Imperial College London
Department of Computing

Security and Efficiency of Collateral in Decentralized Finance

Dominik Lucas Harz

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of the Imperial College, January 2022

Copyright

Statement of Originality

I declare that this thesis was composed by myself and that the work it presents is my own except where otherwise stated.

Abstract

Decentralized Finance (DeFi) promises to be a new contender for a radically new financial system. Its foundations are censorship-resistant, non-custodial, and transparent financial protocols. Securing these protocols is achieved by combining cryptographic primitives with economic incentives instead of relying on trusted intermediaries. In DeFi, financial collateral is the central incentive measure providing repercussions against "misbehaving" agents. However, requiring collateral introduces security and efficiency concerns. (i) Securing DeFi protocols using price-volatile and complex assets requires careful risk management. (ii) Efficiency of capital is diminished since locking assets is an opportunity cost and restricts access to DeFi to agents with sufficient funds. We tackle these issues by developing new protocols to optimize collateral requirements in existing DeFi protocols safely. Our contributions are threefold.

First, we provide a risk-based classification of collateral applied in DeFi protocols. Specifically, the classification serves as the starting point to develop a model capturing the security property of financial collateral with unique risks in DeFi.

Second, we present two protocols that can be integrated into existing DeFi protocols. *Promise* transforms suitable DeFi protocols into a subscription mechanism lowering the initial capital locking requirements thus tackling the capital efficiency of collateral. *Balance* is a protocol to reduce collateral in DeFi protocols safely. Balance is similar to a credit scoring system where "well-behaving" agents enjoy a lowered collateral. As such, Balance can be used both to tailor security of protocols by required per-agent collateral requirements instead of per-protocol requirements and, at the same time, increase capital efficiency of collateral. We demonstrate the practical applicability of Promise and Balance by decreasing collateral in the XCLAIM cross-chain communication protocol by up to 10% under conservative assumptions.

Third, we discuss the practical security of financial collateral. We outline new types of attacks on DeFi protocols secured by collateral through trustless coordination of rational agents and so-called flash loans with the example of the popular Maker protocol. We conclude by noting the perils of constructing collateralized DeFi protocols and outlining strands of future work to increase their security and efficiency.

Acknowledgements

I would like to thank:

- My supervisor, Professor Will Knottenbelt who has been an extraordinary supervisor to me. He gave me both the freedom and trust to explore avenues of research to arrive at one of the most interesting fields I have ever come across.

- Professor Francesca Toni for her input on game theory and mechanism design that played a large role in the final direction of this thesis. Professor Sophia Drossopoulou and Susan Eisenbach for discussions around smart contract programming languages, holistic smart contract specifications, and many more great research pointers. Professor Magnus Boman, who instilled in me passion for research and always encouraged me to explore the boundaries of agents in blockchains.

- ACE358 and all the great people in it and their guests including João, Dimitrios, Paul, Sam, Lewis, Daniel, Alexei, Sirvan, Dragos, Katerina, Rami, and Francesco. Monday funday and Franco Manco will stay a fond memory. My collaborators Rami Khalil, Ryuya Nakamura, Magnus Boman, Andreea Minca, Sam Werner, Benjamin Livshits, Ariah Klages-Mundt, Panayiotis Panayiotou, Josh Lind, Daniel Perez, Alexei Zamyatin, Lewis Gudgeon, and William Knottenbelt.

- Lewis for being an outstanding research partner and friend. Balance would not have seen the light of the world without his contributions. Reliable in good as in bad times. Ariah for being one of the brightest minds I had the chance to collaborate with. Daniel for being always up for social activities and for convincing half of the lab to switch to tiling window managers. My co-conspirator, co-founder, and great friend Alexei. It's incredible to have the chance to bring research to life with our own company. To many more successful launches and a bright future at Interlay.

- My parents, Petra and Albert, for always believing in me and supporting me in any way they possibly can. My grandmother, Else, who is a true inspiration for consistently maintaining a joyful and positive attitude. My sister, Lilli, for being dedicated and caring

while leading the way by getting her PhD first. My friends Daniel, Adrian, and John who came to visit me wherever I went on my journeys. My friend Adam with whom I had the joy to explore many trails. And last, my beautiful wife, Anna, who's support has been unwavering across the globe. I love you with all my heart.

Dedication

To Lilli, for being a true inspiration to her older brother. And to Anna, for life-long companionship.

‘A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools.’

Douglas Adams

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The 2008/2009 financial crisis shattered trust in institutions representing the traditional financial system [Har+17]. Consequently, a group of programmers created what was to become the most successful peer-to-peer electronic cash system to date. Bitcoin ingrained its cause in the genesis block [Nak08]:

The Times 03/Jan/2009 Chancellor on brink of second bailout for banks[1]

Now, more than ten years later, a flourishing DeFi ecosystem promises to offer users a comprehensive set of financial services. The services range from financial lending and asset exchanges to price-stable currencies, derivatives, and portfolio management products [Wer+21].

The DeFi space is in a nascent stage. As of December 2021, DeFi protocols have combined market capitalization of USD 264bn, dwarfed by the USD 11tn market capitalization of Gold, USD 3tn of Apple alone, and even compared to the USD 1.1tn market capitalization of Bitcoin. Nonetheless, the growth of DeFi protocols is impressive. As shown in Fig. 1.1, DeFi experienced exponential growth measured by its Total Value Locked (TVL). TVL refers to the amount of capital deposited in DeFi protocols, such as collateral in lending protocols and stablecoins and

[1] Bitcoin genesis block https://blockstream.info/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f (Accessed: 26.02.2021).

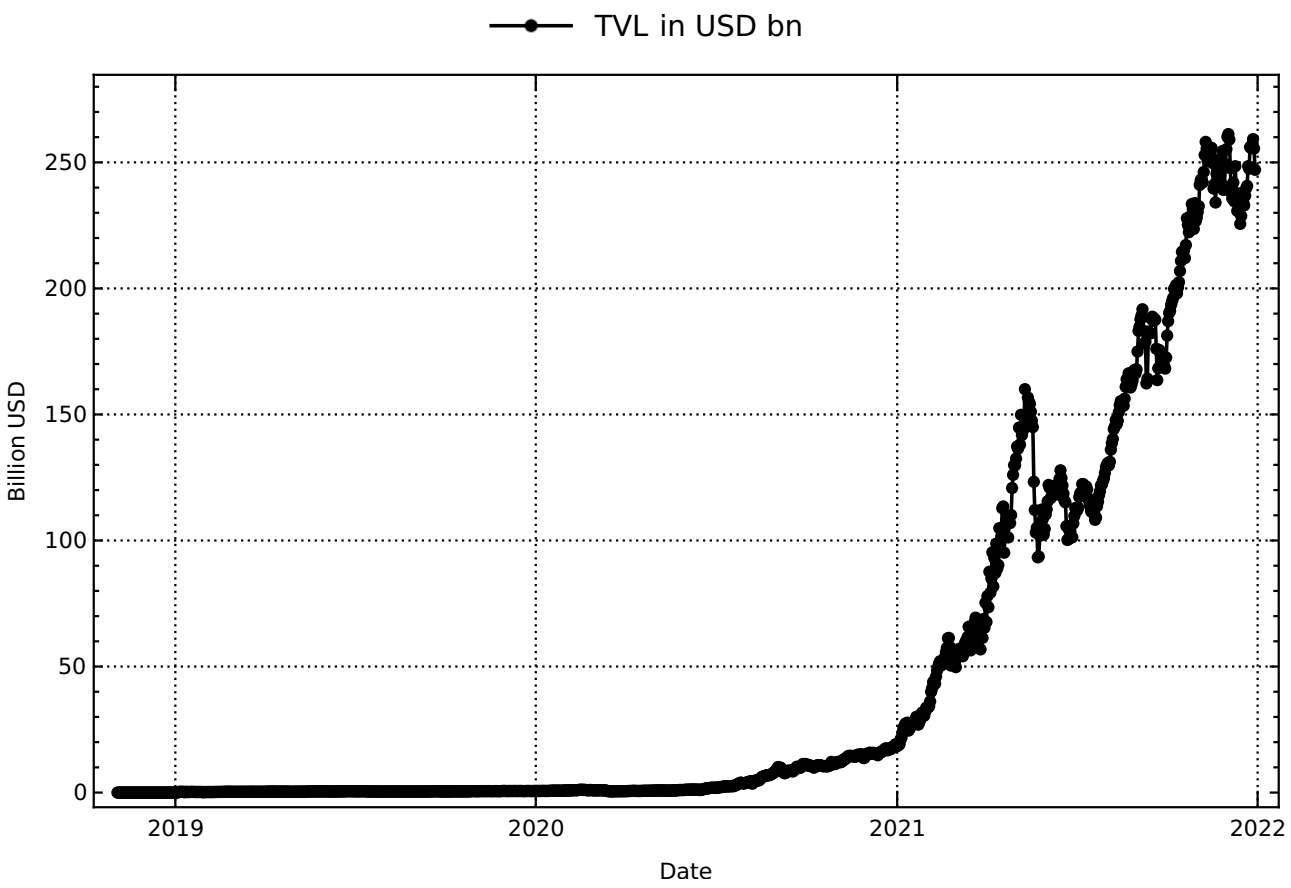

Figure 1.1: DeFi's Total Value Locked (TVL) in billion USD from November 2018 to December 2021. Data from https://defillama.com/ (Accessed: 29.12.2021).

liquidity in Automated Market Makers (AMMs). DeFi's TVL grew from less than USD 1mm in January 2020 to over USD 264bn in a mere two years.

Moreover, CoinGecko lists over 370 DeFi protocols[2]. Uniswap, a DeFi AMM, overtook one of the largest centralized cryptocurrency exchanges, Coinbase, by trading volume in August 2020[3]. Due to DeFi's pseudonymous nature, assessing the total number of users is challenging. Estimates state that on Ethereum alone more than 4mm unique addresses are interacting with DeFi protocols[4]. Notably, a single user can have more than one address.

Large companies are joining this new space as well. IBM and Meta have dedicated blockchain teams, and the number of new blockchain projects is ever increasing. Likewise, Venture Capital firms are pouring money into promising start-ups. Sotheby's Metaverse marketplace for Non-

[2] https://www.coingecko.com/en/categories/decentralized-finance-defi (Accessed: 29.12.2021)
[3] https://cryptobriefing.com/uniswaps-daily-volume-overtook-coinbase-more-80-million/ (Accessed: 29.12.2021)
[4] https://dune.xyz/rchen8/defi-users-over-time (Accessed: 29.12.2021)

Fungible Tokens (NFTs) generated USD 100mm in sales within eight months of operation[5].

DeFi Use Cases. DeFi aspires to change the financial system's fundamental principle: replace trust in institutions and other market participants with decentralized, (pseudo)anonymous networks based on cryptographic primitives and economic incentives. To understand the potential impact of DeFi, we are illuminating exemplary applications used by the author of this thesis. For a complete overview, we refer to [Lau+20]. The common factor across all these use cases is that merely the accessibility to funds on a blockchain like Ethereum suffices to perform these use cases. None of the protocols listed below requires any centralized user accounts or logins; anyone can execute them anywhere, anytime using a cryptocurrency wallet with available funds.

Passive income generation. DeFi allows generating passive income via depositing capital into protocols. A low-risk variant of a passive income strategy starts with acquiring stablecoins pegged to a fiat currency like the USD. A popular strategy is to acquire USDC or Dai (one centralized and one decentralized stablecoin) and lend it to a protocol like Compound [Com21a] or Aave [Aav21]. As of December 2021, this allows generating around 3% of Annual Percentage Yield (APY), a measurement that considers the accrued interest, including compounding. In return, borrowers have to pay back around 4% of Annual Percentage Rate (APR) when returning their loan. Moreover, the borrowers have to over-collateralize their borrowed asset with another asset. The main risks in this operation are borrowers defaulting on their loans, which can significantly increase in "Black Thursday" like events [Gud+20b], and technical smart contract bugs, e.g., the risks described in [Tsa+18].

Decentralized lending. One of the first and still most widely used protocols is Maker [Mak21a]. In its first protocol iteration, Maker allowed the creation of a stablecoin, Dai, for a collateralized position with ETH, Ethereum's native asset. Dai is a synthetic asset, i.e., users cannot exchange it for "real" USD from within Maker, but it is widely accepted on exchanges. Dai became popular as it allows to generate a price-stable asset while maintaining exposure to ETH. A user

[5] https://www.bloomberg.com/news/articles/2021-12-15/sotheby-s-makes-100-million-in-nft-sales-with-younger-audience (Accessed: 29.12.2021)

deposits on day one USD 1000 worth of ETH (0.25 ETH) into Maker to generate USD 500 worth of Dai, i.e., at a collateralization rate of 200%, which is 50 percent-points above Maker's enforced liquidation threshold. Next, users can invest USD 500 worth of Dai in a passive income strategy. In December 2021, the APR of paying back the Dai loan was 2.75%. The high APR renders the passive income strategy unprofitable considering substantial transaction costs on Ethereum. The alternative is to use Dai as a payment method or convert it to fiat to spend the money in the "real world". The underlying assumption of this strategy is that ETH's price will appreciate over time. Assume that on day 100, the initial 0.25 ETH is now worth USD 5000. The borrower can withdraw 0.2 ETH (USD 4000) to keep the collateralization at 200% and sell 0.025 ETH (500 USD) for 500 Dai and pay back the initial loan. Instead of selling 0.125 ETH on day one, the borrower had to sell 0.025 ETH for the same outcome. Apart from smart contract risks, the ETH price might fall such that the borrow is liquidated with the consequence that the Dai debt resets while all ETH is lost.

Asset exchanges. Some refer to the success of Automated Market Makers as DeFi's "zero to one" moment [Ber20a]. Popular exchanges like Uniswap [Uni21] use a Constant Function Market Maker (CFMM). Specifically, Uniswap V1 pioneered the $x \quad y = k$ equation to calculate the exchange price between an amount x of an asset X and amount y of an asset Y based on a constant k. So-called Liquidity Providers (LPs) provide a pair of assets in a trading pool. Other users can then trade against the pool. Having CFMMs allows trading even assets with low liquidity with limited, or, at least, predictable slippage. AMMs form the most popular asset exchanges in DeFi with a range of different functions, e.g., for trades between assets with approximately the same price as Curve [Cur21] and generalized AMMs that allow the creation of any equation including asset baskets and risk-weighted assets [Bal21]. The main critique against AMMs is their capital inefficiency and the LP risk to be worse off in USD terms for providing liquidity instead of passively holding one of the assets (so-called *impermanent loss*).

Portfolio managers. A popular strategy to convince users to provide liquidity is to pay them via "airdropping" governance tokens [Ber20b]. In this process, a DeFi protocol will automatically allocate a token that usually grants the holders to participate in governance, e.g., [Com21a; Aav21], and in rare circumstances also let them participate in the revenue generated by the

protocol, e.g., [Cur21; Oly21]. As projects with new tokens appear often, it can be hard to find an optimal strategy to provide liquidity. The established term in DeFi for providing liquidity, extracting governance tokens, and redeploying liquidity where yield is highest is "yield farming". Portfolio managing protocols simplify the yield farming process by automatically redeploying liquidity across protocols or selling the received tokens for the originally provided asset to increase the yield generation, e.g., [Yea21a].

Two compelling use cases not listed here are (i) structured products that combine acquiring assets together with a derivative to allow building more advanced investment strategies, e.g., [Rib21] and (ii) insurance-like products that protect against protocol or custodian failures, e.g., [Nex21].

1.1 Motivation

In its ideal form, DeFi seeks to deliver censorship-resistant, non-custodial, and transparent financial services. However, what does this mean? In today's financial system, where one entrusts third parties like banks and other institutions with controlling their assets, tomorrow's DeFi world sets out to give users back control. This control comes in three essential forms.

1. **Censorship-resistance:** Users have direct access to financial services. In contrast to the traditional financial system, DeFi offers free participation in any on-chain financial protocol.

2. **Non-custodial:** Anyone has complete control over their assets by holding them in a digital wallet. Like holding cash in a physical wallet, third parties cannot seize the assets of the user. Even when locking assets, DeFi can enforce that solely the user with the correct key maintains access.

3. **Transparent:** Anyone is able to verify the details of financial services. Instead of institutions and governance bodies conducting credit checks or opening bank accounts based on

privately conducted processes, access to DeFi protocols is given based on public criteria encoded in smart contracts.

The majority of DeFi projects today are operating on Ethereum [Woo14]. Ethereum provides a Turing-complete distributed virtual machine entitled the Ethereum Virtual Machine (EVM). Despite its scalability and cost inefficiencies, it allows protocol designers to create a wide variety of DeFi protocols that resemble the services found in the traditional financial system.

Importantly, DeFi has introduced and popularized financial innovations of its own. Innovations include AMM exchanges, the encoding of governance using tokens, and new ways to raise funding, e.g., via Initial Coin Offerings (ICOs), lock-drops, or Decentralized Autonomous ICOs (DAICOs), cf. [But18].

However, one of the most innovative principles of DeFi is its composable nature. It has become almost a standard to refer to different DeFi protocols and services as Lego blocks. This analogy works well since, similar to Lego blocks, DeFi services often use a standard set of APIs (e.g., ERC20 token, ERC721 NFT), and protocol designers can freely combine services as they are accessible by anyone. Composability allows for almost unhindered innovation and network effects as a protocol designer merely needs to deploy a contract that can readily integrate with other existing protocols without writing custom integrations or explicitly granting access[6].

Consequently, an open financial system creates a large attack surface. The ease of creating new financial assets based on ERC20 tokens led to the 2017/2018 ICO hype where a range of highly volatile assets flooded the market, including outright scams. AMMs allow anyone to create new trading pairs, enabling scammers to create fake token pairs. Also, Dai, the highest capitalized decentralized stablecoin to date, suffered severely from the "Black Thursday" event in March 2020. Last, the open and unrestricted interaction with protocols combined with projects rushing to release their protocols without regulatory oversight leads to a range of exploited protocols. While being transparent, it is alarming to see millions of USD being lost due to security flaws[7].

[6] A excellent example for this is Uniswap. Trading pairs can be added at will to the AMM-based exchange without control.

[7] For a current list of the most expensive vulnerabilities and their exploitation, see https://rekt.eth.link/leaderboard/.

1.1.1 Collateral Usage in DeFi

Agents govern decentralized systems. Since purely cryptographic security is often impossible or impractical, DeFi protocols are designed as cryptoeconomic systems. DeFi protocols combine cryptographic proofs and economic incentives to securely provide financial services to (pseudo)anonymous agents in an open-world system. This unrestricted access requires the protocol designers to tackle their protocols' technical and economic security.

While there is a realm of literature on ensuring technical security, such literature is still sparse on the economic side. Specifically, one of the most common instruments in DeFi security — financial collateral — has not yet been analyzed in depth.

Financial collateral in DeFi serves as a repercussion for misbehaving agents and ensures that agents have "skin in the game". There are five main areas where financial collateral is applied:

1. **Collateralized Loans**: Popular applications such as Aave and Compound allow anyone to take out a loan in cryptocurrencies. The borrower has to provide collateral in one currency to borrow another asset. Since the system is permissionless with unknown identities, the loans must be *over-collateralized*, i.e., the value of the collateral asset must be higher by a protocol-defined threshold than the value of the borrowed asset [Gud+20b].

2. **Stablecoins**: Due to the volatile prices of cryptocurrencies, stablecoins are popular to facilitate payments. Centralized stablecoins like USDC or USDT function similar to central banks where a trusted custodian holds the backing asset. Decentralized stablecoins, on the other hand, employ collateral to derive value for the stablecoin. The largest stablecoin by market capitalization to date, Dai by Maker, allows agents to generate new Dai by backing it with various other assets. Contemporary designs employ different collateral constructions, with a more detailed overview found in [Kla+20].

3. **Proof-of-Stake**: We summarize protocols that employ collateral for the core consensus layer, like Proof-of-Stake protocols such as Ethereum 2.0, Polkadot, or Cosmos. These protocols require validators to stake collateral to punish them for consensus violations, cf. [BGM16; DPS19; But16; KN12; Fan+19].

4. **Bridges**: Cross-chain bridges between different Layer-1 (L1) systems or between L1 and Layer-2 (L2) systems may employ collateral to safeguard assets being locked on the system. For an informal introduction, [Ber21] is recommended, while [Zam+21] provides a formal overview proving that cross-chain communication is impossible without a trusted third party.

5. **Governance**: Collateral is also used in on-chain governance. On-chain governance systems require the proposer of a change to lock a certain amount of governance tokens for the proposal. Governance systems like Aragon [Ara21] or Polkadot [Web21] destroy the locked collateral if the proposal is rejected (commonly referred to as "slashing"), thus enforcing proposers to ensure beforehand that their requested change has enough community support.

The above categories form a substantial value of DeFi. The top two collateralized lending protocols, Aave and Compound, make up USD 23.2bn of financial collateral locked in DeFi alone [8].

1.1.2 Collateral Liquidations

The usage of collateral implies that if something goes wrong, the collateral can be used to reimburse a mistreated party, punish the offender, or both. Punishments frequently happen in DeFi. The act of taking away an agent's collateral is often called "slashing". The underlying reason for slashing collateral can take two different forms.

First, **external events**, like the price fluctuation of an asset, can lead to liquidations. This applies to the *Collateralized Loans*, *Stablecoins*, and *Bridges*. These protocols require that users maintain an over-collateralized position. If the collateralization drops to a certain liquidation threshold, the position is liquidated. On the Black Thursday event in March 2020, USD 10mm of assets were liquidated in a single day in the Maker protocol [Gud+20b]. The total amount of liquidations in Aave accumulate to around USD 352mm[9].

[8] From https://defillama.com/protocols/lending (Accessed: 1.1.2022)
[9] From https://aavewatch.vercel.app/liquidations (Accessed: 1.1.2022)

Second, **protocol rules** lead to liquidations. The difference to external events is that the liquidation is a direct result of the action or inaction of an agent. For example, governance systems like Polkadot or Aragon require collateral to submit a change proposal. If the proposal is rejected, the proposer loses its collateral. Hence, the action of suggesting a system change without a majority leads to the slashing event. Inaction can also cause liquidations, for example, in cross-chain bridges like XCLAIM [Zam+19], where a collateralized intermediary must release tokens to their rightful owner within a specific time limit. If the intermediary fails to release the tokens in time, they are slashed.

In both cases, liquidations are necessary to stabilize the system towards its desired state as intended by the designer of the DeFi protocol. Concrete examples include (i) keeping the overall system collateralized in stablecoin and lending protocols and (ii) punishing proposers of undesired changes, e.g., proposing changes that are not in the interest of the majority.

Liquidations are a last-resort technique. Hence, the question arises how collateral can be best determined to keep the system itself secure, i.e., maintain a stable state, while keeping liquidations to a necessary minimum.

1.1.3 Collateral Inefficiency

Locked capital in the form of collateral is expensive: being unable to access funds, agents face an opportunity cost of foregone returns that they could have accrued in alternative investments. In blockchain-based systems, agents do not need to trust each other to interact. Implicitly, identities between different agents are unknown and irrelevant for the context of interaction. For example, taking out a loan must be executed without any other trust than the financial collateral provided by the agent.

Nevertheless, sources of uncertainty, such as exchange rate volatility, can require *over-collateralization*, where more capital than necessary is locked up as a buffer. Hence, *cryptoeconomic* protocol design plays a vital role in balancing security with locked capital costs.

A utility-maximizing agent chooses which action to perform based on self-interest, ignoring

the impact of their action on other agents. However, cheating or other malicious behavior by self-interested agents may harm the protocol as a whole. Therefore, protocols typically involve two types of incentives, e.g., [PD16; KGF19; TR17; Kos+16; Kal+18; DEF18; Zam+19]: *(i)* payouts motivate agents to act in the best interest of the protocol, and *(ii)* deposits prevent malicious actions by punishing misbehaving agents. Choosing the appropriate collateral is challenging due to private information and event dependency:

Definition 1.1 (Private information). *A valuation $v_A(\ )$ encodes the preference of an agent A for a speci c outcome in a protocol depending on an action* [10]*. We de ne this information as private, i.e., only known to the agent*[11].

Definition 1.2 (Event-dependency). *Collateral provided in cryptocurrencies can be subject to external events. Event dependency is caused by events that occur outside of the underlying blockchain and a ects the level of required collateral.*

In practice, over-collateralization is common. In Maker, the average Dai position is collateralized with around 300% of ETH while the requirement is 150% [Mak19].

1.2 Objectives

DeFi does not come without risk. The ease of accessibility of DeFi protocols combined with the stark increase in value makes DeFi systems appealing targets for adversaries. In Table 1.1, we select exploits that happened in 2021 to emphasize the risk of using DeFi. While DeFi offers tangible custodial and censorship-resistance benefits, building these protocols securely and efficiently is a demanding endeavor. Estimations of the capital lost in blockchain-related security incidents range close to USD 24bn [Slo21]. Moreover, as Table 1.1 illustrates, even massively capitalized and widely used DeFi protocols are at risk.

The road for DeFi to become the basis for a new financial system is long. As part of this journey, we aim to steer the development of DeFi protocols in a more secure and efficient direction. The

[10] For brevity, we write $v_A(\ )$ as v_A in the rest of the thesis.
[11] See also Chapter 9.4.1 on games with incomplete information in [Nis+07].

Table 1.1: Selected DeFi security incidents of popular DeFi protocols including TVL as of 29 December 2021 sorted by disclosure date.

Protocol	TVL	Loss	Disclosure	Attack vector
BadgerDAO	USD 1.02bn	USD 120mm	2.12.2021	Malicious JS injection on the BadgerDAO UI allowing adversary to withdraw funds from user wallets [Bad21].
Lido Finance	USD 11.84bn	USD 0	8.10.2021	Node operators of the Lido staking system could have stolen up to 20,000 ETH. The vulnerability was disclosed as part of a bug bounty program and not exploited [Imm21].
Compound	USD 9.49bn	USD 68mm	4.10.2021	A smart contract bug resulted in more COMP governance tokens being claimable by certain users than intended [Com21b].
yearn. nance	USD 5.66bn	USD 11mm	4.2.2021	Applying flash loans to imbalance the exchange rate in another protocol, impacting the ability to withdraw funds from the yearn protocol [Yea21b].

work conducted as part of this thesis follows two ambitions:

Increasing the Security of DeFi Protocols. For DeFi to become the standard for financial infrastructure, it should be *safe* to use for any participant. We seek to equip creators and users of DeFi protocols with tools to understand and reason about their protocol security. At the same time, we strive to make DeFi more secure by proposing new protocols that can help increase the economic security of the protocols as a whole.

