-estimation results, we find the optimum lag length to be 3. The results are presented in full in Table 3.9.

It appears that Compound has market power again, with the borrowing rates on Aave and dYdX adjusting to match the Compound interest rate level. Aave appears to adjust faster at 0.607, compared to dYdX at 0.115. In terms of long-run relationships, we find that Compound and dYdX share a long-run relationship and that Aave and dYdX share a long-run relationship. We obtain the following long-run cointegrating relationships:

	(1)
D_c_usdc	
L._ce1	0.0146
	(0.83)
L._ce2	0.0271
	(1.89)
D_a_usdc	
L._ce1	0.607***
	(3.42)
L._ce2	-0.720***
	(-4.97)
D_d_usdc	
L._ce1	0.115**
	(2.75)
L._ce2	0.0200
	(0.59)
Long-run (_ce1)	
Compound USDC	1
Aave USDC	5.55e-17
	.
DyDx USDC	-1.353***
	(-7.77)
Constant	0.0066
Long-run (_ce2)	
Compound DAI	-2.78e-17
	.
Aave DAI	1
DyDx DAI	-1.347***
	(-7.95)
Constant	0.00283
Observations	119

t statistics in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Table 3.9: Vector Error Correction Model Results — USDC.

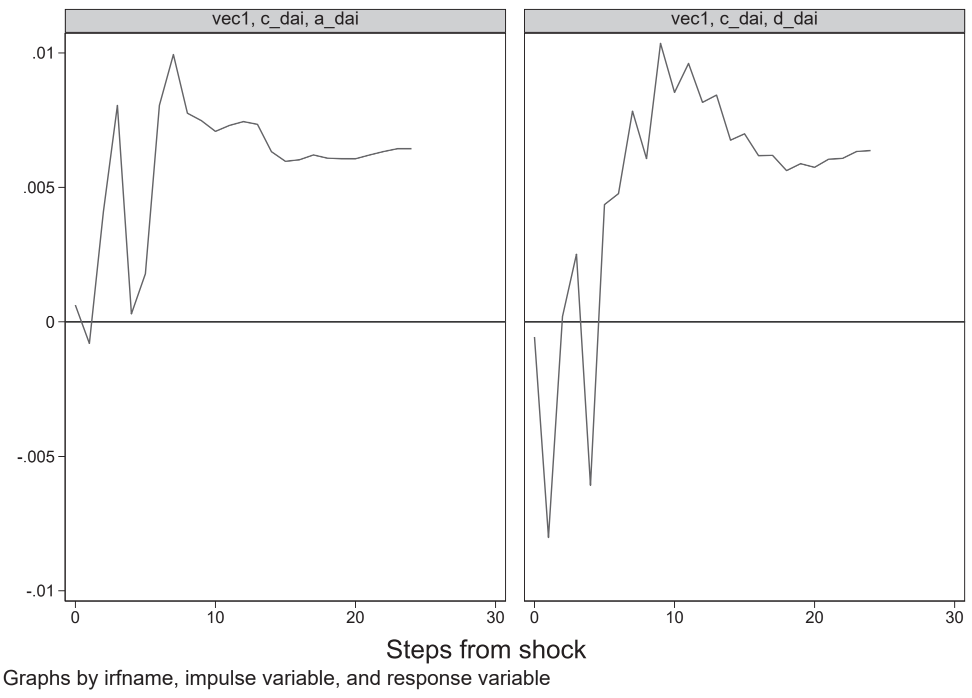


Figure 3.13: Impulse Response Function: impact of a shock (at the beginning) to Compound's DAI borrow rate on Aave and dYdX's DAI borrow rate.

$$USDC_{Compound} = -1.353USDC_{dYdX} - 0.007 \tag{3.24}$$

and

$$USDC_{Aave} = -1.347USDC_{dYdX} - 0.003 \tag{3.25}$$

For USDC, like DAI, dYdX has a long-run cointegrating relationship with Compound and Aave.

We plot the impact of a change in the USDC borrowing rate in Figure 3.14. A shock to Compound's borrowing rate on USDC permanently affects the interest rates in Aave and dYdX.

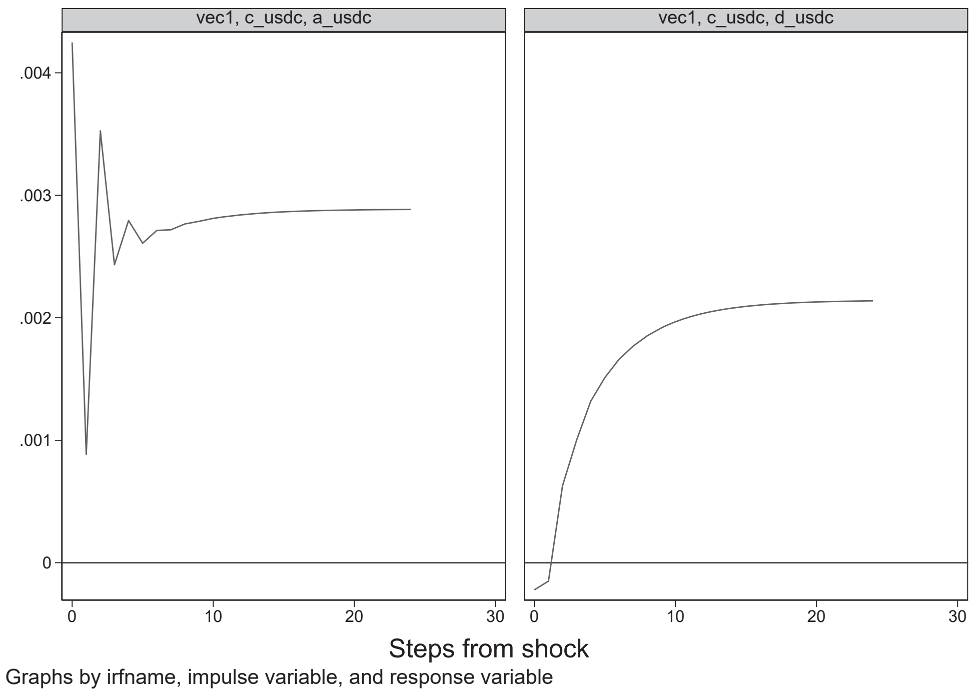


Figure 3.14: Impulse Response Function: impact of a shock to Compound's USDC borrow rate on Aave and dYdX's USDC borrow rates.

3.6.3 Robustness checks

We performed extensive robustness checks on the fitted VECM models. Since our ability to draw sound inference on the adjustment parameters depends on the cointegrating equations being stationary, we plot the cointegrating equations over time (see Figures 3.15, 3.16, 3.17 and 3.18.)

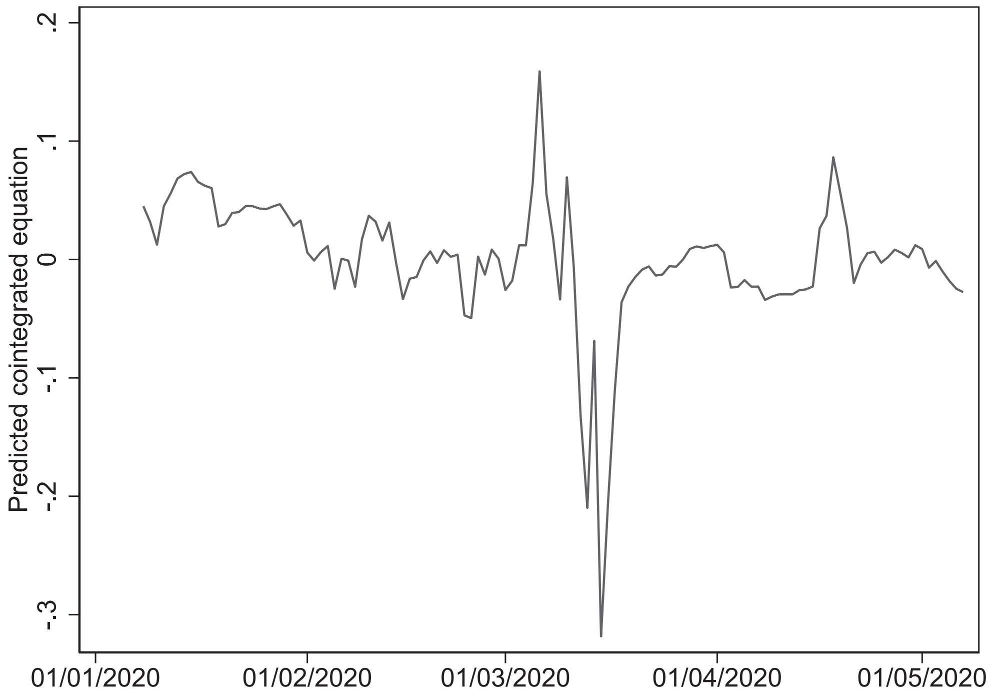


Figure 3.15: DAI cointegrating equation 1.

The cointegrating equations appear overall without significant trends, though note the presence of a large negative shock in the DAI specifications mid-March, and therefore broadly stationary. Furthermore, we check that we have correctly specified the number of cointegrating equations in Figures 3.19 and 3.20.

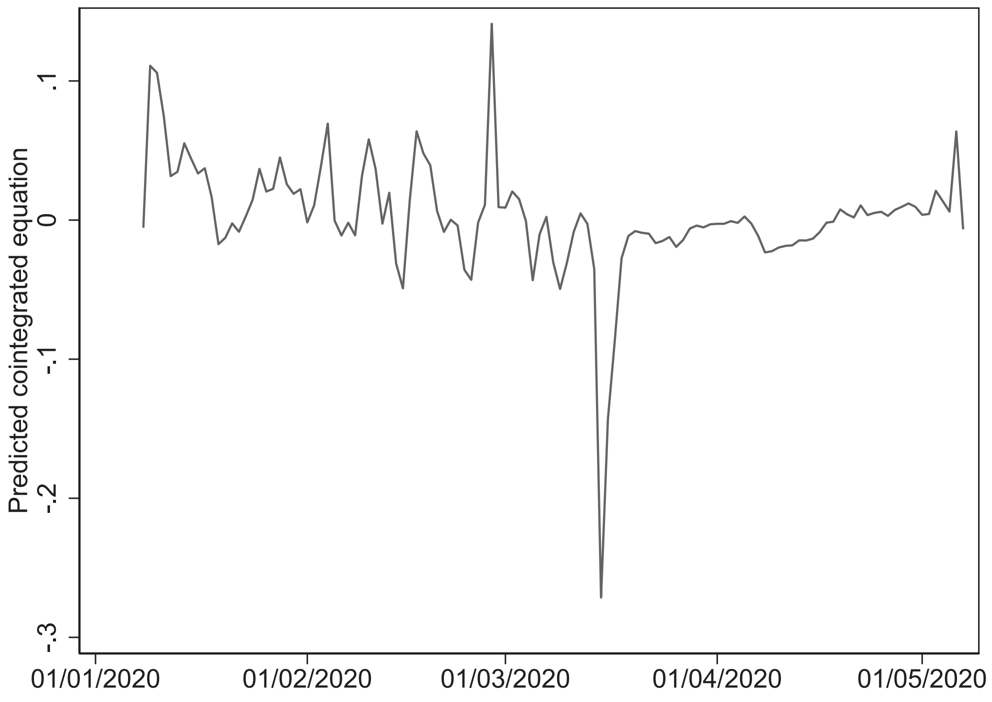


Figure 3.16: DAI cointegrating equation 2.

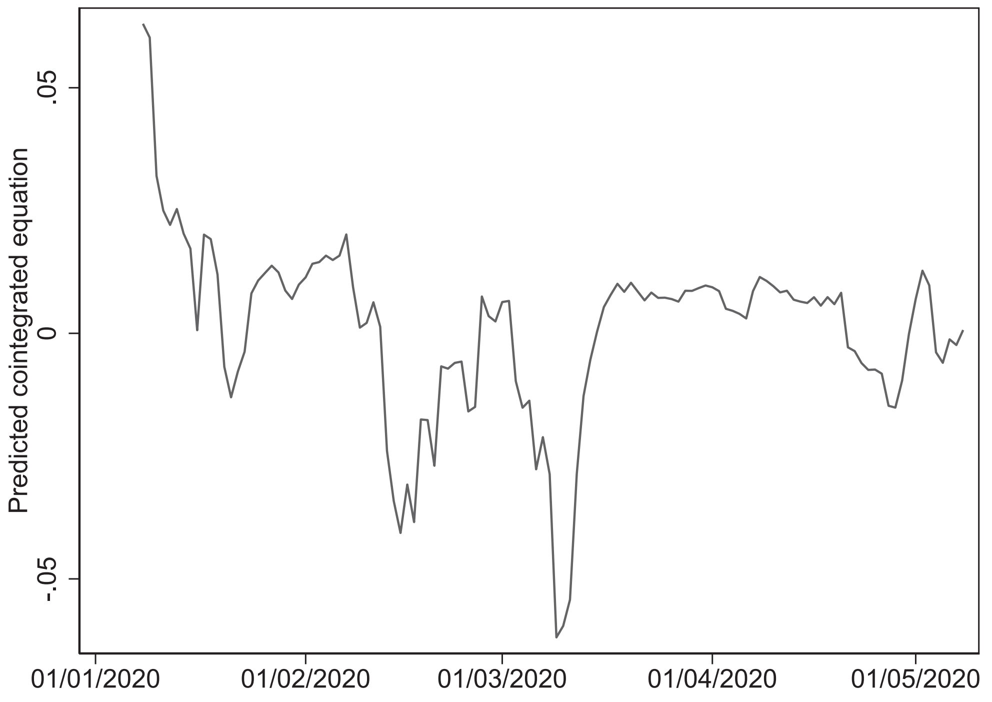


Figure 3.17: USDC cointegrating equation 1

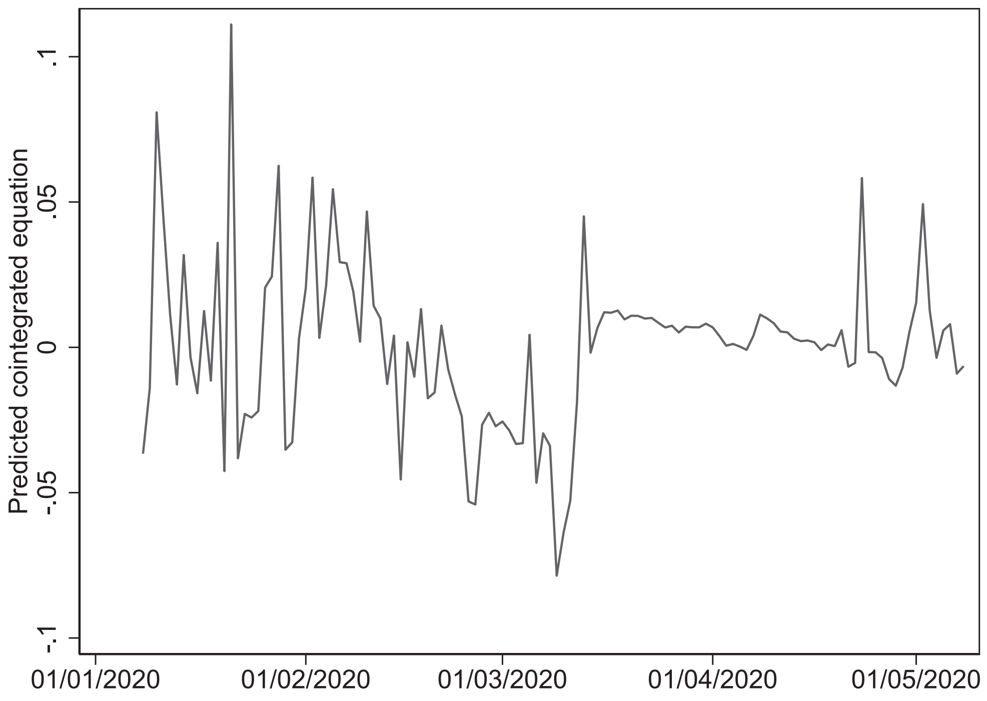


Figure 3.18: USDC cointegrating equation 2.

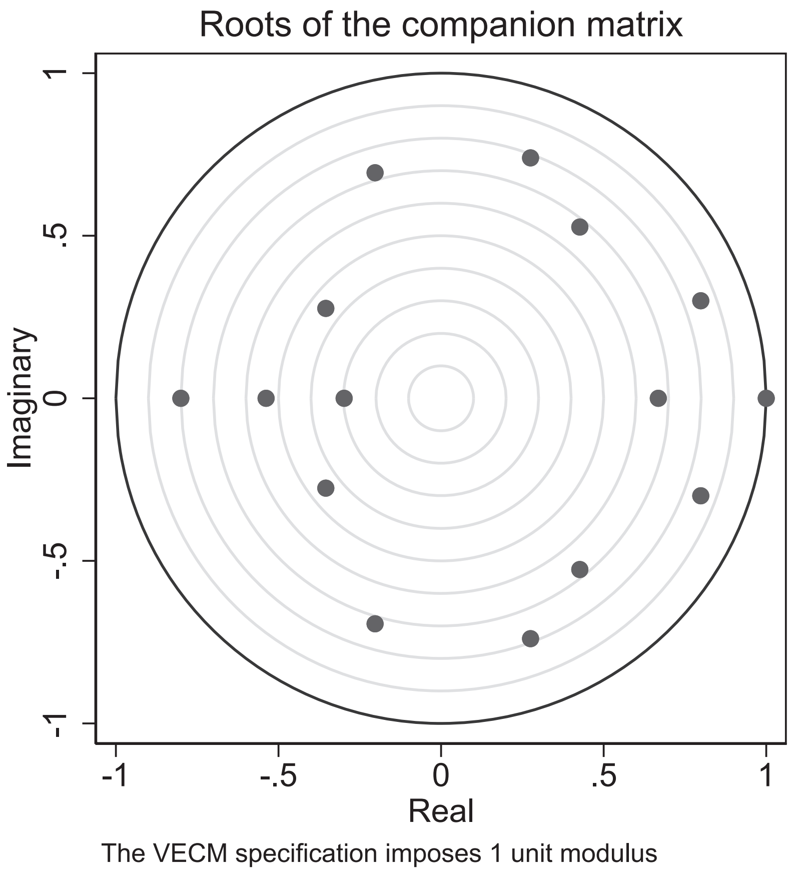


Figure 3.19: DAI cointegrating equations misspecification test.

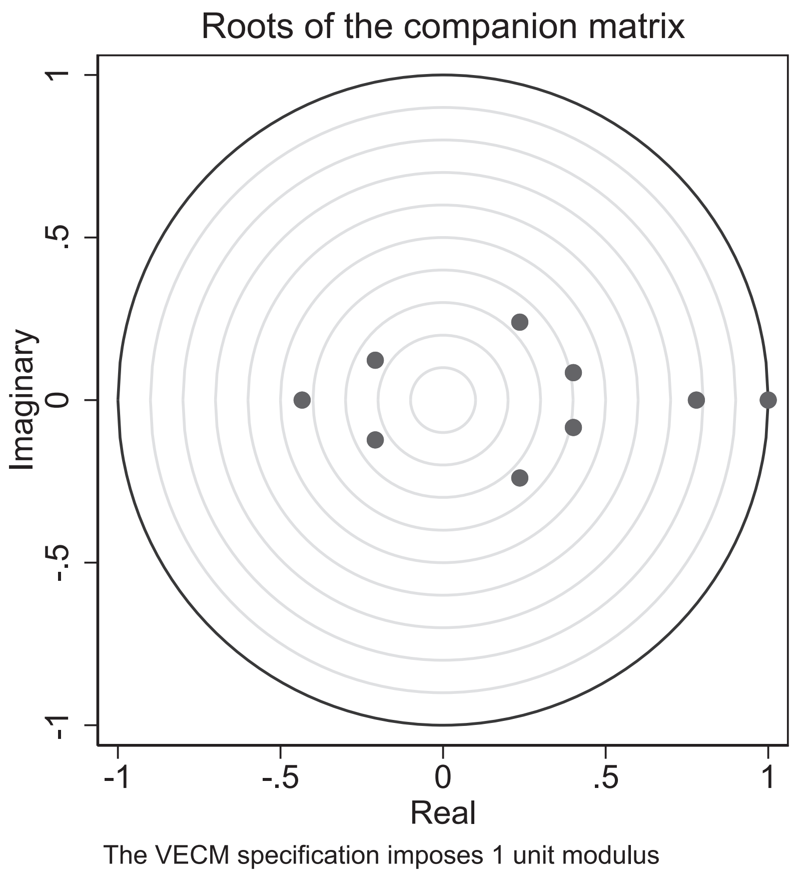


Figure 3.20: USDC cointegrating equations misspecification test.

We find no evidence that any of the eigenvalues are close to the unit circle, and therefore no evidence that the model is misspecified (see [Sta13] for details on this test.) Additionally, we test for serial correlation in the residuals of the regressions and find little evidence of this. A test for the normality of the errors in our models does suggest that the errors are non-normally distributed, which may affect our standard errors but should not result in parameter bias. This panel of robustness tests makes us confident that the VECM models are reasonably well specified.

3.6.4 Summary

Overall we find evidence of cointegrating relationships between markets for DAI and USDC. This suggests that interest rate changes in one protocol are associated with interest rate changes in others, providing evidence of agents being incentivized to change protocol by the rates they observe. Moreover, we also find some evidence of Compound having market power.

3.7 Related work

In this section, we present related work about interest rates in both traditional finance and DeFi protocols. Since, to the best of our knowledge, this chapter is the first academic work to investigate PLFs in detail; we include some non-academic work which covers some aspects of PLFs interest rates.

The authors of [KCCM20] focus on the Compound protocol and present an overview of the market risks and liquidation mechanism. They perform agent-based simulations to investigate the economic security of the protocol and find that the protocol can scale to a larger market size while maintaining a low probability even when markets are volatile.

In [Li19], the author describes how the interest rate models work in PLFs. The author first defines the utilisation ratio of a PLF, then describes linear and polynomial interest rate models and finally presents how these different models are used by three significant PLFs, namely, Compound, dYdX and Aave.

The author of [Ale19] analyses Compound to show the risks inherent to decentralised lending. In particular, they focus on the risks associated with illiquidity and bank runs. The authors analyse the SAI market on Compound and find several periods of near-illiquidity and actual illiquidity. They present instances where illiquidity is created because of large loans in a short period and others made by the lenders withdrawing large amounts of funds they had locked. In particular, they show that on five occasions, a single transaction was sufficient to withdraw more than a quarter of the available liquidity. In the worst case, a single transaction drained more than 95% of the available liquidity.

The author of [Fra20b] focuses on how Black Thursday [Reu20] in March 2020 affected the Aave market. They first show that the amount of money borrowed through flash loans increased by more than 10,000% in only a few hours because users were leveraging these to liquidate their collateralised debt positions [Mak19b, Mak]. The author also highlights that the number of borrows on Aave during Black Thursday was more than 100 times higher than the typical amount liquidated, reaching more than 550k USD in a single day. Finally, the author shows that during the Black Thursday crisis, some design flaws of MakerDAO's protocol [Mak] caused Maker to lose more than 4m USD worth of collateral.

In [JK15], the authors elucidate the difference between the intermediation and financing roles of traditional banks and show that when modelling banks with financing models as opposed to intermediation models, identical shocks have much more significant effects on the real economy.

Finally, [BHM19] presents a work about interest rates in the context of cryptocurrencies but centred on a different problem. Their work focuses on how cryptocurrencies could set an interest rate for their holders, such that they accumulate these interest rates continuously.

3.8 Conclusion

In this chapter, we coin Protocol for Loanable Funds to describe DeFi equivalents of Intermediaries for Loanable Funds in traditional finance, providing a classification framework for the extant interest rate models. We analyze three of the largest PLFs in terms of market liquidity, efficiency and dependence.

Regarding market liquidity, we find that individual PLFs often operate at times of high utilization, and often, these moments of high utilization are shared across protocols. Moreover, we find that token holdings can be concentrated to a very small set of accounts, such that at any time were a small number of suppliers to withdraw their funds, perhaps in concert, they could significantly reduce the liquidity available on markets and possibly make such markets illiquid.

In terms of market efficiency, we consider whether uncovered interest parity holds. Overall, it does not, suggesting that token markets are, at present, relatively inefficient. This also suggests that, at present, agents may need to be more responsive to interest rate incentives.

Regarding market dependence, we find that the borrowing rates on these protocols influence each other. In particular, that Compound has some market power to set the prevailing borrowing rate for Aave and dYdX.

Chapter 4

Decentralized Financial Crisis

4.1 Introduction

As explored at the outset of this thesis, blockchain technology emerged in part as a response to the Financial Crisis of 2007–8 [Nak08].[1] The perception that banks had misbehaved resulted in a deterioration of trust in the traditional financial sector [Ear09]. The causes of the crisis were several, but arguably chief among them was a lack of transparency regarding the amount of risk major banks were accumulating. When Lehman Brothers filed for bankruptcy, it had debts of 613bn USD, bond debt of 155bn USD and assets of 639bn USD [BBC09]. Central to its bankruptcy was its exposure to subprime (i.e., bad quality) mortgages. This exposure was compounded by the fact that the bank had a leverage ratio[2] of 30.7x in 2007 [Bro07].

From their inception, blockchain-based cryptocurrencies sought to provide a remedy to such crises: facilitating financial transactions without reliance on trusted intermediaries, shifting the power, and therefore, the ability to cause crisis through the construction of opaque and complex

[1]The first Bitcoin block famously outlines: "The Times 03/Jan/2009, Chancellor on brink of second bailout for banks".

[2]Defined as $\frac{\text{total assets}}{\text{equity}}$; the total assets were more than 30 times larger than what shareholders owned, indicating substantial debt.

financial instruments, away from banks and financial institutions [Nak08]. Ten years later, a complex financial architecture—the architecture of Decentralised Finance (DeFi)—is gradually emerging on top of existing blockchain platforms. Components in this architecture include those that pertain to lending, decentralised exchange of assets, and markets for derivatives [Fou19, Com19, Syn20, Pro20b, dYd20, Ins20].

DeFi architectures for lending require agents to post *security deposits* to fully compensate counter-parties for the disappearance of the agent. We assume that when an economically rational agent faces a choice between the repayment of a debt or the loss of collateral, given the absence of reputation tracking—on account of agent pseudonymity and the possibility of an agent using multiple addresses—the agent will choose the least costly option. Security deposits serve to guard against (i) *misbehaviour* of agents, where the action that would maximise individual utility does not maximise social welfare, and (ii) external events, such as large exogenous drops in the value of a particular cryptocurrency [HGGK19]. Of all DeFi protocols, those with the most locked capital are for lending. As of April 15th, 2020, the largest protocol by capitalisation, Maker [Fou19], has c. 65% of all capital locked in DeFi, corresponding to 342.9m USD [Pul19a].

Governance is another crucial facet of DeFi protocols, and we observe differing degrees of governance decentralisation. For example, Maker uses its own token (MKR) to allow holders to vote on a contract that implements the governance rules. In contrast, Compound [Com19], the third largest protocol by market share, is centrally governed, and a single account can shut down the system in case of failure. Moreover, as in traditional finance, these protocols do not exist in isolation. Assets that are created in Maker, for example, can be used as collateral in other protocols such as Compound, dYdX [dYd20], or in liquidity pools on Uniswap [Pro20b]. Indeed, the composability of DeFi — the ability to build a complex, multi-component financial system on top of crypto-assets — is a defining property of open finance [Pul19b]. However, if the underlying collateral assets fail, all connected protocols will be affected as well: there is the possibility of financial contagion.

This chapter. We focus on DeFi *lending* protocols, which constitute c. 76% of the DeFi market in capital terms. We consider two distinct but interconnected aspects of the attack and risk surface for collateral-based DeFi lending protocols: (i) attacks on the governance mechanism and (ii) the economic security of such protocols in “black swan” [Tal07] financial scenarios.

In relation to attacks on the governance mechanism, we examine an attack on Maker [Fou19], which consists of an adversary amassing enough capital to seize full control of the funds within the protocol. Herein, we further consider two distinct attack strategies. We engaged in a process of responsible disclosure with Maker, which we detail, who have since modified their governance parameters to mitigate the two attack strategies we present. The first attack strategy, crowdfunding, inspired by [Mic19], covertly executed, was feasible within two blockchain blocks and required the attacker to lock c. 27.5m USD of collateral. This would have enabled an attacker to steal all 0.5bn USD of locked collateral in the protocol and mint an unlimited supply of DAI tokens. The second novel attack strategy utilised so-called flash loans and allows an adversary to amass the Maker collateral within a single transaction. This attack did not require the locking of collateral and only required a few US dollars to pay for gas fees.

With respect to the economic security of collateralised DeFi lending protocols, with reference to our stylised model, we present the possibility of a DeFi lending protocol becoming undercollateralised (or insolvent) — where security deposits become smaller than the issued debt — as the result of a drying up of liquidity. Assuming rational economic agents, in such an undercollateralisation event, the borrower would default on their debts since the amount they have borrowed has become worth more than the amount they escrowed. Starting from formal definitions of the economic security constraints for DeFi lending platforms, we then use Monte Carlo simulation to stress-test their financial robustness. We submit that such stress testing constitutes an important approach to bounding the economic security of DeFi lending protocols when formal security proofs are not obtainable, and their security primarily depends on economic properties. We find that for plausible parameter ranges, a DeFi lending system could find itself undercollateralised. To the extent that other DeFi protocols allow agents to lend or trade the undercollateralised asset, financial contagion—where an economic shock spreads to other protocols—would be expected to result.

Contributions.

- **Governance attack on Maker (Section 4.2).** With a specific focus on the largest DeFi project by market share, Maker, we show how, prior to Maker implementing a parameter change, it was feasible to successfully steal the funds locked in the protocol covertly and within two blocks or within two transactions. By exploiting recent flash loan pool contracts, we show how an attacker with no capital (besides gas fees) would have been able to execute such an attack if the flash loan pools would provide sufficient liquidity (which they did not at the time of writing).

- **Formal modelling of DeFi lending protocols (Section 4.3).** We provide definitions for economically-resilient DeFi lending protocols, introducing overcollateralisation, liquidity, and counter-party risk as formal constraints. These definitions serve to formalise financial risk constraints for more than 93% of the funds locked in DeFi lending protocols as of April 15th, 2020 [Pul19a].
- **Financial stress-testing (Section 4.4).** We develop a methodology to quantitatively stress-test a DeFi protocol with respect to its financial robustness, inspired by risk assessments performed by central banks in traditional financial systems. We simulate a price crash event with our stress-test methodology to a stylised DeFi lending protocol that resembles the largest DeFi lending protocols to date by volume: Maker, Compound, Aave and dYdX. We find, for plausible parameter ranges, that a DeFi lending protocol could become undercollateralised within 19 days.

4.2 Governance attack on Maker

In this section, we first present an attack on the governance mechanism of the Maker protocol [Fou19]. We use a representation of the state of the Ethereum main network on February 7th and the Maker contract to simulate how such an attack could take place as realistically as possible. While focusing on a specific protocol, we submit that such a governance attack represents a new element of the attack surface for DeFi protocols more generally. Since we first analysed this attack vector, the Maker protocol has been modified to mitigate this attack: we detail our interaction with the Maker team below. Although the basic idea of the attack had been briefly presented in a blog post [Mic19], the *feasibility* has not been analysed.

4.2.1 Disclosure to Maker

We engaged in a process of responsible disclosure with Maker, as detailed below.

- On February 7th, 2020, we reached out to the Maker team regarding our exposition of the feasibility of the governance attack.
- On February 14th, the authors had a conference call with Maker, where we described our work. We agreed to give the Maker team sight of this chapter before publicising it; we subsequently sent a draft on February 17th.
- On February 18th, the authors further contacted Maker to describe how the use of flash loans increased the risk of the governance attack, offering a response window before publicising this result, within which Maker provided helpful feedback.

After this exchange, on February 21st, Maker announced that the Governance Security Module had been activated, implementing a delay period before proposals took effect of 24 hours [Mak20], mitigating the vulnerability.

4.2.2 Background and threat model

The governance process relies on the MKR token, where participants have voting rights proportional to the amount of MKR tokens they lock within the voting system. MKR can be traded on exchanges [Coi20a].

Executive voting. Using executive voting, participants can elect an *executive contract*, defining a set of rules to govern the system by staking (i.e., locking up) tokens on it. Executive voting is continuous, i.e., participants can change their vote at any time, and a contract can be newly elected as soon as it obtains a majority of votes. The elected contract is the only entity allowed to manipulate funds locked as collateral. If a malicious contract were to be elected, it could steal all the funds locked as collateral.

Defense mechanisms. Several defence mechanisms exist to protect executive voting. The *Governance Security Module* encapsulates the successfully elected contract for a certain period, after which the elected contract takes control of the system. At the time of first writing on February 7th, this period was set to zero [Wee20]. This has subsequently been increased to 24 hours [Mak19a], see Section 4.2.1. The *Emergency Shut Down* allows a set of participants holding a sufficient amount of MKR to halt the system. However, this operation requires a constant pool of 50k MKR tokens, worth 27.5M USD as of February 7th.[3]

Threat model. We assume the existence of a rational adversary, i.e., one who would only engage in the attack if the potential returns are higher than the costs. In this attack, the costs are the amount of money that the adversary has to pay to have their contract elected as the executive contract. The returns are the amount of money the contract could steal or generate once elected. There are two ways in which electing an adversarial executive contract can financially benefit the adversary. First, the contract can transfer all the ETH collateral to the adversary's address. Second, the contract can mint new DAI tokens and transfer them to the adversary. The DAI tokens can then be traded until the DAI price crashes and effectively destroy the Maker system.

As of February 7th, there were c. 150k MKR tokens used for executive voting, and the current executive contract had 76k MKR tokens staked. We observed that the staked amount changes relatively often, and the number of tokens staked on the elected contract often dropped below 50k MKR tokens (eq. 27.5M USD). As of February 7th, there was c. 470M USD worth of ETH locked as collateral of the DAI supply, which an executive contract can dispose of freely. This shows that the attack would have been financially attractive even before trading the DAI tokens.

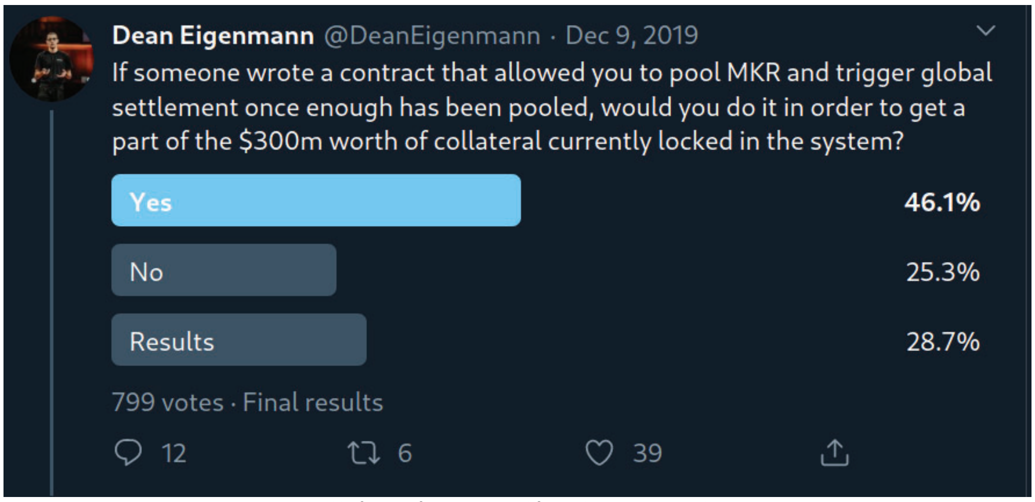
4.2.3 Crowdfunding and flash loans

An adversary can choose between the following strategies to amass the capital required for the governance attack.

Crowdfunding. Crowdfunding MKR tokens may allow users to lock their tokens in a contract and program the contract so that when the required amount of MKR tokens is reached, it stakes all its funds on a malicious executive contract. This would allow multiple parties to collaborate trustlessly on such an attack while keeping control of their funds and being assured they will be compensated for their participation. [4]

[3]We use the price of MKR on 2020-02-01, which was 550 USD.

[4]An (admittedly informal) poll on Twitter from late 2019 conducted by a user soon after this attack first appeared shows that several participants would be interested in such crowdfunding. See Figure 4.1.


Dean Eigenmann @DeanEigenmann · Dec 9, 2019

If someone wrote a contract that allowed you to pool MKR and trigger global settlement once enough has been pooled, would you do it in order to get a part of the $300m worth of collateral currently locked in the system?

Yes	46.1%
No	25.3%
Results	28.7%

799 votes · Final results

Figure 4.1: Twitter poll for of a crowdfunding attack on MKR governance.

Liquidity pools and flash loans. A shortcoming of the crowdfunding attack is the required coordination effort between the participants and the likely alerting of benevolent MKR members. Instead, an attacker could use liquidity pools offering *flash loans* [Aav20c]. A flash loan is a *non-collateralised* loan valid only within one transaction. In the Ethereum Virtual Machine (EVM), a transaction can be reverted entirely if a condition in one part of the transaction is not fulfilled. A flash loan then operates as follows: a party creates a smart contract that (i) takes out the loan, (ii) executes some actions, and (iii) pays back the loan and, depending on the platform, interest.

The interesting aspect for our purposes is that if step (ii) the execution of the actions fails or step (iii) the loan repayment cannot be completed, the EVM treats this loan as if it never took place. Hence, under the assumption there is enough liquidity available in protocols such as dYdX [dYd20] and Aave [Aav20b], an attacker could execute the MKR governance attack in step (ii), and, if successful, repay the flash loan in step (iii). Since the flash loan requires no collateral, the capital lock-up cost for the attacker is significantly reduced. If enough liquidity is available in these pools, the attacker might not have to lock any tokens. Furthermore, the liquidity provider may have also profited from the execution of the attack, depending on in which protocol their tokens were locked. For example, in Aave, as of February 7th, they would have received an interest rate of 0.09% for each flash loan.

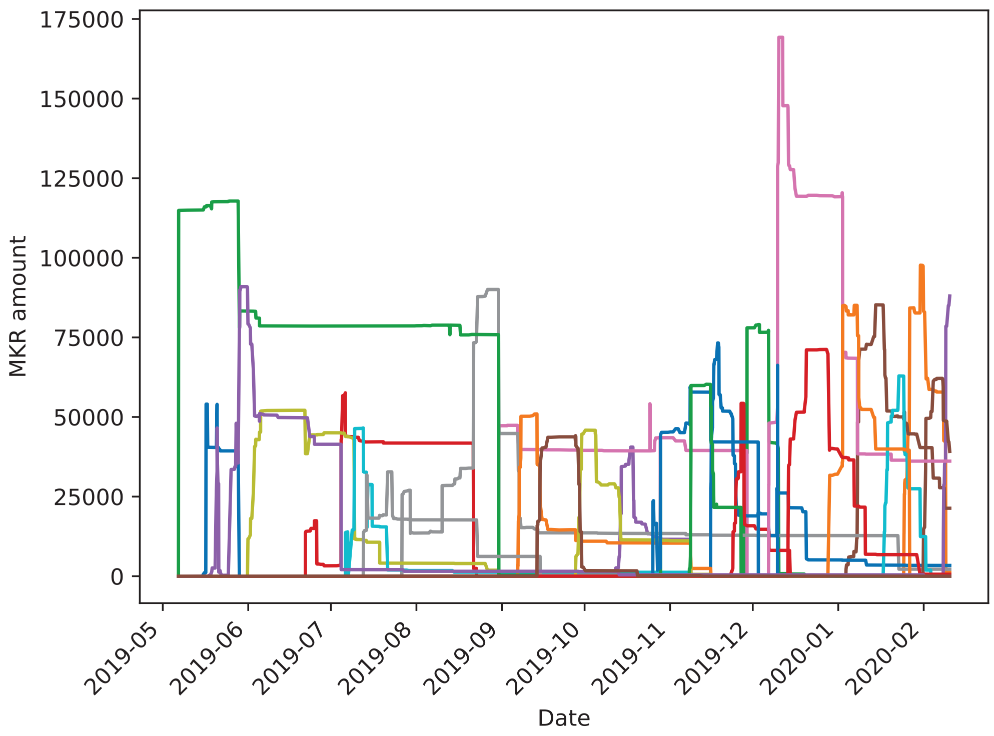


Figure 4.2: Evolution of the number of MKR tokens staked on different executive (i.e., governing) candidate contracts. We observe that at times the MKR amount of the executive contract dropped below 50k MKR.

4.2.4 Practical attack viability

In this section, we use empirical data to show how such an attack could take place and describe potential shortcomings. We first analyse all the transactions received by Maker's governance contract of as February 7th: 0x9ef05f7f6deb616fd37ac3c959a2ddd25a54e4f5. Since the deployment of this contract in May 2019, there were 24 different contracts elected as the executive contract (see Figure 4.2).

When a contract is elected as the executive contract, the total amount of staked MKR is, for a short period, distributed almost equally between the old and the new executive contract.[5] This reduces the number of tokens required for the attack by more than 50%.

One day after the first blog on this attack was published [Mic19], there was a sharp increase in the MKR staked on the executive contract, rising from c. 75k to c. 160k MKR at the beginning of December 2019. One token holder [i3n20] in particular injected a large number of tokens—

[5]This was particularly visible at the end of November 2019 (80k MKR to 40k MKR) and in the middle of January 2020 (120k MKR to 45k MKR).