To this end, we can differentiate between *technical security* and *economic security* [Wer+21]. Technical security ensures that a DeFi protocol's implementation follows its specification accurately. A wide range of literature is concerned with this specific aspect, i.e., smart contract security and verification [BP17; BZ18; Tsa+18] as well as consensus algorithms [Bad+17; Aiy+05; GKL15]. On the other hand, economic security seeks to ensure that the economic mechanism underlying the DeFi protocol is stable for the desired outcome. If we take, for example, a lending protocol, a marketplace where agents are supplying and borrowing assets,

then the desired outcome might be described by three goals: (i) Agents supplying assets profit from the supply of the assets under consideration of risks, (ii) agents borrowing assets can pay back their assets and can borrow at reasonable rates, and (iii) the operator, e.g., a foundation, company, or Decentralized Autonomous Organization (DAO), of the platform can maintain the business sustainably. From this example, it becomes apparent that economic security revolves around agents' incentives. As part of these incentives, financial collateral plays a significant role in DeFi.

This work aims to provide suitable models for financial collateral in DeFi protocols. This should serve users and DeFi protocol creators alike to determine safe collateralization bounds that minimize liquidations. Moreover, it should reduce the overall risk of using DeFi. To prevent the mistakes from the 2008/2009 financial crisis, combining financial products (i.e., DeFi protocols) should consider the risk holistically to prevent contagion of failures.

Increasing the Capital Efficiency of DeFi Protocols DeFi enthusiasts pioneered new products that skyrocketed in popularity over the last two years. When building and shipping new products, decisions about the feature set must be made, and the current DeFi protocols opted to generalize capital requirements over the entire protocol rather than on a user level. As a comparison, this is as if a bank would offer the same conditions for a loan to anyone. Naturally, the bank offering the same conditions to anyone would have to model the risk according to the highest-risk individuals.

These considerations are taken by collateral requirements in DeFi today. Maker, Compound, Aave, et al. require the same collateralization rates for everyone. Generalizing collateral rates across all users might result from two considerations: (i) time-to-market is an important consideration and integrating a user-specific credit scoring takes effort, and (ii) in an anonymous system with Sybil-identities, building a credit scoring or reputation system is a challenging task.

While this thesis outlines a model to argue about the security of collateral in DeFi, it also explores the creation of new systems to increase collateral efficiency.

1.3 Contributions

DeFi as Cryptoeconomic Protocols The first contribution of this thesis is to model DeFi protocols as cryptoeconomic protocols [But17]. Essentially, cryptoeconomic protocols are a subset of the algorithmic mechanism design space, cf. [NR01; ST13; Rou15; Nis14]. A set of economically rational agents interact with a mechanism with quasi-linear utilities, i.e., the values of outcomes of participating agents can be measured in monetary units like tokens. Further, we use a dominant strategy model where we assume worst-case outcomes instead of a distribution employing a Bayesian prior. This approach is in line with system models and cryptographic proving approaches where the goal of the system design is to prevent worst-case outcomes under (bounded) worst-case assumptions. In addition, we can utilize cryptographic proofs in the mechanism design to ensure that agents may omit but not falsely report information.

Categorization of Collateral Risks in DeFi We generalize the risk-based model in [Kla+20] for DeFi protocols and confine it towards collateralization risks. Based on this model, we can reason about collateralization risks in DeFi within single protocols. Moreover, in future work, this model can be extended to reason about risks for composing DeFi protocols. For example, a lending protocol may accept a token that encodes a collateralized debt position as collateral itself. This construction is similar to the Collateralized Debt Obligation (CDO) structures. As the financial crisis of 2008/2009 has shown, understanding the risk of such composing products is essential to prevent cascading failures, cf. [Gud+20b].

Reducing Initial Collateral Lockup To increase the efficiency of collateral, we introduce the `Promise` protocol [Har+20]. `Promise` can be integrated into existing DeFi protocols to reduce the initial capital requirements by 1% to 3.5%. We introduce the system, show how it can be applied to XCLAIM [Zam+19], and prove its security.

Dynamically Adjusting Collateral The core contribution of this thesis is the `Balance` protocol [Har+19]. In `Balance`, we combine the insight of modeling DeFi protocols as cryptoeco-

nomic protocols together with reputation systems, cf. [PS13] to create a "credit-scoring" system for anonymous agents. In particular, we allow agents to have *individual* collateralization levels based on the classification of their objective interactions with a DeFi protocol categorized into desired and undesired actions. Intuitively, agents that perform desired actions in a protocol like taking out and paying back loans receive a good reputation, which qualifies them for a lower collateralization level. In `Balance`, we model agents' interactions with a collateralized protocol as a sequential game of self-interested agents. We show that collateral can be reduced by 10% for the XCLAIM protocol. The `Balance` protocol can be integrated into collateralized DeFi protocols, and we argue how it withstands reputation system attacks [KC09].

Attacking Collateralized DeFi Protocols Our last contribution is the practical attack description of the Maker protocol using a novel technique called flash-loans [Gud+20b]. In particular, we leverage the ability of Ethereum to loan capital within a single transaction to acquire enough governance tokens that would have allowed us to drain all collateral from Maker and mint arbitrary amounts of new Dai. The attack vector on the governance structure was initially discovered by [Zol19] and quantified by the authors of [Gud+20b]. Our core contribution is the discovery of executing the attack within a single Ethereum transaction that rendered the attack practically cost-free, impossible to stop without a prior security measure, and removed the necessity to coordinate the attack with other adversaries. We disclosed the attack to Maker in February 2020, and adequate measures have been taken to prevent executing of the attack. The author has published the findings and the resolve in greater detail in a blog post [Har20].

1.4 Publications

The following works form the foundation of this thesis.

- Dominik Harz, Lewis Gudgeon, Arthur Gervais, and William J. Knottenbelt. 2019. Balance: Dynamic Adjustment of Cryptocurrency Deposits. In Proceedings of the 2019 ACM

SIGSAC Conference on Computer and Communications Security (CCS '19). Association for Computing Machinery, New York, NY, USA, 1485–1502.

- Dominik Harz, Lewis Gudgeon, Rami Khalil, Alexei Zamyatin. 2020. Promise: Leveraging Future Gains for Collateral Reduction. In Mathematical Research for Blockchain Economy. Springer Proceedings in Business and Economics. Springer, Cham.

Moreover, the following papers include relevant work by the author that is incorporated into this thesis where indicated.

- Sam M. Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J. Knottenbelt. 2021. SoK: Decentralized Finance (DeFi). Retrieved from http://arxiv.org/abs/2101.08778
- Ariah Klages-Mundt, Dominik Harz, Lewis Gudgeon, Jun-You Liu, and Andreea Minca. 2020. Stablecoins 2.0: Economic Foundations and Risk-based Models. In Proceedings of the 2nd ACM Conference on Advances in Financial Technologies (AFT '20). Association for Computing Machinery, New York, NY, USA, 59–79.
- Lewis Gudgeon, Daniel Perez, Dominik Harz, Ben Livshits, and Arthur Gervais. 2020. The Decentralized Financial Crisis. In Proceedings of the IEEE 2020 Crypto Valley Conference on Blockchain Technology (CVCBT 2020).
- Alexei Zamyatin, Dominik Harz, Joshua Lind, Panayiotis Panayiotou, Arthur Gervais and William J. Knottenbelt. 2019. XCLAIM: Trustless, Interoperable, Cryptocurrency-Backed Assets. In Proceedings of the IEEE Symposium on Security and Privacy, May 2019., 1254–1271.

Last, the following papers include work by the author that is not present in this thesis:

- Ryuya Nakamura, Takayuki Jimba and Dominik Harz. 2019. Refinement and Verification of CBC Casper. In Proceedings IEEE 2019 Crypto Valley Conference on Blockchain Technology, CVCBT 2019.

- Dominik Harz and William J. Knottenbelt. 2018. Towards Safer Smart Contracts: A Survey of Languages and Verification Methods. Retrieved from https://arxiv.org/abs/1809.09805

Chapter 2

Background Theory

This chapter introduces the main DeFi protocol types that utilize collateral. For a complete overview of all DeFi protocol types, we refer to [Wer+21; Lau+20]. Next, we outline the foundations of DeFi protocols, including the blockchain layer, governance, and oracles. Moreover, we introduce the concept of cryptoeconomic protocols and treat DeFi as a collection of such protocols.

2.1 DeFi Protocols

Asset Exchanges Decentralized asset exchanges enable two main features: First, they allow anyone to create a new trading pair between any asset. This enables a maximum amount of possible trading combinations as there are no restrictions to list new assets. Second, DeFi exchanges allow anyone to trade assets without having to fulfill Anti-Money Laundering (AML) or Know Your Customer (KYC) requirements compared to traditional exchanges. While this raises regulatory concerns, DeFi exchanges work well in practice [AKN19]. The most popular type is AMMs, cf. [Uni21; Cur21; Bal21]. AMMs omit collateral as trades are executed directly with the liquidity provided by LPs within a single atomic transaction. However, alternative exchanges exist like order-book exchanges, cf. [WB17], and auction-type exchanges. These might employ collateral to motivate agents to fulfill orders. However, due to the technical

complexity of fully decentralizing order-book exchanges and auction exchanges having a slight advantage over AMM-type exchanges, they have not seen much adoption.

Loanable Funds In PLFs [Gud+20c], agents are brought together to supply and borrow capital in a marketplace. Suppliers receive interest on their funds, whereas borrowers receive capital. The main functionality of these protocols is to find a stable equilibrium between the supply and demand of certain assets. Borrowing assets are fully collateralized such that the value of the borrowed asset is always smaller than the provided collateral. Prevalent examples are Aave and Compound.

Stablecoins The main two types of stablecoins are custodial like USDC and USDT as well as non-custodial stablecoins like Dai [Kla+20]. Centralized stablecoins can be modeled using traditional financial models, including reserve funds, fractional reserve funds, and central bank models. They are not collateralized, or the collateral is not placed within a programmatic smart contract. Hence, we exclude custodial stablecoins from this work. Custodial stablecoins, specifically USDC and USDT, have a prominent role in DeFi, though with a combined market capitalization of USD 1.2tn.

Non-custodial stablecoins utilize collateral to back the issuance of new tokens. A detailed introduction to the different types of non-custodial stablecoins is given in [Kla+20]. We note that our model in Ch. 3 results from this work.

Bridges Bridges connect different systems to make assets native to one system usable in the other system. Examples include wBTC, where BTC is locked on the Bitcoin blockchain to make a "wrapped" representation of BTC available on Ethereum. Moreover, so-called layer-two systems facilitate interactions with assets without necessarily employing their consensus but inheriting their security from the layer-one system [Gud+20a]. Bridges are used when using an asset in another system is desirable. Concretely, using BTC on other blockchains like Ethereum is desirable since Ethereum supports generic DeFi protocols and is not limited to asset transfers. Moreover, using Ethereum native assets on layer-two systems like Arbitrum, Polygon,

or Optimism or other blockchains like Avalanche, NEAR, Solana or Polkadot is desirable since transaction cost on these systems are lower than on Ethereum. Solving the scalability of blockchains will require a heterogeneous set of different blockchains that are specialized for particular applications [Cro+16; Zam+21]. Collateral is used in bridges in two ways [Ber21]: If the bridge is custodial, the custodian of the asset may have to provide collateral to prevent the custodian from stealing the asset, cf. [Zam+19]. If the bridge is non-custodial, collateral may still be employed to prevent liveness failures.

Proof-of-Stake Strictly speaking, Proof-of-Stake (PoS) might not be considered a DeFi protocol. After all, PoS is part of the core blockchain responsible for ensuring incentives for reaching consensus and promoting rational miners or validators to produce blocks correctly. PoS protocols have come a long way since their first discussion in Bitcoin talk forums [KN12] with major blockchains like Cardano [Dav+18; Bad+17], Cosmos [Ten16], and Polkadot [Web21] already employing PoS and Ethereum planning on switching from PoW to PoS [Eth19].

PoS can be considered DeFi if we look at it from a functional perspective. Validators stake funds to participate in the PoS system and produce new blocks. This locked collateral can be slashed if the validators violate the rules of the staking protocol. For correctly producing blocks, validators receive, depending on the system, block rewards and transaction fees. While this operation is not feasible to the average user as minimum availability must be guaranteed, many PoS protocols, e.g., [Web21], allow regular users to nominate or delegate stake to validators, e.g., [Lid21]. In turn, users receive a staking token that can be further integrated into other DeFi protocols. The staking derivative is an interest-bearing asset that can be further utilized, e.g., as collateral in PLFs.

2.2 Foundations

2.2.1 Blockchains

For the purpose of this work, we will use *blockchains* and *distributed ledgers* as synonyms. This blockchain provides a ledger functionality as, for example, defined in [PSS17; Bad+17; Dav+18; Bad+18]. The ledger functionality allows anyone to submit transactions. These transactions are validated and eventually added to a new block to be included in the global state. The state might not change either after being explicitly finalized, e.g., [CL99; Ten16] or given that under a *common pre x*, the probability of changing the state is negligible [GKL15; Ger+15].

Conceptually, blockchains consist of three primary components [Bon+15]:

Transactions and Smart Contracts On a high level, blockchains can be split into account-based (e.g., Ethereum [But13]) and Unspent Transaction Output (UTXO) models (e.g., Bitcoin [Nak08]). In both cases, the state of the entire system is encoded via transactions. In the UTXO model, due to a lack of accounts, the balance of each "user" is represented as the sum of the UTXOs controlled by their private keys. In account-based models, the balance of each user is represented by the current balance itself encoded in the state. The state can be modified via transactions: these are sent by users of the blockchain to miners to be included in a block. A particular form of a transaction is a program (often called smart contract). Smart contracts encode conditional handling of transactions. For example, they could encode that user A pays user B a number of coins X if user B submits a pre-image of a hash to the smart contract. Instead of relying on a middle man, the smart contract enforces the conditions of checking the hash pre-image and processing of payments. However, smart contracts need to be triggered by a transaction. Smart contracts allow the creation of a wide range of DeFi and other protocols. Further reading on transactions and smart contracts can be found in, e.g., [Luu+15; Luu+16].

Consensus and Mining The critical problem in any decentralized ledger is the *double-spending* attack. Any rational holder of funds in the blockchain is incentivized to spend their

existing funds more than once. Hence, the blockchain must find a way to ensure that funds are spent exactly once. Bitcoin, and most other ledgers, achieve this by logging all transactions into a chain of blocks, with each block containing a set of transactions. Each miner should only include valid transactions into a block, including the solution of a challenging computational puzzle (i.e., Proof-of-Work) [Nak08; Bon+15]. The chain with the most blocks, i.e., the longest chain with correct blocks, is deemed canonical. Other nodes in the network that receive blocks with incorrect transactions, should discard them and not append them to their longest chain. This is often referred to as *Nakamoto consensus*. Other chains may replace the PoW part with having miners stake funds on the longest chain [Bad+18; KN12].

Nakamoto consensus and related consensus protocols do not have *nality*. Instead, after a common prefix, often termed security parameter k, blocks are considered final [GKL15; Ger+15]. Other consensus protocols have explicit finality. However, they have to restrict the set of miners to a known set [CL99]. In either case, less than 1=3 of miners are assumed to be malicious [ES14].

Last, miners receive transaction fees and block rewards for performing this work. While this ensures that rational miners have an incentive to continue to produce blocks, mining rewards are a challenging problem in itself. First, the structure of mining incentives might motivate miners to delay blocks [Car+16]. Second, in PoS, wealth compounding effects increase the centralization of validators [Fan+19]. Last, miners can front-run transactions to make profitable trades themselves as they are free to re-order, delay, or censor transactions in any way they see fit [Dai+20]. Specifically, the last issue, termed Miner Extractable Value (MEV), has a significant impact on DeFi, as we will explore in Ch. 3.

Peer-to-Peer Communication Bitcoin, Ethereum, and most other blockchains use ad-hoc, peer-to-peer (p2p) communication to broadcast blocks and transactions. While looking at the different construction of the p2p layer and attack vectors, cf. [AZV17], is interesting; we are primarily concerned with the impacts to DeFi and collateral. If we assume that the blockchain layer and its p2p network is secure, looking at the p2p layer becomes interesting when considering the security of cross-chain assets. Here, trustless communication requires the

usage of chain relays [Zam+21]. Due to the p2p communication, these relays can be attacked and might be more vulnerable to p2p-level attacks than the canonical network. For example, a relay poisoning attack on a Bitcoin light client, like the Ethereum BTC Relay, is easier to execute than to convince the entire Bitcoin network of invalid blocks. This can have an impact when cross-chain collateral is used, e.g., wBTC in DeFi on Ethereum.

2.2.2 Governance

DeFi protocols like asset exchanges, PLFs, or bridges evolve over time. To steer this development, some form of governance exists that decides which changes are introduced to a protocol. This might include adding new features, changing existing parameters, or fundamentally changing how the protocol works [ROH16; Yer17].

Governance systems are hard to build: Bitcoin, for example, does not have an on-chain governance system, instead a purely social governance mechanism in the form of BIPs, and consent by developers and miners is used to steer Bitcoin development. As Bitcoin Cash and Bitcoin SV forks show, this process is not without pain. "The Blocksize War" by Jonathan Bier gives detailed insight into this subject [Bie21].

While Ethereum adjusted a similar social consensus mechanism, many DeFi protocols opted for a new form of governance. Protocols including Uniswap, Maker, Compound, Synthetix, and Aave use a dedicated governance token to quantify each community member's voice by representing their tokens. Governance participants can vote on changes to the protocol using their tokens.

However, this also introduces issues: Parties that hold the majority of tokens can change protocol rules without any other party blocking this. Moreover, token allocations can be skewed towards a few parties like early investors of a project that can control the majority of the protocol. However, one of the most considerable matters is the lazy voter problem. Governance participants need to come online to vote in a proposal and sometimes send an on-chain transaction incurring significant transactions costs (e.g., on Ethereum) to participate. For example,

in Maker, a protocol responsible for about USD 15bn of locked collateral, about 20 individual accounts vote on governance proposals, and about 20% of the available MKR tokens are locked in the governance contract to make them eligible for voting[1].

Two directions are being taken to tackle the lazy voter problem.

1. **Delegation** Maker, Uniswap, Polkadot, and other protocols use delegation of votes. The idea is that one can allocate their governance tokens to someone else on behalf of them. Polkadot further formalizes this process by allowing a fixed number of "councilors" to have delegated votes. Councilors can further make decisions like spending from the protocols treasury by only having the council itself vote on it. However, the approach of using delegates or councilors raises legal concerns where delegates might be seen as the official representatives of the protocol and might be liable in some tax jurisdictions for the protocol earnings.

2. **Optimistic Voting** Aragon and other protocols go a different route using optimistic governance. In this approach, by default, changes to a protocol are accepted unless they are voted against. This can take the form of having a turnout bias on votes for protocols, e.g., a much higher number of "nay" votes is required compared to the "aye" votes to stop a proposal. The alternative is challenge protocols. For example, in Aragon, a proposal must be backed by a set number of tokens. If the same number of tokens challenge the proposal, a randomly sampled committee of members from all governance token holders is created. The committee then has to reach a decision on the proposal to either accept or reject it.

2.2.3 Oracles

DeFi protocols can require data that is not present readily on the blockchain they reside on. For example, calculating the collateralization rate of an asset A backing an asset B often requires

[1] See https://forum.makerdao.com/t/improving-governance-participation-rate-at-makerdao/11706 (Accessed 20.12.2021)

price data. This price data can be obtained either from on-chain data, e.g., prices from a DEX, or from off-chain data, e.g., prices from a CEX. Oracles are generally centralized and trusted entities. They feed off-chain data into a smart contract where other smart contracts can query it.

Since oracles are trusted entities, a failure in price reporting can severely impact DeFi protocols. For example, oracles might cause mass liquidations. Protocols like Maker, Acala, and others typically deploy some mitigating measures, including aggregating and medianize oracle sources and adding a time delay between the reported oracle price and when an oracle price "goes live" in the protocol. Maker, e.g., leaves a one-hour time delay to allow agents to react to price changes from reported to "go-live" price.

2.3 DeFi as Cryptoeconomic Protocols

Cryptoeconomic protocols combine cryptographic primitives and economic theory to create applications on a decentralized ledger. They enable participants to interact without the need to trust each other or a third party. Without trust, decentralized lotteries, escrow contracts, multi-party computations, and other protocols can be realized [BP17; BZ18]. We define a cryptoeconomic protocol as follows.

Definition 2.1 (Cryptoeconomic protocol). *A cryptoeconomic protocol implements agreements between agents. The agreements are publicly known speci cations veri ed through cryptographic primitives in a decentralized ledger. An agreement encodes an incentive mechanism that can consist of payments, collateralization, or a combination thereof, seeking to promote honest behavior.*

Cryptoeconomic protocols can be generalized into three states (cf. Fig 2.1).

1. *Committed*: an agent A commits to perform that fulfills a specification defined by an agreement. The commitment requires collateral C or a "soft" commitment to provide payments at a future state.

2. *Executed*: an agent A performs an action . This action either complies with the specification or violates it[2].

3. *Concluded*: if the action complies with the specification, the collateral plus any payments are transferred to the agent. Otherwise, the agent's collateral is destroyed or transferred to another agent that suffered from [3].

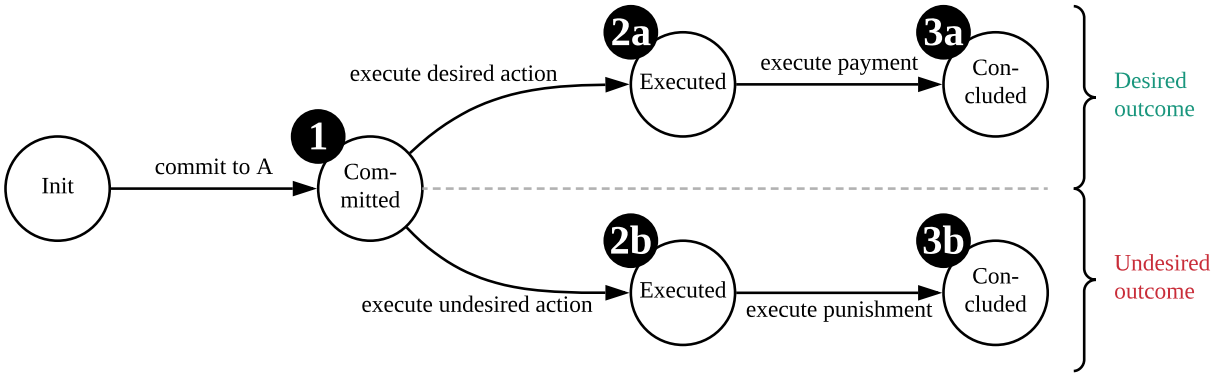

Figure 2.1: A cryptoeconomic protocol consists of three states. (1) an agent commits to the fulfillment of a specification . (2) the agent executes an action that either evaluates to true or false w.r.t. the specification. (3) the protocol is concluded by rewarding or punishing the agent for its action.

In this work, we use notation as summarized in Table A.1.

2.4 Contracts and Agreements

A cryptoeconomic protocol is implemented through smart contracts. A contract includes agreements A in the form of (similar to [SL01]):

$$A = h \ ; p; C i \tag{2.1}$$

[2]protocols are implementing more complex veri cation steps, e.g., multiple rounds with accusations. The principle remains the same at a high level: the protocol implements a method to determine if a speci cation holds.

[3]Once the protocol is concluded, agents are free to redeem their collateral (if it has not been destroyed or sent to another agent).

The specification represents a condition that needs to be fulfilled by an action and evaluates to either true or false, i.e., : ! $f0;1g$. p are the payments made from a receiving agent B to a performing agent A as stated in the agreement. For example, an agent receiving a service would have to make a payment, i.e., $p < 0$. An agent providing a service would receive the payment in return, i.e., $p > 0$.

Further, agents need to provide collateral to *commit* to a protocol. Collateral C is part of the agreement. We note that a single collateral c_i 2 C provided by an agent i can be used for multiple agreements. C is a set of collateral $fC_1;:::;C_ng$ paid by agents to participate in an agreement implemented in a smart contract. The smart contract acts as an escrow of the collateral, encoding rules that affect the collateral. If an agent complies with the specification in an agreement, i.e., ( ) = 1, the collateral is refunded. If an agent performs an action ( ) = 0 (or no action), the smart contract will not refund the collateral to the agent.

2.5 Specifications

The specification describes the *task* that an agent needs to provide and the *proof* that serves as evidence that the task has been provided. There are several approaches to encode the specification. In the BitML calculus [BZ18], a specification consists of (i) a model describing a contract and agent choices symbolically and (ii) a model encoding a sequence of transactions that form a smart contract in Bitcoin computationally. For example, Alice can deliver a digital good to Bob in return for a payment. The contract would then specify that if Bob receives the good, payment to Alice is being made. Specifications are also advantageous when exchanging digital goods with the FairSwap protocol [DEF18]. In FairSwap, Alice sends a digital good to Bob and provides proof of sending by providing a witness (hashes of the transferred data) to a smart contract as proof. The specification in FairSwap is encoded as a boolean circuit that evaluates whether the provided witness (the hash) satisfies the specification. In case the circuit evaluates to true, Alice is paid for delivering the data.

Abstractly expressing the specification gives us the freedom to leave the encoding and imple-

mentation up to the cryptoeconomic protocol. For the remainder of this thesis, we assume the specification can either be fulfilled, i.e., $= 1$ or not, i.e., $= 0$.