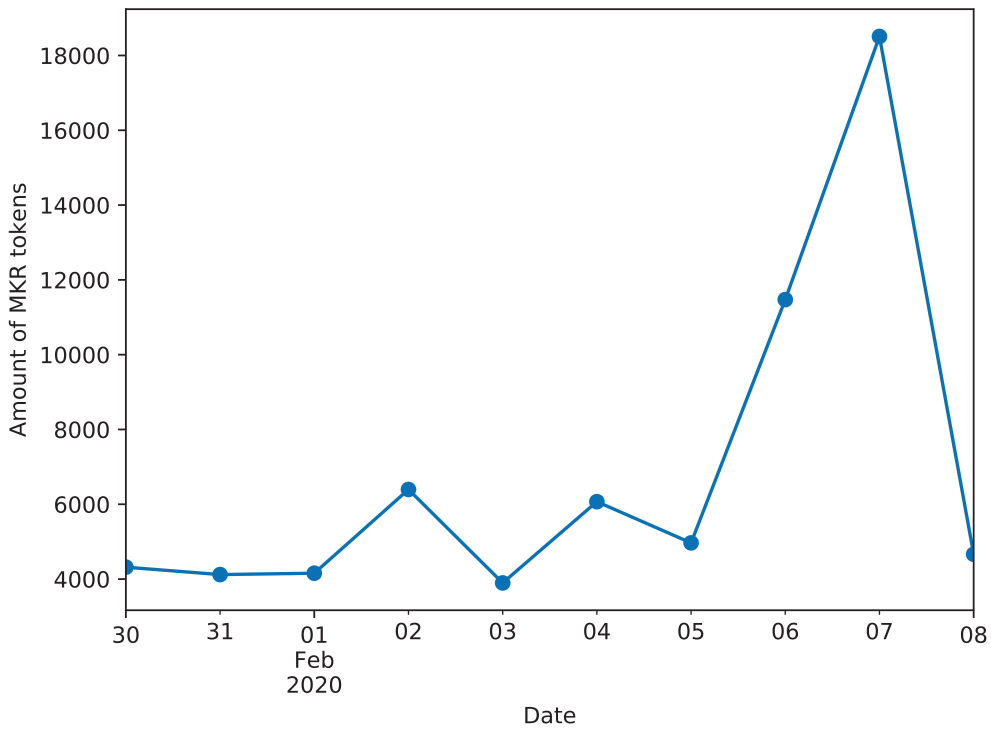


Figure 4.3: Daily traded volume of MKR tokens between 2020-01-30 and 2020-02-08.

c. 66k MKR—potentially to help prevent an attack from occurring. [6]

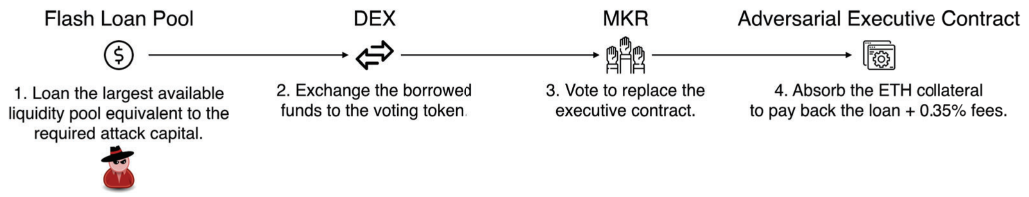
4.2.5 The attacks

The crowdfunding strategy. We inspected the amount of MKR transferred between January 1st, 2020 and February 8th, 2020. See Figure 4.3.

We find a mean MKR transaction volume of c. 9k MKR tokens per day, corroborated by e.g. [Coi20a]. Given such volumes, an attacker accumulating 1k MKR tokens per day, for instance, would have sufficient tokens (i.e., more than 50k MKR) in less than two months. However, accumulating all the tokens in a single account would attract attention. Indeed, from our discussions with the Maker team, the large MKR token holders are known.

To be covert, an attacker could try accumulating tokens to multiple accounts without perceptibly changing the distribution of MKR tokens. On February 8th there were c. 5k accounts,

[6]It is unclear if the token holder was the Maker Foundation or some other party; in our discussion with Maker, they stated they knew the identity of the token holder. The holder staked their tokens on the currently elected contract, making the attack more difficult to execute, before releasing the staked tokens approximately one month later. The token holder had more than sufficient tokens to execute the attack: were they malicious, they could have stolen the funds.



Flash Loan Pool
DEX
MKR
Adversarial Executive Contract
1. Loan the largest available liquidity pool equivalent to the required attack capital.
2. Exchange the borrowed funds to the voting token.
3. Vote to replace the executive contract.
4. Absorb the ETH collateral to pay back the loan + 0.35% fees.

Figure 4.4: Example flash loan attack against Maker. All steps can be executed within one transaction, assuming the flash loan pool and DEX have sufficient liquidity. To execute the attack, the adversary would not need upfront capital besides the gas fees (estimated to amount to c. 15 USD).

holding a total of c. 272k MKR tokens.[7] Given that the attack is possible with 50k MKR tokens, an adversary could spread their tokens across, e.g. 100 accounts with an average of 500 tokens each. However, one drawback of this approach is the requirement to vote from these 100 accounts. Voting for a contract costs, on average 69k gas. Given the gas limit per block on Ethereum is 10 million, filling half of a block with voting transactions would allow votes from 10M/69k $\approx$ 72 contracts. Doing so would be inexpensive [PL20], meaning that an attacker would have been able to perform the whole attack in two blocks easily. In the second block, the attacker would finish voting for his malicious contract and execute the attack from the contract, leaving only one block for anyone to react to the attack.

The flash loan strategy. Alternatively, to execute the governance attack without amassing tokens, the attacker could utilise liquidity pools to borrow the required tokens via a flash loan (e.g. via dYdX [dYd20] or Aave [Aav20c]). [8] The attacker makes two transactions (see Figure 4.4).

Transaction 1: Deploy the malicious governance contract and deploy the attack contract executing the flash loan.

Transaction 2: Call the attack contract deployed in Transaction 1 that executes the following steps.

1. Take out flash loan(s)(e.g. from Aave and dYdX) in the currency with the deepest markets for buying MKR tokens. As of February 7th, this was ETH.
2. Sell the ETH loan for 50k MKR tokens on decentralised exchange(s).

[7]Excluding the holders with a low balance (less than 1 MKR token), and a large balance (more than 5k MKR tokens).

[8]Aave is a protocol deployed on the Ethereum mainnet on January 8th, 2020, https://etherscan.io/tx/0x4752f752f5262fb11733e0136033f7d53cdc90971441750f606cf1594a5fde4f.

3. Vote with the 50k MKR tokens to replace the current Maker governance contract with the malicious contract deployed in Transaction 1.
4. Mint DAI into an account chosen by the attacker.
5. Take out enough ETH from the Maker system to repay the flash loan.
6. Repay the flash loan with the required 0.09% interest to Aave and repay the flash loan to dYdX with minimal (1 WEI) interest.

In a naive approach, we could utilise the exchange rate for ETH to MKR to obtain that an attacker requires 114,746 ETH to execute the attack. However, in practice, an attacker seeking to buy such a large quantity of MKR tokens would pay a greater price than this, forced to buy the tokens at the best remaining market price for each unit. As of February 14th, an attacker sourcing the required 50k MKR tokens from three different DEXs—38k MKR from Kyber, 11,500 MKR from Uniswap, and 500 MKR from Switcheo—would need a total of 378,940 ETH, 3.3x that of the naive estimate. [9] As of February 14th, the flash loan providers had insufficient pool liquidity: dYdX had c. 83,590 ETH and Aave had c. 13,670 ETH. However, on February 14th, the ETH growth rate of Aave was 5.18% per day. Assuming the growth rate continued, it would have only taken *66 days* until enough liquidity was available in Aave.

4.2.6 Profitability analysis

The crowdfunding strategy. With the crowdfunding strategy, the profits from the attack could be split equally between the funding parties. The only cost is the 20 USD for including the transactions [PL20]. In return, the attackers could take away the 434,873 ETH in collateral in Maker plus 145m DAI, amounting to a net profit of 263m USD (as of February 7th). Additionally, the attackers could mint unlimited new DAI and use this to buy other cryptocurrencies available at centralised and decentralised exchanges.

The flash loan strategy. Assuming dYdX's and Aave's liquidity pools had accumulated the required 378,940 ETH to execute the attack, we can calculate the profitability as follows. The attacker obtains a total of 434,873 ETH in collateral from Maker as well as the 50k MKR tokens and the 22m DAI currently in circulation. The attacker needs to repay the 378,940 ETH loan

[9] For current liquidity and rates see https://dexindex.io/.

with minimal interest (1 WEI for dYdX and 0.09% for Aave (265.82 ETH)). Furthermore, the attacker must pay the gas fees for the two transactions. The second transaction involves various function calls to other contracts and will cost c. 15 USD equivalent of gas. However, by the end of the attack, the attacker has c. 55k ETH, 50k MKR, and 145m DAI. This amounts to a net profit of 191m USD. Moreover, the attacker can design the attack smart contract to revert the transaction if it becomes unprofitable. This makes the attack *risk-free* from a cost perspective for the attacker. As pointed out, the attacker can create unlimited DAI to buy up existing liquidity on decentralised and centralised exchanges.

4.3 DeFi lending protocols

After presenting a specific attack vector, we now turn to a generalisation of the financial risk for DeFi lending protocols. This section provides a formal system model for a DeFi lending system and characterises system constraints. Table 4.1 details the parameters for existing DeFi lending protocols we seek to generalise in this section.

4.3.1 DeFi lending protocol model

Overcollateralised borrowing allows an agent to provide asset A as collateral to receive or create another asset B, of lower value, in return. Asset B, typically with the payment of a fee, can be returned, and the agent redeems its collateral in return. However, borrowed asset B may have different properties to asset A: for example, an agent might provide a highly volatile asset A and receive a price-stable asset B in return. Furthermore, a third asset, C, can serve as a governance mechanism, such as MKR. Asset C Holders can influence the rules of the DeFi lending protocol. In the absence of a governance asset, DeFi lending protocols typically replace this function with a central privileged operator introducing counterparty risk.

At the agent level, a DeFi lending protocol permits agents $k \in K$ to escrow cryptocurrency i, c_i and borrow (or issue) units of another cryptocurrency d against that value. We formulate the constraints herein such that $i \in I$, where I denotes all the permissible collateral types. Table 4.1 provides the collateral assets and liquidation ratios for DeFi protocols that account for 93% of the DeFi lending market. The prices of escrowed and borrowed assets are typically quoted regarding an agreed quote currency, e.g. USD.

At the system level, a DeFi protocol is the aggregation of the individual acts of borrowing by agents, such that the system collateral of type i is given by $C_i = \sum_{k=1}^{K} c_{i,k}$ for K agents. We formally define an economically secure DeFi lending protocol as follows:

Definition 4.1 (Economically Secure DeFi lending protocol)**.** *Assuming rational agents, a DeFi lending protocol is economically secure if it ensures that $\forall t$, with reference to a basis of value (e.g. USD), the total value of the system debt D at time t is smaller than the total value of all backing collateral types I ($\sum_{i=1}^{I} C_i$) at time t.*

4.3.2 Economic security constraints

We now provide three constraints on the economic security of a DeFi lending protocol. These constraints apply to DeFi protocols that feature one or several collateral assets and may also feature a reserve asset.

The overcollateralisation constraint. Since the values of both the collateral assets and debt are subject to price fluctuations, overcollateralisation seeks to ensure that there is *always* sufficient collateral to cover the debt, i.e., to avoid insolvency.

Definition 4.2 (Overcollateralisation)**.** *Escrowed collateral c_i has a greater value with respect to a basis of value than an issued loan d.*

Denoting the overcollateralisation factor as $\lambda_i \geq 0$, such that each collateral type has its minimum collateralisation ratio, and the price and quantity of an asset as $P()$ and $Q()$ respectively, the margin M of overcollateralisation at time t at the system level[10] is as follows (summing over agents $k \in K$ and collateral types $i \in I$). A protocol designer faces a trade-off. If the parameter λ_i is too low, volatile markets may mean the protocol becomes undercollateralised.

[10]The system level perspective looks at the aggregates of assets and liabilities; depending on the protocol the ability to use one asset to cross-subsidise an undercollateralised another asset may be restricted.

However, if it is too high, there is significant capital market inefficiency, with more capital than necessary in escrow, leading to opportunity costs.

$$M_t = (1 + \lambda_i) \sum_{k=1}^{K} \sum_{i=1}^{I} P_{c_{i,k,t}} Q_{c_{i,k,t}} - \sum_{k=1}^{K} d_{k,t} \tag{4.1}$$

Clearly, $M_t \geq 0 \iff \sum_{k=1}^{K} d_{k,t} \leq (1 + \lambda_i) \sum_{k=1}^{K} \sum_{i=1}^{I} P_{c_{i,k,t}} Q_{c_{i,k,t}}$. Should $M < 0$, the margin of overcollateralisation is negative, and therefore, the system as a whole is undercollateralised.

In addition, a protocol may have another pool of reserve liquidity available, enabling it to act as a lender of last resort.[11] For example, one such pool of collateral could be constituted by governance tokens Π for the protocol itself.[12] In a DeFi protocol, participants can have voting power proportional to the number of governance tokens they hold. The total value of this pool of collateral is given by $P(\Pi)Q(\Pi)$, and thus adding this into the margin of overcollateralisation for the system yields:

$$M_t = (1 + \lambda_i) \sum_{k=1}^{K} \sum_{i=1}^{I} P_{c_{i,k,t}} Q_{c_{i,k,t}} + P_{\Pi,t} Q_{\Pi,t} - \sum_{k=1}^{K} d_{k,t} \tag{4.2}$$

Therefore, at the system level, the necessary condition for economic security in terms of over-collateralisation is $M_t \geq 0$. In the event that $(1 + \lambda_i) \sum_{k=1}^{K} \sum_{i=1}^{I} P_{c_{i,k,t}} Q_{c_{i,k,t}} < \sum_{k=1}^{K} d_{k,t}$, the reserve asset Π of a protocol can be used as a *lender of last resort* to buy the collateral value. If $M < 0$, even the liquidation of the primary collateral and reserve assets would be insufficient to cover the total system debt. Since the collateral and reserve assets may be correlated[13], the ability of a reserve asset to recapitalise a system may be limited in the event of a sharp price drops. Without additional protocol-specific defence mechanisms, this would constitute a catastrophic system failure since the borrowed funds would become worthless as they would no longer be redeemable.

The liquidity constraint. In an illiquid market, liquidating a collateral asset may only be possible with a significant *haircut*, where the collateral is sold at a discount. Following [Nik09], we define market liquidity as follows.

[11] If a protocol does not have this, $Q_{\Pi,t} = 0$.

[12] MKR tokens in the case of Maker.

[13] There is evidence that crypto-assets display high intra-class correlation, limiting the advantage of diversification [KPA+19].

Definition 4.3 (Market liquidity)**.** *A measure of the extent to which a market can facilitate the trade of an asset at short notice, low cost and with little impact on its price.*

The liquidity available in a market implies a security constraint: in expectations, over a certain time horizon, DeFi marketplaces can offer enough liquidity that in the event of a sustained period of negative price shocks, a protocol will be able to liquidate its collateral quickly enough to cover its outstanding debt liabilities.

For a time interval $[0, T]$ this can be expressed as:

$$\int_0^T \mathbb{E}[\Omega] d\Omega \leq \mathbb{E}[\Omega_{max}] \tag{4.3}$$

where Ω denotes the total notional traded value, i.e., the (average) price multiplied by the quantity for each trade. For a given trade ω of size q, $\omega = \bar{p}q$; aggregating these trades for a total number of trades J provides $\Omega = \sum_{j \in J}^{J} \omega_j$. Ω_{max} denotes the maximum notional value that could be sold off during a period of distress in the financial markets.

In the event of a severe price crash, on the assumption that a protocol is collateralised to a representative 150%, we assume a protocol will seize 100% of the debt value from the collateral pool and seek to sell this collateral as quickly as possible on a market pair to the debt asset. Once a buyer has traded the debt asset d for the collateral, the protocol could burn the debt d, effectively taking it out of circulation and offsetting the liability. Therefore, negative price shocks' impact on a DeFi lending protocol and how quickly they materialise depend on the liquidity available on all collateral/debt pairs. In the event of a liquidity crisis, the demand for liquidity outstrips supply[14], such that equation (4.3) is binding. Indeed, if equation 4.3 is binding, there are not enough buyers in the market to buy the ETH for sale.

The counterparty risk constraint. DeFi lending protocols need to be fully decentralised because of, for instance, the possibility of Oracle attacks (which could cause a flash crash) and privileged access to smart contracts. Therefore it is necessary to either assume the "operator" of the protocol is honest or that the operator only offers the services of the protocol provided they are profitable for them. We formally model this counterparty risk by assuming that its existence in a given protocol creates a risk premium, ψ, such that for an agent deciding between earning a

[14]Indeed, such liquidity crises were at the heart of the Financial crisis of 2007-8, as the value of many financial instruments traded by banks fell sharply without buyers [Ell12].

return in a DeFi lending protocol vs elsewhere, the expected return in the DeFi protocol (r_D), once adjusted for the risk premium (ψ), must be higher than an outside return r_f. Formally, we have participation constraint $r_D - \psi > r_f$. This constraint is a participation constraint, and in Section 4.4, we assume that this inequality holds, such that agents have already chosen to participate in the protocol.

There exists an inherent trade-off in counterparty risk. On the one hand, governance mechanisms implemented through voting allow for a certain degree of decentralisation whereby multiple protocol participants can influence the future direction of a protocol. Depending on the distribution of tokens, this may reduce the risk of one party becoming malicious. However, it also opens the door to attacks on the voting system, as we introduced in Section 4.2. On the other hand, a single 'benevolent dictator' who controls the governance mechanism can prevent the attacks introduced in Section 4.2. Yet this requires trusting that this central entity does not lose or expose its private keys controlling access to the smart contracts governing the protocol and that this central party cannot be bribed to behave maliciously.

4.4 Stress-testing DeFi lending

This section considers the financial security of a generic DeFi lending protocol, stress-testing the architecture to quantitatively assess its robustness as inspired by central banks [oE19, Res19].

4.4.1 Stress-testing framework

Central banks conduct stress tests of banking systems to test their ability to withstand shocks. For example, in an annual stress test, the Bank of England examines the potential impact of an adverse scenario on the banking system [oE19]. The hypothetical scenario is a "tail-risk" scenario, which seeks to be broad and severe enough to capture a range of adverse shocks. Following such best practices, we devise and implement a stress test of the DeFi architecture.

4.4.2 Simulation approach

We leverage the generic DeFi lending protocol architecture developed in Section 4.3.1. We focus on a single collateral asset here for tractability. Still, this analysis can be extended to lending protocols which rely on overcollateralisation by multiple volatile collateral assets in combination with reserve assets.

Protocol	**Collateral asset (liquidation ratio)**	**Reserve asset**
Maker [Fou19]	ETH (150%), BAT (150%), USDC (125%)	MKR
Compound [Com19]	ETH (133%), BAT (167%), DAI (133%)	
	REP (250%), USDC (133%), ZRX (167%)	
Aave [Aav20c]	DAI (125%), USDC (125%), TUSD (125%)	
	ETH (125%), LEND (154%), BAT (154%)	
	KNC (154%), LINK (143%), MANA (154%)	
	MKR (154%), REP (154%), WBTC (154%)	
	ZRX (154%)	
dYdX [dYd20]	ETH (115%), USDC (115%), DAI (115%)	

Table 4.1: Parameters of DeFi lending platforms, comprising 93% of DeFi market as of April 15th, 2020.

In part reflecting Table 4.1, we make the following assumptions about the system's initial state.

1. The lending protocol allows users to deposit ETH as their single source of collateral c_i.

2. The lending protocol has 1m tokens of a generic reserve asset. The simulation's start has the same price as ETH but with exactly half of the historical standard deviation of ETH taken over the sample period.

3. By arbitrage among borrowers, before the crash, the lending protocol is collateralised to $\lambda_i + \epsilon$, i.e., just above the minimum collateralisation ratio.

4. At the start of the crisis, the protocol had a collateralisation ratio of exactly 150%, such that every USD of debt is backed by 1.50 USD of collateral.

5. Each unit of debt d^k maintains a peg of 1:1 to the US dollar, allowing us to abstract from the dynamics of maintaining the peg.

6. At the start of the sell-off, it is possible to sell 30,000 ETH per day without impacting the price.[15].

[15]This assumption is based on the 24-hour volume of ETH/DAI across markets listed on CoinGecko on February 7th, 2020 and, as such, is only a rough proxy for the market liquidity. We use this figure only as a parameterisation baseline and highlight the theoretical possibility of illiquidity causing a default.

7. The amount of reserve asset Π is fixed at the start of the sell-off at 1m units.

8. System debt levels range from 100m USD to 400m USD.

Next, we detail the methodology we follow to obtain our simulation results.

Price simulation. Firstly, as inputs into our model we obtain OHLCV data at daily frequency [Cry20], focusing on January 1st, 2018 to February 7th, 2020, incorporating the significant fall in the ETH price in early 2018.