Chapter 3

Risk-based Model

In this chapter, we outline a risk-based model for collateral usage in DeFi protocols. In particular, we generalize the model presented in [Kla+20] to reflect collateral-secured protocols, including stablecoins, PLFs, bridges, and derivatives (like PoS assets). Our model builds upon an algorithmic mechanism design putting agents and their types into the model's center. While a range of work describes collateral in DeFi protocols, this chapter aims to develop an overview of the risk over the design space of applying collateral and their trade-offs.

3.1 Agents and Mechanisms

3.1.1 Action Choices

Following Definition 2.1, an agent commits to a protocol and can only then act. The agent has three action choices, each with an associated utility.

1. An agent $A\ 2\ P$ performs a *desired action* $_A\ 2\ \Sigma_d$, where Σ_d is defined as the set of all actions that evaluate to true with respect to the specification of an agreement, i.e. $(\Sigma_d) = 1$.

2. An agent $A \in P$ performs an *undesired action* $a_A \in \Sigma_u$, where Σ_u is the set of all actions that evaluate to false for the specification of an agreement, i.e. $(\Sigma_u) = 0$.

3. An agent $A \in P$ performs *no action*, i.e., $a_A = ;$.

3.1.2 Synchrony Assumption

In an asynchronous system model, no action is not a choice since eventually, the agent acts within the agreement. In a synchronous system model, no action is a valid choice. In practice, we see both the synchronous and asynchronous models. XCLAIM uses a synchronous model where an agent commits to returning funds held in custody within a set period [Zam+19]. Maker, Aave, and Compound use an asynchronous model where eventually the agent returns borrowed funds with interest or defaults on its loan [Mak21a; Aav21; Com21a]. We assume that agents eventually act in the asynchronous setting. Thus, we consider only the desired and undesired action going forward.

3.1.3 Utility Parameters

Each of the two action choices results in a utility for the agent expressed by $u(\ )$. The utility of an action depends on five parameters.

1. Payment p: determines how much agent B pays for action in agreement A and how much performing agent A receives by fulfilling the specification in A.

2. Cost c: captures all costs associated with acting. This includes, for example, transaction costs and costs for executing an action a in A.

3. Collateral C: summarizes the collateral by agents in A.

4. Expected future return E[rC]: describes the opportunity cost for locking collateral C within an agreement that could be used in another protocol to earn interest at rate r[1].

[1] We assume the future return rate is stochastic such that the opportunity cost is not known with certainty. We, therefore, denote the return rate as the expected return rate, E[r].

5. Valuation v: encodes the *private* preference of an agent for an outcome depending on an action .

The private valuation expresses that an agent A prefers performing an action $2 \Sigma_u$. Reflecting our security focus, we focus on the worst case, strategic behavior, where agents derive a valuation from performing an undesired action. Assume A is given the task to perform a computation in a protocol like TrueBit [TR17] and provided a collateral C in the agreement A to perform a computation that fulfills the specification . If A has a positive valuation for v_A that does *not* fulfill the specification, and the provided collateral C is less than v_A, the agent prefers the undesired action.

Note that this private valuation is not limited to pecuniary value: it can include the non-pecuniary value that an agent may derive from performing an undesired action. In practice, v might be hard to approximate. We reflect this by introducing agent types depending on the relation between v and the protocol incentives C and p (cf. Section 3.1.6). We assume that whatever the relative share of pecuniary as opposed to non-pecuniary value in v_A is, it is possible to approximate v_A overall by finding an amount of money that would leave an agent indifferent between receiving v_A or the money. Also, we account for an undesired agent type in which the valuation to act maliciously is much larger and can be infinite.

For example, consider a protocol that facilitates exchange between currencies. An agent in such a protocol might prefer holding Bitcoin over Ether regardless of the exchange rate. Therefore, such an agent might decide to renege on performing a Bitcoin to Ether trade if the agent derives more value v_A from the outcome in which they continue to hold Bitcoin. In general, if an agent's private value of an outcome v_A becomes greater than their collateral C, the agent might decide to perform an undesired action.

3.1.4 Performing Agent Utilities

The following are the utilities an agent $A \; 2 \; P$ *performing* an action. Due to the worst-case assumption, we omit the valuation v_A for the desired action $2 \Sigma_d$ as we assume agents do not

have an intrinsic or external motivation to comply with the specifications of agreements in .

$$u_A(\ _A) = \begin{cases} p \quad c_A \quad \mathrm{E}[rC_A]; & \text{if } \ _A \in \Sigma_d \\ v_A \quad C_A \quad c_A \quad \mathrm{E}[rC_A]; & \text{if } \ _A \in \Sigma_u \end{cases} \tag{3.1}$$

The agent tries to maximize its utility by choosing an appropriate action. We note three observations from equation (3.1).

First, a committed agent receives negative utility for performing no action. Hence, the agents need to expect positive utility from performing any action. The choice to enter any cryptoeconomic protocol depends on the expected loss of interest on the collateral and the possible future utility.

Second, performing an undesired action becomes a rational choice if $\ _{A;i} \in \Sigma_d; \ _{A;j} \in \Sigma_u : u(\ _{A;i}) < u(\ _{A;j})$. In a protocol agents can enter or leave the game at any time, and their past actions are not considered. Hence, the game consists of single, disconnected rounds, i.e., it is a *single-shot game*. We also note that only if the value v_A of the agent is greater than the collateral provided, the utility of performing an undesired action can be positive.

If the value of the task is higher than the collateral, then cheating is a rational choice. As later pointed out, in XCLAIM, when a Vault should send 2 BTC to a user, but the collateral provided in ETH is lower than the 2 BTC, a rational agent will take the BTC instead of transferring it to the user. Another example can be made with TrueBit: if the value of providing a false solution to a computation problem is higher than the collateral, then a rational agent should provide a false solution. This might be the case when a competitor of the party requesting a computation wants to sabotage the party requesting the computation in the first place. One could quantify the worth of risking to lose the collateral provided to get a false solution submitted.

Third, executing a desired action depends on the publicly known payout p as well as the agent-specific costs c_A caused by performing the action and the expected loss of interest $\mathrm{E}[rC_A]$. An agent will choose this action if it yields higher utility than the other actions.

This is a deterministic model where we assume that if an agent "cheats", it loses its collateral. However, one could extend the model to include a probability of the agent being caught if the specification function includes a prior.

3.1.5 Receiving Agent Utilities

An agent $B \in P$ *receiving* a specific action has a different utility from the performing agent A. This agent does not influence the performance of agent A if we assume that the payout cannot be changed after commitments are made. We assume that the agent who receives an action is reimbursed with the performing agent's collateral. Depending on A's actions, the utilities are calculated as in (3.2). If the performing agent executes the desired action, the receiving agent utility consists of the value of the agreement minus the payment and the costs to interact with the contract. If the performing agents perform an undesired action, the receiving agents are reimbursed; however, the value and costs need to be deducted from the utility. If no action is performed, the receiving agent may incur costs.

$$u_B(\ _A) = \begin{cases} v_B \quad p \quad c_B; & \text{if } \ _A \in \Sigma_d \\ C_A \quad v_B \quad c_B; & \text{if } \ _A \in \Sigma_u \\ \cdot \quad c_B; & \text{if } \ _A = ; \end{cases} \tag{3.2}$$

3.1.6 Agent Types and Adversaries

The performing agent A influences the utility for both performing and receiving agents. Most importantly, the security of a cryptoeconomic protocol depends on a core assumption: adversaries are economically rational and computationally bounded. Considering Eq. (3.1), an agent A chooses to perform the desired action if the utility of the desired action is higher than the utility of the undesired action:

$$p \quad c_A \quad E[rC_A] > v_A \quad C_A \quad c_A \quad E[rC_A] \tag{3.3}$$

Inspired by the BAR model [Aiy+05], this notation allows us to express three types of agents, from which we can deduct the adversaries.

Definition 3.1 (Performing agent types)**.** *We categorize performing agents into the following three types.*

1. *Type $_d$: the honest type will always perform the desired action as its utility resulting from the desired action is larger than the valuation of an undesired action, i.e., v_A $C_A + p$.*
2. *Type $_u$: the malicious type will always perform an undesired action as v_A $C_A + p$. By de nition, the collateral is never large enough to prevent an undesired action by A.*
3. *Type $_r$: the rational type is undecided in the single-shot game setting which decision to take as v $C_A + p$. Only if $v < C_A + p$ is true will 3.3 hold true.*

By this definition, a cryptoeconomic protocol can ensure that economically rational adversaries, i.e., type $_r$, and honest participants perform desired actions. As expressed by the type $_u$, malicious adversaries cannot be economically motivated to perform the desired action. These agent types will perform the undesired action as their valuation for the undesired outcome (from the perspective of the protocol designer) is much greater than the economic damage due to the loss of their collateral.

3.1.7 Security

Following Definition 3.1 we define the security of cryptoeconomic protocols as:

Definition 3.2 (Security under rational agents)**.** *Assuming agent type $_r$ with a private valuation V, a cryptoeconomic protocol implementing a speci cation is secure if $_r$'s utility u for ful lling the speci cation is higher than its utility for violating the speci cation , i.e., $u(\Sigma_d) > u(\Sigma_u)$.*

These agent types are comparable to the BAR model introduced in [Aiy+05]. The $_d$ type is similar to the altruistic agent, $_u$ to the Byzantine, and $_r$ to the rational agent. Further, we

note that by Definition 2.1 and its single-shot game nature, agents behave individually rational and truthful once they are committed in a protocol. Since an agent cannot gain additional utility from hiding its preference, the agent truthfully reveals its preference by performing the utility-maximizing action.

Formally, we note that in the single-shot setting, agents' reported type $\hat{}$ is equivalent to its actual type .

3.1.8 Mechanism Utilities

Social welfare describes the sum of all agent utilities in a mechanism [Woo09]. If we combine equations (3.1) and (3.2), we can calculate the sum of utilities for performing and receiving agents $fA; Bg\ 2\ P$ by considering the actions of *performing* agents A. Assuming that every A is matched with at least one B, we split P into two groups corresponding to whether the performing agents perform desirable or undesirable actions. P comprises a group Λ_D, for the pairs of agents containing performing agents who perform desirable actions and Λ_U for undesirable actions. Thus summing for pair of agents $x\ 2\ P$, where there are X pairs, such that $2X = jPj$:

$$u(\ _a) = \begin{cases} \sum_{x=1}^{x} (v_{x;b} \quad c_{x;b} \quad c_{x;a} \quad e[rc_{x;a}]); & \text{if } a\ 2\ _d \\ \sum_{x=1}^{x} (v_{x;a} \quad c_{x;a} \quad v_{x;b} \quad c_{x;b}); & \text{if } a\ 2\ _u \\ \sum_{x=1}^{x} (\ e[rc_{x;a}] \quad c_{x;b}); & \text{if } a\ 2\ _; \end{cases} \tag{3.4}$$

Hence, if we take all actions within a specific time, we can specify the utility for a protocol . The goal for a protocol is to find a function that maximizes the utility for all agents given the distribution of agent types. Notably, agents acting individually rationally and truthfully after commitment does not imply that the mechanism as a whole maximizes social welfare.

3.2 Risk Dimensions

Fundamentally, collateralized protocols are comparable to Collateralized Debt Obligations (CDOs) and Contracts For Difference (CFDs). A CDO is backed by a pool of assets classified into tranches. In its applied form in the financial world, CDOs have multiple tranches sorted according to their risk. Higher-risk assets receive a higher interest (yield) but are liquidated first if the loan defaults (= junior tranche). Lower risk assets generate a lower yield than the junior tranche but are only liquidated if the junior tranche assets are insufficient to cover the outstanding debt (= senior tranche).

In the nascent DeFi ecosystem, most protocols use single asset collateral constructions, e.g., [Mak21a; Com21a; Aav21] or treat pooled assets as a single tranche, e.g., Uniswap LP tokens in Maker. While this simplifies the actual implementation in the current landscape, the below model accounts for an extension of pooled collateral assets.

The model outlines seven dimensions of risk.

1. **Primary Value**: Effect of the type of collateral used on the underlying risk.
2. **Data Feed**: Implications of external (trusted) data sources.
3. **Governance**: The process of introducing changes to the protocol.
4. **Base Layer**: The impacts of the incentives from the base layer, i.e., Miner Extractable Value (MEV) [Dai+20].
5. **Technical Security**: Risks that express the discrepancy between the (formal) specification of the protocol and its actual implementation.
6. **Censorship and Counterparty**: The dependency of a protocol on third parties and their ability to affect the security or liveness of the protocol.
7. **Market Conditions**: This includes liquidity of markets, arbitrage opportunities, and conditions for liquidations.

The design space for collateralized protocols is subject to these risk dimensions. However, they are often "hidden" behind the nominal value N of the collateral C. If we take the *worst-case assumption* for the risks above, any risk will already be priced into C.

Though there is a catch: The economic security of specific constructions does not hold considering economically rational agents with strategic behavior (let alone malicious agents). We outline these cases below. For Ch. 4, 5, and 6 we assume that the cryptoeconomic protocol manages risk under worst-case assumptions. In Ch. 7, we lift this assumption and show how the selected risks can be exploited to attack collateralized protocols.

3.2.1 Primary Value

The primary value describes the "source" of value for the asset. In the explicit sense, this can be the price that an asset is trading for, i.e., for exogenous and endogenous assets. In the implicit sense, it can be the value or trust that agents put into the system, i.e., for implicit collateral. We differentiate between three different types of primary value.

- **Exogenous**: the asset used for the collateral can be used outside the system; it serves as collateral. Typically, only a fraction of the asset is used as collateral in one particular system. An example of this is ETH used in the Dai stablecoin system [Mak21a] or ETH used in XCLAIM [Zam+19].
- **Endogenous**: the asset is used purely in the context of the system as a means of collateral. Typically, the creator of the systems has also created the asset and set up the initial distribution mechanism. Examples include SNX used in Synthetix to create synthetic assets tied to the USD, BTC, or shares in the stock market [Syn20].
- **Implicit**: rather than using an explicit asset, protocols are able to use their reputation itself as collateral. For example, this is similar to how Tether issues USDT without the ability for everyone to verify that the USD-backing is sufficient for minting new USDT [Kla+20].

The type of collateral plays a significant role in the way the price of the collateral asset should be modeled. Collateral can thereby be viewed on a scale from exogenous to endogenous. Collateral assets that are on the exogenous side have advantageous properties in crisis: Agents wishing to liquidate an exogenous asset due to a failure in a protocol will be able to do so without having to suffer a discount on the collateral price. Endogenous collateral has advantages when the protocol performs well. If the demand for the endogenous asset is high, its price may appreciate leading to the system having higher liquidity capacity.

Implicit collateral can be modeled similarly to endogenous collateral but is not explicitly tokenized. NuBits and other stablecoins are examples of such systems where miners control the supply of a stablecoin without explicitly being collateralized [Kla+20].

Typically, exogenous assets show lower price volatility whereas endogenous assets show higher volatility, i.e., ${}^2N_{\text{exogenous}} < {}^2N_{\text{endogenous}}$. In practice, the collateralization rate f for exogenous assets is higher than for endogenous assets, for example, ETH, USDC (exogenous), and SNX (endogenous).

3.2.2 Data Feed

Data feeds are used to determine the value of a collateral asset when it is priced outside of the protocol. Concretely, DeFi protocols use oracles to import the price of assets from centralized exchanges. Several issues with using oracles exist, and finding ways to decentralize oracles remains a challenging open research problem [Kla+20]. For widely adopted oracle providers, incentives might be skewed. Long-term, the profitability from (slightly) misreporting exchange rates might be higher than providing data honestly. Using decentralized exchanges as an oracle source is non-trivial as well as short-term price movements (especially considering lash loans [Wer+21]) can skew exchange rates.

Thus, the value N of C might be incorrectly reported. Misreporting has an impact on the collateralization rate f such that agents might either not be liquidated if they should or liquidated when they should not.

3.2.3 Governance

The governance function has wide-ranging control in a range of collateralized DeFi protocols, including Maker, Aave, and Compound. It can add new assets as collateral, has full access to the issuance of debt, access to collateral assets, and set new parameters for collateral stability, i.e., f.

As such, governance is one of the critical areas of focus for DeFi risk. Analogous to how miners may extract value from protocols, governance participants can be incentivized to extract value from protocols, termed Governance Extractable Value [Kla+20]. Fundamentally, the question is whether profits from long-term participation, e.g., via stake-to-vote rewards and profits from a DeFi protocol, i.e., in-protocol revenue, are sufficient for governance holders not to attack the system. DeFi has seen several such "rug-pulls". When launching a new governance token, the issuer of the token can sometimes extract all value from the system by selling the tokens to investors and subsequently abandoning the project. Such perverse incentives form in poorly designed governance systems.

However, this might not just be a concern with young projects. If the long-term expected value of a governance token suffers a crisis of confidence, governance token holders acting rationally might want to extract locked liquidity from the protocol they (partially) control. Hence, DeFi protocols might rely on benevolent actors, i.e., protocol foundations that hold token reserves to prevent such attacks. Less severe risks include governance miscalculating the required collateral ratio f for a collateral asset or allowing debt limits beyond the liquidity of assets; see Section 3.3 for a more in-depth discussion.

3.2.4 Base Layer

Due to lack of finality, miners can re-order or censor transactions in upcoming and *past* blocks [GKL15]. We can relax this assumption to allow any agent to re-order and censor transactions by considering bribing [MHM18; Jud+19]. This allows agents to order transactions in the way most optimal for them in terms of collateral. Thereby, agents can influence

collateral through impacting governance, data feeds, censoring transactions, or changing the market conditions. Commonly, this is referred to as MEV [Dai+20]. The "flashbots" project has professionalized miner value extraction and cultivated techniques to prevent these forms of miner interventions.

3.2.5 Technical Security

Technical security risk is concerned chiefly with the discrepancy of DeFi protocols *intentions* and its actual implementation. Where present, a specification of the protocol might be used to reason about its intentions. However, most specifications leave edge cases answered in its concrete implementation. The infamous "code-is-law" argument often discussed in the crypto scene emphasizes the dichotomy across the community.

Technical security has an impact on collateral in two ways: First, a severe security bug might directly lead to the loss of all collateral C in a protocol. If users can withdraw funds without authorization, the protocol will fail. The cause of such a bug can be manifold. The most famous is likely The DAO hack with its reentrancy bug, but several others include integer overflows, misuse of privileged accounts, and simple bugs [Atz+17].

Second, technical security risk might come in the form of counterparty risk. If a DeFi protocol utilizes a collateral asset C minted by another protocol, it relies on its technical security. A critical security bug in a protocol like Dai would have severe consequences for the entire DeFi ecosystem employed in various other protocols. While DeFi's composability allows it to innovate and integrate fast, it also comes with severe composition risk.

3.2.6 Censorship and Counterparty

Centralized assets can censor DeFi protocols. For example, if Circle decided to ban Maker from using USDC or BitGo decided to ban wBTC from being used in Maker, more than USD 2.7bn of collateral in Maker would become inaccessible[2]. Worse, Maker uses Dai as part of its price

[2] From https://daistats.com/#/collateral (Accessed 4.1.2022)

stabilization module. If the price of Dai exceeds USD 1, users can mint 1 Dai with 1 USDC to execute arbitrage trades to stabilize the price back to USD 1.

3.2.7 Market Conditions

In case of liquidations, an agent might want to exchange C for another asset. For example, a custodian might not release an asset from custody in bridges, and the agent can slash the custodian. The agent, wishing to keep the initially bridged asset, seeks to exchange C.

Due to market conditions, including overall available liquidity of C and slippage, liquidating C might be impacted. Specifically, in extreme cases such as described in [Gud+20b], the cryptoeconomic protocol as a whole might become under-collateralized since agents are not able to deleverage the system fast enough due to limited liquidity and slippage.

3.3 Risk Equivalence

DeFi protocols are not limited to using a single collateral asset. In practice, having more than one collateral asset *should* diversify risk and increase the overall liquidity of the system. Increasing liquidity in a collateralized system often goes hand in hand with increasing revenue for the protocol.

Technically, collateral assets can be combined into a single asset to reflect a CDO structure with different risk levels. As of December 2021, prevalent protocols including Aave, Compound, and Maker have isolated collateral assets. In such a setup, each asset should be *risk equivalent*, i.e., two collateral assets C_x and C_y should have equivalent risk profiles.

To achieve this risk equivalence, protocols including [Aav21; Com21a; Mak21a; Zam+19] use three basic parameters.

1. **Debt ceiling**: A debt ceiling determines how much collateral of a particular asset can be deposited.

2. **Liquidation threshold**: The liquidation threshold represents the collateral rate f_{liq} at which the collateral of an agent is liquidated due to external events (e.g., price volatility).

3. **Secure threshold**: The secure threshold represents the collateral rate f_{sec} until an agent can borrow assets against the provided collateral.

Determining Risk Parameters Setting these three parameters is subject to governance. Determining the parameters has ample impact on the economic security of a DeFi protocol. In [Gud+20b], collateral asset prices are modeled as a stochastic process to analyze tail-risks in ETH collateral in Maker. An alternative approach is to use Value-at-Risk (VaR), cf. [DP97], to determine the potential financial loss. Both basic methods can be extended to account for additional risk parameters: First, the pricing of the collateral can be exogenously or endogenously modeled. Next, it is possible to account for market conditions such as changing slippage deepening on the size of trades and bid-ask spreads in VaR models. In practice, analysis is accomplished under certain assumptions, often choosing pessimistic market conditions to minimize system-wide liquidation risk, cf. [Fu+21].

Related to the determination of risks parameters, [Gud+20c] examines the interest rate competition across PLFs. The interest rate within a protocol is equivalent to the payments p in our cryptoeconomic protocol, whereas the interest an agent can earn in another lending protocol is represented by r in our model. However, [Gud+20c] explores the efficiency of interest rates (impacting p) within a PLF and the interrelation across protocols (impacting r).

Chapter 4

PROMISE

Since their creation, blockchains' most significant property has been facilitating trustless exchange between entities with weak identities [Böh19]. Yet the trustless nature of the systems means that parties *may* transact without trusting each other and that they *should not trust* each other. This creates a design challenge for interactions that would typically involve such trust. In this chapter, we focus on blockchain protocols which, at least in part, encode trust by monetary collateral. Here, collateral is value escrowed by a service provider, Alice, to guarantee the user, Bob, that Bob cannot lose funds regardless of Alice's behavior. In particular, payment, cross-chain, and generic computation protocols can be designed such that Bob is guaranteed to receive from Alice at least the amount of funds that are at risk if she misbehaves. Protocols involving collateral include cross-chain communication [Zam+19], scalable off-chain payments [KGF19], state channels [DFH18], watchtowers [McC+18; AKW19; Ava+19], and outsourcing of computation and verification games [TR17]

Problem. Relying on collateral as trust is itself associated with a set of challenges. Collateralization requires a substantial amount of funds upon protocol initialization, limiting the set of participants to a selected few. Leaving participation to a small set of agents can lead to phenomena like the "rich are getting richer" through wealth compounding [Fan+19]. While it is impossible to grant less wealthy agents proportionally higher rewards due to Sybil identi-

ties [Dou02], we can lower the entry barrier for agents to join a protocol. Finally, locked funds result in opportunity costs for the agent who could use their collateral for participating in other protocols [Har+19].

Promise. We present Promise, a simple but effective mechanism to lower entry barriers for intermediaries in protocols relying on collateral for secure operation. Promise is a subscription mechanism: Instead of locking up a significant amount of funds as collateral, Promise allows intermediaries to stake future payments (e.g., service fees) with the *promise* the payments will be disbursed upon the correct provision of the service. Similar to online platforms, users can choose to *subscribe* to a service and pay fees upfront — for some pre-agreed service period (the "subscription period"). However, instead of transferring these payments directly to the intermediary, users lock pre-paid fees in an escrow smart contract, preventing theft by either party. The intermediary needs to provide the service correctly for the entire period set by the user. The benefit of this scheme is two-fold: (i) the intermediary is incentivized to act honestly while enjoying lower initial collateral, and (ii) the user can reduce transaction cost and pay if the service was provided honestly over his defined period. As long as (i) the initial collateral is higher than the potential gain from not delivering the service, (ii) the expected future revenue from correct operation exceeds potential gains by the intermediary, (iii) users have the option to leave the protocol, (iv) and misbehavior can be proved to the smart contract, Promise incentivizes correct behavior.

Implementation. We show that implementing Promise in an Ethereum Solidity smart contract only adds USD 0.03 to setup Promise between a user and a service provider, USD 0.01 to provide the deposit for the service provider, and USD 0.01 to supply the prepayment. During the subscription period, each service delivery adds a cost of USD 0.01. Finally, the deposit and accumulated payments withdrawal add USD 0.01 for the service provider.

Outline. We adopt the system model from Ch. 3 and make necessary extensions in Section 4.1. First, we start with a description of Promise in Section 4.2. Next, we discuss the

security of Promise and argue in which cases Promise can benefit users and intermediaries in Section 4.4. We discuss related work in Section 4.6 and conclude in Section 4.7. We discuss how Promise can be applied to XCLAIM in Ch. 6.

4.1 System Model

In Promise, a user Bob engages a service provider Alice to fulfill a task valued at V_B on his behalf. Bob pays Alice p each period t for performing the task. Given the absence of strong identities, the total value of the task to Bob (V_B) needs to be fully collateralized via a collateral C, such that $C \quad V_B$. For example, suppose a particular task involves Alice offering a service and Bob having a $100 exposure to Alice — in the form of counter-party risk. In that case, Alice will need to post at least $100 as collateral to *insure* the exposure, such that Bob does not stand to lose funds if Alice behaves maliciously.

Promise is a mechanism to reduce initial collateral locking. However, Promise is not a stand-alone protocol but rather serves as a "plug-in" to existing cryptoeconomic protocols. Given a generic cryptoeconomic protocol that satisfies the assumptions of Section 4.1.2, we can apply Promise and write the protocol as $_P$. We note that the agreement A is given by the generic protocol . We assume that Alice and Bob have entered into agreement A and have agreed on the specification , payments p, and the deposit D.