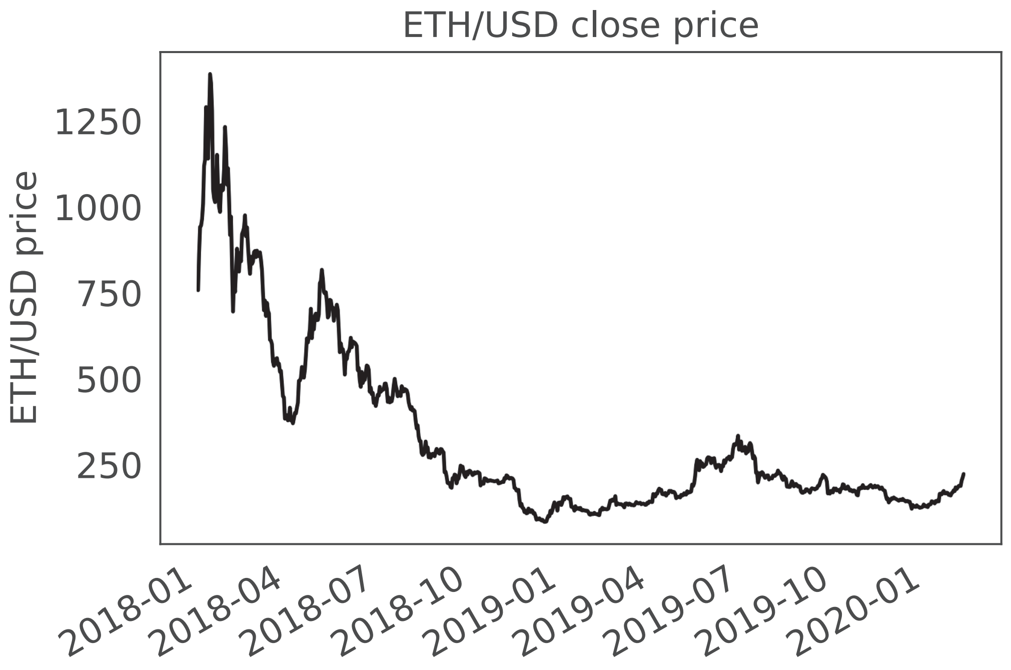


Figure 4.5: Close prices for ETH/USD over the period January 1st, 2018 to February 7th, 2020.

We present the evolution of ETH close prices in Figure 4.5 and a histogram of log returns in Figure 4.6. The most notable element is the decline in the ETH/USD price over the course of 2018, with the price of ETH falling from an all-time high of 1,432.88 USD to c. 220 USD as of February 7th, 2020. Taking parameters from this historical data[16], we use Monte Carlo simulation to capture how the ETH and reserve prices may evolve over the next 100 days. Monte Carlo simulation leverages randomness to produce a range of outcomes of a stochastic system. We simulate 5,000 randomly generated paths using a geometric Brownian motion, specified with the following equation.

[16]For the daily ETH/USD price data, we find mean log returns of -0.001592 and standard deviation 0.050581, parameter values which have been independently verified.

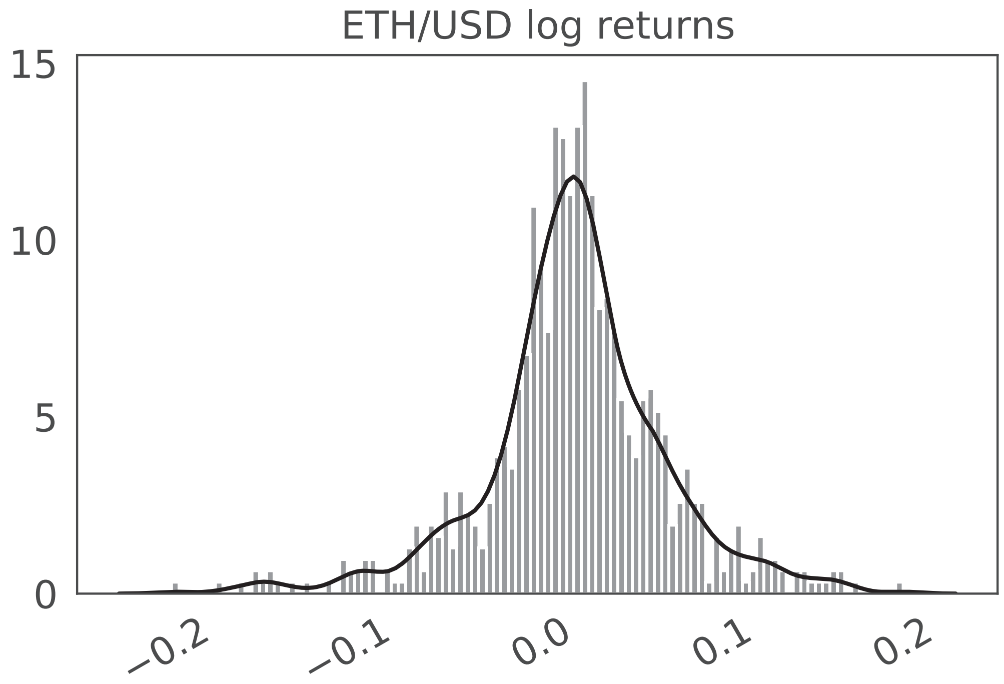


Figure 4.6: Broadly symmetric log returns for ETH/USD over the period January 1st, 2018 to February 7th, 2020.

$$P_{c_{i,t}} = P_{c_{i,0}} \exp\left[(\mu_i - \frac{\sigma_i^2}{2})t + \sigma_i W_t\right] \tag{4.4}$$

W_t denotes a Wiener process [Wie76] and for collateral type i μ_i denotes the drift and σ_i denotes the volatility.[17] Of the 5,000 simulations, our subsequent analysis is focused on the iteration which yields the fastest undercollateralisation event. By focusing on this worst-case, we test the DeFi lending protocol with a "black swan" event, representing a severe challenge to its robustness.

System simulation. We propose a simple model for the decline in liquidity over time as follows.

$$L = L_0 \exp(-\rho t) \tag{4.5}$$

where L_0 denotes the initial amount of ETH that can be sold daily and ρ is a coefficient. Intuitively, this equation captures the notion that if the protocol attempts to sell large volumes

[17]In this estimation, we draw shocks from the normal distribution, as is standard in GBM. Since performing a Jarque-Bera test [JB87] over the sample period suggests that the log-returns are non-normal, it is possible that, in our estimation, we underestimate the impact of heavy tails. Therefore, we present a best-case upper bound; in practice, undercollateralisation could precipitate more quickly.

each period, the liquidity available in the next period will be lower.

In this simulation approach, we simplify by not modelling the impact that selling large volumes of collateral will have on the price of the collateral asset. In such a sell-off scenario, selling large volumes would endogenously push the price lower. Therefore what we present here represents an upper bound on the price behaviour: in reality, the price drop may be even worse than the one we examine.

4.4.3 Simulation results

We start with the Monte Carlo simulation of the correlated asset paths before considering how this would impact a DeFi lending protocol and an ecosystem of multiple lending protocols.

Monte Carlo Price Simulation. To capture the effects of different correlations between the collateral asset and the reserve asset, we consider three different extents of correlation between the collateral and reserve asset: (i) strong, positive correlation (0.9), (ii) weak, positive correlation (0.1) and (iii) strong negative correlation (-0.9). We then generate correlated asset paths during the Monte Carlo simulation process. In this section, we report results for strong correlations but include those for weak and strong negative correlations.

Figure 4.7 shows the results of 5,000 runs of the Monte Carlo simulator for the ETH price, and Figure 4.8 shows the results for the reserve asset price in the presence of strong positive correlation in the asset price returns. The starting price of assets used in the simulator is the close price of ETH/USD on February 7th, 2020.

We isolate the simulation, which yields the fastest undercollateralisation event.[18]

In Figure 4.9, it is clear that in this worst-case scenario for the ETH/USD price, the price of the reserve asset similarly falls. This illustrates the risk of using a reserve asset which is positively correlated with the collateral asset: if the price of the collateral asset falls relative to the same basis of value, the reserve asset value is likely to fall, limiting the ability of a DeFi lending protocol to recapitalise itself.

Impact on Collateral Margin. We take the simulation yielding the fastest undercollateralisation

[18]We plot the co-evolution of the asset price paths for strong correlation in Figure 4.9.

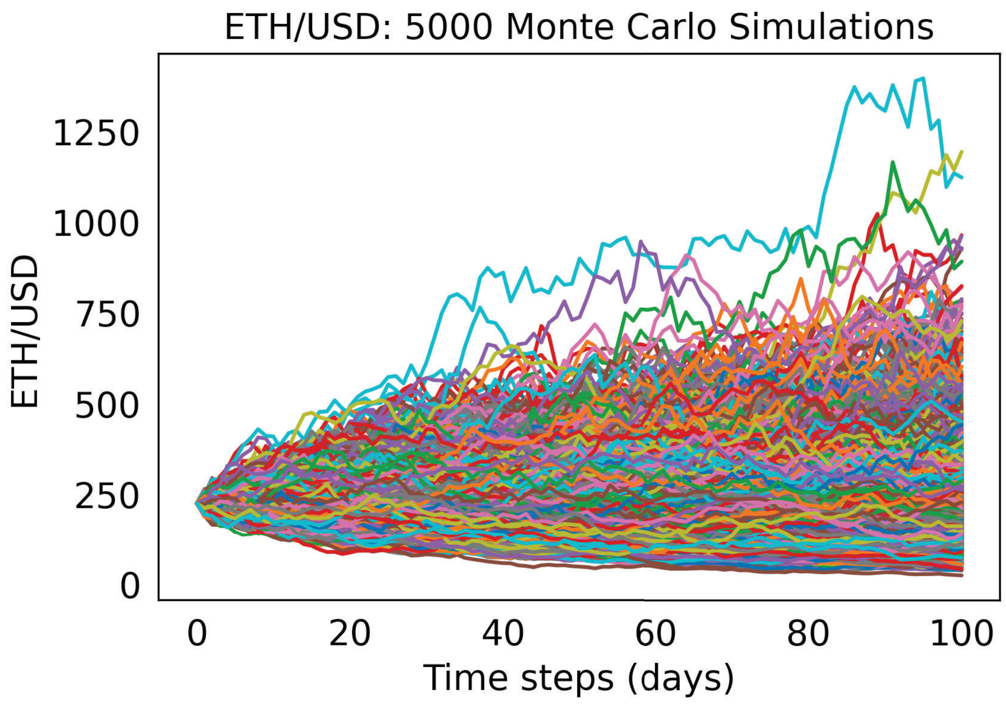


Figure 4.7: Monte Carlo forecast of ETH prices over the next 100 days from February 7th, 2020.

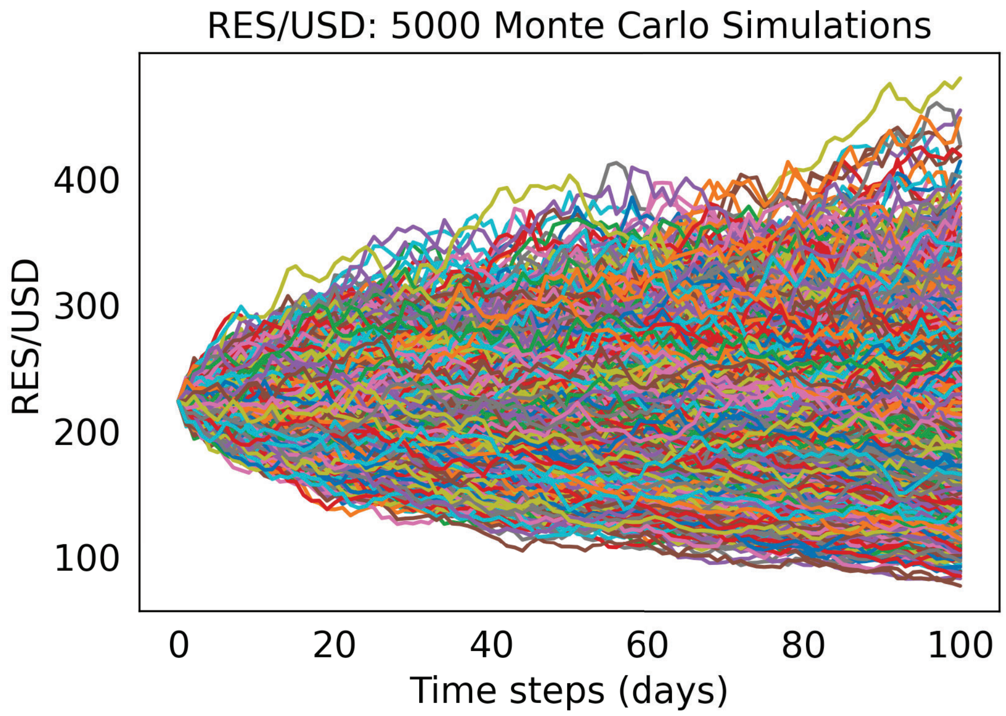


Figure 4.8: Monte Carlo forecast of the reserve asset price over the next 100 days from February 7th, 2020.

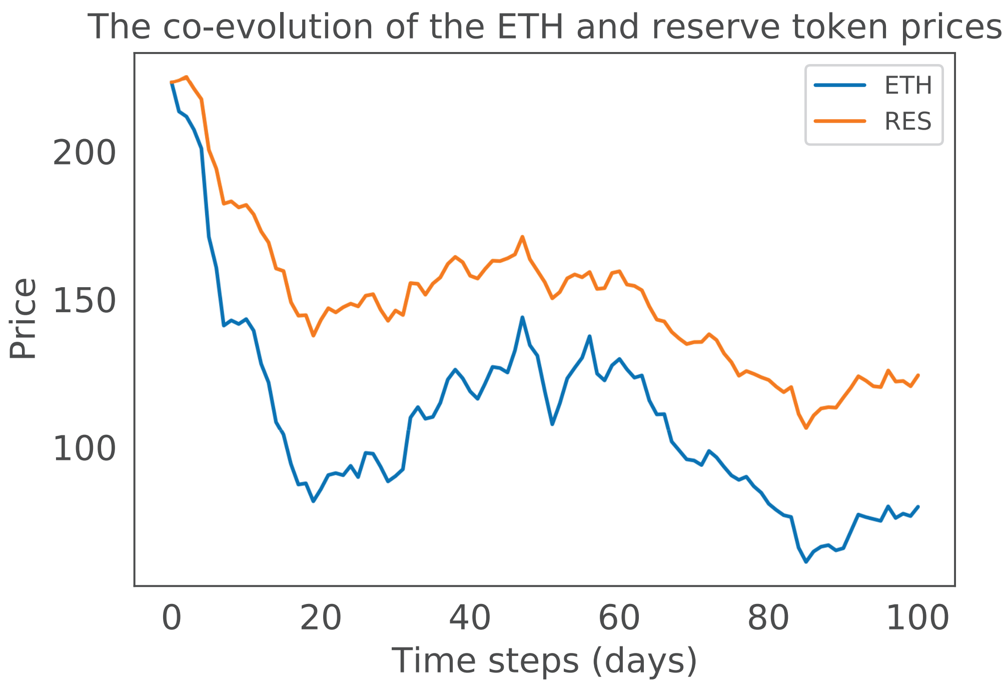


Figure 4.9: For the simulation yielding the fastest undercollateralisation event, the co-evolution of the ETH and reserve asset prices where the asset price returns are strongly positively correlated.

event and consider the impact this would have on the collateral margin of a DeFi lending protocol. The main results of this are presented in Figure 4.10.

Plotted with solid lines is the evolution of the total collateral margin (comprising the collateral and the reserve asset) over time as the prices of the collateral asset and reserve asset decline. The dashed lines indicate how the amount of system debt evolves through time, on the assumption that at the start of the 100 days, the protocol seeks to sell off all of the debt. The speed at which the debt can be liquidated through the sale of its backing collateral, in turn, depends on the available liquidity in the market for which we consider 3 cases:

1. Constant liquidity (such that it is possible to sell a constant amount of ETH every day at the average daily price)
2. Mild illiquidity (where the illiquidity parameter is arbitrarily set to some low level $\rho = 0.005$)
3. Illiquidity, such that $\rho = 0.01$.

Where the initial system debt level is 100m USD, regardless of the liquidity parameter, the collateral margin does not become negative. However, at higher debt levels, the margin gets

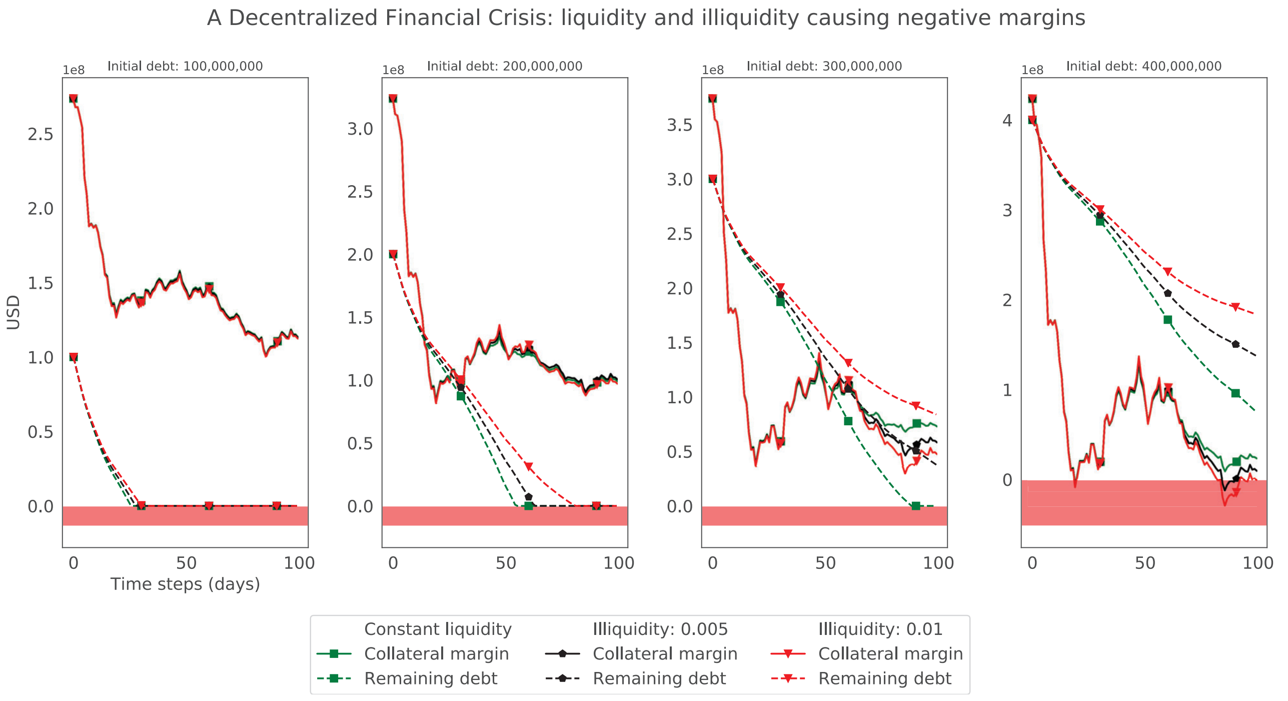


Figure 4.10: A DeFi lending protocol is experiencing a sharp decline in its collateral and reserve assets price. Panels correspond to four different levels of system debt, with each panel showing the evolution of the collateral margin (solid lines) and the total debt outstanding (dashed lines). Each panel also shows the consequences of different liquidity parameters. The margin becomes negative in panels 3 and 4— entering the red region below zero—the situation in which the lending protocol has become undercollateralised.

closer to 0. Once the debt level reaches 400m USD, the margin falls below 0, so the protocol is undercollateralised overall. In the fourth panel of Figure 4.10, we see that after 19 days of the protocol attempting to liquidate as much debt as possible, due to illiquidity, it is unable to liquidate in time, and the margin becomes negative. This would constitute a crisis in a DeFi protocol. Each unit of debt would not have sufficient collateral backing, and rational agents would walk away from the protocol without repaying their debt.[19] The results show that a weakly correlated reserve asset can slow or prevent the collateral margin from becoming negative (see Figure 4.11). In contrast, a strongly negative correlation between the assets can bolster the collateral margin (Figure 4.12).

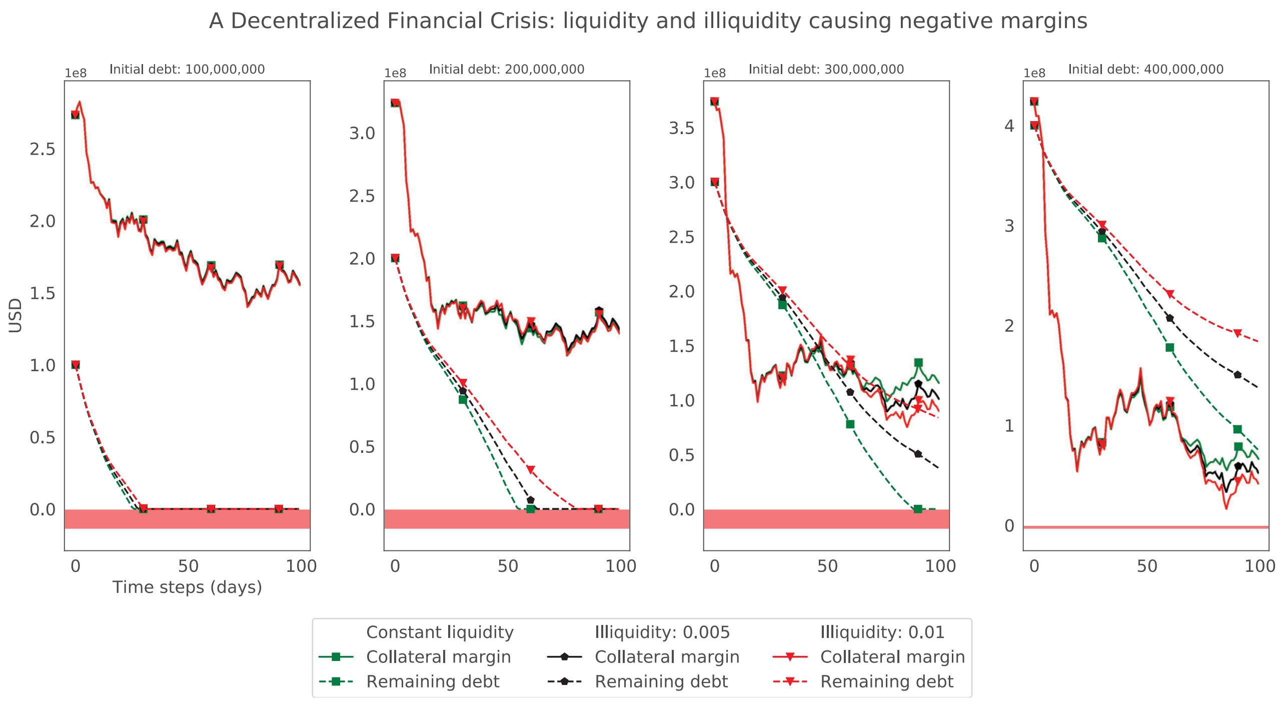


Figure 4.11: A DeFi lending protocol is experiencing a sharp decline in the price of its collateral and reserve assets, where the assets correlate 0.1. Panels correspond to 4 different levels of system debt, with each panel showing the evolution of the collateral margin and the total debt outstanding. Each panel also shows the consequences of different liquidity parameters.

For the case where the collateral and reserve assets are strongly positively correlated, we consider how *quickly* a crisis may materialise for varying starting values of ETH liquidity and initial debt in Figure 4.14. Figure 4.14 shows that for a given amount of debt, the lower the starting liquidity

[19]If strong identities (i.e., where the mapping between an agent and online identity is one-to-one and time-invariant) are enforced on-chain, this calculus may change for agents, reducing the probability of a crisis by increasing the costs to the agent of reneging on their debt commitments. In this chapter, we assume that strong identities are not enforced.

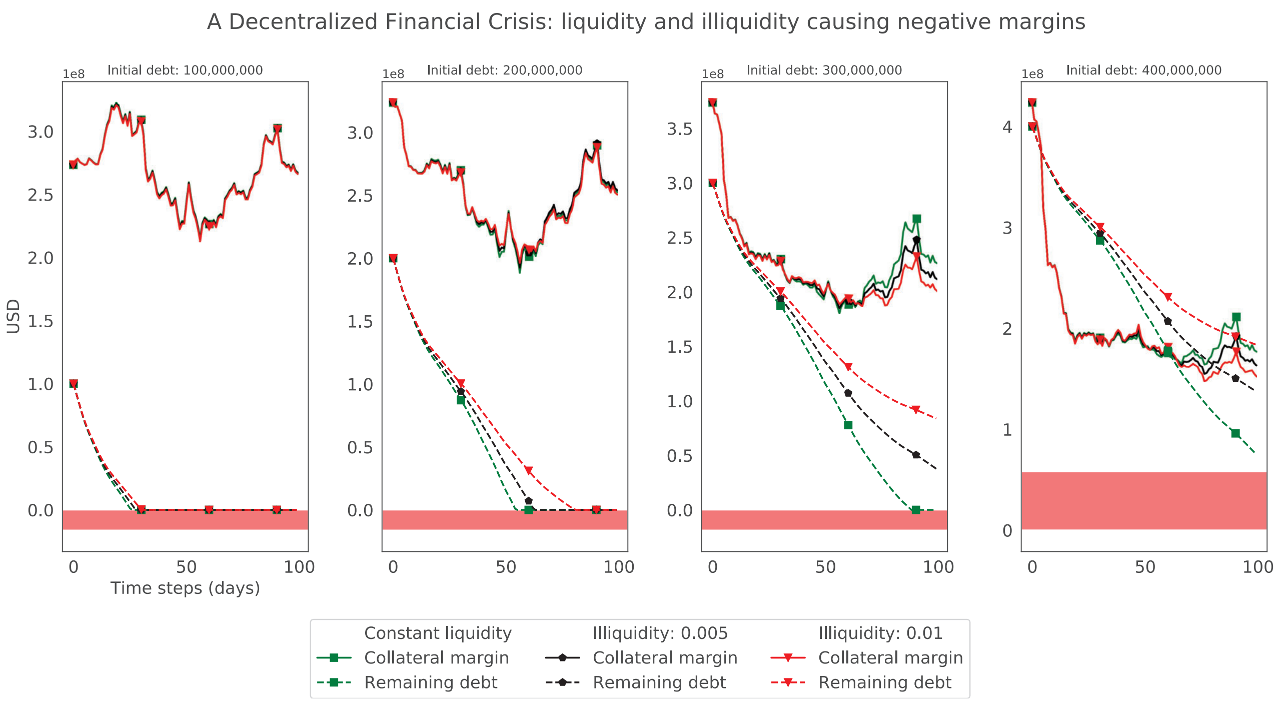


Figure 4.12: A DeFi lending protocol is experiencing a sharp decline in the price of its collateral asset with a negatively correlated reserve asset, where the assets correlate -0.9. Panels correspond to 4 different levels of system debt, with each panel showing the evolution of the collateral margin and the total debt outstanding. Each panel also shows the consequences of different liquidity parameters.

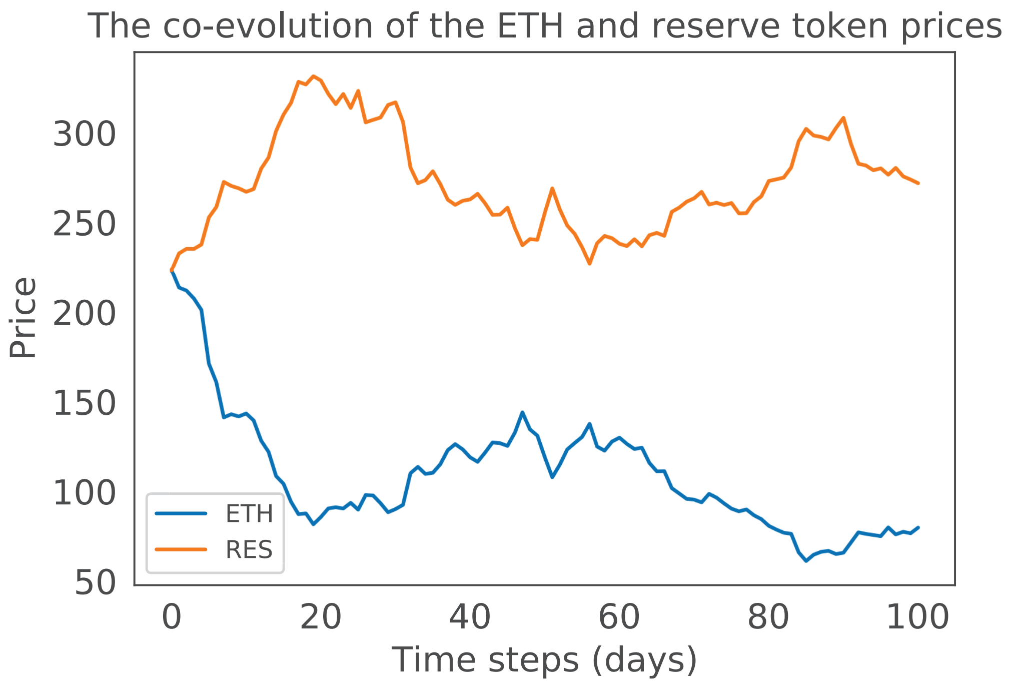


Figure 4.13: For the simulation yielding the fastest undercollateralisation event, the co-evolution of the ETH and reserve asset prices where the asset price returns are strongly negatively correlated.

(i.e., the amount that can be sold within 24 hours), the faster a negative margin precipitates. Similarly, for a fixed initial starting liquidity, the more debt in the system, the faster the margin will become negative, down to below 15 days.

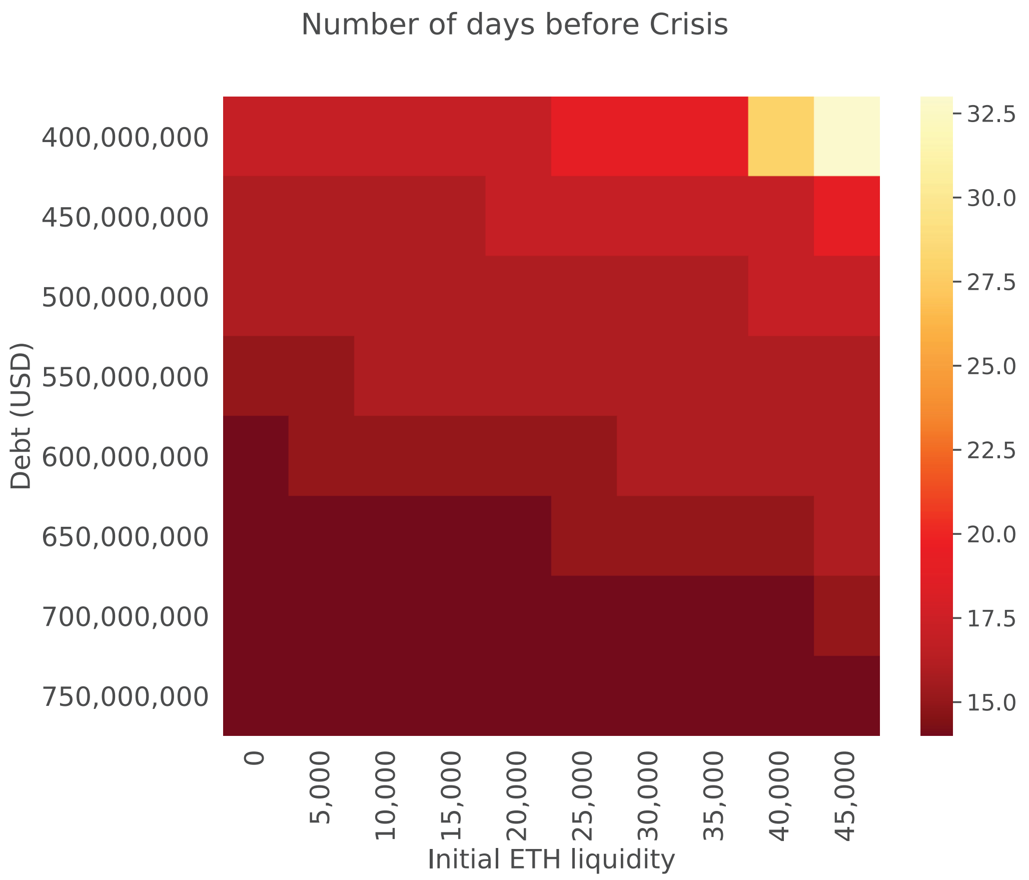
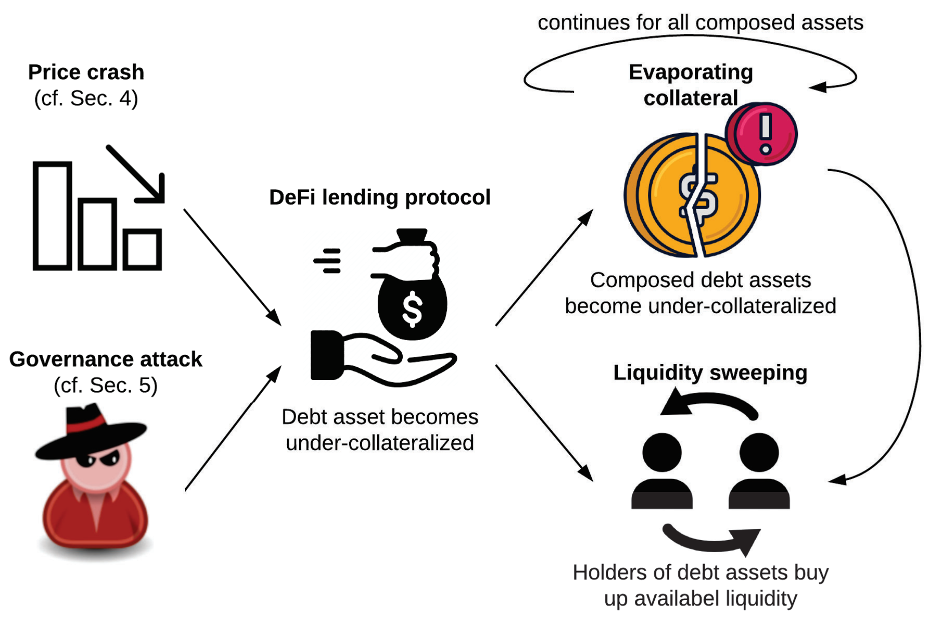


Figure 4.14: Number of days before the collateral margin becomes negative, depending on the amount of system debt and the initial amount of ETH that can be sold within 24 hours.

4.5 Composability and contagion

The possibility of financial contagion in DeFi is significant given the unrestricted composability of protocols [Pul19b]. In Sections 4.4 and 4.2, we considered an exogenous price drop and a governance attack in each case on a single protocol. Now we argue why these two design weaknesses can lead to contagion to other protocols that might ultimately lead to a decentralised financial crisis.

Both weaknesses, exogenous price drops and governance attacks result in the debt asset being



continues for all composed assets
Price crash
(cf. Sec. 4)
Evaporating collateral
DeFi lending protocol
Composed debt assets become under-collateralized
Governance attack
(cf. Sec. 5)
Liquidity sweeping
Debt asset becomes under-collateralized
Holders of debt assets buy up availabel liquidity

Figure 4.15: Both the price crash and the governance attack lead to an under-collateralisation of a DeFi lending protocol. In turn, agents will try to exit the protocol by buying other assets (liquidity sweeping). Any agent in a protocol using the now under-collateralised asset as debt will face the same situation and, in turn, try to exit by buying other assets (evaporating collateral).

under-collateralised. Assuming rational agents with weak identities, it is not individually rational for agents to settle their debts, and thus, the under-collateralised asset eventually reaches a value of 0. Hence, an agent holding such an asset can execute a strategy called *liquidity sweeping* to use his *existing holdings* to buy other assets while the price is not yet 0. A special sub-case occurs in the governance attack scenario where the attacker can additionally mint an unlimited supply of the debt asset to buy up *all* the available liquidity of other assets.

Further, any protocol that uses the now under-collateralised asset as a collateral asset to issue another asset also becomes under-collateralised. Consider the example of Maker and Compound: Maker uses ETH as collateral to issue DAI. Compound allows agents to collateralise DAI to issue a cDAI token. As DAI becomes under-collateralised, cDAI loses its debt-backing. We refer to this as *evaporating collateral*. Holders of composed assets like cDAI then have to buy up existing liquidity once they are aware that individually rational agents will not settle their debts. We illustrate this cycle in Figure 4.15 and give a total estimate of the financial damage in Table 4.2.

Liquidity sweeping. Contagion occurs when the under-collateralised debt asset is used to

Weakness	**Financial damage (USD)**
Under-collateralisation (price crash) of MakerDAO	145m
Under-collateralisation (governance attack) of MakerDAO	211m
Contagious under-collateralisation (price crash) of MakerDAO	180+m
Contagious under-collateralisation (governance attack) of MakerDAO	246+m

Table 4.2: Financial damage for an initial failure of the MakerDAO protocol either through a price crash or a governance attack.

"soak-up" as much liquidity as possible before buyers of the debt asset can react. There are two cases: First, if, as detailed in Section 4.4, the debt asset is under-collateralised due to an exogenous price crash, the agent can trade all their under-collateralised debt (such as DAI) for other assets on DeFi lending protocols which offer liquidity pools such as Compound [Com19], Uniswap [Pro20b], and Aave [Aav20c]. This way, the agent can dispose of the debt asset if it assumes the debt asset will fall to 0. Second, if, as detailed in Section 4.2, an attacker gains the ability to create new tokens without collateral backing, the attacker can use these tokens to buy *all existing liquidity* on available exchanges.

However, both cases assume that the price of the under-collateralised debt asset is not 0 at least as long to complete the trades. We support this assumption with the halting of the IOTA cryptocurrency network that occurred on February 13–15, 2020 [IOT20b]. Following a security incident, the IOTA foundation stopped the centralised coordinator. The IOTA network could not process transactions containing value and advised its users not to access their funds in their wallets [IOT20a]. During the incident, the trading volume on exchanges of IOTA fell from 41m USD to 21m USD Notably, the price of IOTA only moved from 0.34 USD to 0.31 USD within 14 hours of the announcement and recovered back to around 0.32 USD after another two hours. Hence, the inability to transfer the IOTA cryptocurrency and access funds on the IOTA network for an extended period caused only a maximum price drop of 9%.

To quantify the impact of liquidity sweeping, we took an instantaneous snapshot of major marketplaces offering a DAI pair at approximately noon GMT+8 on 15 February, as presented in Table 4.3. In the price crash case, agents could trade the whole of their $145m holdings. However, an agent who had successfully undertaken the governance attack would be able to soak up all of the order book liquidity since, with, in effect, an unlimited DAI supply, they can pay any price to acquire other assets. We estimate the total USD value, aggregating across markets and pairs, to be c. 211m USD[20].

Marketplace	**Total orders for DAI (USD)**
CoinbasePro [Dat20]	7,343,000
OasisDEX [MEGH20]	2,483,000
Kraken [Dat20]	6,085,000
Bitfinex [Dat20]	2,661,000
Bittrex [Dat20]	39,957,000
KuCoin [KuC20]	2,442,000
Probit [Lim20]	10,000
Switcheo [Swi20]	176,000
VCC Exchange [Exc20]	21,000
OkEx [OKE20]	11,000
Exmo [EXM20]	61,000
DDEX [DDE20]	46,000
Coinut [Coi20b]	6,000
Compound [Com19]	143,283,000
Uniswap [Pro20a]	2,176,000
Kyber [Pro20a]	2,176,000
Bamboo Relay [Pro20a]	979,000
Eth2DAI [Pro20a]	971,000
Airswap [Pro20a]	99,000
TOTAL	210,986,000

Table 4.3: Assuming DAI:USD peg maintained at 1:1, total USD liquidity on all DAI pairs by marketplace. Rounded to the nearest thousand.

Evaporating collateral. Liquidity sweeping is not contained to agents that participate in a single lending protocol if the assets of the protocol are used in others as well. Instead, a vicious cycle occurs where any debt-backed asset issued in another protocol, e.g., DAI would be used to issue cDAI on Compound, loaned in Aave, and used as a margin trading collateral in dY/dX. Considering the DAI example in the price crash case, 145m USD of DAI would be exchanged, 35m USD of cDAI and any other protocol that uses DAI as backing collateral.

Collateral composition. As a stylised example, assume that there are N protocols with different minimum collateralisation ratios, and each uses D/N as collateral. If this collateral becomes less than 100% collateralised, we assume that the value of the debt would quickly fall to 0. We further assume that agents choose a particular protocol for reasons exogenous to the system and that each agent obtains the maximum leverage they can on a particular protocol by using their issued debt to purchase additional collateral, which in turn is used to issue more debt. To reflect heterogeneity in collateralisation ratios among protocols, we assign each of the N protocols a collateralisation ratio based on random sampling from a uniform distribution over a specified range. The maximum possible leverage for a given protocol π when an agent allocates D/N units of collateral to it is given by $D/N_\pi/\lambda_\pi$. Therefore the maximum systemic loss is

[20]see Figure 4.3

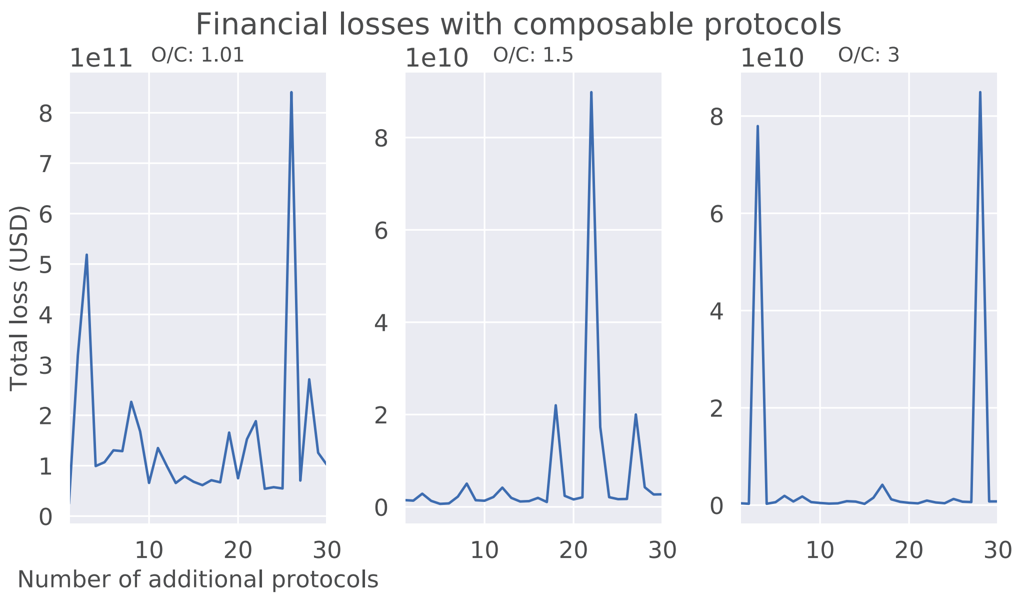


Figure 4.16: Financial losses for 30 DeFi protocols which use the debt asset as collateral. "O/C" stands for overcollateralisation.

given by $\sum_{\pi=1}^{\Pi} D/N_\pi/\lambda_\pi$.