4.1.1 Roles

We define the following roles.

- **Performing Agent**, the Intermediary: Alice is economically rational and entrusted with executing a task. She provides collateral C into the escrow before performing the task and receives m payments p upon successful completion. Alice prefers to adhere to the specification if her utility for doing so is greater than other action choices.

- **Receiving Agent**, the User: Bob represents the user requesting execution of a task by Alice. A user provides payments $\{p_1, \ldots, p_m\}$ into the escrow. The user is assumed to be honest and correctly reports the behavior of Alice.

- **Escrow**: The escrow is a smart contract responsible for holding deposits by Alice and payments by Bob.

- **Verifier**: The verifier detects malicious behavior of Alice. In practice, this role is fulfilled by a smart contract, a dedicated third party, or the user.

4.1.2 Assumptions

The verifier in the system detects faults by Alice and proves that Alice was at fault. This means that the specification of the protocol has some "proof". For example, this could be the hash input of a boolean circuit as in FairSwap, a transaction inclusion proof as required by XCLAIM, or fraud-proofs [ASB18].

We further assume that the protocol utilizing Promise implements payments and deposits through a ledger functionality (e.g., as described in [GKL15]). Also, there is a one-to-one mapping between the collateral and a user, such that the collateral of an intermediary is not split between multiple users. Agents in the system can be identified with their public/private key pair. Finally, time is denoted with t.

We adopt the utility definitions for Alice from Eq. 3.1 and for Bob from Eq. 3.2. Moreover, we adopt the definition of security from Def. 3.2.

4.2 System

In Promise we allow Bob to provide multiple payments in advance and delay the receipt of the payments by Alice. In turn, Alice can reduce the initially provided collateral from C to C_l such that $C_l < C$. At $t = 0$, Bob is able to lock m payments $\{p_1, \ldots, p_m\}$ in escrow and

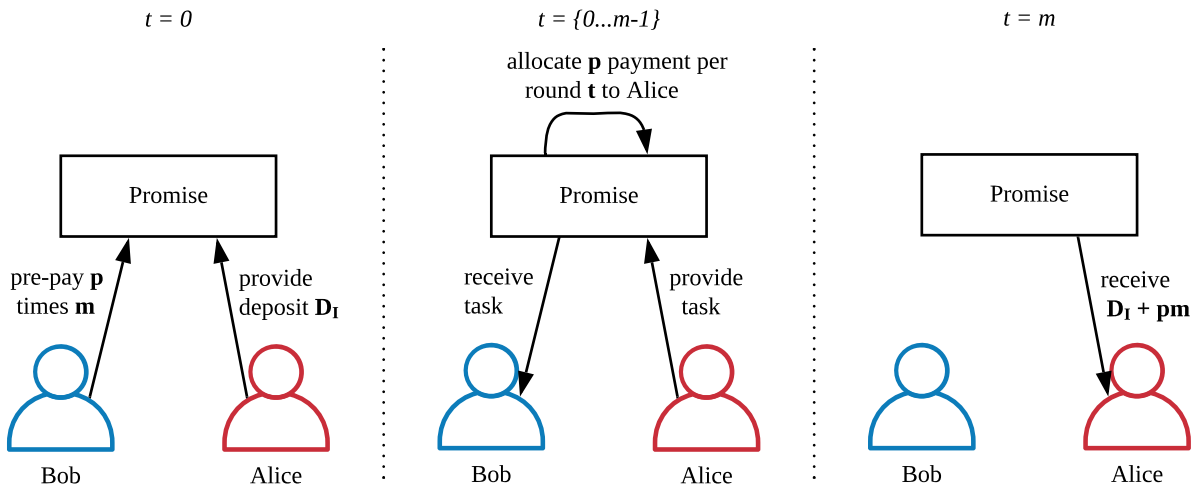

Figure 4.1: Promise allows intermediaries (Alice) to lock less initial collateral C_I and use payments p_i provided by users (Bob) as additional collateral. The initial collateral and payments are locked until time m determined by Bob. Only when Alice fulfills the specification until $t = m$ can Alice withdraw her initial collateral C_I and the total payments pm.

determine a period after which Alice can receive the payments. When $t <$, Alice continues to accumulate collateral as time passes by keeping the cumulative total of her payments p_i in escrow. We provide an intuition in Fig. 4.1. Promise has the following advantages for Alice and Bob.

Alice: the barrier to entry as an intermediary is lowered, as Alice only needs to provide a lower initial collateral C_I as opposed to C in the first period. Further, instead of expecting a single next payment p, Alice has, in expectation, p m payments lined up as part of Bob's subscription to her services.

Bob: the aggregation of multiple payments allows Bob to reduce transaction costs and guarantees Bob that he only pays Alice if she fulfills all tasks for the given period m.

4.3 Protocol

The Promise protocol consists of three steps. We denote the service provider as A, the user as B, and the smart contract implementing Promise as P. We assume that A and B have

agreed on the total payment and the period over which the payment is to-be-paid in advance.

1. At $t = 0$: B locks m payments in P. A locks the initial collateral C_I in P.
2. At $t = \{1; \dots; m\}$: A provides m times the agreed task to B. P allocates one payment p to A if (i) A provides proof to P that fulfills the specification , or (ii) B does not provide a fraud proof that A did not provide the task within a determined time [ASB18].
3. At $t = m$: A withdraws $p(m + 1)$ and C_I from P.

To argue about the security of Promise, we introduce two concepts: (i) sequential games with discounting and (ii) a likelihood of users exiting the system upon the service provider not adhering to the specification of the agreement.

4.3.1 Sequential Games and Discounting

Introducing Promise transforms the single-shot game of the agreement between Alice and Bob into a sequential-round game. Instead of Alice and Bob treating each game in isolation, they need to consider the utilities for the sequence m of the game.

Without Promise, at each round t, Alice decides if she prefers to fulfill the specification based on the utilities denoted by Eq. (3.1) and (3.2).

With Promise, Alice needs to consider that if she does not adhere to in any round t, she does not receive any of the payments. For example, if Alice provided the services according to for n rounds but fails to do so in a round $t < m$, she does not receive pn payments but instead loses C_I and receives 0 payments.

Hence, Alice's decision needs to account for all p m payments. Furthermore, payments are made in the future. A promised payment in the future is less valuable to Alice today, which we denoted with the parameter . $0 <$ < 1 denotes the discount factor of an agent's valuation of future utility. We argue that an agent can spend received payments elsewhere or potentially

invest the payment for a profit. Hence, the service provider faces an opportunity cost for delayed payments. If she follows the same course of action over every round, the pay-offs to Alice are as follows.

$$u_A(t) = \begin{cases} \sum_{t=0}^{m-1} \left(\frac{1}{1+r}\right)^t (p - (t+1)\mathrm{E}[r]p - \mathrm{E}[r]C_I); & \text{if } = 1 \\ \sum_{t=0}^{m-1} \left(\frac{1}{1+r}\right)^t (V_A - \mathrm{E}[r]C_I - C_I); & \text{if } = 0 \end{cases} \tag{4.1}$$

Bob receives the following pay-off, depending on Alice's behavior.

$$u_B(t) = \begin{cases} \sum_{t=0}^{m-1} \left(\frac{1}{1+r}\right)^t (V_B - p - c - (m - t)\mathrm{E}[r]p); & \text{if } = 1 \\ \sum_{t=0}^{m-1} \left(\frac{1}{1+r}\right)^t (C_I - V_B - c); & \text{if } = 0 \end{cases} \tag{4.2}$$

4.3.2 Termination Probability

Lowering Alice's initial collateral to C_I increases the risk of Alice not fulfilling the specification of the agreement. Specifically, Alice's collateral is the lowest in the first round since she has not provided the service yet and has not added any payment into her collateral pool.

We argue that Bob exits a protocol after Alice not adhering to , encoded in the function $(n) \to [0;1]$ describing the likelihood that Bob remains in the protocol. The variable n describes the number of times Bob tolerates Alice not delivering the service. Each time Bob does not receive V_B due to Alice not providing the service as agreed, the lower the probability that Bob continues to participate. Each user can have its own (n) function. For example, users might choose to never participate with a service provider again, i.e., $(1) = 0$, and others might tolerate a higher number of incidents. This changes Alice's pay-off for the protocol as follows.

$$u_A(t) = \begin{cases} \sum_{t=0}^{m-1} \left(\frac{1}{1+r}\right)^t (p - (t+1)\mathrm{E}[r]p - \mathrm{E}[r]C_I); & \text{if } = 1 \\ \sum_{t=0}^{m-1} \left(\frac{1}{1+r}\right)^t ((n)V_A - \mathrm{E}[r]C_I - C_I); & \text{if } = 0 \end{cases} \tag{4.3}$$

As decreases, the payoff to Alice can become negative for not fulfilling the specification if $(n)V_I < \mathrm{E}[r]C_I \quad D$. For Alice, we increase the motivation to follow the specification by (i) providing a sum of payments pm that Bob locks in the protocol, and (ii) the fear of Bob leaving the protocol altogether if she does not provide the service the entire period. As Bob chooses m, he directly influences Alice's expected pay-off. By setting large m and quitting the protocol upon Alice's misbehavior, he can motivate "rational Alice" to act in his interest.

4.4 Analysis

The core argument of Promise is that service providers can reduce their initial collateral by locking multiple payments. Specifically, introducing Promise to a protocol does not incentivize an intermediary A to not adhere to the specification. This means that $_{\mathrm{P}}$ and are equivalent in terms of security considering an economically rational intermediary A under Def. 3.2. More formally, we state that:

Theorem 4.1 (Security equivalence)**.** *Given a protocol that has a veri able speci cation and an economically rational service provider A that provides more initial collateral C_I than an incentive V_A to violate the speci cation, introducing* Promise *is secure if A does not gain additional utility by not ful lling the speci cation considering A participates in at least two rounds in .*

4.4.1 Action Choices

Alice's utility for choosing a specific course of action, i.e., fulfilling vs. not fulfilling the specification, is given by Eq. (4.3). However, this makes an implicit assumption: Alice considers the entire period m as a basis for her decision. We depict her added utilities for an example of two rounds in Fig. 4.2.

Showing that Thm. 4.1 holds requires considering that Alice might not participate for m rounds. Specifically, Alice might still consider the agreement a single-shot game with a precisely one-

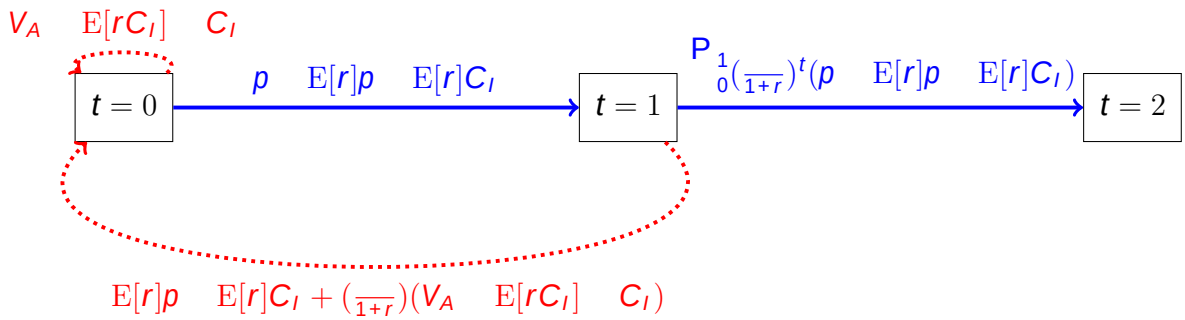

Figure 4.2: Depicting the sum of utilities depending on different action choices made by Alice. At $t = 0$ Alice can choose between fulfilling the specification and receive the utility depicted in blue or choose the opposite and receive the utility depicted in red. If Alice at any point prefers to violate the specification, the game restarts and the action choices are essentially back to the $t = 0$ state. Furthermore, at $t = 1$, Alice will already have committed to adhering to the specification. In case Alice decides to misbehave at this point, she will not receive p that she was allocated when she transitioned to t_1. However, if she decides to continue to fulfill the specification, she will be rewarded with an additional payment allocation. This game continues until $t = m$. Payoffs corresponding to no action, $_p\ 2\ ;$, are excluded as doing nothing would yield negative utility for an agent and therefore a rational agent would not choose this option.

round decision horizon. Following this, we can use $m = 1$ and Eq. (4.1) to conclude that Alice prefers to fulfill the specification if:

$$p \quad \text{E}[r]p > \quad (n)V_A \quad C_I \tag{4.4}$$

Collateral Condition Comparing Alice's decision without Promise in Eq. (B.11) and her decision with Promise in Eq. (4.4) without considering clearly shows that if Alice only evaluates a single round, introducing Promise *weakens* the security of as Alice is not paid immediately and the initial collateral is reduced. Moreover, even if Alice considers multiple rounds, if Eq. (4.4) does not hold, Alice *has a higher utility not to ful l the speci cation* if Bob is willing to continue to enter into agreements with her. Even worse, if Bob decides to continue using the protocol and black-lists Alice for her violating , Bob still might end up with a Sybil identity of Alice. Hence, for Promise to not weaken security, the initial collateral needs to be set above $(n)V_A \quad p + \text{E}[r]p$.

In practice, this is achieved by over-collateralization or state-reversal. Over-collateralization is used in XCLAIM, where a vault has to provide 200% collateral of the value it stands to

obtain by violating the specification. In NOCUST and, generally, payment channel networks, participants cannot steal funds since an older state can be committed that reverses the stealing of funds.

4.4.2 Security Proof

We are now proving Thm. 4.1.

Proof. Under the assumption that A is economically rational, wants to participate in at least two rounds, e.g., $t > 0$, and Eq. (4.4) holds, we prove that Π is secure. If Eq. (4.4) holds, Alice should fulfill the specification in the first round. We now show that if this holds, Alice should continue with the same course of actions in any subsequent round $t \geq m$. If if at any point $k \geq m$, Alice decides to stop adhering to the specification she will receive the following pay-off:

$$u_A(t;k) = \sum_{t=0}^{k-1} \left(\frac{}{1+r}\right)^t \left(\quad (t+1)\mathrm{E}[r]p \quad \mathrm{E}[r]C_I \right) + \left(\frac{}{1+r}\right)^t \left(\quad (n)V_A \quad \mathrm{E}[r]C_I \quad C_I \right) \tag{4.5}$$

Alice has locked her collateral C_I for multiple rounds as well as the payments she should have received. Due to her actions, she gains V_A, but for each round, she has locked more payments and collateral, the higher her cost to change her choices of action w.r.t. the specification. Moreover, if Alice plans to participate in multiple rounds, she stands to decrease her probability of providing services for other users depending on . Hence, assuming Alice is economically rational, Alice has the highest pay-off when fulfilling the specification until she has completed the service within the entire subscription period m, i.e., incentive compatible. □

4.4.3 Cost Reduction for Service Providers

Service providers can reduce their initial collateral to the lower bound under the condition of Eq. (4.4). From this equation, we can determine the reduction if Alice only considers a single game round. We express C_I as C where is the reduction of the initial collateral and solve for This yields:

$$= p \quad \mathrm{E}[r]p + C \qquad (n)V_A \tag{4.6}$$

However, if we consider that Alice wants to participate in m rounds, we can express this based on Eq. (4.3) and solving for . However, we argue that under the assumption of Eq. (4.4), Alice's decision is essentially between participating in a single round and not fulfilling the specification or participating in multiple games over the pre-agreed period m while adhering to the specification. To calculate Alice's decision bound, we assume from Eq. (4.6) the first reduction is set to the lowest possible value. This means that the term $(n)V_A \quad p+\mathrm{E}[r]p \quad C_l = 0$, i.e., at the decision bound, Alice is undecided if she should fulfill the specification since the utilities for both choices are equal. Thus, can be expressed as:

$$= \sum_{t=0}^{m-1} \left(\frac{}{1+r}\right)^t (p \quad (t+1)\mathrm{E}[r]p \quad \mathrm{E}[r]C) \tag{4.7}$$

In Section ,6.1 we provide an example of how collateral is lowered given a set of parameters. Note that to calculate the collateral reduction, both the service provider and the user only need to know the prior collateral requirement C as defined by . For example, in XCLAIM, this is 200%.

4.4.4 Cost Reduction for Users

Assume that Alice behaves honestly. If a user pays every round t for the service provided by Alice, then his pay-off per round is $V_B \quad p \quad c$ as described in Eq. (3.2). However, locking multiple payments incurs an opportunity cost. This cost is lowered at every time step as the payments are assigned to the intermediary, as expressed in Eq. (4.2).

Bob starts with an opportunity cost of $\mathrm{E}[r]pm$ at $t = 0$. The opportunity cost is reduced to $\mathrm{E}[r]p(m \quad 1)$ at $t = 1$ as the payment is allocated to Alice. Generalizing this for t rounds, leaves us with $\mathrm{E}[r]p(m \quad t)$ at every time step t from today's perspective.

The user locks future payments when the sum of the transaction costs c for m payments is

greater than the opportunity cost for locking additional payments plus the single transaction cost for making the prepayments. Hence, the boundary for a user to choose Promise as individually rational choice maximizing his pay-off is given by:

$$\sum_{t=1}^{\infty} \left(\frac{}{1+r}\right)^{t} c = \sum_{t=0}^{\infty} \left(\frac{}{1+r}\right)^{t} \mathrm{E}[r]p(m \quad t) \tag{4.8}$$

$$c = \mathrm{E}[r]p(m \quad t) \tag{4.9}$$

Provided the right hand of Eq. (4.9) is smaller, Bob should use Promise to lock multiple payments pm as it is his individually rational choice that maximizes his pay-off u_B.

4.5 Implementation

We implement Promise in Solidity in around 100 lines of code. We use the implementation to assess the cost of executing the contract functions experimentally. Our cost calculations are summarized in Table 4.1 based on an Ether exchange rate of USD 172.61 and 1.5 Gwei gas price. The implementation is available as an open-source project[1].

Table 4.1: Overview of Promise functions and their cost.

Function	Description	Gas cost	Cost
create	Setup function.	112196	USD 0.02895
collateral	Called by intermediary to provide collateral.	43291	USD 0.01116
payment	Called by user to provide pre-payment.	43770	USD 0.0113
deliver	Called as part of task provision.	50703	USD 0.01309
withdraw	Called by intermediary after the service period is up to receive payment and collateral.	31788	USD 0.0082

[1] https://github.com/nud3l/Promise/tree/master/src

4.6 Related Work

There are two strands of related literature. The first comes from the financial world covering (advance) payments for financial contracts. The second strand comes from the more recent work in decentralized ledgers. A wide range of work focuses on secured debt in the economics literature, such as [Sco77; SJ85]. However, these concepts rely on trust in third parties to maintain security in the debt and payment positions. Promise replaces this third-party trust by holding advance payments in a smart contract escrow. [BT94] considers moral hazard and secured lending in the context of an infinitely repeated credit market game. In addition, the issuance of secured debt has been argued to increase the value of a firm [Sco77]. Other work focuses on the efficiency of secured debt under conditions of imperfect information [Tri92]. Another work branch relates to down payments (a payment made for something bought on credit) [Eng96].

Balance is a protocol that allows intermediaries to lower their collateral over time [Har+19]. It operates at the other end of Promise: instead of lowering the initial collateral, the more an agent behaves honestly, the higher the collateral reduction. Balance requires the highest collateral to be provided at the start of the interaction between agents and assumes that payments are close to 0 (i.e., there is perfect competition). Promise and Balance can be combined to reduce initial collateral when bootstrapping a new protocol and lower collateral requirements for established agents over time. Teutsch et al. discuss bootstrapping a token for verifiable computations [TMB19]. This work discusses how to enable users, like Bob, to obtain the required funds to participate in TrueBit. Their proposal includes a governance game that allows exchanging unique governance tokens into collateral tokens (for intermediaries) and utility tokens (for users). Lastly, the idea of bundling payments together is also introduced in [Ber18] to create subscriptions for the services of agents. Promise extends this idea to allow collateral reduction for intermediaries.

4.7 Conclusion

We present Promise, a subscription mechanism that allows users to lock payments for future services for a period of time. The locked payments are added to the initial collateral of a service provider, Alice, each time a service is delivered. The core assumption for the security of Promise is that a user, Bob, can lock several payments upfront and exit the protocol when Alice misbehaves, receiving back all of his payments over the subscription period and the initial collateral provided by Alice. On the other hand, Alice can utilize Bob's future payments as collateral throughout the subscription period. Promise is a mechanism to reduce initial collateral locking. We have introduced a semi-formal model for Promise. We discuss the security and the effect of the parameter but leave formal proofs of the security properties as future work.

Chapter 5

BALANCE

An agent A performing an action in a protocol can *(i)* be intrinsically motivated to harm the protocol or *(ii)* receive an external payment to act maliciously due to, e.g., bribing [MHM18]. This preference is encoded in v_A and is unknown to the designer of . Since such a valuation is *private information*, the required deposit for secure operation is ambiguous. Further, a deposit can be used as insurance for tasks performed outside the underlying ledger, e.g., performing verifiable computations [KB14; TR17; HB19], cross-chain assets exchanges [Zam+19], or exchanging digital goods [DEF18]. For such usage, the protocol is *event-dependent*, e.g., affected by exchange rate fluctuations. Private information and event dependency require protocol designers to calculate the required deposit levels dynamically. However, this leads to a dilemma: if the required deposit is too high, the agent is likely to refuse to participate in the protocol. A requesting agent fears that the providing agent has an incentive to cheat if the deposit is too low. Whether the deposit is set "too low" and "too high" is subjective in both cases. Thus, our central research question arises: *how can deposits in over-collateralized protocols be dynamically adjusted when assuming that agents have private information and depend on external events?*

We present Balance which provides dynamic collateral adjustment in over-collateralized protocols, improving agents' financial welfare without compromising security. Balance is applicable to protocols that *(i)* are implemented on a decentralized ledger, *(ii)* use a deposit to prevent economically rational agents from performing undesired actions, *(iii)* require agents to over-

collateralize due to at least one of the two identified sources of uncertainty, and *(iv)* verify interactions with agents through objective specifications . Suitable examples for integrating Balance are FairSwap [DEF18], TrueBit [TR17], NOCUST [KGF19], and XCLAIM [Zam+19].

Security intuition. Balance allows agents to reduce their capital deposits over a sequence of periodic rounds while preventing the addition of incentives to act maliciously. In fact, malicious agents that aim to misbehave in a protocol obtain more utility if they do so early on before their deposits are reduced. Conversely, honest and rational agents receive a higher utility by *consistently* acting in the interest of the protocol. Balance achieves this property for protocols that feature over-collateralization by reducing deposits to a lower bound. Above the lower bound, the *additional* utility gained from reducing the deposit is *less* than the opportunity cost for locking up the deposit. Hence, a malicious agent gains no additional utility from cheating in a later round with a reduced deposit. However, honest and rational agents gain additional utility by reducing the opportunity cost of the locked deposit.

Contributions

- **Introduction of Balance , a mechanism for collateral reduction:** the mechanism maps agents to layers according to their behavior, where each layer is associated with a different level of deposit. The higher the layer, the lower the required deposit, mitigating over-collateralization. To the best of our knowledge, this is the first mechanism allowing for the dynamic adjustment of cryptocurrency deposits while maintaining the same level of security.

- **A formal analysis of agents' incentives:** we explicitly model agents' utilities with and without Balance. For agents committed to a protocol, we characterize the conditions on incentive compatibility for Balance . Further, we show that given a distribution of agents over honest, rational, and malicious agent types in a protocol, introducing Balance leads to an increase in social welfare. We show that with a *decrease* of the

provided deposit, we can achieve a *greater* incentive for economically rational agents to behave honestly.

- **An integration with XCLAIM**: we integrate Balance with XCLAIM [Zam+19], a protocol that allows two cryptocurrencies to be exchanged without trusted intermediaries, using Solidity[1]. The clear exhibition of event dependency and private information in XCLAIM makes it a highly suitable Balance use case. We show that Balance reduces deposits by 10% while maintaining the same payoff level for complying with the protocol's specification. The implementation has linear complexity for storing the score of agents depending on the number of layers with a maximum cost of updating a score of a single agent of 55,287 gas (USD 0.07). Further, curating the agents to layers during the transition to the next round has constant complexity with 54,948 gas (USD 0.07).

Structure As with Promise, we adopt the system model from Ch. 2. We provide an overview of Balance and its security assumptions in Section 5.1.1. Next, we introduce the design and functionality of Balance in Section 5.2. We define a model to prove incentive compatibility and social welfare increase in Section 5.3. Balance's security is evaluated in Section 5.4. Finally, we present related work in Section 5.6 and conclude in Section 5.7. We apply Balance to XCLAIM in Ch. 6.

5.1 Preliminaries

5.1.1 System Overview

We introduce a mechanism called Balance, a verifiable layered curated registry. The idea is to use Balance as an extension to an existing cryptoeconomic protocol, i.e., Balance, to control the amount of collateral an agent has to provide to resolve the dilemma of having "too high" or "too low" collateral. The general intuition of the mechanism is displayed in Figure 5.1 and is as follows:

[1] The implementation is available at https://github.com/nud3l/balance.