Figure 4.16 demonstrates how for three different maximum overcollateralisation parameter ranges, namely (101–105%), (101–150%), and (101–300%), what the maximum system-wide losses could be, assuming that the protocol in which the crash occurs has a debt of 400m USD, and the liquidity parameter is $\rho = 0.01$. As the minimum overcollateralisation ratio falls, the maximum potential losses for the system grow: from right to left, as the minimum overcollateralisation requirement falls from $3\times$ (or 300%) to $1.01\times$ (101%), the maximum possible loss increases. Given our uniform splitting of debt, what also emerges is that the minimum collateralisation ratio drives the result.

Crisis. Agents seek to exit their under-collateralised asset positions by buying other available assets. Individually rational agents should seek assets uncorrelated with the asset they are disposing of. However, this leads to a spread of the initial DeFi lending protocol failure to any other asset available for trading on exchanges that accept the initial lending asset. Hence, the DeFi crisis can spread across multiple blockchains and affect centrally-backed assets like USDT [Tet20].

4.6 Related work

There is a paucity of directly related work. However, existing work can be divided into the following categories. A series of fundamental results in relation to the ability of non-custodial stablecoins to maintain their peg is provided in [KMM19]. It is shown that stablecoins face deleveraging spirals, which cause illiquidity during crises, and that stablecoins have *stable* and *unstable* domains. The model primarily involves the assumption of two types of agents in the marketplace: the stablecoin holder (who wants stability) and the speculator (who seeks leverage). The authors further demonstrate that such systems are susceptible to tail volatility. While unpublished, [CDK+18] uses option pricing theory to design dual-class structures that offer fixed-income stablecoins that are pegged to fiat currency. Further, [Pen18] considers how one might build an asset-backed cryptocurrency through the use of hedging techniques.

4.7 Conclusions

This chapter has sought to demonstrate that, as they stand, DeFi lending protocols are liable to various attack vectors. Firstly, we show the feasibility of an attack on the governance mechanism of Maker, finding that, before the fix implemented by Maker, provided an attacker was able to lock 27.5m USD of governance tokens, they would have been able to steal all 0.5bn USD worth of collateral within two blocks. Therein we presented a novel strategy that would have enabled an attacker to steal the collateral within two transactions without the need to escrow any assets.

Secondly, after providing formal constraints on the robust operation of a DeFi lending protocol, we use simulations to show that a DeFi lending protocol may become under-collateralised for given parameters. We describe the interrelation of market liquidity and outstanding debt, showing how the larger the debt, or the less liquid a market, the faster insolvency can occur. We also consider different levels of correlation between the collateral and the reserve asset in a DeFi lending protocol and show that having a reserve asset that is weakly positively correlated or indeed negatively correlated can help to ensure protocol solvency.

These two failure modes in a DeFi protocol are potentially mutually reinforcing. If the collateral and reserve assets of a DeFi lending protocol experience a sharp decline in price, the cost of acquiring enough governance tokens to undertake the governance attack would also likely fall. Conversely, should an actor undertake a governance attack, this would send shock waves throughout the DeFi ecosystem, reducing the price of the collateral asset and making under-collateralisation more likely.

Chapter 5

Dissimilar Redundancy

5.1 Introduction

Decentralized Finance (DeFi) Protocol hacks are frequent — with more than 150 to date totalling losses of more than 1.5bn USD. See Appendix A.1 of this thesis for a list of such hacks. This is problematic for a financial infrastructure that is purportedly going to replace traditional finance. The severity of the issue of hacks is exacerbated by the non-custodial nature of DeFi systems. Unlike in traditional financial systems where there are safety nets such as the state or insurers, in the DeFi setting, there are no such safety provisions at scale. Moreover, while there is a nascent insurance market for DeFi insurance, e.g. [Mut21], such solutions provide only a (necessary) second-best solution: providing coverage in the event of a DeFi system failure. A first-best solution is to prevent failure in the first place.

Preventing such failures is challenging. Leaving aside the security of the underlying blockchain layer, the two main pillars of DeFi protocol security are (i) extensive smart contract testing and (ii) code audits. Both of these have drawbacks. Ensuring adequate test coverage is very challenging, not least when the objective is to ensure all edge cases are covered in the tests. While testing is an essential part of smart contract development, we suggest that it is unrealistic to expect even best-practice testing with a very high degree of coverage to be sufficient to prevent *all* bugs. External teams often perform code audits under time pressure, with audits typically lasting 2 to 3 weeks from start to finish, even for teams unfamiliar with the code base. Even for the most experienced software engineer, native to DeFi, it is wishful thinking to believe that they will always be able to catch every bug. The problem is compounded by smart contract composability — where DeFi protocols are snapped together like DeFi Lego — which increases the challenge as tests and auditors now have to anticipate bugs that could arise with unseen smart contracts.

This final major chapter in the thesis presents a new approach to preventing smart contract failure: dissimilar redundancy. In aviation, the safety-critical nature has led to the emergence of a practice of implementing multiple separate and redundant flight control systems. For example, the Boeing 777 [Yeh96] featured a Fly-By-Wire flight control system that had to meet extremely high levels of functional integrity and reliability. To do this, it had three primary flight computers, each containing three *dissimilar* internal computational lanes. The lanes differed in terms of compilers, power supply units and microprocessors, with, for example, lane one using the AMD 29050, lane two the Motorola 68040 and lane three the Intel 80486. Within each of the three flight computers, two lanes acted as monitors while the third lane was in command. In this way, the flight computer features a form of redundancy that is *dissimilar*, with the multiple lanes resistant to bugs induced by microprocessors or compilers.

We apply the core of this idea to smart contracts. We implement and detail a system based on a proxy pattern that relies on dissimilar DeFi protocol implementations and cross-checks one against the other before effecting any on-chain state change. On Ethereum, this approach has already been taken for client implementations, with the community maintaining multiple open-source clients developed by different teams and using different programming languages [Eth21]. This approach aims to strengthen the network and make it more diverse, with a view to avoiding a single client dominating the network to remove single points of failure. We extend this concept to the smart contract layer itself.

Our contributions are as follows.

- We introduce the notion of dissimilar redundancy for DeFi protocols.
- We provide an implementation of a protocol for dissimilar redundancy for a DeFi proto-

col[1].

- We evaluate the protocol on a smart contract auction system implemented in both Solidity and Vyper, verify that a fuzzing approach would be able to detect purposefully introduced bugs and provide the costs in USD of using a protocol for dissimilar redundancy.

Outline

We provide our methodology in Section 5.2, evaluate it in Section 5.3, consider related work in Section 5.4 and conclude in Section 5.5.

5.2 Methodology

5.2.1 Overview

We now turn to how we use the approach of dissimilar redundancy in the context of Ethereum. At the centre of our approach is a proxy architecture pattern. With a proxy pattern, all message calls to a contract $\mathcal{C}$ first go through a proxy contract $\mathcal{P}$ that serves to direct the message calls to contract $\mathcal{C}$. With this pattern, contract $\mathcal{C}$ contains the actual implementation logic while contract $\mathcal{P}$ provides a storage layer. At present, a common use of this pattern is to provide contract upgradeability [BG19, Pal19]: while contracts cannot be directly upgraded once deployed, upgradeability can be mimicked by changing where contract $\mathcal{P}$ delegates calls to from contract $\mathcal{C}$ to contract $\mathcal{C}_1$.

We expand on this pattern: in pursuing dissimilar redundancy, we allow $\mathcal{P}$ to delegate to multiple implementations simultaneously. This contrasts with the standard pattern, which only permits delegation to a single implementation at a time. In a nutshell, our proxy contract $\mathcal{P}$ sequentially calls two different implementations - supposedly identical in logic - $\mathcal{C}_1$ and $\mathcal{C}_2$, and ensures that the data returned by function calls to each implementation as well as return values

[1]https://github.com/danhper/smart-contract-dissimilar-redundancy

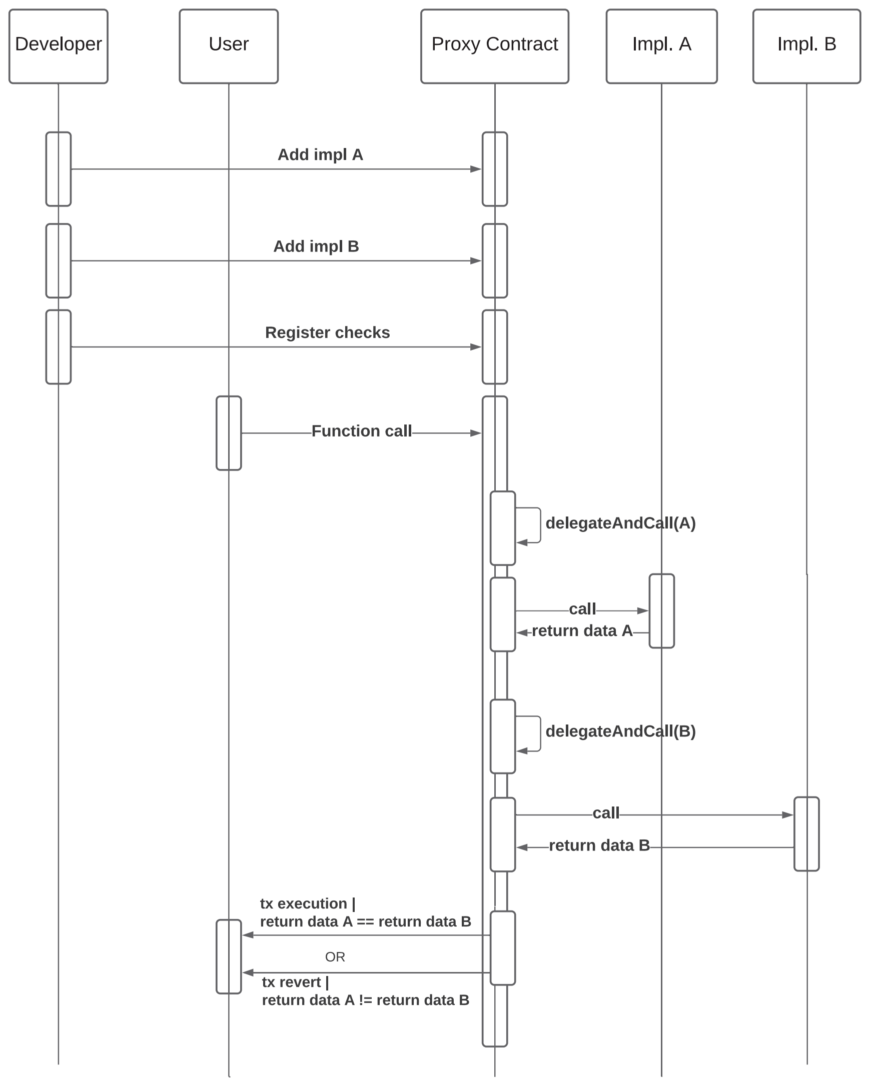


Figure 5.1: Overview of our dissimilar redundancy framework

from an arbitrary number of *checks* provided by the contract developer match. In this context, a check is a call to a contract's function of which the return value is deemed relevant to the function called. For example, in the context of an ERC-20 token [VB15], the developer might want to add balanceOf(from) and balanceOf(to) as a check for transferFrom(from, to, amount). The proxy will then call balanceOf twice after each call to transferFrom and ensure consistent results among the implementations.

Although the idea is straightforward, the implementation involves overcoming several technical challenges. Figure 5.1 shows an overview of the framework.

5.2.2 Technical challenges

Calling implementations with the same state. The first challenge is that we need to call both implementations with the same initial state; however, sequential contract calls typically result in state changes. To overcome this, all state changes made by a call to an implementation must be rolled back before we call the next implementation. We leverage the atomic nature of Ethereum's message calls to achieve this: if a transaction raises an error, the state reverts to the initial state.

Algorithm 1 Dissimilar redundancy framework call delegation

```
function CallDelegate(impl, data, checks, isLast)
    (ok, retData) ← DelegateTo(impl, data)
    checkResults ← RunChecks(checks)
    checksHash ← HashChecks(checkResults)
    if isLast then
        return (ok, retData, checksHash)
    else
        revert (ok, retData, checksHash)
    end if
end function

function RedundantCall(implementations, data)
    $n$ ← Length(implementations)
    signature ← GetSig(data)
    checks ← GetChecks(signature)

    for $i \leftarrow 0, n-1$ do
        impl ← implementations[$i$]
        last ← $i == n-1$

        callData ← Encode(CallDelegate, impl, data, checks, last)
        (_, delegateRet) ← DelegateTo(this, callData)
        (ok, retData, checksHash) ← Decode(delegateRet)

        if IsDefined(previousOk) then
            Assert(previousOk == ok)
            Assert(previousRetData == retData)
            Assert(previousChecksHash == checksHash)
        else
            previousOk ← ok
            previousRetData ← retData
            previousChecksHash ← checksHash
        end if
    end for

    return (ok, retData)
end function
```

Instead of the proxy directly delegating to an implementation, our approach first delegates to *itself*, passing in the call data as an argument along with the implementation to call and the checks to perform. In this delegated call, the proxy then delegates to the implementation executes all the checks and combines their result in a single hash value. It then reverts the execution to roll back the changes made by the implementation and returns the checks hash as the revert data. The only exception is the call to the last implementation, where it returns the checks hash normally instead of reverting to persist the changes. We provide a high-level overview of the delegation logic in Algorithm 1. Low-level implementation details can be found in our open-source implementation.

Check encoding. A second challenge concerns how the checks performed after each execution should be encoded. A check is a call to a contract that needs to be consistent across implementations after each execution. To make the proxy retain the interface of a proxied implementation, we must pre-register the checks with the proxy rather than specifying the checks with each call. A naive implementation could involve the developer registering a function with pre-encoded arguments to be called after any call. However, such an implementation would have severe limitations. Pre-encoding arguments make calls such as the one described above for balanceOf *impossible*, as these depend on call data and transaction information—this information is only available at runtime. Since these calls need to be well targeted for them to be effective and find potential discrepancies among implementations, there needs to be a different approach.

Instead, we implement an approach which achieves the following objectives:

1. the registration of per-function checks without pre-encoded arguments. For example, transferFrom and approve could have a different set of checks registered.
2. call data and transaction information should be accessible during the checks.

The first objective is easily achieved by storing a mapping from function signature to checks and retrieving the relevant checks depending on the function signature called by the current call to the proxy. The second objective is more challenging, as the developer must be able to register checks upfront that rely on information only available when the function is executed. To allow for this, rather than registering checks by passing in the arguments themselves, we design a simple byte encoding for the arguments that allows abstract arguments, registered with the checks, to be mapped to the concrete arguments that are computed when executing the checks. For example, an abstract argument could be “the first argument of the current call” or “the sender of the current transaction”. When executing the checks, the proxy will map these to the actual value of the first argument or the transaction’s sender and encode these when calling the check function.

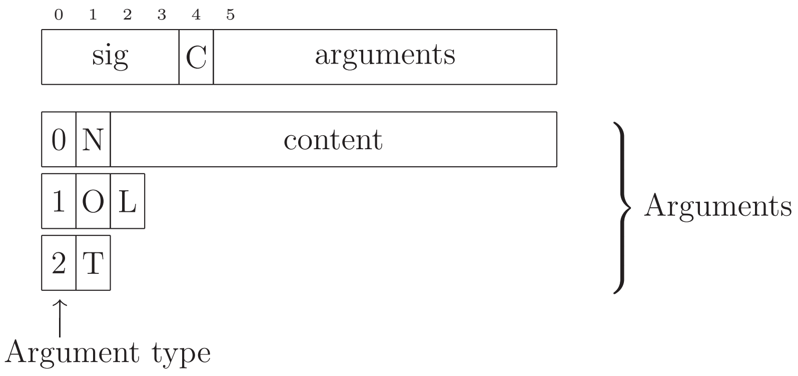


Figure 5.2: Encoding for registering checks. C is the number of arguments. The three types of arguments are shown. Type 0 represents a static argument, and N is the length of the static content. Type 1 represents the call data argument where O and L are the offset and length to retrieve from call data. Type 2 represents the environment argument, and T is the type of environment variable (e.g. msg.sender) to use

In Figure 5.2, we show how we encode abstract arguments to register the checks. As for regular contract calls, the first four bytes represent the signature of the function to be called. The next byte is the number of arguments to pass to the function. Each argument can then be one of three types: a static argument, a call data argument or an environment argument. These types determine how the concrete argument will be computed when the contract is called:

- A static argument: simply passed through to the function as a concrete argument.
- A call data argument: the given bytes from the transaction call data are extracted and passed as an argument.
- An environment argument: looks up the concrete argument in the current transaction. The argument to be looked up in the environment, e.g., the transaction’s sender or the current block timestamp, is specified by a single byte in the abstract argument.

Once all the abstract arguments are converted into their concrete counterpart, they are encoded along with the function signature, and the check can be performed.

```
tokenProxy.registerCheck(
    sig["transferFrom"],
    tokenProxy,
    encode_args(
        Token,
        "balanceOf",
        [(ArgumentType.CallData, (4, 32))]
        )
    )

tokenProxy.registerCheck(
    sig["transferFrom"],
    tokenProxy,
    encode_args(
        Token,
        "balanceOf",
        [(ArgumentType.CallData, (36, 32))]
        )
    )

tokenProxy.registerCheck(
    sig["transferFrom"],
    tokenProxy,
    encode_args(
        Token,
        "allowance",
        [(ArgumentType.CallData, (4, 32)),
        (ArgumentType.Env, EnvArg.Sender)]
        )
    )
```

Figure 5.3: Example checks registration for an ERC-20 token transferFrom function

This encoding provides enough flexibility to perform a wide variety of different checks.

5.2.3 Implementation

We show an example of such checks in Figure 5.3, where we register three different checks for the transferFrom function of an ERC-20 token. The two first checks are for the balance of the first argument (the address sending tokens) and the second argument (the address receiving tokens) of the transferFrom. In both cases, we extract these arguments from the call data. The third check is for the allowance and uses the first argument but also the sender of the transaction, as their allowance is expected to decrease after a successful call to transferFrom.

5.3 Evaluation

We use a smart contract that implements a simple auction system to evaluate our solution. The rules of the auction system are as follows:

1. A seller starts an auction with an NFT of their choice and sets an end time
2. The NFT is transferred to the auction contract
3. Any user can bid in the auction, and the bid must be strictly greater than the previous highest until the auction ends
4. After the auction ends, anyone can "finalise" the auction, which will either transfer the NFT to the winner of the auction or transfer it back to the owner if there were no bids

We implement the auction contract with the same behaviour in Solidity and Vyper but purposefully introduce implementation bugs in one of the two contracts. We then check whether these bugs would be detected by fuzzing the contract, i.e., passing many arbitrary input values, causing the proxy to fail due to inconsistencies in the evaluation results.

The bugs. We introduce two bugs to the Vyper version of the contract. First, rather than checking that the bid is strictly greater than the previous one, we check that the bid is greater or equal to the previous one. This means that in this particular case, the Solidity version will revert the transaction while the Vyper one will successfully execute. Second, in the Vyper implementation, we omitted to implement the case without bidders. As a result, both versions will successfully execute, but for the Vyper implementation, the ownership of the NFT will still be the auction contract in the case there are no bidders.

5.3.1 Development-time testing

We first test our code locally to show how our approach can make bug detection at development time significantly easier.

```
auction_proxy.registerCheck(
    sig["finalize"],
    nft_collection.address,
    encode_args(
        NftCollection,
        "ownerOf",
        [(ArgumentType.Static, ("uint256", NFT_ID))]
        )
    )

@given(bids=st.lists(st.tuples(st.integers(min_value=0),
        addresses())))
def test_bid(bids):
    with ensure_consistent():
        for value, account in bids:
            auction_proxy.bid({"from": account, "value": value})

@given(bids=st.lists(st.tuples(st.integers(min_value=0),
        addresses())))
def test_finalize(bids):
    with ensure_consistent():
        for value, account in bids:
            auction_proxy.bid({"from": account, "value": value})
        chain.sleep(3600)
        auction_proxy.finalize()
```

Figure 5.4: Testing code using dissimilar redundancy

To test our code, we use Python combined with the Brownie framework [2] for testing and the Hypothesis library [MHD+19] to generate test cases. We show the most critical part of the code we use in our auction contract in Figure 5.4. We note that the ensure_consistent context manager is implemented as part of our tooling and will only fail if a call reverts because implementations did not behave similarly.

In the tests, we first register a check for finalise that will look up the owner of the NFT that was on sale. For testing both bid and finalise, we generate random bids, where a bid is an account and value (price to pay for the auction) pair. For bid, we only execute all the bids, while for finalise, we also ensure that the auction has ended and execute finalise.

Using this approach, fuzzing the contract requires performing differential fuzzing on the different implementations of the contract logic registered by the proxy. This makes it possible to quickly identify the cases where the two implementations do not behave in the same way.

[2] https://eth-brownie.readthedocs.io/

```
Falsifying example: test_bid_consistency(bids=[
    (1, <Account '0x66aB6...5871'>),
    (1, <Account '0x66aB6...5871'>)],
)
================================================================
FAILED tests/test_auction.py::test_bid -
brownie.exceptions.VirtualMachineError: revert: all
implementations must return the same success
```

(a) Failing test case for the bid function showing a failure after two bids with the same value were placed

```
Falsifying example: test_finalize_consistency(
  bids=[]
)
================================================================
FAILED tests/test_auction.py::test_finalize -
brownie.exceptions.VirtualMachineError: revert: all
implementations must return the same checks
```

(b) Failing test case for the finalize function showing a failure when no bids have been placed

Figure 5.5: Failing tests when fuzzing the auction contract with a correct and an incorrect implementation

In Figure 5.5, we show the results of these tests. Note that we fix the first failing test before proceeding to the second one.

The test framework correctly outputs a failure for the bid function when two bids are placed with the same value in a row. The test output makes the failure scenario clear, as the two bids of the input have a value of 1. The error message mentions that all implementations should return the same "success", which means that one of the implementations was successfully executed (the bug-ridden version) while the other failed to execute.

The test for the finalise function also correctly fails with an example containing no bids. This is consistent with the bug in the Vyper version of the Solidity, which does not transfer back the token properly to its original owner. The failing test also mentions that all implementations must return the same checks, which means that all implementations had the same success status (they all succeeded in this particular case), but the checks did not return the same value. Indeed, a check was registered to look up the NFT owner.

Overall, with this example, with only a few lines of code, it was possible to have extensive coverage of the tested function that can automatically find discrepancies among implementations and indicate to the developer the test cases that would yield different results.

Name	**Address**
Auction proxy	0xAd837BDD116C14aA82311Db7D1879C7cDDCfd283
Auction (Sol, proxied)	0xdb85f3DB2aA6E5e294485972ABE921be188b6A37
Auction (Vy, proxied)	0x9FD31161360B5E772f2b9C469D4A35E679273Dbf
Auction (Sol, standalone)	0xE575CCb0213393eBFc9258013af1c43e9E416544
Auction (Vy, standalone)	0xEe164319fE07127Efc8fdf8b3e99ea736F8c955E

Table 5.1: Contracts deployed on Polygon mainnet. "Sol" are Solidity contracts. "Vy" are Vyper contracts. "Proxied" are contracts used by the proxy. "Standalone" are contracts interacted with directly.

	Start	First bid	Subsequent bid	Finalize
Proxied	0.0229	0.0092	0.006	0.0144
Solidity	0.0094	0.0045	0.0029	0.0052
Vyper	0.0108	0.0045	0.0029	0.0071

Table 5.2: USD cost[3] of calling different functions of the auction contract with and without dissimilar redundancy proxy. The gas price is fixed at 30 Gwei.

5.3.2 Real-world deployment

An essential strength of our approach is that it is possible to utilise the two implementations not only at development and testing time but also after the contract is deployed, ensuring that all the transactions executed will always be consistent across the different implementations provided. To demonstrate how this would perform on a real-world blockchain, we deploy our auction and its two implementations on the Polygon main network (an Ethereum side-chain suitable since it offers lower transaction fees), ensure that the calls that would trigger revert correctly and measure the cost overhead of our approach. We list all the deployed contracts in Table 5.1.

To give comparable results, we deploy our proxy using the Solidity and Vyper implementations and a standalone version of each implementation. During our interactions with the contracts, we maintained a fixed gas price of 30 Gwei, which was enough at the time of writing for near-instantaneous inclusion in a Polygon block.

We summarise the cost nominated in US dollars of all the interactions with our auction contracts in Table 5.2. We split the costs of the first and the subsequent bids since the first bid allocates

[3]We use the December 12th 2021, price of 2.15 USD per MATIC token, Polygon's native token used to pay for gas fees

storage, using more gas than the subsequent ones. Since the proxy uses both the Solidity and the Vyper implementation, a lower bound for the cost is the sum of the cost of each implementation. The difference between the sum of these costs and the cost of calling the proxy is the overhead of the proxy itself, including the cost of calling checks after each call and checking consistency among results. In our example, only finalise has a check registered to check for the owner of the NFT after execution. For the calls to start and bid, respectively, about 1.2% and 2.2% of the total cost is part of the proxy overhead, the rest of the cost being used by the actual underlying functions. For finalise , the overhead is understandably higher as a check is registered. Indeed, about 14.5% of the cost is used by the proxy itself.

We also check that a transaction that would cause our correct and our buggy implementation to diverge would correctly revert. To do so, we bid using subsequently twice the same amount, which fails for the proxied version (tx 0x5f840a...6dc334c) by returning the same error as the one we saw during the tests. As a sanity check, we try the same sequence of bids on the standalone implementations, and the Solidity version reverts correctly (tx 0x406ad7...7759673) while the Vyper one succeeds (tx 0x888189...7df41e8).

5.4 Related work

We first present how a similar approach has been helpful for Ethereum clients, then in the context of bug bounties, before finally presenting some related research discussing differential fuzzing.

5.4.1 Ethereum clients

The Ethereum network has been running using different client implementations. The goal of having multiple clients has always been to make the network more robust and ensure that it does not fail in case there is a bug, a vulnerability, or an avenue for a potential denial-of-service (DoS) attack in one of the implementations. In the case of Ethereum, this allows for several failure modes. If one of the implementations had a bug that would make it crash on certain transactions, the network would continue to operate with other implementations that do not contain this bug. This would be similar in the case of a DoS attack only effective on one type of implementation. On the other hand, if a bug or exploit results in a different state transition after executing a transaction, it would create a network split where the victim implementation would only manage to reach consensus with nodes using the same implementation. This is riskier than the previous case but remains safe as long as exchanges and other entities bridging on-chain activity to off-chain assets ensure all implementations are in a consistent state before accepting to process a transaction. Despite the high maintenance cost, this diversity of clients has benefited Ethereum and allowed it to operate smoothly for over six years.

5.4.2 Application to bug bounties

In [BDTJ18], the authors develop the Hydra Framework, which is the first general approach to modelling bug bounties in a way that seeks to incentivize bug disclosure. This framework relies on a concept the authors refer to as an exploit gap, with their framework transforming programs via N-of-N version programming, running multiple program instances. This framework has similarities to the approach we develop but differs in some crucial respects. In Hydra, the authors rely on an instrumentation approach, instrumenting opcodes, whereas ours relies on an argument encoding scheme and proxy contracts.

5.4.3 Differential fuzzing

Having two implementations that should behave in the same way allows to perform differential fuzzing: fuzzing both implementations trying to look for cases where they would behave differently. This technique has already been used in multiple domains such as cryptography [BJR+14], programming languages [CSS+16], and blockchain consensus [Eth19, FRM+19]. A recent work leveraging differential fuzzing to find bugs in Ethereum clients [YKC21] has managed to find not only the most known consensus bugs but also two new ones, including a bug that led to a fork in the consensus due to only part of the full nodes in the network having upgraded to the latest version [Cop21]. Overall, this shows the potential of differential fuzzing and how it can help find bugs and zero-day exploits.

5.5 Conclusion

We have argued that the high financial stakes in the context of DeFi merit an approach to program redundancy inspired by avionics: the utilization of dissimilar redundancy. Through implementing the same program logic more than once, ideally with different programming languages and even by different engineering teams, and then using an on-chain execution logic that ensures that the dissimilar implementations must *agree* before the on-chain state can update, redundancy is brought into the smart contract ecosystem. Such redundancy should make smart contracts and DeFi, as a whole, less vulnerable to exploits from implementation bugs.

We hope this chapter can offer one step toward a more robust and secure DeFi. As in avionics, in DeFi, the stakes are high, and the risks material.

Chapter 6

Conclusion

6.1 Summary of thesis achievements

The first part of the thesis examines the interconnection of DeFi protocols, focusing on — and coining the term — Protocols for Loanable Funds (PLFs). Our empirical examination showed that nascent DeFi protocols display substantial degrees of interconnection, existing as a series of connected nodes. We presented a liquidity study on markets for a set of assets. We found that liquidity plays a critical role in the performance of DeFi protocols, with interest rate rules behaving differently in response to varying degrees of liquidity. We find that periods of illiquidity are common, often shared between protocols, and the sources of liquidity reserves are concentrated. In some cases, three accounts control as much as 50% of the liquidity. On investigating one particular PLF, Compound, we found that the no-arbitrage condition of Uncovered Interest Parity did not hold for the sample period, suggesting that markets associated with the protocol may be relatively inefficient. Further, we investigated the extent of market dependence between different PLFs. The borrowing rates appear to be interdependent, with protocols seeming to influence the interest rates on other protocols. These findings highlight the importance of understanding the complex and dynamic interconnections between DeFi protocols to assess better and manage systemic risk within the ecosystem.

The second part of the thesis investigates the fragility of an individual DeFi node, using MakerDAO's DAI as a case study. Our analysis first revealed a design weakness in the governance system of MakerDAO's DAI that made it feasible for an attacker to take complete control of the protocol, exposing users to significant risk. We presented a novel strategy utilising flash loans that would have allowed the execution of the governance attack in just two transactions without locking any assets. We then developed a stress-testing framework for a stylised DeFi lending protocol and simulated a price crash event with this stress-testing methodology. In this simulation, we found that for plausible parameter values, a DeFi lending protocol could become undercollateralised within 19 days.

In the final major chapter, we proposed an approach to making DeFi protocols less dangerous from a smart contract perspective through leveraging redundancy for smart contracts: using multiple smart contracts that are supposed to implement the same program logic to reduce the chance of software bugs occurring. Our analysis of over 70 exploits of DeFi protocols and the total loss of approximately 1.5bn USD highlighted the urgent need for new approaches to minimise such attacks' frequency and severity. We developed a new self-delegation proxy pattern approach, with novel argument encoding for arbitrary function checks, to We showed that dissimilar redundancy could reduce the likelihood of successful attacks on DeFi protocols that exploit the code-is-law property.

6.2 Looking toward the future

In direct relation to this thesis, the most pressing avenues for future work seem to be:

- Developing formal models that examine financial contagion in the DeFi setting. Such work could extend that in the economics literature [EGJ14, AG00], which investigates cascades of failures in a network of interdependent financial organisations. DeFi protocols could be considered along the key axes of integration (how dependent each protocol is on counterparties) and diversification. In particular, in the economics literature, such work considers the role of discontinuities in asset values (for example, defaults) on cascades: in the DeFi setting, smart contract risk is a very plausible source of such discontinuities in asset values. Therefore extending the analysis of the impact of discontinuities in asset prices into the DeFi realm is of great importance.

- Perhaps of all the contributions of this thesis, there remains the most work to do concerning chapter 5. The extent of DeFi protocol exploits, both in terms of frequency and value, is staggering; clearly, significant effort must be invested in reducing the frequency of such exploits. One immediate area for future work relates to improving the efficiency of schemes that leverage dissimilar redundancy. In particular, reducing the computational resources required on-chain since these seem likely to be prohibitively expensive. However, the extent to which computational efficiency (especially in terms of gas on Ethereum) matters long-term will depend on advances made at the layer-one level. If gas costs fall significantly on Ethereum, the need for computationally efficient dissimilar redundancy regimes is less important. The existence of such exploits prevents DeFi from safely scaling and supplanting traditional financial mechanisms since consumers seem likely to be exposed to significant welfare losses.

Aside from future work that pertains most directly to this thesis, it is clear that how privacy is approached on-chain must evolve for DeFi to scale. On Ethereum, DeFi grants pseudoanonymity. But if an address is linked to an individual, the entire transaction history of that address is visible to anyone in the world. For individuals, such possible privacy breaches may result in significant personal danger: for example, the wealthy could be targeted. For businesses, exposing all of their payments to other companies makes modern commerce prohibitive on-chain and, in some cases, may encourage the formation of oligopolies to the detriment of consumers.

Appendix A

A list of exploits in DeFi

A.1 Exploits

Table A.1: Funds lost in protocol hacks. Source: DeFiLlama, https://defillama.com/hacks

Date	**Name**	**Technique**	**Loss (m, USD)**
2022-03-23	Ronin	Private Key Compromised (Social Engineering)	624.0
2021-08-10	Poly Network	Access Control Exploit	611.0
2022-10-06	Binance Bridge	Proof Verifier Bug	570.0
2022-11-12	FTX	Private Key Compromised (Unknown Method)	450.0
2022-02-02	Wormhole	Signature Exploit	326.0
2018-04-21	Gate.io	Private Key Compromised (Unknown Method)	235.0
2021-12-04	Bitmart	Private Key Compromised (Unknown Method)	196.0
2022-08-01	Nomad	Trusted Root Exploit	190.0
2022-04-17	Beanstalk	Flashloan Governance Attack	181.0

Continued on next page

Table A.1 – continued from previous page

Date	Name	Technique	Loss (m, USD)
2022-09-20	Wintermute	Private Key Compromised (Brute Force)	160.0
2017-11-09	Parity Multisig	Contract not initialized	150.0
2021-09-29	Compound	Math Mistake Exploit	147.0
2021-12-13	Vulcan Forged	Private Key Compromised (Unknown Method)	140.0
2021-10-27	Cream	Flashloan Price Oracle Attack	130.0
2021-12-02	Badger	Frontend Attack	120.0
2022-10-11	Mango Market	Price Oracle Attack	115.0
2022-06-23	Harmony Bridge	Private Key Compromised (Unknown Method)	100.0
2021-10-08	Mirror Protocol	Duplicate Call Exploit	90.0
2022-05-01	Fei Rari	Flashloan Reentrancy Attack	80.0
2022-01-28	Qubit Finance	Transfer Logic Exploit	80.0
2021-10-29	AnubisDAO	Drained Contracts	60.0
2016-06-17	The DAO	Reentrancy	60.0
2021-04-19	EasyFi	Private Key Compromised (Unknown Method)	59.0
2021-04-28	Uranium Finance	Math Mistake Exploit	57.2
2021-11-05	bZx	Private Key Compromised (Phishing)	55.0
2022-03-23	Cashio	Collateral Validation Exploit	48.0
2020-09-29	Kucoin	Private Key Compromised (Unknown Method)	45.0
2021-05-19	PancakeBunny	Flashloan Price Oracle Attack	45.0
2021-02-13	Alpha Finance	Flashloan Pool Shares Exploit	37.5
2021-09-21	Vee Finance	Flashloan Price Oracle Attack	34.0
2022-01-18	Crypto.com	Private Key Compromised (Unknown Method)	33.7
2021-03-04	Meerkat Finance	Drained Contracts	32.0

Continued on next page

Table A.1 – continued from previous page

Date	Name	Technique	Loss (m, USD)
2021-11-30	MonoX	Swap Function Exploit	31.4
2021-05-02	Spartan Protocol	Flashloan Pool Shares Exploit	30.5
2021-12-18	Grim Finance	Flashloan Reentrancy Attack	30.0
2022-11-01	Deribit	Private Key Compromised (Unknown Method)	28.0
2022-06-05	Wintermute	Cross Chain Multisig Deployment Exploit	27.6
2021-03-05	Paid Network	Infinite Mint and Dump	27.0
2021-06-23	StableMagnet	Drained Contracts and User Wallets	27.0
2020-04-19	dForce	Reentrancy	25.0
2020-10-26	Harvest Finance	Flashloan Price Oracle Attack	25.0
2021-05-12	XToken	Flashloan Price Oracle Attack	24.0
2021-04-12	Elephant Money	Flashloan Price Oracle Attack	22.2
2022-11-06	Pando Rings	Price Oracle Attack	22.0
2022-05-13	Blizz Finance	Outdated Oracle Exploit	21.8
2021-08-03	Popsicle Finance	Flashloan Incentive Rewards Exploit	20.0
2020-11-22	Pickle Finance	Vault Swap Exploit	19.7
2021-08-30	Cream Finance	Flashloan Reentrancy Attack	18.8
2021-11-25	Snowdog	Drained Contracts	18.1
2021-05-17	bEarn	Flashloan Withdraw Logic Exploit	18.0
2021-10-14	Indexed Finance	Flashloan Price Oracle Attack	16.0
2022-10-27	Team Finance	Migrate Function Mistake Exploit	15.8
2022-04-02	Inverse Finance	Price Oracle Attack	15.6
2020-09-28	Eminence	Flashloan Price Oracle Attack	15.0
2021-02-27	Furucombo	Delegatecall Exploit	14.0
2022-04-28	Deus DAO	Flashloan Price Oracle Attack	13.4
2020-12-02	Compounder Finance	Drained Contracts	12.0
2021-02-05	Yearn	Flashloan Price Oracle Attack	11.0
2021-05-07	Value DeFi	Math Mistake Exploit	11.0

Continued on next page

Table A.1 – continued from previous page

Date	Name	Technique	Loss (m, USD)
2021-12-02	Saddle Finance	Flashloan Swap Logic Exploit	11.0
2021-05-08	Rari Capital	Price Oracle Attack	10.0
2021-05-05	Value DeFi	Access Control Exploit	10.0
2022-01-04	Arbix Finance	Drained Contracts	10.0
2022-02-10	Dego Finance	Private Key Compromised (Unknown Method)	10.0
2020-12-29	Cover	Incentives Function Mistake	9.4
2021-08-10	Punk Protocol	Delegatecall Exploit	8.95
2022-07-02	Crema	Access Control Exploit	8.8
2022-02-08	Superfluid	CTX Exploit	8.7
2023-02-16	Platypus Finance	Flashloan Reentrancy Attack	8.5
2022-10-18	Moola Market	Price Oracle Attack	8.4
2021-12-21	Visor Finance	Access Control Exploit	8.199999
2021-07-22	Thorchain	Refund Logic Exploit	8.0
2020-11-17	Origin Protocol	Flashloan Reentrancy Attack	8.0
2022-01-08	LCX	Private Key Compromised (Unknown Method)	7.94
2021-07-10	Anyswap	Private Key Compromised (Bad ECDSA Implementation)	7.9
2020-12-18	Warp Finance	Flashloan Price Oracle Attack	7.8
2022-02-06	Meter	Deposit Function Exploit	7.7
2017-07-18	CoinDash	Frontend Attack	7.7
2022-02-01	BNS	Private Key Compromised (Unknown Method)	7.5
2021-05-28	BurgerSwap	Flashloan Reentrancy Attack	7.2
2021-08-12	DAO Maker	Private Key Compromised (Unknown Method)	7.0
2020-11-14	Value Defi	Flashloan Price Oracle Attack	7.0
2022-12-10	Lodestar Finance	Flashloan Price Oracle Attack	6.9

Continued on next page

Table A.1 – continued from previous page

Date	Name	Technique	Loss (m, USD)
2021-06-16	Alchemix	Borrow Logic Exploit	6.5
2021-05-29	Belt	Flashloan Price Oracle Attack	6.3
2022-03-15	Hundred Finance	Flashloan Reentrancy Attack	6.2
2021-07-15	Bondly	Private Key Compromised (Unknown Method)	5.9
2022-06-16	Inverse Finance	Flashloan Price Oracle Attack	5.8
2021-03-14	Roll	Private Key Compromised (Unknown Method)	5.7
2022-03-15	Agave DAO	Flashloan Reentrancy Attack	5.5
2022-08-02	Slope Wallet	Private Key Compromised (Stored Publicly)	5.3
2022-12-02	Ankr	Access Control Exploit	5.0
2021-07-15	THORChain	Over-ride Logic Exploit	5.0
2022-11-03	pGALA *white hack	Misconfiguration	4.589449
2021-06-22	Eleven Finance	Flashloan Burn Function Exploit	4.5
2021-08-29	X-Token	Flashloan Price Oracle Attack	4.5
2022-12-16	Raydium	Private Key Compromised (Unknown Method)	4.4
2021-11-07	ChainSwap	Exploit Weak Authentication Check	4.4
2021-08-17	SurgeBNB	Flashloan Reentrancy Attack	4.0
2021-09-04	DAO Maker	Access Control Exploit	4.0
2022-11-10	DFX	Reentrancy	4.0
2022-03-31	Ola Finance	Reentrancy	4.0
2023-02-10	dForce Network	Reentrancy	3.65
2022-08-08	Dragoma	Drain Vaults	3.5
2022-07-28	Nirvana Finance	Flashloan Price Oracle Attack	3.5
2021-09-03	Siren Protocol	Reentrancy	3.45
2020-11-16	CheeseBank	Flashloan Price Oracle Attack	3.3
2021-09-17	JayPegs Automart	Redirected Deposits	3.1

Continued on next page

Table A.1 – continued from previous page

Date	Name	Technique	Loss (m, USD)
2022-05-08	Fortress Protocol	Governance and Oracle Exploit	3.0
2021-03-15	Deus DAO	Flashloan Price Oracle Attack	3.0
2022-11-02	Skyward Finance	Reentrancy	3.0
2023-02-02	Orion	Reentrancy	3.0
2021-03-20	Turtle Dex	Drained Contracts	2.5
2021-07-16	PancakeBunny	Flashloan Incentive Rewards Exploit	2.4
2022-10-11	TempleDAO	Exploit Lack of Input Authentication	2.3
2022-06-08	Gym Network	Deposit Function Exploit	2.1
2022-03-27	Revest Finance	Reentrancy	2.009999
2020-11-12	Akropolis	Flashloan Reentrancy Attack	2.0
2021-03-09	DODO	Access Control Exploit	2.0
2022-05-04	MM Finance	DNS Spoofing	2.0
2023-02-20	Dexible	Arbitrary External Call	2.0
2021-10-08	Mirror Protocol	Outdated Oracle Exploit	2.0
2023-02-20	Hope Finance	Router Exploit	1.86
2021-12-08	8ight Finance	Private Key Compromised (Unknown Method)	1.75
2023-02-01	BonqDAO	Price Oracle Attack	1.7
2022-03-13	Paraluni	Flashloan Reentrancy Attack	1.7
2021-08-13	Acala Network	Incentives Function Mistake	1.6
2021-07-30	Levyathan	Private Key Compromised (Stored Publicly)	1.5
2022-12-25	Rubic	Router Exploit	1.41
2022-03-03	Treasure DAO	Buy Function Exploit	1.4
2021-02-09	Growth DeFi	Flashloan Price Oracle Attack	1.3
2022-11-02	Solend	Price Oracle Attack	1.26
2021-07-14	ApeRocketFi	Flashloan Incentive Rewards Exploit	1.26
2020-04-19	Lendf.me	Reentrancy	1.2
2022-12-13	ElasticSwap	Price Oracle Attack	0.854

Continued on next page

Table A.1 – continued from previous page

Date	Name	Technique	Loss (m, USD)
2022-08-23	SudoRare	Drained Contracts	0.8
2021-05-24	Autoshark	Flashloan Incentive Rewards Exploit	0.745
2021-05-26	Merlin Labs	Flashloan Incentive Rewards Exploit	0.68
2023-01-17	Midas Capital	Flashloan Reentrancy Attack	0.66
2020-02-18	bZx	Flashloan Price Oracle Attack	0.65
2022-08-10	Blur Finance	Drained Contracts	0.6
2022-08-09	Curve Finance	DNS Spoofing	0.575
2022-09-18	GMX	Price Oracle Attack	0.565
2021-05-26	Merlin Labs	Math Mistake Exploit	0.55
2020-02-15	bZx	Flashloan Slippage Attack	0.35
2021-06-29	Merlin Labs	Incentive Rewards Exploit	0.33
2022-10-21	Bond Protocol	Arbitrary External Call	0.3
2021-06-28	Safe Dollar	Incentive Rewards Exploit	0.248
2023-03-02	ArbiSwap	Infinite Mint and Dump	0.1
2023-01-11	ROE Finance	Flashloan Pool Shares Exploit	0.08

Bibliography

[AAV20a] AAVE. Aave, 2020.