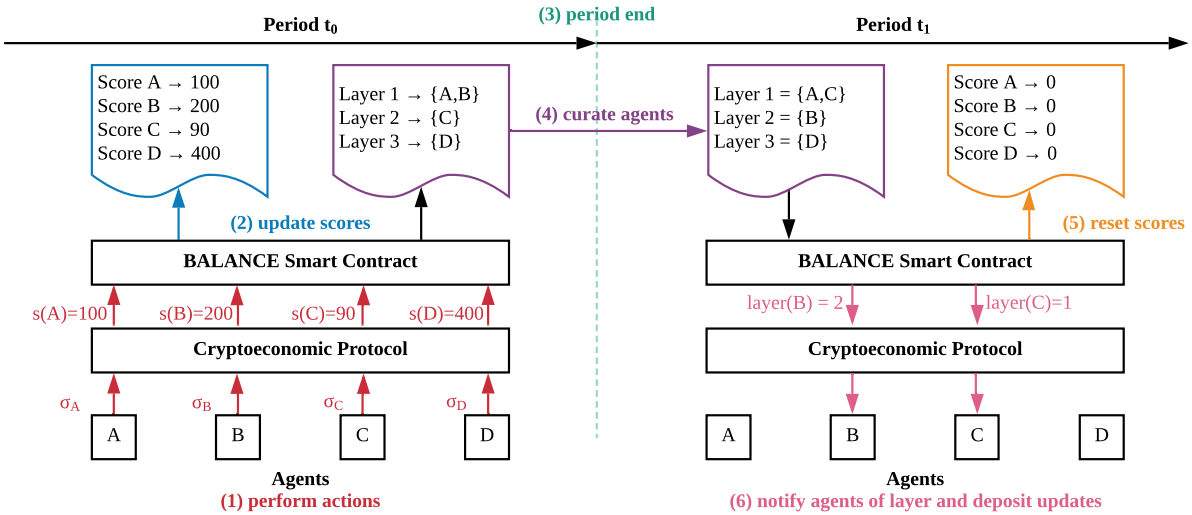

Figure 5.1: Balance is implemented as a smart contract and integrated into an existing cryptoeconomic protocol. Agents A, B, C, and C are assigned to layers in the registry representing their collateral level. In step **(1)**, they are performing actions as part of a cryptoeconomic protocol. By performing these actions, they obtain a score that is forwarded to the Balance smart contract. Step **(2)** is triggered by an agent's actions and updates its scores. Scores reflect a reputation within a round, where the higher the score, the better. In step **(3)**, the current round ends. The next interaction of an agent with the cryptoeconomic protocol triggers step **(4)**, the curation of agents to layers. The score of each agent is used to update the mapping of agents to layers, whereby agents can either stay at the same layer, get promoted or demoted by one, or get removed from the registry. Updating the assignment of an agent to a layer is determined by the upper and lower bounds of the layer an agent is currently assigned to. Within that update, step **(5)** resets the scores of the agent for the next round. Last, in step (6), agents that changed their layer are notified (e.g., via events emitted from a smart contract). From there, agents can start interacting with the contract again to influence their scores.

- *Commitment*: an agent, A, has to provide collateral C_1 to commit to a protocol encoded by a smart contract. For protocols with private information and event-dependency, the collateral is typically set relatively high[2]. An agent is not trusted at any point in time. However, an agent can reduce its collateral by executing actions desired by the protocol.

- *Execution*: agents are tracked by a decentralized registry that stores a ranking of their contributions towards the protocol. Agents are initially assigned to the lowest layer, which results in the highest required collateral. When performing desired actions, the agent collects a score within a round.

- *Curation:* if the agent's score is high enough by the end of a round, it moves to the next

[2]For example, in the Dai stable coin, users have to provide at least 150% collateral to issue Collateralized Debt Positions (CDPs).

layer (provided there is a higher layer). The mechanism defines a factor for each layer multiplied by the base collateral. This will affect the collateral an agent has to provide. Further, after each round, the score of an agent is reset to retain an incentive to act in the best interest of the protocol. Thereby, this mechanism introduces a *sequential game*.

When an agent is assigned to a lower layer, i.e., providing higher collateral, the agent must ensure that the collateral requirement is met within the current round. Otherwise, agents will move down a layer or be removed from the registry.

5.1.2 Actors

Balance includes three actors with the following roles.

1. **Performing agent** A: an agent A is committed to Balance and chooses action $2 \{\Sigma_d; \Sigma_u\}$ evaluating to true or false with respect to the specification of an agreement A. The cost experiments are conducted under the assumption that all agents in the current registry change their layer (i.e., maximum updates in the smart contract)[3].

2. **Receiving agent** B: a receiving agent B requested the performance of an action $2 \Sigma_d$ as expressed by the specification in the agreement A. The execution of the action is left with an agent A.

3. **Registry** R**:** the registry is a smart contract implemented on a ledger. The registry keeps a mapping of scores of performing agents in a round t, a mapping of agents to layers in a round t, and updates the mapping of agents to scores and layers during the transition to the next round $t+1$. The registry has direct access to the result of a performing agent's action $( ) = \{0; 1\}$ to update the agent's score.

Each actor is identified on the ledger using a private/public key pair. We note that agents can take several identities on the blockchain, i.e., Sybil identities are possible. The adjustment of

[3] All USD conversions are made with an Ethereum exchange rate of USD 230 and a gas price of 6 GWEI.

collateral and consequences of performing actions is bound to a public key. An agent A that performs several desired actions to reduce its collateral can have a Sybil identity A'. However, the identity A' is in the view of the registry R a separate agent and hence will not benefit from the reduced collateral of the desired action sequence by identity A. Further, we show in Section 5.4.2, that agents in Balance cannot profit from Sybil attacks.

5.1.3 System properties

Balance achieves the following properties.

1. **Auditability**: performing and receiving agents A and B have public insight into the amount of collateral C and layer assignment of performing agents $A \; ! \; L$ (Section 5.1.4).

2. **Reduction of opportunity costs**: Balance reduces the opportunity costs of locked-up collateral $\mathrm{E}[rC_A]$ for performing agent A by reducing collateral depending on the layer assignment $A \; ! \; L$ (Section 5.2.1).

3. **Strategy-proofness**: a performing agent A behaves truthfully concerning its valuation of an outcome v_A independent of its type (Section 5.3.2, 5.3.4).

4. **Transparency**: assuming performing agents A_1 and A_2 provided the same initial collateral C, but are now assigned to different layers such that $C_{A_1} \not= C_{A_2}$, B does not need to choose between A_1 and A_2 as Balance ensures that each collateral provides the same utility for A_1 and A_2 to act in the interest of B (Section 5.3.5).

5. **Social welfare increasing**: the joint welfare of performing agent A and receiving agent B is increased with Balance (Section 5.3.6).

6. **Sybil resistance**: a performing agent A cannot gain additional utility from Balance by creating Sybil identities A' (Section 5.4.2).

5.1.4 Blockchain model

Balance operates as part of implemented on a blockchain. This blockchain provides a ledger functionality as, for example, defined in [PSS17; Bad+17; Dav+18; Bad+18]. We assume that the ledger has *nality* as in consensus protocols like [Mil+16; Ten16; CL99]. Additionally, we assume given the number of participants n in a consensus protocol, the number of Byzantine faults is $f < n/3$. In cases where a consensus protocol does not provide finality, like Nakamoto consensus [Nak08], we assume that a security parameter k is set such that with high probability the transaction is securely included [Ger+15]. Apart from that, we assume that less than 1/3 of miners are malicious [ES14]. Setting an appropriate parameter for k and having less than 1/3 miners malicious, prevents the impact from selfish-mining attacks.

Aside from the ledger, we assume there is an execution environment and a scripting language available that supports the creation of secure smart contracts to implement agreements A and the registry R, for example, the Ethereum Virtual Machine [Woo14; Tsa+18]. We assume that the agreement A including its specification , payments p, and collateral C is implemented by a smart contract SC such that SC , A. Further, we require the possibility of creating generic data structures to store the information about agents' scores and layers as part of the registry R and that such information is readable by performing and receiving agents A and B. Moreover, we assume that the cryptographic primitives in the ledger are correctly implemented.

We make no further assumptions regarding the blockchain. Miners and adversaries can re-order transactions, censor transactions, and read the information from transactions before and after they are included in the blockchain.

Property 1 Auditability

The collateral C and mapping of performing agents to layers A ! L is publicly available within R.

5.1.5 Agent assumptions

We make the following assumptions. First, for clarity of exposition, we simplify that if an agent performs a single desired action at a specific time, they transition to the higher layer. This is a simplification because, in the complete specification, agents would need to perform several actions to change the layer. This assumption strengthens the adversary, as collateral reduction requires only a single action. Second, we assume that Balance unambiguously detects an undesired action, i.e. ( ) = 0. In this case, the agent takes on a new identity in the protocol and can continue to interact with the protocol starting again from the lowest layer. Third, we assume that in the long run, the market for a protocol is perfectly competitive in the sense that the price (or payments) made in the system will be equal to the marginal cost of each transaction. This assumption provides a higher incentive to perform undesired actions as $p = 0$ *reduces* the utility of performing the desired action as described in Eq. (3.1). Fourth, we assume that performing an undesired action is free of direct costs[4]. This assumption increases the utility for performing an undesired action as we set $c_A = 0$ for the undesired action utility.

Overall, our assumptions are pessimistic: we assume agents gain no payments from performing desired actions and can perform undesired actions free of direct cost. Hence, we ensure that incentive compatibility holds even under these pessimistic assumptions (cf. Section 5.3). In practice, we expect that these assumptions can be modified to include positive payment vectors and costs for executing undesired actions. This would allow more aggressive collateral reductions; however, calculating these are left as future work.

5.2 Balance Protocol

Extending with Balance adds two properties. First, we introduce the *registry* R consisting of layers as well as functions to update the score of agents and to curate agents into layers.

Second, we include a *score* s into A such that:

[4]For example, an agent might go o ine and not perform any action in A.

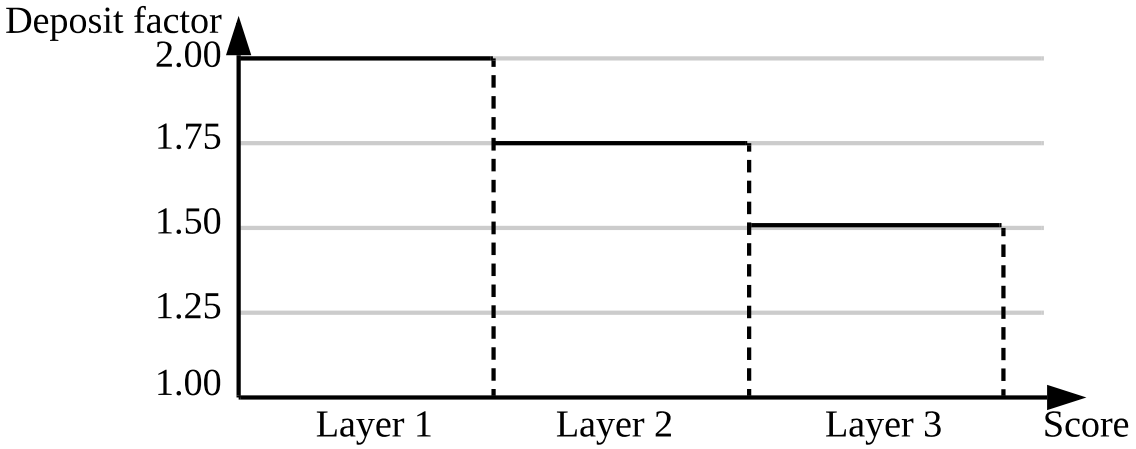

Figure 5.2: An intuition of Balance with three layers. The more an agent contributes to the integrated protocol (measured by a score), the further it moves up the layers (from 1 to 3). The higher the layer, the lower the relative collateral needed.

$$A = h\ ; p; s; C i \tag{5.1}$$

s is added to the current agent's score if the specification of the agreement evaluates to true. All details of the agreement are public knowledge.

5.2.1 Integrating layers

A layer is a set of agents that have to provide the same relative collateral. We use Balance to assign agents to layers, making this a verifiable layered curated registry as illustrated in Figure 5.2. Intuitively, the higher the score of an agent, the higher the layer an agent is assigned to.

While one could argue that the score could directly be used to calculate the collateral (for example, similar to bonded curves), layers have advantages. Having layers allows for multiple equilibria from the perspective of the protocol. Further, collateral can be more predictable by programming a pre-defined number of layers with specific collateral levels. In an open market, the protocol designers might not know how many interactions an agent has with the

protocol. Thus, layers give protocol designers the chance to set bounds on actions that reduce the collateral. For agents contributing to the protocol by performing desired actions, they can expect their required collateral to fall as they progress through the layers. This expectation motivates agents to contribute to the protocol while still keeping the collateral relatively high.

We formally define a finite order of layers $fL_1 \quad ::: \quad L_!g$, where each layer $L_m \ 2 \ L$ has a lower bound l, an upper bound u and a collateral factor $f \ 2 \ R^+$. We explain the use of the bounds in Section 5.2.3. The factors determine the ordering of layers. Each layer maps to a collateral level in order, $L \ ! \ C$. For instance, L_2 requires collateral C_2.

$$L_i = hl; u; f\, i \tag{5.2}$$

The set of agents are mapped to the layers, i.e., $P \ ! \ L$. Each agent is only mapped to a single layer at any point in time. The *layer* function returns the layer an agent is currently assigned to. We define the layer(A) as the current layer of an agent A, layer(A) + 1 as the higher layer with a lower collateral factor, and layer(A) 1 as the lower layer with a higher collateral factor.

The layers are used to calculate the collateral an agent needs to provide. We define a base collateral C_{base} (for the lowest layer) and a factor f that are used to calculate C_m, where $m \ 2 \ L$. Further, the collateral C_A of an agent A is determined by the base collateral and the factor of the layer the agent is currently assigned to.

$$C_m = C_{\text{base}} f_m \tag{5.3}$$

$$C_A = C_{\text{base}} f_{\text{layer}(A)} \tag{5.4}$$

Factors are ordered similarly to the layers. Formally, factors are a finite order $ff_! < \quad < f_1g$ where the factor corresponding to the highest layer $L_!$ is the *smallest*.

Property 2 Reduction of opportunity costs
Balance reduces collateral lock-up.

5.2.2 Updating scores

An agent that just committed to an agreement starts at the lowest layer with the highest collateral factor. An agent can move up or down the layers in the registry depending on its score s_A.

Agents can increase their scores in any round by performing actions. The score for an action is determined from the agreement, as defined in (5.1). The *update* function takes an action by an agent and updates the agent's score.

$$\text{update}: \quad _A \; ! \; s_A \tag{5.5}$$

5.2.3 Curating agents

We define a function *curate*. This function takes as input all agents in an agreement A and results in a new assignment of agents to layers depending on their scores. If an agent's score is higher than the upper bound of its currently assigned layer, the agent progresses to the higher layer. If an agent's score is lower than the lower bound of its currently assigned layer, the agent falls back to the lower layer.

$$\text{curate}(A) = \begin{cases} \text{layer}(A); & \text{if } l_{\text{layer}(A)} \quad s_A \quad u_{\text{layer}(A)} \\ \text{layer}(A) + 1; & \text{if } u_{\text{layer}(A)} < s_A \\ \text{layer}(A) \quad 1; & \text{if } s_A < l_{\text{layer}(A)} \end{cases} \tag{5.6}$$

As layers are a finite order, an agent cannot progress above the highest layer or below the lowest layer. An agent can leave the registry by de-registering. This detail is left open to the actual underlying cryptoeconomic protocol. We note that most protocols require the agent to perform the action they initially committed to. If an agent leaves prematurely, i.e., before the concluded

phase, the agent's collateral is usually destroyed. However, an agent can certainly participate in multiple agreements in the same contract. The scores of these multiple agreements are added to the overall score of the agent. An agent already in the highest layer can, in the best case, remain there. An agent currently in the lowest layer and misses its lower bound is excluded from the registry. Further, any agent that performs an undesired action in any agreement part of a smart contract is immediately excluded from the registry, and the collateral is destroyed or refunded to another agent.

5.2.4 Time period

According to our blockchain assumptions, Balance requires a minimum number of blocks to consider transactions confirmed depending on the security parameter k [GKL15]. Further, Balance requires a common period t over all its agreements A. We can determine a *minimum* time period by considering the agreement that requires the longest time even in the case of multiple chains (e.g., XCLAIM as in Section 6.1). However, the time between blocks Δ is not constant on all chains (e.g., [DW13]). In the case of a single chain, it is sufficient to express the time as a number of blocks, i.e., t depends solely on the security parameter k and the number of transaction n_{tx} to execute an agreement such that $t = kn_{\mathsf{tx}}$.

If Balance is used in the multi-chain case, we need to include a buffer b to account for deviations in Δ. The time period in seconds for *one* chain is expressed by $t_{\mathsf{chain}} = b\ (n_{\mathsf{tx}}k\Delta)$. We can then calculate the total time by summing the time periods for all involved chains within an agreement. Last, the time period can be expressed as a number of blocks by dividing the total time by the average Δ of the chain on which Balance is implemented.

However, we note that the time period also determines how often agents need to perform desired actions to remain or progress in the layers. Hence, protocol designers need to consider how often agents execute actions within the cryptoeconomic protocol.

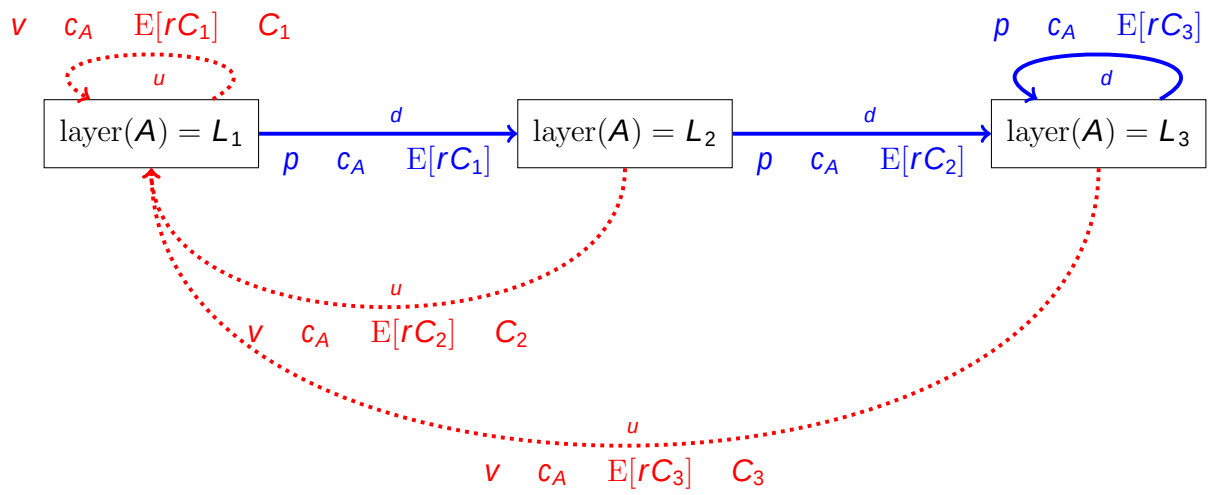

Figure 5.3: Layer transitions for agent A in a simplified 3-layer system. The red dashed lines correspond to an undesirable action($_u$); the black solid branches correspond to a desirable action ($_d$). The agent starts in L_1. If the agent chooses $_d$ they are moved into layer L_2, receiving payoff p c_A E[rC_1], thus reducing the agent's collateral to C_2. In contrast, if the agent chooses $_u$ in L_1 they receive payoff v c_A E[rC_1] C_1 and are removed from the registry. Assuming agents can rejoin the protocol, this is equivalent to being returned to L_1. This process is repeated analogously in L_2 and L_3. Payoffs corresponding to no action, $_p$ 2 ;, are excluded as doing nothing would yield negative utility for an agent and therefore a rational agent would not choose this option.

5.3 Incentive Compatibility

The performing agent commits to the protocol by providing collateral C_m. The collateral required by each layer is such that $C_1 > C_2 >$ $> C_!$: Balance rewards the performance of desired actions by letting agents move to the next layer, thereby decreasing the amount of collateral they need to lock up. The utility that the agent gets depends on their choice of action from $_A$ 2 $f\Sigma_d; \Sigma_u; \Sigma_; g$. Given that the performing agent is committed to the protocol, a utility-maximizing agent will never choose to do nothing ($_A$ 2 $\Sigma_;$) as they would receive a positive utility from committing either a desired or an undesired action (depending on their valuation, v_A).

Figure 5.3 presents the payoffs the receiving agent would receive by deciding to perform either a desired or undesired action. The performing agent receives a payoff after each move. If we consider a single-shot game at each round, the agent must decide between the resulting utility of two actions, i.e., the desired and undesired action.

5.3.1 Action choice

We can express the condition for choosing a desired action for an agent A at layer $m \in [1; !]$ with the following Eq.s:

$$v \quad c_A \quad \mathrm{E}[rC_m] \quad C_m < p \quad c_A \quad \mathrm{E}[rC_m] \tag{5.7}$$

This amounts to requiring that

$$v < C_m + p \tag{5.8}$$

Theorem 5.1. *Considering a single-shot game, if the decision to perform the desired action holds for the highest layer $L_!$, then it will also hold for all previous layers.*

Proof. If the agent in the highest layer decides to perform the desired action, Eq. (5.8) holds. As $C_! < \quad < C_1$, the utility of the agent at layer $L_!$ is higher than at the previous layers. Hence, a rational agent deciding to perform the desired action at layer $L_!$ makes the same decision in the previous layers. This also allows it to reach the highest layer. □

5.3.2 Incentive Compatibility for Honest and Malicious Types

We now extend the analysis to consider the intertemporal payoffs resulting from the agents' decisions. If an agent always does the desired action, i.e., an agent of type $_d$, they get a payoff of $p \quad c_A \quad \mathrm{E}[rC_1]$ in the first period. In the second period, invoking a discount factor $0 < \quad < 1$ and a real rate of interest r, agents receive a payoff of $(\frac{}{1+r})(p \quad c_A \quad \mathrm{E}[rC_2])$ in today's terms. The payoff is discounted, reflecting that future payoffs are less valuable than present ones because a payoff today would receive a rate of return r. This continues until the agent is in the highest layer, $L_!$. In the highest layer, agents would receive a payoff of $(\frac{}{1+r})^{! \ 1}(p \quad c_A \quad \mathrm{E}[rC_!])$ in today's terms. Thus summing to infinity yields the following payoff $u(\ _d)$ for agents who always performs a desired action each round:

$$u(\ _d) = \sum_{t=0}^{!-1} (\frac{}{1+r})^t (p_A \quad c_A \quad \mathrm{E}[rC_{t+1}]) + \sum_{t=!}^{\infty} (\frac{}{1+r})^t (p_A \quad c_A \quad \mathrm{E}[rC_!]) \tag{5.9}$$

Where convergence of the latter term requires that $|(\frac{}{1+r})| < 1$.

If we assume that an agent is of type $_u$, the agent will indefinitely perform an undesired action: at the lowest layer L_1, the utility of the undesired action is higher than the utility of the desired action. Hence, this agent's utility is as follows.

$$u(\ _u) = \sum_{t=0}^{\infty} (\frac{}{1+r})^t (v \quad c_A \quad \mathrm{E}[rC_1] \quad C_1) \tag{5.10}$$

Assuming that $|(\frac{}{1+r})| < 1$, such that the series of payoff converges, this can be expressed as:

$$u(\ _u) = \frac{(1+r)(v \quad c_A \quad \mathrm{E}[rC_1] \quad C_1)}{1+r} \tag{5.11}$$

Theorem 5.2. *Agents of types $_d$ and $_u$ in* Balance *act individually rational and truthful to their valuation.*

Proof. By Definition 3.1, agents committed to the protocol cannot increase their utility by hiding their true valuations: the protocol is strategy-proof. The utility of each agent in an infinite time horizon is given by Eq.s (5.9) and (5.10), respectively. Thus, Balance is incentive compatible for types $_d$ and $_u$. □

Property 3 Strategy-proofness
Balance *offers strategy-proofness for types $_d$ and $_u$.*

5.3.3 Decision Boundary for the Rational Type

The rational agent of type $_r$ is indifferent between performing an undesired or desired action in the highest layer L. Hence, consider the following strategy $S_{\text{layer cycling}}$ for an agent:

Definition 5.1 (Layer-Cycling)**.** *The agent performs a desired action in the rst ! 1 layers but commits an undesired action once in layer !.*

We index these cycles with , such 0 1 . For instance, the first iteration of this strategy is denoted by = 0; the next iteration by = 1. If we consider a one-shot game, whether the agent optimally commits a desired or undesired action is determined by (5.8). However, this does not consider that once the agent performs an undesired action, they need to work back up through the layers, with greater opportunity costs for their locked-up collateral, by performing ! 1 desired actions to reach the highest layer again. Thus, we provide a framework that captures the intertemporal trade-off that agents face in Balance in committing an undesired action once in the highest layer.

For each cycle complete starting at $t = 0$, the payoff of such a strategy can be expressed as

$$u_A = v \quad c_A \quad C_! \quad \mathrm{E}[rC_!] + \sum_{t=1}^{!-1} \left(\frac{}{1+r}\right)^t fp \quad c_A \quad \mathrm{E}[rC_t]g \tag{5.12}$$

where $C_!$ denotes the collateral made in the highest layer $L_!$. The corresponding expression for infinite cycles is provided in Eq. (B.4) in Appendix B.1.

Now consider the stream of payoffs to an agent who always commits the desired action once in the highest layer. We can group the stream of payoffs into sections of size $!$, to correspond to the number of rounds required for a player playing $S_{\text{layer cycling}}$ to complete one entire cycle. Thus the equivalent utility for performing desired actions for a single cycle is as follows:

$$u_A = \sum_{t=0}^{!-1} \left(\frac{}{1+r}\right)^t fp \quad c_A \quad \mathrm{E}[rC_!]g \tag{5.13}$$

Similarly, the corresponding expression for infinite cycles is provided in Eq. (B.2) in Appendix B.1. Lemma B.1 in the same Appendix shows that comparing the payoffs of Eq. (5.13) and (5.12) over infinite cycle repetitions ! 1 is the same as comparing those equations for a single cycle. Thus, equating (B.4) and (B.2) provides us with an expression for the value of v at the boundary.

$$v = p + C_! + \sum_{t=1}^{!-1} \left(\frac{}{1+r}\right)^t (\mathrm{E}[rC_t] \quad \mathrm{E}[rC_!]) \tag{5.14}$$

Eq. (5.14) captures several salient aspects of Balance. Firstly, the boundary valuation depends on the levels of p and C_{ℓ}. An increase in the payment that the agent receives or an increase in the collateral in the final layer serves to increase the value of v required such that the agent is indifferent between committing a desired or undesired action. The minimum valuation v also depends on the difference in opportunity costs (forgone returns) on collateral, with the opportunity cost of funds in lower layers contributing more: $(\frac{}{1+r})^t$ declines as t increases.