[Aav20b] Aave. Aave Liquidity Pools. https://app.aave.com/borrow, 2020.

[Aav20c] Aave. Aave Protocol. https://github.com/aave/aave-protocol, 2020.

[Aav20d] Aave. Borrow Interest Rate. https://docs.aave.com/risk/liquidity-risk/borrow-interest-rate, 2020.

[Aav20e] Aave. Optimizedreserveinterestratestrategy. https://etherscan.io/address/0x247227714bd121c528310e3bbff401ae34c9f9f6#code, 2020.

[ABC17] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on ethereum smart contracts (sok). In *International conference on principles of security and trust*, pages 164–186. Springer, 2017.

[AG00] Franklin Allen and Douglas Gale. Financial contagion. *Journal of political economy*, 108(1):1–33, 2000.

[Ale01] Annika Alexius. Uncovered interest parity revisited. *Review of international Economics*, 9(3):505–517, 2001.

[Ale19] Alethio. Illiquidity and Bank Run Risk in Defi. https://medium.com/alethio/overlooked-risk-illiquidity-and-bank-runs-on-compound-finance-5d6fc3922d0d, 2019.

[Ava23] Avalanche. Avalanche. https://www.avax.network/, 2023.

[BBC09] BBC. BBC World Service, Aftershock Timeline. http://www.bbc.co.uk/worldservice/business/2009/09/090902_aftershock_timeline_noflash.shtml, 2009.

[BDTJ18] Lorenz Breidenbach, Phil Daian, Florian Tramèr, and Ari Juels. Enter the hydra: Towards principled bug bounties and exploit-resistant smart contracts. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1335–1352, 2018.

[BG19] Gabriel Barros and Patrick Gallagher. Eip-1822: Universal upgradeable proxy standard (uups), 2019.

[BGK+18] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 913–930, 2018.

[BHM19] Dorje C. Brody, Lane P. Hughston, and Bernhard K. Meister. Theory of cryptocurrency interest rates, 2019.

[Bin23] Binance. Binance Smart Chain. https://www.bnbchain.org/en, 2023.

[BJR+14] Chad Brubaker, Suman Jana, Baishakhi Ray, Sarfraz Khurshid, and Vitaly Shmatikov. Using frankencerts for automated adversarial testing of certificate validation in ssl/tls implementations. In *2014 IEEE Symposium on Security and Privacy*, pages 114–129. IEEE, 2014.

[BMTZ17] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: A composable treatment. In *Annual International Cryptology Conference*, pages 324–356. Springer, 2017.

[Boa20] Federal Reserve Board. FOMC's target federal funds rate or range, change (basis points) and level. https://www.federalreserve.gov/monetarypolicy/openmarket.htm, 2020.

[Bro07] Lehman Brothers. Lehman brothers holdings inc. plan trust – '10-k' for 11/30/07. http://www.secinfo.com/d11MXs.t5Bb.htm#1stPage, 2007.

[Car23] Cardano. Cardano. https://cardano.org/, 2023.

[CDK+18] Yizhou Cao, Min Dai, Steven Kou, Lewei Li, and Chen Yang. Designing stable coins. https://duo.network/papers/duo_academic_white_paper.pdf, 2018.

[CGM09] Jacopo Carmassi, Daniel Gros, and Stefano Micossi. The global financial crisis: Causes and cures. *JCMS: Journal of Common Market Studies*, 47(5):977–996, 2009.

[Cir20] Circle. USDC. https://www.circle.com/en/usdc, 2020.

[CL99] Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, pages 173–186, 1999.

[CM05] Menzie D Chinn and Guy Meredith. Testing uncovered interest parity at short and long horizons during the post-bretton woods era. Technical report, National Bureau of Economic Research, 2005.

[Coi20a] CoinMarketCap. Maker (MKR) price, charts, market cap, and other metrics. https://coinmarketcap.com/currencies/maker/, 2020.

[Coi20b] Coinut. Coin Ultimate Trading. https://coinut.com/, 2020.

[Com19] Compound. Compound finance. https://compound.finance/, 2019.

[Com20] Compound. DAI rate model. https://etherscan.io/address/0xfed941d39905b23d6faf02c8301d40bd4834e27f#code, 2020.

[Cop21] Tim Copeland. Bug impacting over 50% of ethereum clients leads to fork. https://www.theblockcrypto.com/post/115822/bug-impacting-over-50-of-ethereum-clients-leads-to-fork, 2021.

[Cry20] CryptoCompare. Cryptocompare. https://www.cryptocompare.com/, 2020.

[CSS$^+$16] Yuting Chen, Ting Su, Chengnian Sun, Zhendong Su, and Jianjun Zhao. Coverage-directed differential testing of jvm implementations. In *proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 85–99, 2016.

[Dat20] Cryptowatch Market Data. Cryptowatch DAI pair market data. https://cryptowat.ch/assets/dai, 2020.

[DDE20] DDEX. DDEX: Decentralized Margin Trading. https://ddex.io/, 2020.

[DGKR18] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018.

[dYd19] dYdX. Polynomial Interest Setter. https://etherscan.io/address/0xaEE83ca85Ad63DFA04993adcd76CB2B3589eCa49#code, 2019.

[dYd20] dYdX. dYdX. https://dydx.exchange/, 2020.

[Ear09] Timothy C Earle. Trust, confidence, and the 2008 global financial crisis. *Risk Analysis: An International Journal*, 29(6):785–792, 2009.

[EGJ14] Matthew Elliott, Benjamin Golub, and Matthew O Jackson. Financial networks and contagion. *American Economic Review*, 104(10):3115–3153, 2014.

[Ell12] Larry Elliott. Three myths that sustain the economic crisis. *The Guardian*, Aug 2012.

[ES14] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.

[Eth19] Ethereum. EVM lab utilities. https://github.com/ethereum/evmlab, 2019.

[Eth21] Ethereum. Ethereum Nodes and Clients - Client diversity. https://ethereum.org/en/developers/docs/nodes-and-clients/client-diversity/, 2021.

[Exc20] VCC Exchange. VCCExchange. https://vcc.exchange/, 2020.

[EXM20] EXMO. Exmo. https://exmo.com/, 2020.

[Fis30] Irving Fisher. *Theory of interest: as determined by impatience to spend income and opportunity to invest it.* Augustusm Kelly Publishers, Clifton, 1930.

[For08] Financial Stability Forum. Report of the financial stability forum on enhancing market and institutional resilience. *Working Papers of the Financial Stability Forum*, 2008.

[Fou19] The Maker Foundation. MakerDAO. https://makerdao.com/en/, 2019.

[Fou20] Ethereum Foundation. Solidity v0.8.0 documentation. https://docs.soliditylang.org/en/v0.8.0/index.html, 2020. Accessed: 12-01-2020.

[Fra20a] Emilio Frangella. Aave Borrowing Rates Upgraded. https://medium.com/aave/aave-borrowing-rates-upgraded-f6c8b27973a7, 2020.

[Fra20b] Emilio Frangella. Crypto Black Thursday: The Good, the Bad, and the Ugly. https://medium.com/aave/crypto-black-thursday-the-good-the-bad-and-the-ugly-7f2acebf2b83, 2020.

[FRM+19] Ying Fu, Meng Ren, Fuchen Ma, Heyuan Shi, Xin Yang, Yu Jiang, Huizhong Li, and Xiang Shi. Evmfuzzer: detect evm vulnerabilities via fuzz testing. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1110–1114, 2019.

[GMSR+20] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 201–226. Springer, Cham, 2020.

[GPH+20] Lewis Gudgeon, Daniel Perez, Dominik Harz, Benjamin Livshits, and Arthur Gervais. The Decentralized Financial Crisis. In *2020 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 1–15. IEEE, 2020.

[GWPK20] Lewis Gudgeon, Sam M. Werner, Daniel Perez, and William J. Knottenbelt. DeFi Protocols for Loanable Funds: Interest Rates, Liquidity and Market Efficiency. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, page 92–112, 2020.

[HGGK19] Dominik Harz, Lewis Gudgeon, Arthur Gervais, and William J Knottenbelt. Balance: Dynamic adjustment of cryptocurrency deposits. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1485–1502, 2019.

[HGKZ20] Dominik Harz, Lewis Gudgeon, Rami Khalil, and Alexei Zamyatin. Promise: Leveraging future gains for collateral reduction. In *Mathematical Research for Blockchain Economy*, pages 143–160. Springer, Cham, 2020.

[i3n20] i3nikolai. Note on MKR voting and distribution. https://www.reddit.com/r/MakerDAO/comments/enpad1/note_on_mkr_voting_and_distribution/, 2020.

[Ind] Index. Index: A comprehensive list of decentralized exchanges (DEX). https://distribuyed.github.io/index/.

[Ins20] InstaDApp. InstaDApp: Trustless smart wallet for DeFi. https://instadapp.io/, 2020.

[IOT20a] IOTA. IOTA Coordinator halted. https://twitter.com/iotatoken/status/1227990537799524352?s=20, 2020.

[IOT20b] IOTA. IOTA Status. https://status.iota.org/, 2020.

[Isa06] Mr Peter Isard. *Uncovered interest parity.* International Monetary Fund, 2006.

[JB87] Carlos M Jarque and Anil K Bera. A test for normality of observations and regression residuals. *International Statistical Review/Revue Internationale de Statistique*, pages 163–172, 1987.

[JK15] Zoltan Jakab and Michael Kumhof. Banks are not intermediaries of loanable funds–and why this matters, 2015.

[KCCM20] Hsien-Tang Kao, Tarun Chitra, Rei Chiang, and John Morrow. An analysis of the market risk to participants in the compound protocol. In *Third International Symposium on Foundations and Applications of Blockchains*, 2020.

[KMHG$^+$20] Ariah Klages-Mundt, Dominik Harz, Lewis Gudgeon, Jun-You Liu, and Andreea Minca. Stablecoins 2.0: Economic foundations and risk-based models. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 59–79, 2020.

[KMM19] Ariah Klages-Mundt and Andreea Minca. (in) stability for the blockchain: Deleveraging spirals and stablecoin attacks. *arXiv preprint arXiv:1906.02152*, 2019.

[KPA$^+$19] Aikaterina Koutsouri, Francesco Poli, Elise Alfieri, Michael Petch, Walter Distaso, and William Knottenbelt. Balancing cryptoassets and gold: A weighted-risk contribution index for the alternative asset space. *Mathematical Research for Blockchain Economy, Forthcoming*, 2019.

[KuC20] KuCoin. Kucoin. https://trade.kucoin.com/spot, 2020.

[LBC$^+$19] Lindsay X Lin, Eric Budish, Lin William Cong, Zhiguo He, Jonatan H Bergquist, Mohit Singh Panesir, Jack Kelly, Michelle Lauer, Ryan Prinster, Stephenie Zhang, et al. Deconstructing decentralized exchanges. *Stanford Journal of Blockchain Law & Policy*, 2019.

[Li19] Tian Li. How lending pool interest rates actually work. https://medium.com/hydro-protocol/how-lending-pool-interest-rates-actually-work-375794e71716, 2019.

[Lim16] Tether Limited. Tether: Fiat currencies on the bitcoin blockchain, 2016. Accessed: 08-06-2020.

[Lim20] Probit Global Service Limited. Probit. https://www.probit.com/en-us/, 2020.

[Mak] Maker. The Maker Protocol: MakerDAO's Multi-Collateral Dai (MCD) System. https://makerdao.com/en/whitepaper/. Accessed: 08-06-2020.

[Mak19a] Maker. The Governance Security Module (GSM). https://blog.makerdao.com/governance-security-module-gsm/, 2019.

[Mak19b] MakerDAO. Makerdao. https://makerdao.com/en/, 2019.

[Mak20] Maker. Governance Security Module (GSM) vote. https://forum.makerdao.com/t/signal-request-should-we-have-another-executive-vote-regarding-the-governance-security-module/1209/25, 2020.

[Mal89] Burton G. Malkiel. *Efficient Market Hypothesis*, pages 127–134. Palgrave Macmillan UK, London, 1989.

[MEGH20] Inc. Maker Ecosystem Growth Holdings. Oasis trade. https://oasis.app/, 2020.

[MG20] Toshiko Matsui and Lewis Gudgeon. The speculative (in) efficiency of the cme bitcoin futures market. In *Mathematical Research for Blockchain Economy*, pages 91–103. Springer, Cham, 2020.

[MHD$^+$19] David R MacIver, Zac Hatfield-Dodds, et al. Hypothesis: A new approach to property-based testing. *Journal of Open Source Software*, 4(43):1891, 2019.

[Mic19] Micah Zoltu. How to turn $20M into $340M in 15 seconds. https://medium.com /coinmonks/how-to-turn-20m-into-340m-in-15-seconds-48d161a42311, 2019.

[Moh09] Rakesh Mohan. Global financial crisis: Causes, impact, policy responses and lessons. *Reserve Bank of India Bulletin*, pages 879–904, 2009.

[MSS20] Amani Moin, Kevin Sekniqi, and Emin Gun Sirer. Sok: A classification framework for stablecoin designs. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*, pages 174–197. Springer, 2020.

[Mut21] Nexus Mutual. A Decentralized Alternative to Insurance. https://nexusmutual. io/, Nov 3, 2021.

[MXC$^+$16] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The Honey Badger of BFT Protocols. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*, pages 31–42, 2016.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://ww w.ussc.gov/sites/default/files/pdf/training/annual-national-training-seminar/2 018/Emerging_Tech_Bitcoin_Crypto.pdf, 2008.

[Nik09] Kleopatra Nikolaou. Liquidity (risk) concepts: definitions and interactions, 2009.

[oE19] Bank of England. Stress testing the uk banking system: key elements of the 2019 annual cyclical scenario, 2019.

[oE20] Bank of England. Interest rates and bank rate. https://www.bankofengland.co .uk/monetary-policy/the-interest-rate-bank-rate, 2020.

[Ohl37] Bertil Ohlin. Some notes on the stockholm theory of savings and investments ii. *The Economic Journal*, 47(186):221–240, 1937.

[OKE20] OKEX. Okex. https://www.okex.com/en/, 2020.

[OS03] Arthur O'sullivan and Steven M Sheffrin. Economics: Principles in action, 2003.

[Pal19] Santiago Palladino. Eip-1967: Standard proxy storage slots. https://eips.ethereum.org/EIPS/eip-1967, 2019.

[Pen18] Alex Lipton Thomas Hardjono Alex Pentland. Digital trade coin (dtc): Towards a more stable digital currency, 2018.

[PG23] Daniel Perez and Lewis Gudgeon. Dissimilar redundancy in defi. In *Mathematical Research for Blockchain Economy: 3rd International Conference MARBLE 2022, Vilamoura, Portugal*, pages 109–125. Springer, 2023.

[PL20] Daniel Perez and Benjamin Livshits. Broken metre: Attacking resource metering in EVM. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020.* The Internet Society, 2020.

[PL21] Daniel Perez and Benjamin Livshits. Smart contract vulnerabilities: Vulnerable does not imply exploited. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1325–1341. USENIX Association, August 2021.

[Pou23] The Brixton Pound. The Brixton Pound. https://brixtonpound.org/, 2023.

[Pro20a] DexIndex Protocol. Dexindex. https://dexindex.io/, 2020.

[Pro20b] Uniswap Protocol. Uniswap. https://uniswap.io/, 2020.

[Pro23a] Eos Protocol. Eos. https://eosnetwork.com/, 2023.

[Pro23b] Solana Protocol. Solana. https://solana.com/, 2023.

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 643–673. Springer, 2017.

[Pul19a] DeFi Pulse. The DeFi Leaderboard. https://defipulse.com/, 2019.

[Pul19b] DeFi Pulse. What is DeFi? https://defipulse.com/blog/what-is-defi/, 2019.

[Res19] The Federal Reserve. Dodd-Frank Act Stress Tests. https://www.federalreserve.gov/supervisionreg/dfa-stress-tests.htm, 2019.

[Reu20] Reuters. Black thursday: Wall street stocks plunge 10% in worst one-day loss in 32 years. https://www.amny.com/business/black-thursday-wall-street-stocks-plunge-10-in-worst-one-day-loss-in-32-years/, 2020.

[Rob34] Dennis H Robertson. Industrial fluctuation and the natural rate of interest. *The Economic Journal*, 44(176):650–656, 1934.

[Sch58] Thomas C Schelling. The strategy of conflict. prospectus for a reorientation of game theory. *Journal of Conflict Resolution*, 2(3):203–264, 1958.

[Sta13] LP StataCorp. Stata time-series reference manual, 2013.

[Swi20] Switcheo. SwitcheoNetwork. https://switcheo.network/, 2020.

[Syn20] Synthetix. Synthetix: Decentralized synthetic assets. https://www.synthetix.io/, 2020.

[Tal07] Nassim Nicholas Taleb. *The black swan: The impact of the highly improbable*, volume 2. Random house, 2007.

[Tet20] Tether. Digital money for a digital age. https://tether.to/, 2020.

[Tro23] Tron. Tron. https://tron.network/, 2023.

[VB15] Fabian Vogelsteller and Vitalik Buterin. EIP-20: ERC-20 Token Standard. https://eips.ethereum.org/EIPS/eip-20, 2015.

[W$^+$14] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[Wal87] Donald A Walker. Walras's theories of tatonnement. *Journal of Political Economy*, 95(4):758–774, 1987.

[Wee20] Week in Ethereum News. Week in Ethereum News December 28, 2019. https://weekinethereumnews.com/week-in-ethereum-news-december-28-2019/, 2020.

[Wie76] Norbert Wiener. Collected works, vol. 1, 1976.

[WPG$^+$21] Sam M Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J Knottenbelt. Sok: Decentralized finance (defi). *arXiv preprint arXiv:2101.08778*, 2021.

[XPCF23] Jiahua Xu, Krzysztof Paruch, Simon Cousaert, and Yebo Feng. Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols. *ACM Computing Surveys*, 55(11):1–50, 2023.

[Yeh96] Y.C. Yeh. Triple-triple redundant 777 primary flight computer. In *1996 IEEE Aerospace Applications Conference. Proceedings*, volume 1, pages 293–307 vol.1, 1996.

[YKC21] Youngseok Yang, Taesoo Kim, and Byung-Gon Chun. Finding consensus bugs in ethereum via multi-transaction differential fuzzing. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, pages 349–365. USENIX Association, July 2021.