As detailed in Appendix B.2, explicitly invoking collateral ratios between the layers, such that $C_{\ell} = f_{\ell} C_{\text{base}}$ and $C_t = f_t C_{\text{base}}$, and setting $p = 0$, enables (5.14) to be rewritten as

$$v = f_{\ell} C_{\text{base}} + \sum_{t=1}^{\ell-1} \frac{}{1+r}^{t} \mathrm{E}[r C_{\text{base}}(f_t \quad f_{\ell})] \tag{5.15}$$

Note that on the assumption that an agent can advance a single layer m per time period t, to make this relationship between layers and time explicit, we swap m for t. Rearranging this expression for f_{ℓ} and assuming a linear relationship between the smallest factor f_{ℓ} and the other factors f_m (see Appendix B.1) enables the indifference boundary to be plotted, see Figures 5.6, 5.4, 5.7, and 5.5.

5.3.4 Incentive Compatibility Region for the Rational Type

We can characterize the incentive compatibility region for the rational agent as follows.

Theorem 5.3. *Provided $f_{\ell} > f_{\ell}$, performing $_d$ at the highest layer (and not $S_{\text{layer cycling}}$) is incentive compatible for type $_r$.*

Proof. We set C_{base} as a relative value where $C_{\text{base}} = 1$. Assuming a linear relationship between the smallest factor f_{ℓ} and the other factors f_t, as detailed in Appendix B.2 (B.12), enables us to express the boundary condition for incentive compatibility for agents of type $_r$ f_{ℓ} as follows:

$$f_{!} = \frac{v! \quad v \quad \sum_{t=1}^{!} \frac{1}{1+r}^{t} \mathrm{E}[r f_1(! \quad t)]}{! \quad 1 \quad \sum_{t=1}^{!} \frac{1}{1+r}^{t} \mathrm{E}[r(! \quad t)]} \tag{5.16}$$

Thus, provided $f_{!} > f_{!}$, incentive compatibility holds for $_{r}$. □

Property 3 Strategy-proofness

Balance *offers strategy-proofness for type* $_{r}$.

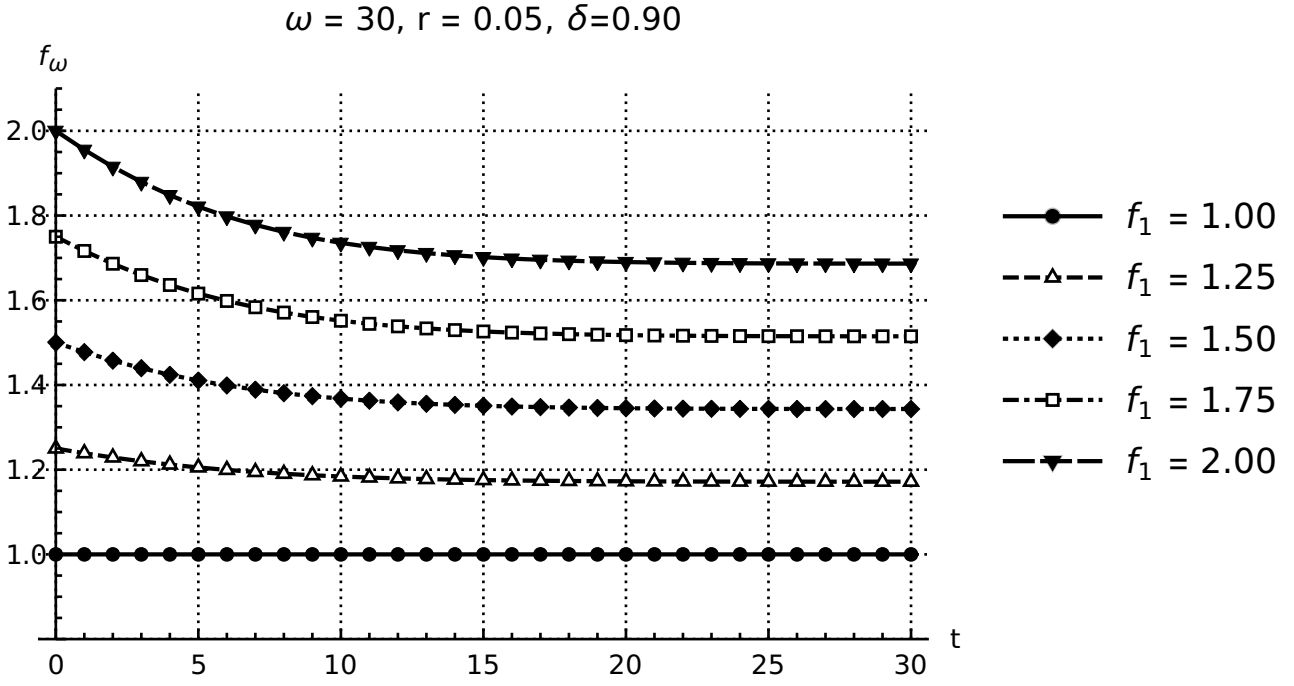

Figure 5.4: The boundary $f_{!}$ depending on five different initial factors of f_1. If $f_{!}$ is chosen above $f_{!}$, an economically rational agent has a higher utility to choose the desired action. Below $f_{!}$, the agent chooses an undesired action.

5.3.5 Transparency

Balance seeks to be transparent to receiving agents B. B does not have to choose between performing agents A based on their layer assignment but selects A as without Balance.

Theorem 5.4. *Given an initial collateral C_1 for a performing agent A of type* $_{r}$ *and* Balance *with ! layers, A cannot decrease its utility for performing a desired action and does not have a*

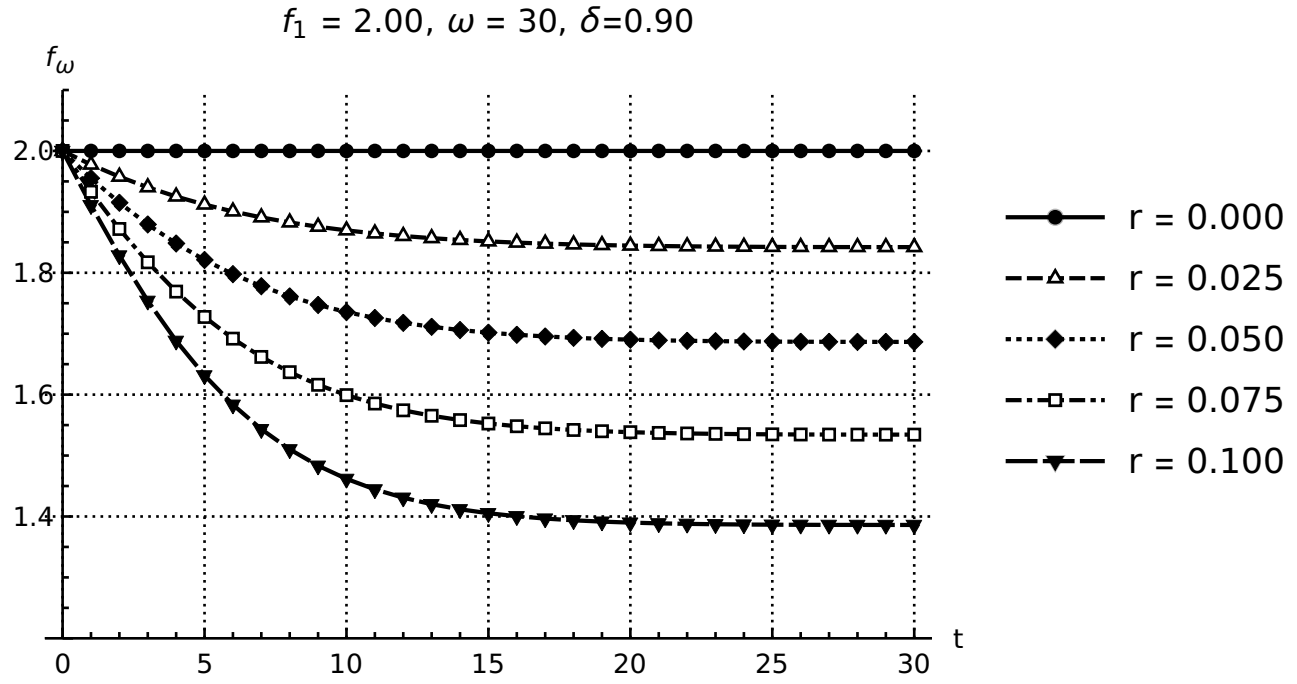

Figure 5.5: The boundary $f_!$ depending on five different return rates r. If $f_!$ is chosen above $f_!$, an economically rational agent has a higher utility to choose the desired action. Below $f_!$, the agent chooses an undesired action.

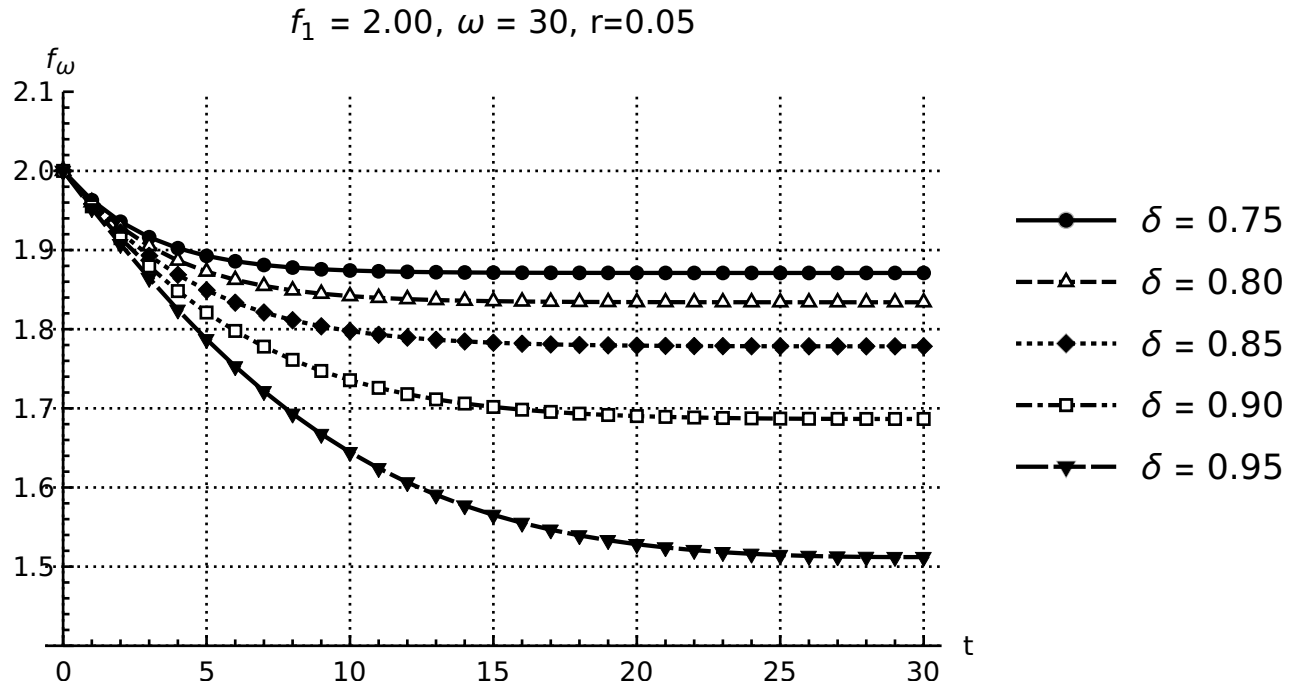

Figure 5.6: The boundary $f_!$ depending on five different discount factors . If $f_!$ is chosen above $f_!$, an economically rational agent has a higher utility to choose the desired action. Below $f_!$, the agent chooses an undesired action.

higher utility for an undesired action assuming $f_! > f_!$.

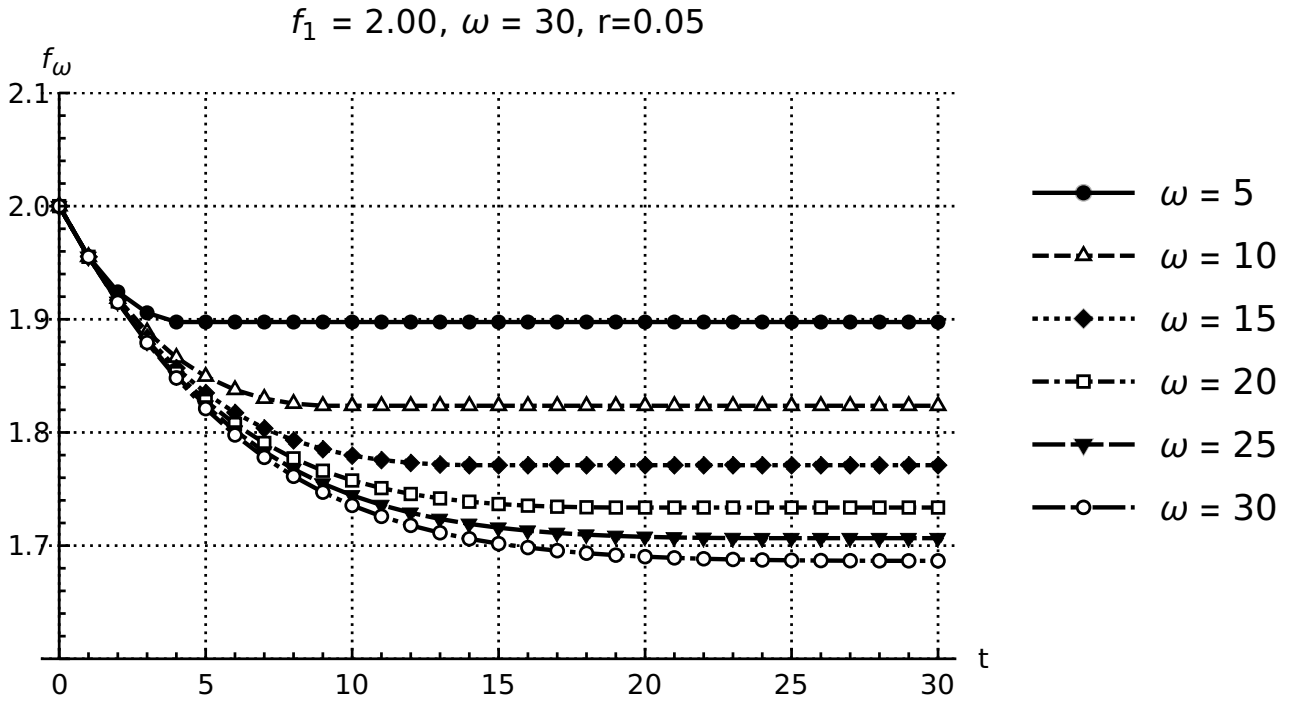

Figure 5.7: The boundary $f_{!}$ depends on six different numbers of layers $!$. If $f_{!}$ is chosen above $f_{!}$, an economically rational agent has a higher utility to choose the desired action. Below $f_{!}$, the agent chooses an undesired action.

Proof. If A $!$ L_m at a point $t = 0$, we consider two cases: First, A can consider an *infinite* horizon. By Theorem 5.3, we know that if $f_{!} > f_{!}$, the utility for performing a desired action is greater than the undesired action for A starting at L_1. By the infinite time horizon, the starting layer L_m becomes irrelevant. Hence, A has the same utility for performing the desired action at layer L_m as at L_1.

Second, by Theorem 5.1, if agent A performs a desired action at the highest layer, A also performs desired action at previous layers. Further, the utility at the lowest layer L_1 is smaller than that at the subsequent layers. If A only considers a single step, and A has performed the desired action at layer L_1, A can only increase its utility by performing a desired action in the next layer. □

Property 4 Transparency
Balance *is transparent to a receiving agent B.*

5.3.6 Comparison and social welfare

Next, we show that adding Balance increases social welfare for agents who perform desired actions.

Theorem 5.5. *For a distribution of agent types $p(x)$, where $x \in \{\theta_d, \theta_u, \theta_r\}$, adding* Balance *increases social welfare.*

Proof. As in (3.4), if performing agents perform desired actions, the utility is equal to $\sum_{x=1}^{X}(v_{x;B} - c_{x;B} - c_{x;A} - \mathrm{E}[rC_{x;A}])$, where x is a pair of agents in the set of pairs X. Without Balance, each pair must have collateral C_1 each period. Over a cycle length $t = \ell$, the total utility for all pairs of agents over the cycle is given by

$$\sum_{t=0}^{\ell-1}\sum_{x=1}^{X}(1/(1+r))^t((v_{x;B} - c_{x;B} - c_{x;A} - \mathrm{E}[rC_{x;1}])) \tag{5.17}$$

In contrast, with Balance, total welfare is given by

$$\sum_{t=0}^{\ell-1}\sum_{x=1}^{X}(1/(1+r))^t((v_{x;B} - c_{x;B} - c_{x;A} - \mathrm{E}[rC_{x;t+1}])) \tag{5.18}$$

For agents performing desired actions, Balance improves welfare when (5.18) is greater than (5.17). Nevertheless, since in all but $t = 0$, the opportunity cost term ($[rC_{x;t+1}]$) is smaller in the latter equation, with Balance permitting agents to post smaller collateral at higher layers, (5.18) must be the larger. Therefore, Balance improves social welfare for agents performing desired actions. For agents who always perform undesired actions, social welfare is the same with Balance as without it. □

Property 5 Social welfare increasing
Balance *is social welfare increasing.*

5.3.7 Parameter behavior

Balance has four main parameters that influence the boundary f_{l} over which f_{l} must be set to ensure that agents of type $_{r}$ perform desired actions. Figures 5.4, 5.5, 5.6, and 5.7 demonstrate how f_{l}, the factor corresponding to the highest layer boundary, changes as the parameters f_1, r, and l vary. The purpose of this section is to provide guidance on the behavior of a given implementation when the parameter values are varied. The values of f_1 and l can be configured by the designer of the protocol , whereas and r are external factors that require careful consideration.

- Figure 5.4 shows that, all else equal, for all t, opting for a higher value of f_1 results in a relatively higher value of f_{l} than lower values of f_1. Starting with a higher layer factor f_1 permits a greater reduction in f_{l} over time relative to the starting value of f_{l} at $t = 0$, but results in a higher absolute value of f_{l} relative to lower values of f_1.
- Figure 5.5 shows that, all else equal, $\forall t$, an increase in r results in a lower value of f_{l}. To offset a higher interest rate, f_{l} has to fall more quickly since a higher interest rate results in higher opportunity costs of collateral.
- Figure 5.6 shows that, all else equal, $\forall t$, an increase in the discount factor allows f_{l} to be relatively lower. As the discount factor increases, such that future utility is more valuable today, the factor f_{l} corresponding to the highest layer can be reduced.
- Figure 5.7 shows that all else equal, $\forall t$, using more layers l permits a greater reduction in f_w through time. However, this increase is not linear. Using Eq. 5.16, we can show that if $l \; l \; 1$, collateral reduction has a lower bound.

We note an interesting property for protocol designers when choosing parameters for Balance. If the required collateral factor at each layer is set to the boundary f_{l}, Balance enjoys the same level of security. However, if a designer sets the collateral factors for layers m such that $f_1 > f_m > f_{l}$, Balance has a relatively higher level of security with respect to economically rational agents *and* increases social welfare.

5.4 Security arguments

We discuss a range of attack strategies against Balance and how to mitigate them.

5.4.1 Single-shot attack

In a single-shot attack strategy, $S_{\text{single shot}}$, the only objective of agent A is to attack B through committing an undesired action. A would then consider executing such an attack with the least cost. In particular, A may reduce the cost of an attack by playing a modification of $S_{\text{layer cycling}}$ where A progresses through the layers and performs the undesired action at the highest layer. We show that if $f_{!}$ is set above a bound $f_{!}$, A would not gain additional utility by waiting until it is in the highest layer before performing the attack.

Lemma 5.1. *If A plays $S_{\text{single shot}}$ it cannot gain additional utility given that the $f_{!}$ is set above $f_{!}$.*

Proof. This strategy has two implications. First, A performs $!$ 1 desired actions that contribute to the social welfare of the protocol. Second, A increases its utility at $t = !$ for the undesired action with being punished with the smaller $C_{!}$ instead of C_1. However, for each $!$ 1 round, A has incurred an opportunity cost. As shown in Lemma B.1 in Appendix B.1, $S_{\text{single shot}}$ results in the same utility expression and boundary for $f_{!}$ as $S_{\text{layer cycling}}$. Hence, if $f_{!}$ is set to be greater than $f_{!}$, A does not gain additional utility from playing $S_{\text{single shot}}$. A should commit the undesired action at C_1 if A intends to perform an undesired action. Otherwise, A should perform the desired action. □

Balance provides in the single-shot attack the same security as protocols without Balance, however, offers desired agent types to reduce their collateral over time.

5.4.2 Reputation boosting

An agent A could create Sybil identities to request the performance of actions and fulfill them themselves. This results in improving the agent's score and reducing its collateral. We denote this S_{boosting}, a common attack vector in reputation-like systems [KSG03; JIB07].

Theorem 5.6. *Reputation boosting S_{boosting} is not rational in* Balance *if the cost of this strategy, $c(S_{\text{boosting}})$, is higher than the expected saving from the reduction of collateral.*

Proof. We consider a scenario where A can take the role of a receiving agent B. Further, we assume that the cost for requesting a service within an agreement A by $c(S_{\text{boosting}}) = c_B$. Without loss of generality, we assume that an instance of Balance includes two layers, L_1 and L_2. A performs the desired action by fulfilling its request (per role B). Next, A can either execute a desired or an undesired action. In either case, the increase of utility is determined by the reduction in opportunity cost at each layer, i.e., $(\frac{}{1+r})\mathrm{E}[rC_2]$ $\mathrm{E}[rC_1]$. A sufficient condition for S_{boosting} to not be a rational strategy is:

$$c_B > \mathrm{E}[rC_1] \qquad \frac{}{1+r} \quad \mathrm{E}[rC_2] \tag{5.19}$$

Therefore, we prevent S_{boosting} by limiting the maximum collateral that can be provided by a single identity of A to $C_{\max}$. If A wants to exceed $C_{\max}$, it needs to commit to an agreement with an additional identity A'. □

The *commit* stage of cryptoeconomic protocols typically requires an on-chain transaction which incurs a cost, e.g., [KGF19; PD16]. If requesting is *free* in a protocol, boosting would be rational. Thus, to prevent *grie ng* by B, cryptoeconomic protocols typically require a small amount of collateral [DEF18] as friction. Since B also has an expected loss of interest on this collateral, this incurs a cost expressed by $\mathrm{E}[rC_B]$.

Property 6 Sybil resistance
Balance *is resistant to the creation of Sybil identities.*

5.4.3 Action delay

In strategy, S_{delay}, A, who has already performed sufficient desired actions to move up a layer, delays the desired action from t until $t+1$. In the new round, with a reset score, the delayed action would count towards the new score. If there are significantly more requesting agents B than performing agents A, a backlog of requests will occur. We propose two remedies. First, A could be required to perform the actions within a time limit enforced by the cryptoeconomic protocol (e.g., HTLCs). Second, Balance could be modified to use *dynamic time rounds*. Dynamic time could be based on the number of *active* performing agents A and the number of requests by receiving agents B. For instance, the next round could be started after a certain fraction (or all) of the performing agents A could have at least fulfilled one request by B.

5.4.4 Competition and cooperation

Performing agents compete to fulfill agreements, receive payments and reduce their collateral requirements. However, the number of requests may be insufficient for A to maintain its assignment to the high layers of Balance. In such cases, two strategies for A are as follows.

First, A can be *non-cooperative* and play a strategy S_{compete} in which A (i) tries to fulfill requests as fast as possible by, e.g., detecting request transaction early through memory pool sniffing and (ii) actively prevent other performing agents from fulfilling requests by, e.g., executing eclipse attacks [Hei+15; Ger+15].

Second, A can be *cooperative*, executing a strategy $S_{\text{cooperate}}$ to collaborate with other agents. A could form groups with shared interests through, e.g., multi-signature wallets with other agents. A set of agents can act as a single entity to reduce their pooled collateral and fulfill agreements as a whole. This concept is similarly applied to layer-one protocols as so-called mining pools. Lewenberg et al. formulate an analysis for cooperative games in the Bitcoin mining network [Lew+15]. A cooperative game for cryptoeconomic protocols would consider the number of performing agents A, their overall collateral C, and the likelihood of having open requests by B.

In both cases, `Balance` does not change the possible strategies in the cryptoeconomic protocol. Agents can play either strategy with and without `Balance` being present.

5.5 Implementation

We implemented `Balance` for Ethereum in Solidity v0.5.0[5]. The implementation consists of around 250 lines of code. `Balance` can be used as a library, or another protocol can inherit its variables and functions. The core functions of `Balance` are *update* and *curate* defined by (5.5) and (5.6).

The cost experiments are conducted under the assumption that all agents in the current registry change their layer (i.e., maximum updates in the smart contract)[6]. We conducted experiments with 100 to 500 agents and four layers.

The main issue of the implementation is related to the *update* function. In a naive implementation, the algorithm loops through every agent and check if it needs an update to the layer. However, this implementation would result in an *out-of-gas* exception with more than 250 agents. Hence, we adjust our implementation.

We achieve the constant complexity in the `curate` function by executing the assignment of agents for the next round into the `update` function, see implementation 5.9 and 5.8. In Solidity, this is handled by mapping agents to layers for every round. The `curate` function updates the round counter. The `update` function has a linear complexity concerning the number of layers, amounts to 55,287 gas costing around USD 0.05. The `curate` function has constant complexity independent of any parameters. The execution of the `curate` function costs 54,948 gas which is equivalent to around USD 0.05.

[5] Implementation available at https://github.com/nud3l/balance.
[6] All USD conversions are made with an Ethereum exchange rate of USD 106.00 and a gas price of 9 Gwei.

```
function curate() public onlyOwner returns (bool) {
    require(_start != 0, "period not started");
    require(block.number >= _end, "period not ended");

    // update start and end times for next round
    _start = block.number;
    _end = block.number + _blockperiod;

    // switch to the next round
    _round++;

    emit Curate(_round, _start, _end);
    return true;
}
```

Figure 5.8: The *curate* function of Balance implemented in Solidity. When a new round starts, all agents are assigned to their respective layer by incrementing the round.

5.6 Related Work

To the best of our knowledge, Balance is the first reputation-based system for the dynamic adjustment of cryptocurrency deposits. There are two discernible strands of related literature.

5.6.1 Token Curated Registries

The first strand relates to Token Curated Registries (TCRs) [Mik17], inspiring the layered aspects of Balance [McC18]. A TCR formally represents a set R in which elements n can be included in a set through a token-based voting mechanism. A variety of different TCR types have been proposed [Mik17; Rou17; Rud18; Gaj18; McC18].

Notably, ranked TCRs R_O enable agents to vote on the rank (i.e., position) of an element in a set of elements such that $n_i \quad n_{i+1}$. Further, layered TCRs are a set R_L, consisting of distinct subsets where $[_{i=1}^{n} R_i \quad R_L \quad \bigvee \; \setminus_{i=1}^{n} R_i = ;$. However, while Balance takes inspiration from a ranked and layered TCR construction, TCRs require voting by individuals with tokens, adding significant and potentially unwarranted complexity [AK18].

Roles *Consumers* of R use a registry to support their decision process. A registry could consist of a list that contains valuable information. For example, students deciding which

```
function update(address agent, uint256 action) public returns (bool) {
    _scores[_round][agent] += _rewards[action];

    // assignment in the current round
    uint8 assignment = getAssignment(agent);

    require(assignment > 0, "agent not assigned to layer");

    // promote the agent to the next layer
    if (_scores[_round][agent] >= _upper[assignment] && assignment !=
        _layers.length) {
         // assignment in the next round
         _assignments[_round + 1][agent] = assignment + 1;
    // demote the agent to the previous layer
    } else if (_scores[_round][agent] <= _lower[assignment] && assignment >
        1) {
         // assignment in the next round
         _assignments[_round + 1][agent] = assignment - 1;
    // agent layer remains the same
    } else {
         _assignments[_round + 1][agent] = assignment;
    }

    emit Update(agent, _rewards[action], _scores[_round][agent]);

    return true;
}
```

Figure 5.9: The *update* function of `Balance` implemented in Solidity. An agent's score is updated if it performs an action.

university to choose could use a decentralized publicly voted version of a top 100 university list instead of choosing from a range of centralized lists.

Candidates are agents who want to enlist an element n in R. To register an element, the candidate has to provide an application including collateral used to vote whether or not to include the element into the list.

Members are agents that own one or multiple elements n in R. Depending on the TCR type, their interests can be different. Generally, we assume that members try to optimize the ranking (if any) of their elements to remain part of the registry.

Curators own tokens that are used to vote on accepting elements of candidates, ranking of elements already in the registry, or challenging existing elements. Their objective is to maintain a "good" registry. However, it is not commonly defined what a "good" registry is as this is entirely subjective.

Types of TCRs *Unordered TCR* A set R_U, where each element n was voted into the set by a range of agents P. An agent needs to provide collateral to trigger a voting procedure to include an element in the registry. Upon application, any curator of the list can challenge the contribution providing collateral. If the application is not challenged, n enters R_U. Otherwise, a vote is cast by all curators. The winner of the vote gets a share of the provided deposit by the other party, and his deposit is returned. The voters receive the other share of the deposit [Mik17].

Ranked TCR A set R_O, where each element n_i was voted into the set and given an index i representing its position. This leaves a set with order n_i n_{i+1}. Agents apply to include elements n to the registry in the same way as in unordered TCRs. However, a ranking is determined by a mechanism including, for example, pairwise voting by the curators of the list, a ranking during the application process, or a curation market [Rou17; Rud18].

Graded TCR A set R_G, where each element n_i has a specific weight w. This leaves a partially ordered set where elements might have the same weights n_i n_{i+1}. Elements n is again added by voting. Weights are determined by voting and introducing order to the TCR [Gaj18]. The weights can be used to combine the TCR with, e.g., bonding curves.

Layered TCR A set R_L, consisting of distinct subsets where $\bigcup_{i=1}^{n} R_i$ R_L $\bigvee$ $\bigcap_{i=1}^{n} R_i = \emptyset$. Elements n are added by voting. However, in layered TCRs, any new element is added to the lowest layer. A set of rules defines which elements can progress to the next layer. Similarly, there exists a set of rules that determines if an element is removed from a certain layer or the registry altogether [McC18].

The TCRs described above can be combined to form nested or combinatorial TCRs. For example, the layered TCR could consist of R_U or R_G or a mixture thereof.

Parameters TCRs can be parameterized through the *minimum deposit* any agent needs to pay for suggesting an element for application, the *application period* an element can be challenged, a *commit period* in which agents can commit to a vote, and a *reveal period* in which agents can present their votes. Next, a *bonus* can be determined for the winning agent of a

challenging process, and last, the *voting quorum* that decides the outcome of a challenge. For example, the bonus could be all staked deposits minus transaction costs, and a voting quorum could be a simple majority or a 2=3 majority vote.

Limitations of TCRs The incentive design for voting a new element into the list can be challenging. Ideally, maintainers of a list are interested in having a "good" list and would therefore vote to include elements that are deemed worthy. Previous analysis on TCRs shows limitations by employing a game-theoretic analysis [AK18]. First, during the bootstrapping of a TCR, little to no comparison between the elements is possible. There is a chance that low-quality elements enter a registry close to its inception. Next, the authors show that unordered TCRs have two Nash equilibria, where all voters vote either "accept" or "reject" on challenged items. Further, they show that the likelihood of a candidate element entering the list depends on the ordering of arriving candidates. The fewer items already on the list, the higher the likelihood of a new candidate entering. Finally, TCRs are little used in practice at the moment. This might arise from the issues mentioned above and their yet unclear added benefit. The most significant obstacle for adoption is the potential to game the consensus-by-voting mechanism. TCRs might also suffer from the fact that curators are not voting on elements. An inflation mechanism can encourage curators to vote by incentivizing active and informed voters [WK18].

5.6.2 Reputation Systems

The second strand concerns *reputation* aspects of Balance. Yu et al. use a notion of reputation to define a miner's power in terms of its work performed over the lifetime of a blockchain, as opposed to instantaneous mining power, in order to mitigate the vulnerability of blockchains to 51% attacks, where an adversary momentarily possesses more than 50% of the total mining power [Yu+19]. Another system for reputation management in decentralized systems, but this time for users of the system as opposed to miners, is [LZ17]. The mechanism uses cryptocurrencies to measure trust: using deposits between different agents, the authors construct a web-of-trust-like architecture. Balance is different, creating only direct trust

relationships between agents and actions are directly evaluated through the agreements in a smart contract. Taxonomies of reputation-based systems [JIB07; PS13; HBC15] indicate comparators for Balance and other reputation-based systems. Balance is a *quantitative* as opposed to qualitative trust system, expressing a reputation in terms of a deposit factor. Reputation itself is accrued by agents through direct experience, i.e., direct interaction with a smart contract. In particular, there is no transitive reputation between peers: the trust that *others* place in an agent does not confer trust onto the agent [7]. Some systems such as Pisa [McC+18] allow for transitive trust, allowing a reputation to be indirectly established. There is considerable literature in economics concerning reputation and trust (e.g., [FLM09; MR82; KW82]), but the authors are unaware of any work focusing on the economic aspects of reputation formation concerning cryptocurrency deposits.

5.7 Conclusion

Balance is an application-agnostic system, intended as an extension to existing cryptoeconomic protocols, that allows for the reduction of cryptocurrency deposits without compromising security. By explicitly modeling agents' utilities, we show that it features an incentive-compatible mechanism that rewards agents for the performance of desired actions by reducing their required deposits, and therefore the opportunity costs of those deposits. Moreover, we show that the addition of Balance increases social welfare.

The primary motivating force for agents in our construction is the expected reduction in opportunity costs resulting from forgone returns on deposits. If we modify the assumption of perfect competition in Section 5.3, such that agents receive a positive payment from performing the desired action, payments p could also constitute a motivating force. For protocols where a reduction in the deposit is not practical or not desired by the protocol designers, the factor in each layer could be used to calculate the payment. In this case, the factor would increase with every layer so that agents in the lowest layer receive a payment of pf_1, where, e.g., $f_1 = 0.6$.

[7] One caveat is that in one sense trust is transitive across agreements since an agent may participate in multiple agreements within the same smart contract, retaining the same deposit factor.

To the best of our knowledge, this work is the first to use a curated registry to enable dynamic deposit requirements. One question that arises is that given v is private information, at what level should deposits and factors be set such that the proportion of agents who find it incentive-compatible to perform desired actions is optimal for the protocol (or society) as a whole? In addition, we plan to explore different parameter configurations such as overlapping boundaries and restricting the number of agents per layer, as well as extend the model to cover probabilistic formulations of the specification.

Chapter 6

Applications of Promise and Balance to XCLAIM

6.1 XCLAIM

XCLAIM is a generalized protocol that allows the exchange of cryptocurrencies across different blockchains without a trusted intermediary [Zam+19]. The protocol employs third parties that provide collateral to participate in the protocol. Further, XCLAIM reduces the cost of atomic cross-currency swaps compared to atomic swaps realized with hashed time-lock contracts (HTLC). We choose XCLAIM as an example of applying `Promise` and `Balance` as it is subject to exchange rate risk (event-dependency) and heterogeneous valuations of individual cryptocurrencies (private information).

6.1.1 Protocol Overview

XCLAIM is divided into three sub-protocols. First, the *issue protocol* enables the creation of cryptocurrency-based assets (CbAs) from a backing chain onto an issuing chain. Second, in the *swap protocol*, senders and receivers on the issuing chain can exchange the issued CbAs. Finally, in the *redeem protocol*, a redeemer exchanges his CbA for the original coin on the

backing chain. For example, Alice could issue Bitcoin-backed tokens on the Ethereum chain. Alice could then send her Bitcoin-backed tokens to Bob on Ethereum in exchange for Ether. Bob can take his Bitcoin-backed tokens to receive the equivalent amount of Bitcoins on Bitcoin. XCLAIM includes six roles.

In the example, Bob takes the first four roles (issuing and redeeming backed tokens) and Alice has role five (the Vault).

1. *CbA Requester*: creates a backed token $i(b)$ on an issuing chain from locking a coin b on the backing chain.
2. *CbA Sender*: sends a backed token $i(b)$ on the issuing chain.
3. *CbA Receiver*: receives a backed token $i(b)$ on the issuing chain.
4. *CbA Redeemer*: destroys the backed token $i(b)$ on the issuing chain to redeem the coin b on the backing chain.
5. *CbA Backing Vault (Vault)*: a non-trusted third party enabling the issue protocol and executing the redeem protocol. The CbA Requester locks her coins b with the Vault while the CbA Redeemer upon the destruction of $i(b)$ redeems b from the Vault.
6. *Issuing Smart Contract* (`iSC`): implements the sub-protocols on the issuing chain.

The Vault only becomes active during the redeem sub-protocol as any actions in the issuing sub-protocol are executed by the CbA Requester and the `iSC`. To ensure correct behavior, the Vault has to provide collateral on the issuing chain, allowing the issuing and redeeming of CbAs. The `iSC` serves as an escrow for the collateral of the Vault. The collateral is a promise by the Vault to release the backing tokens b when a valid redeem request is made by destroying $i(b)$. Under the assumption of an economically rational Vault, the Vault releases b as otherwise its collateral is refunded to the CbA Redeemer. Further, the `iSC` verifies the correct execution of the issue and redeem process by verifying transactions occurring in the backing chain on the issuing chain via a chain relay. An overview of the protocol is given in Figure 6.1.

Vault agreements

Roles: A mapping from XCLAIM to `Balance` and `Promise` roles.

1. **Performing agent** A: Vault
2. **Receiving agent** B: CbA Requester and CbA Redeemer
3. **Registry** R: Integrated in the `iSC`
4. **Escrow and Verifier**: Integrated in the `iSC`

Events: An update to the exchange rate at time t changes the required ideal collateral dC_Ae and bC_Ac. The current collateral C_A might fall below dC_Ae or bC_Ac.
Private information: A might value a currency higher such that even if $C_A > bC_Ac$, A behaves economically rational when performing an undesired action.

A_1: **Issue commitment**

True, if A provides a collateral C_A dC_Ae or **False**, if $C_A < dC_Ae$.

p If of $A_{1;4}$ is true, A receives p_A paid by B in its issue request.

C A's collateral C_A is refunded to A if of A_1 is false.

A_2: **Redeem request** (*Precondition:* of A_1 is true.)

True, if A fulfills a *valid* redeem request by B within a time w or **False**, if A fails to provide a proof of executing B's request within w.

p If of A_2 is true, A receives payment p_A[a].

C A's collateral C_A is transferred to B if of A_2 is false.

A_3: **Liquidation** (*Precondition:* of A_1 is true and asset price update by oracle O).

True, if C_A is above bC_Ac or **False**, if C_A is below bC_Ac.

p There are no payments for performing this action.

C A's collateral C_A is used to perform a redeem request as stated in A_2.

A_4: **Buffered collateral** (*Precondition:* of A_1 is true and asset price update by oracle O).

True, if C_A is above dC_Ae or **False**, if C_A is below dC_Ae.

p There are no payments for performing this action. A cannot participate in the issue protocol while is false.

C There is no impact on the collateral.

[a] XCLAIM does not de ne how payments are made during redeem (cf. Section VII-F [Zam+19]). We assume that a fee is already paid during issue.

Figure 6.1: An overview of the agreements of a Vault in XCLAIM.

6.1.2 Collateral Dilemma for XCLAIM Vaults

Vaults in XCLAIM provide collateral in the currency of the issuing chain to insure against risk on the backing chain. Exchange rate fluctuations are external events that affect the security assumption of a Vault fulfilling a redeem request. Sudden drops in the exchange rate result in insecure protocol states where the provided collateral is less than the value of the coin b on the backing chain. Hence, rational Vaults are incentivized to refuse redeem requests even under the threat of having their collateral taken away. Further, Vaults have a private valuation for the outcome of the redeem protocol. If a Vault values the backing coin b higher than the required collateral, the Vault has an incentive not to fulfill the redeem request. XCLAIM mitigates exchange rate fluctuations and, implicitly, the potentially detrimental effects of private information through staged collateral.

1. In *secure operation*, a CbA Requester can issue new CbAs as C_A is greater than the ideal collateral $\lceil C_A \rceil$.

2. In the *bu ered collateral* stage, $C_A < \lceil C_A \rceil$ but C_A is still above a lower bound collateral $\lfloor C_A \rfloor$. In this stage, a CbA Requester cannot issue new CbAs based on this Vaults collateral. The Vault can provide additional collateral to increase its collateral rate back to the ideal rate.

3. In the *liquidation* stage, a Vault's collateral is automatically liquidated if it falls below $\lfloor C_A \rfloor$. The collateral rate is *dynamically* adjusted using an oracle O importing the exchange rate.

6.2 Integrating Balance

The objective of Balance is to reduce collateral up to the boundary f_l. Integrating Balance is, therefore, an exercise in determining external factors and choosing the parameters such that f_l can be optimized. This process includes the following steps:

1. Determining the return rate r includes finding an approximate value that agents could earn by participating in protocols other than .

2. Determining the discount factor comprises of finding the discount agents apply to future returns.

3. Choosing the scores s in A needs careful consideration of the agreement. Reducing collateral introduces a positive incentive for agents. Therefore, only agreements that already have a positive incentive, e.g., payments, should be associated with a positive score.

4. Choosing a period length t depends on how often a protocol designer expects agents to interact with the protocol. Since agents need to perform desired actions continuously, they should have a long enough period.

5. Choosing the number of layers l determines (i) the overall reduction of collateral possible and (ii) how many periods it takes to reach the lowest collateral.

6. Choosing the initial factor f_1 is a trade-off between the "buffer" of over-collateralization and the amount of collateral reduction. The higher f_1, the higher the reduction possible. However, also the higher the absolute collateral will be.

We provide detail on determining external factors r and in Section 6.2.3. However, a thorough study of parameters specifically for decentralized ledgers is left as future work.

In the following, we describe in detail the application of Balance to the XCLAIM protocol.

6.2.1 Vault Agreements

We integrate Balance with XCLAIM by defining the agreements of Vaults. These include their specification, payments, and collateral, as well as the score. Further, we define the length of the period t in which an agent can perform actions. Last, we set the initial collateral factor f_1 and lower bound f_l. Our definitions are described in Figure 6.2. We leave the quantification

Parameters of Balance

Assumptions:

r 0:05. A earns 5% of its collateral by participating in other protocols.

0:9. A discounts future income by 10%.

Time constraints:

t 528 blocks (on Ethereum)

Actions: Actions depend on the agreements and have an associated score.

A_1 **Desired** $_d$: A provides $C_A \quad dC_Ae$.
Undesired $_u$: A provides $C_A < dC_Ae$.
Score s: 0. A should not receive a reduction of collateral for adding collateral in small quantities to reduce its overall collateral. By setting $s = 0$ for A_1 we prevent such behaviour.

A_2 **Desired** $_d$: A fulfils B's redeem request within w.
Undesired $_u$: A fails to execute B's redeem request within w.
Score s: > 0. A receives an additional incentive to execute redeem requests and can in return reduce its collateral.

A_3 **Desired** $_d$: If $C_A < bC_Ac$, A adds additional collateral $C_A^{\emptyset}$ such that $C_A + D_A^{\emptyset} \quad bC_Ac$.
Undesired $_u$: A does not add collateral new collateral or an insufficient amount such that $C_A + D_A^{\emptyset} < bC_Ac$.
Score s: 0. A should not receive a reward for being temporarily below the liquidation bound.

A_4 **Desired** $_d$: If $C_A < dC_Ae$, A adds additional collateral $C_A^{\emptyset}$ such that $C_A + D_A^{\emptyset} \quad dC_Ae$.
Undesired $_u$: A does not add new collateral or an insufficient amount such that $C_A + D_A^{\emptyset} < dC_Ae$.
Score s: > 0 should receive an incentive for staying above the ideal collateral, however, this might result in purposely falling under the ideal collateral.

Layers: The layer factors express the boundary for dC_Ae. We leave bC_Ac as suggested by XCLAIM at 1:05.

$!$ 12. Based on considering a time window of 24 hours and the minimum time of 528 blocks.

f_1 2:06. Based on our analysis of exchange rate and order books.

$f_!$ 1:85. Based on applying f_1 to (5.16) and considering the exchange rate and order book analysis.

Figure 6.2: An overview of the parameters for an integration of Balance into XCLAIM.

of scores for actions and determining lower and upper bounds of layers as future work. We argue that Balance does not affect the security arguments of XCLAIM in Section 6.2.4.

Determining a minimum time XCLAIM assumes a minimum number of blocks to consider transactions confirmed depending on the security parameter k [GKL15]. As outlined in Section 5.2.4, we determine the period by considering the agreement with the most transactions. Agreements $A_{1;3;4}$ depend on one issuing chain transactions while A_2 depends on one backing chain and two issuing chain transactions. We use a buffer of 2 such that the time $t = 2(k_b\Delta_b + 2k_i\Delta_i)$. For the case where Bitcoin is the backing and Ethereum the issuing chain, we assume $k_{\mathsf{ETH}} = 12$, $\Delta_{\mathsf{ETH}} = 15$ s, $k_{\mathsf{BTC}} = 6$, $\Delta_{\mathsf{BTC}} = 10$ min. This results in t 132 min or t 528 blocks on Ethereum.

Determining scores Within XCLAIM, the issue commitment A_1 is equivalent to the commit stage. We choose $s = 0$ for A_1 to prevent agents from splitting up their collateral into smaller parts. Further, we exclude a reward for A_3 resolving $C_A < \lfloor C_A \rfloor$ as ideally C_A should always be greater than $\lceil C_A \rceil$. We define a positive score for the desired action in A_2. Further, we argue that having a positive score in A_4 provides an additional incentive for A to re-balance its collateral in case of exchange rate fluctuations. In turn, this improves the chance of CbA Requesters being able to participate in the issue sub-protocol.

Determining the number of layers We determine the number of layers $!$ by considering which time period a performing agent A is accounting for in its decision to choose between Σ_d and Σ_u. We assume that A considers a 24 hour time period. Next, we calculate $!$ by dividing 24 hours by the time representation of t. Based on $t = 528$ blocks, we set $! = 12$.

Determining factors XCLAIM suggests collateral stages of 2:0 for $\lceil C_A \rceil$ and 1:05 for $\lfloor C_A \rfloor$. We leave $\lfloor C_A \rfloor$ as suggested. We focus on reducing $\lceil C_A \rceil$ for a concrete implementation of XCLAIM for Bitcoin and Ethereum. To verify the suggested collateral $\lceil C_A \rceil$, we perform an analysis of the daily exchange rate fluctuations and private valuations by parsing order books

of exchanges (cf. Section 6.2.3) for Bitcoin and Ethereum. Based on our analysis, and assuming $r = 0{:}05$ and $\quad = 0{:}9$ as well as a linear relation between layer factors, we set $f_1 = 2{:}06$ and $f_{!} = 1{:}85$. Thereby, we achieve a reduction of 10% of the collateral.

6.2.2 Currency Fluctuations and Valuations

ETH-BTC fluctuation To determine the buffer for the deposit to account for exchange rate fluctuations, we collect the daily high and low prices of BTC to USD and ETH to USD from a period of one year (3 May 2018 to 3 May 2019). The data is collected from the Poloniex exchange API[1]. We are interested in the *drop* of the exchange rate, i.e., from the high price to the low price. Therefore we calculate the drop as $\frac{p_{\text{low}} \quad p_{\text{high}}}{p_{\text{high}}}$. Using the exchange data, we find that the highest drop within the year is 17:06%, and the average per day drop is 3:90% with a standard deviation of 2:73%. The maximum daily exchange rate drop is visualized in Figure 6.3.

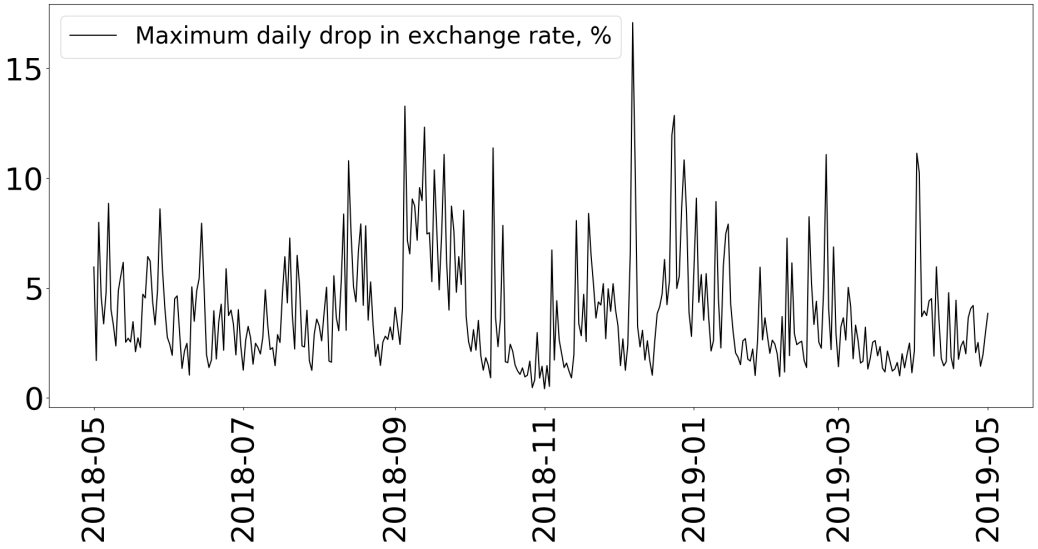

Figure 6.3: Using daily data from the Poloniex exchange, this figure plots the percentage decrease (in absolute terms) of the low price relative to the high price for a given day.

ETH-BTC valuations Agents can express a different preference for cryptocurrencies. To quantitatively approximate this difference, we scanned the order books of the Binance exchange

[1]Taken from https://docs.poloniex.com/

for bids and asks for a period of one week from 3 May to 9 May, 2019[2]. We calculated the valuation difference by comparing the bid/ask price to the exchange rate when the order is placed. We note that most agents place orders close to the actual exchange rate as the mean difference between price and exchange rate is 0:64% with a standard deviation of 2:73%. However, there are significant outliers with up to 76:34% in this data. The valuation difference serves as an orientation to calculate the security buffer for the cross-currency deposit.

We note several shortcomings of our approach. Orders placed on an exchange may not reflect the honest valuation of the agent. Agents might place extreme orders to influence the exchange rate price instead of valuing the currency at that level, and more generally, market manipulation (such as front-running and transaction re-ordering) is cause for concern [Dai+20] [3]. Further, we did not consider a *weighted* approach where the difference between price and exchange rate also considers the size of the order. Further, significant differences can also be caused by human error when entering the order.

6.2.3 Factor Calculation

We can calculate the factors taking the highest deviations from the exchange rate fluctuation and the valuation differences. We define the highest exchange rate fluctuation as $\Delta_{ex} = 0{:}1706$ and the highest valuation difference as $\Delta_v = 0{:}7634$. We calculate a buffer b_{deposit}:

$$b_{\text{deposit}} = (1 + \Delta_{ex}) \quad (1 + \Delta_v) \tag{6.1}$$

We then determine the factor f_1 by using a base deposit of 1: $f_1 = 1 \quad b_{\text{deposit}} = 2{:}06424$. We assume $r = 0{:}05$ [4], $\quad = 0{:}9$ [5], $! = 12$ and linear relationship between factors. Further, we apply

[2] Taken from https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md

[3] However, if the order is matched, an agent will need to ful ll it, providing a constraint on behavior

[4] A risk free rate of 5% approximates, for instance, the long term average rate on 10 year Treasuries of 6.14% [FRE22]

[5] [FLO02] supports a discount factor of 0.9

the buffer b_{deposit} to all subsequent factors f. We apply these assumptions and results to (5.16). This yields $f_{\omega} = 1.85$. Our results are visualized in Figure 6.4.

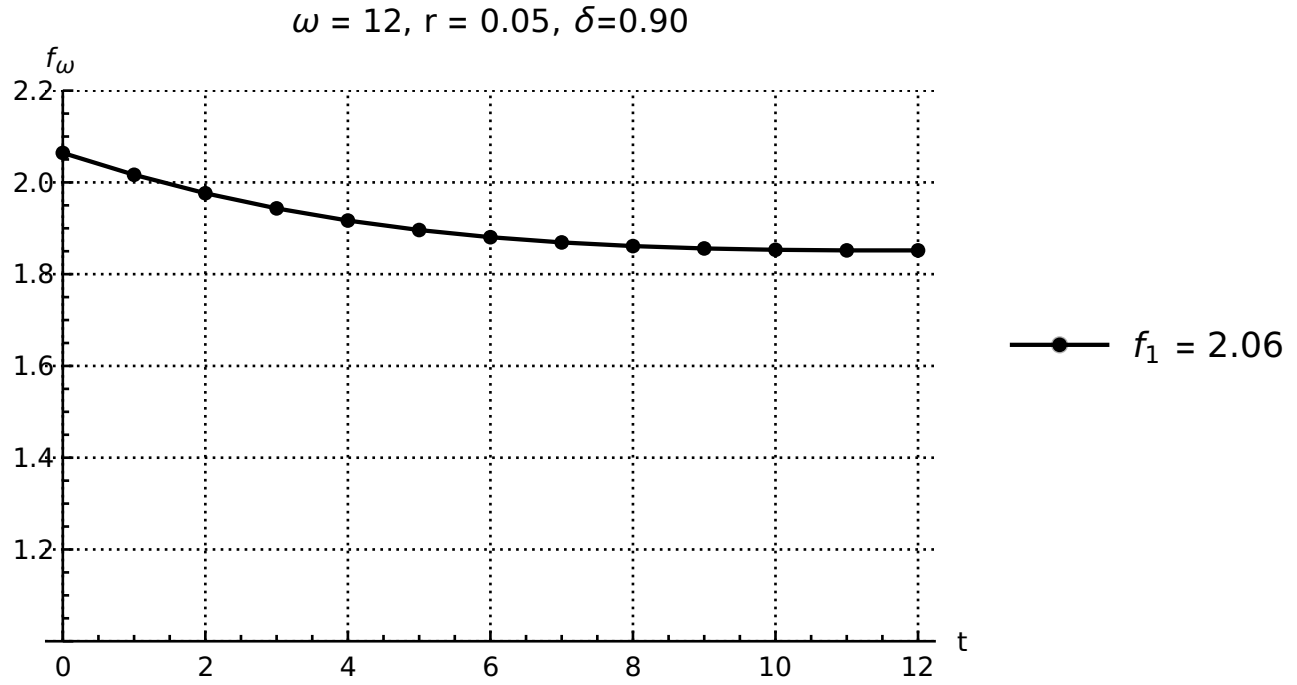

Figure 6.4: Boundary of f_{ω} for the initial factor of $f_1 = 2.06$ with an additional buffer for exchange rate fluctuations and valuation differences. The function is the decision boundary for an economically rational agent to decide between a desired and undesired action given a number of time steps t. Assuming $r = 0.05$, $\quad = 0.9$, $\omega = 12$.

6.2.4 Security Arguments

We discuss possible effects of integrating Balanceinto XCLAIM.

Vaults A take multiple roles to reduce their deposit There is no guarantee that A has an opportunity to perform the desired action within a given period t. A would thus fall to a lower layer. To prevent this, A could ensure a constant flow of redeem requests by creating an artificial demand as A can act as CbA Requester, Vault, and CbA Redeemer, called strategy S_{boosting} as described in Section 5.4.2. Assuming that expected interest on the deposit $\mathrm{E}[r]$ $C_{\text{layer}(A)}$ is higher than the cost $c_A(S_{\text{boosting}})$, this would be a dominant strategy. This leads to an issue: A could ensure that its actions reduce its deposit and perform the undesired action, i.e., not executing a redeem request of B with a lower deposit and punishment. XCLAIM can

prevent this by requiring a *maximum deposit* per Vault A. Given an expected interest Er, the maximum allowed deposit is defined by $C_{max} = \frac{c_A(S_{\text{demand}})}{\mathsf{E}[r]}$. This also holds under Sybil identities since deposit reductions are not transitive between multiple identities.

Conflicting minimum and maximum deposits XCLAIM suggests preventing Sybil attacks by either (i) requiring a minimum deposit from vaults through the `iSC`, or (ii) a fee for issuing based on the total amount and not per issue request. The minimum deposit amount conflicts with the Sybil resistance requirement for `Balance` as defined in Section 5.4.2. Hence, XCLAIM would need to adopt a fee model based on issue amounts rather than requiring a minimum deposit per Vault when integrating `Balance`. As such, Sybil resistance in both XCLAIM and `Balance` is maintained.

Length of time period t motivates agents to delay actions by a vault We assume that an agent can fulfill more than one desired action within a time period t and A has already a high enough score to progress to a higher layer or remain in the highest. In that case, A can play a strategy S_{delay} to delay performing redeem requests (A_2) and balancing its deposit (A_3) until the next period (see also Section 5.4.3). In XCLAIM's case, S_{delay} is limited by enforcing time limits on agreements $A_{2;3;4}$. Thus, vaults do not delay their actions.

Artificial redeem requests Since the score for the redeem request A_2 is positive, A might try to obtain $i(b)$ to execute redeem requests to maintain its ranking. This would require that the cost for redeem is lower than the opportunity cost of remaining at a higher layer. This argumentation is similar to Section 5.4.2 as a maximum deposit will prevent a vault from having a higher gain in opportunity cost compared to the cost of executing redeem requests.

Deposit adjustments to improve score The score to resolve a state where $C_A < \lceil C_A \rceil$ is positive. As such, A has an incentive to let its deposit fall below $\lceil C_A \rceil$ and resolve this state by adding more deposit, expressed by strategy $S_{\text{deposit-bouncing}}$. By having a positive score, `Balance` encourages $S_{\text{deposit-bouncing}}$. However, we can prevent the impact of this strategy by

setting the score for such an action to be below a lower bound of a given layer L_m. For example, XCLAIM might allow A to improve its score with $S_{\text{deposit-bouncing}}$, but sets the score such that A is only able to reach a layer $L_m \quad L_{!}$. Thereby, it *limits* $S_{\text{deposit-bouncing}}$ but does not prevent it.

6.3 Integrating Promise

In this section, we apply Promise to the XCLAIM protocol. We show analytically how Promise is able to reduce the initial amount of locked collateral.

Promise can be applied such that the Vault, Alice, locks some initial collateral C_I and issues backed-tokens using this collateral. Bob, using the service, can lock the future payments of Alice to allow him to transfer more assets between the ledgers.

Initial Parameters XCLAIM uses an initial set of parameters as follows:

- Initial Collateral C: A service provider needs to provide 200% collateral C in comparison to the value held in custody V_A.
- Payments p: Although payments are not specified in the original XCLAIM paper, similar services such as tBTC require users to pay 0.9375% of fees as payment[6].
- Rate of return r: A service provider needs to lock collateral in the ETH currency to participate. Possible alternatives offer a maximum of 3.75% APR rates[7].
- Discount factor : Service providers can discount future payments. As the price of cryptocurrencies is relatively volatile, we adopt a strongly discounted future income at 0:75 from [Har+19].

[6] Based on https://docs.keep.network/tbtc/index.pdf from 3 May 2020.
[7] Based on https://www.coingecko.com/en/earn/ethereum from 3 May 2020.

Integrating Promise For example, we are using BTC as the issuing currency and ETH as the backing currency. This means that the Vault, the service provider, has to lock ETH as collateral to provide security against stealing BTC it holds in custody. Given the parameters, Promise can be integrated as follows.

1. The user and a Vault agree on a subscription period m. For example, the user and the Vault can agree that the Vault will be responsible for the subsequent ten Bitcoin-backed tokens issued or redeemed by this user that are each 1 BTC in size.

2. The user and the Vault set up a Promise contract in which the user pre-pays $m = 10$ fees at 0.9375% of 1 BTC for the following ten requests at a set price of $p = 0.009375$ to issue or redeem Bitcoin-backed tokens.

3. The Vault then provides the initial collateral C_I into the contract.

4. Each time the user issues or redeems tokens with the Vault, the Vault is allocated a part of the payment p.

5. Finally, after ten requests have been made, the Vault can withdraw pm and C_I.

Cost Reduction For simplicity, we are going to denote all monetary amounts in BTC. In XCLAIM, a Vault would have to provide the equivalent of 2 BTC in collateral to hold custody over 1 BTC in value. First, we calculate the possible C_I collateral given Eq. (4.6). This gives us the minimum collateral required to also protect against a *Vault that plays a single-shot game*. For simplicity, we are assuming that the Vault has an incentive of 1 to steal the BTC (the current value of the Bitcoin) as well as a hidden motivation to steal BTC such that $V_A = D = 2$. Moreover, the Vault is not interested in any future collaboration with the user, hence $(n) = 1$. Last, we divide the 3.75% APR through 365 days to get the average return.

$$
\begin{aligned}
&= p \quad \mathrm{E}[r]p + D \qquad (n)V_A \\
&= 0{:}009375 \quad \frac{0{:}0375}{365}0{:}0093750 + 2 \quad 2 \\
&= 0{:}00937404
\end{aligned}
\tag{6.2}
$$

Second, we are using Eq. (4.7) to explore the reduction factor if the *Vault plays a sequential game*. Note that, at a minimum, a Vault has at least a private value of V_A to not follow the specification of XCLAIM: if the Vault can take the 1 BTC and is punished with less collateral C_I being taken away, it is incentive compatible for the Vault to take the 1 BTC. Using the example values above, we calculate as:

$$
\begin{aligned}
&= \sum_{t=0}^{\infty} \left(\frac{}{1+r}\right)^t (p \quad (t+1)\mathrm{E}[r]p \quad \mathrm{E}[r]D) \\
&= \sum_{t=0}^{\infty} \left(\frac{}{1+r}\right)^t (0{:}009375 \quad (t+1)\frac{0{:}0375}{365}0{:}009375 \quad \frac{0{:}0375}{365} \quad 2)
\end{aligned}
\tag{6.3}
$$

Discussion We plot the results from Eq. (6.2) and (6.3) in Figure 6.5. The collateral reduction can be subtracted from C. In practice, the user and the service provider can agree on the desired reduction. We note three findings: (i) If the user and the service provider want to maintain security w.r.t. no additional incentive for the service provider to violate the specification, the maximum reduction is given by the single-shot game reduction from Eq. (6.2) (the orange line in the Figure). (ii) Note that the main reason that the single-shot reduction is comparably low since the user has a "buffer" of 1 unit of BTC that was added to V_A. If the user is willing to accept a lower buffer, say 0.5, the collateral can be consequently lowered. This would still cover the value of the 1 BTC in our example plus a 0.5 potential malicious intent on the Vaults side. (iii) The user and the service provider can agree to lower the initial collateral C_I in a sequential game setting if they agree on a longer period m. However, the collateral reduction is finite: as $m \; ! \; 1$, stabilizes to a constant value.

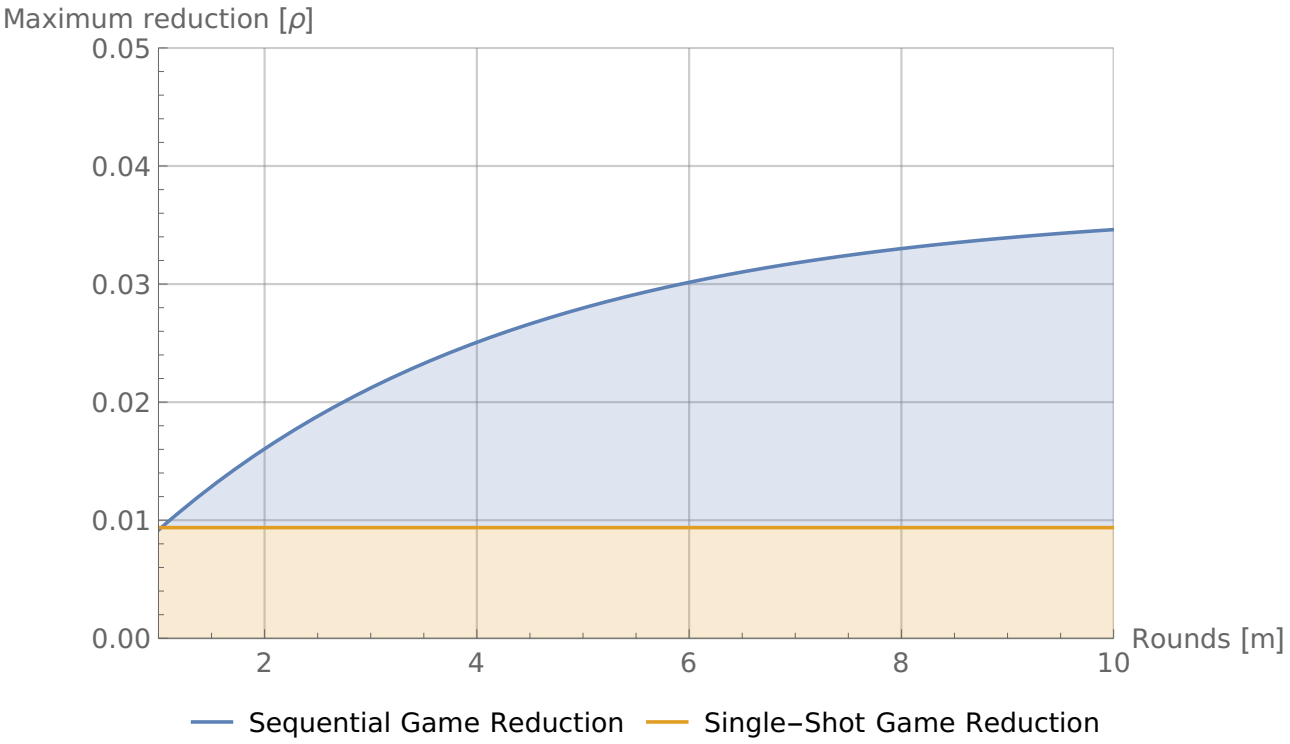

Figure 6.5: The possible collateral reduction assumes that the Vault considers a single round of execution (i.e., a single-shot game) as depicted in the orange line or that the Vault considers a sequential game with multiple rounds as depicted in the blue line. The colored areas show in which collateral reduction ranges the Vault does not receive an additional incentive to violate the specification as agreed with the user. The more rounds m the game last, the higher the collateral reduction can be under the sequential game scheme. Collateral reductions are constant in case the Vault only plays a single-shot game.

Chapter 7

Attacks

Employing financial collateral to secure DeFi protocols is not without risk. In this chapter, we explore practical attack scenarios focusing on governance. In particular, we outline a possible attack scenario on the Maker [Mak21a] protocol. One of the techniques, crowdfunding governance tokens, to execute the attack had been discovered before in a blog post [Zol19] whereas the other attack technique, flash-loans, was introduced as part of our work [Gud+20b]. As part of the discovery of the attack, it was disclosed to Maker in February 2020.

We note that both crowdfunding and flash-loans can be combined as a technique to execute similar attacks. We also emphasize that these techniques can attack other parts of DeFi protocols and are not limited to the governance system itself. However, as governance can control the entirety of the protocol rules, including the controls to safeguard locked capital, a successful governance attack allows gaining control over the entire protocol.

7.1 Governance as a Target

DeFi protocols evolve. Steering the development of DeFi, patterns have emerged to introduce change to otherwise immutable smart contracts [Yer17; ROH16]. A common technique to decide which changes should be applied to a protocol is to let the community of the protocol

vote on the changes. Governance tokens thereby represent voting power. These governance tokens represent partial ownership of the network and, similar to shares in a company, allow agents to vote on proposed changes. For a more detailed background on different governance systems, we refer the reader to Ch. 2.

The crucial part is that whoever controls governance can control any changes to the core system with vast consequences. Examples include changing the share between trading fee distribution between LPs and the core protocol in Uniswap [Uni21], rules that control minting in Dai [Mak21a], or the collateral threshold in PLFs such as Compound [Com21a] and Aave [Aav21]. One can imagine that such comprehensive control makes governance a desirable target for attacks.

7.2 Attack Background

Maker is a protocol that allows agents to lock collateral (e.g., ETH) to mint a stablecoin Dai that is 1:1 soft-pegged to the USD. Following a simple majority voting system, the Maker protocol employs the MKR token to vote on system changes. Agents are required to lock these MKR tokens to be able to vote. An agent can increase or decrease its voting ability by selling or buying MKR tokens from exchanges.

Maker's core governance system is the *executive contract* that controls rules to the core Maker system, including the constraints that concern locking of collateral and minting of new Dai. A malicious executive contract can exploit the Maker system in two ways [Har20].

1. Withdraw locked collateral: The current rule set allows to withdraw collateral if the Dai loan (i.e., the generated Dai) is paid back with interest. Without this requirement, the malicious executive contract could allow a dedicated address to withdraw any locked collateral. As of the discovery of the attack, USD 700mm in assets could have been removed by a successful attacker. As of December 2021, this amount has increased to USD 18.47bn.

2. Mint arbitrary amounts of new Dai: New Dai can be generated if another asset backs it as collateral. However, a malicious execute contract could disable this rule and mint arbitrary new amounts of Dai. Since Dai is widely accepted on exchanges, a successful attacker could mint the new Dai and sell it for other assets on exchanges before the price of Dai would crash due to unlimited supply. Hence, this attack would not be isolated to the Maker system but spread to other DeFi systems. As of December 2021, the 24-hour trading volume of Dai across exchanges is around USD 350mm. Therefore, it is non-trivial to estimate the economic impact of this attack since the attacker would like to be willing to sell Dai for less than USD 1 for other assets.

Defense Strategies Maker implements two defense strategies against malicious executive contracts. First, there is a delay between the successful voting of a new executive contract and its enactment. During our attack reporting, this delay has been increased from zero to 24 hours [Mak20]. This delay should allow members of the Maker team to analyze the new executive contract and take measures against its enactment. This measure is the second defense mechanism: the so-called Emergency Shutdown. In an emergency like detecting a malicious executive contract or other severe system flaw, MKR holders can halt the entire Maker system [Mak21b]. This last-resort action should prevent more extensive damage to the Maker users but results in a complete operational stop of the Maker system and requires a re-deployment of the smart contracts.

Attack assumptions We assume that the adversary is an economically rational agent, i.e., if the execution cost is lower than the estimated income, the adversary will (try to) execute the attack. The execution cost depends highly on the applied technique and will be outlined below. The profitability stems from the impact of the attack. The two avenues are the minting of arbitrary new Dai and extraction of collateral, which puts the potential capital to gain for the adversary at en excess of USD 700mm at the time of discovery of the attack in February 2020 and at an excess of USD 18.47bn as of December 2021.

The attack requires gaining the majority of MKR tokens to elect a new executive contract. The

authors in [Gud+20b] analyzed that the MKR to vote on a new executive contract fluctuates and is frequently below 50,000 MKR. Indeed, inspecting the MKR voting tracker[1], we can see that Maker has a total of around 2,500 voters with 312 addresses being active in votes that accumulate up to approximately 52,000 MKR in the year 2021. The adversary should execute the attack when the locked MKR supply is low to be efficient.

7.3 Crowd-Funding

One way to go about the attack is to buy enough MKR over time to execute the attack. However, Maker would notice this as they track the top MKR holders, and the current top MKR holders are known [Gud+20b]. Maker could prevent such an attack by voting from its reserve.

Instead, the adversary can deploy a smart contract that suggests a new executive contract up for a vote once enough MKR has amassed [Zol19]. In this scenario, adversaries can coordinate trustlessly to execute the attack: If enough MKR is locked in the contract, anyone can receive a profit, which can be encoded in the malicious executive contract.

7.4 Flash Loans

Flash-loans are a new concept that was first introduced on the Ethereum blockchain. First introduced by dYdX [DYd21] and later popularized by Aave [Aav21], flash loans work in three steps as described in Fig. 7.1. These steps are performed within a single transaction. This is achieved by encoding the steps into a smart contract that is then triggered to execute all steps in sequence.

1. Funds are borrowed via a loan function. This is now supported on dYdX, Aave, Uniswap, and Maker, for example.

[1] From https://beta.mcdgov.info/ (Accessed 3.1.2022)

2. Actions are executed, for example, acquiring MKR tokens or conducting trades on exchanges.

3. The borrowed funds are returned plus any interest on the loan.

Importantly, flash loans are currently the only form of loan that requires no collateral. Since the transaction's basic three steps are atomic, the loan is guaranteed to be returned.

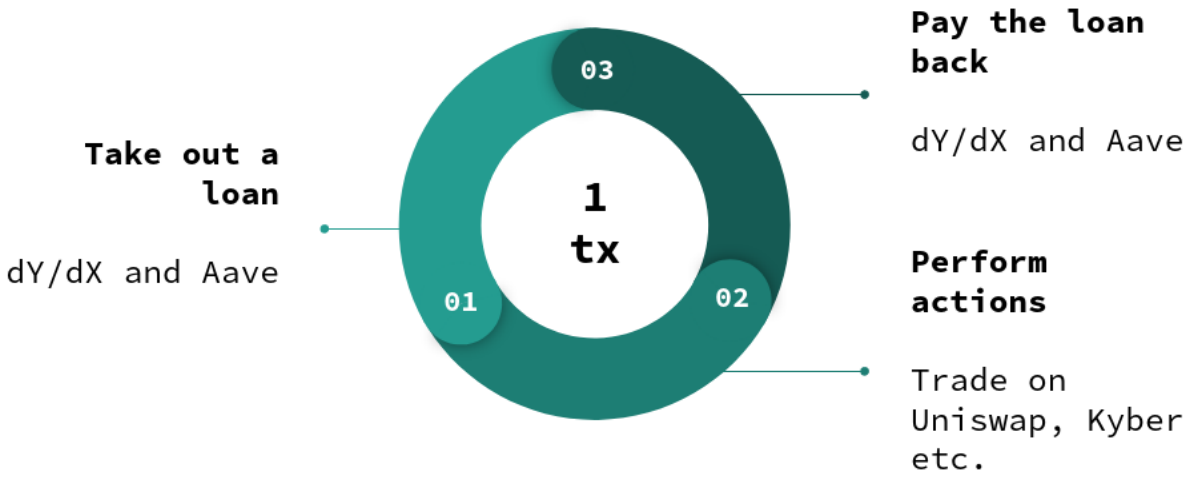

Figure 7.1: Flash loans consist of three steps: taking out the loan, executing actions, and paying back the loan.

Attacking Maker We assume that 50,000 MKR is sufficient to execute the attack. The main problem with acquiring this amount of MKR is the available liquidity on decentralized exchanges. At the time of the attack disclosure in February 2020, the most efficient way to acquire 50,000 MKR was to obtain around 38,000 MKR from Kyber, 11,500 MKR from Uniswap, and 500 MKR from Switcheo. This would have resulted in a cost of 378,940 ETH [Har20].

However, the attack could be combined with another popular flash loan attack strategy: oracle manipulation or so-called sandwich attacks [Wer+21]. In a sandwich attack, an adversary temporarily manipulates a token pair's balance in an AMM exchange temporarily. Some exchanges use AMMs as asset price oracles. Thus the adversary can manipulate the price in a single block to obtain the asset from the vulnerable exchange at a significant discount. This has been used to artificially inflate or deflate prices by up to 285% [Pec20] and used to drain USD 33.8mm from the Harvest protocol [Rek20].

Table 7.1: Overview of cost, impact, and mitigation of governance attacks.

Technique	Cost	Impact	Mitigation
Crowdfunding	High	High	Social: Preventable if majority of tokens vote against malicious change.
Flash-loan	Low	High	Technical: Preventable by enforcing a delay between voting and enactment of governance change.

However, even by combining the governance flash loan attack with a sandwiching attack, a large number of ETH must be borrowed. At the time of disclosure, only dYdX and Aave supported flash loans and had liquidity for about 107,000 ETH combined. As of December 2021, Aave has around 300,000 ETH available liquidity and the required amount of ETH is 233,000[2]. Hence, the attack would have been possible.

Profitability Since the 50,000 MKR stays locked in the executive contract, the 233,000 ETH used to buy them cannot be unlocked and must be paid back. However, the Maker contract has more than 2.6mm ETH locked as collateral[3]. Thus the minimum profit is above 2.3mm ETH, 43,000 stETH, 235mm USDC, 54,000 wBTC, and other tokens.

Feasibility After disclosing the attack to Maker, Maker has taken steps to prevent the attack from happening. In February 2020, Maker introduced a one-day delay until a new executive contract was enacted. This renders the flash loan attack strategy on Maker unfeasible. We summarize the crowdfunding attack and the flash-loan attack in Table 7.1.

7.5 Discussion

Endogenous and Exogenous Assets When using collateral, exogenously priced assets might be preferred when choosing between a highly liquid and an illiquid asset. When an agent wishes to liquidate the collateral asset, it will have a higher chance to achieve this without

[2]From https://dexindex.io/?symbol=MKR&amount=50000&action=buy (Accessed: 3.1.2022)
[3]From https://daistats.com/collateral (Accessed: 3.1.2022)

decreasing N. Conversely, using an exogenous asset as a governance token would not be preferable. When an attacker seeks to acquire the majority in a given system, having a relatively illiquid asset is *an advantage* since adversaries need to pay a high premium to receive the tokens. An excellent example of this is Curve [Cur21]. In Curve, agents have to lock governance tokens to vote. Curve combines this with an innovative incentive model. Governance decides which liquidity pools receive CRV token rewards. Thus, other protocols compete to acquire CRV tokens to influence which liquidity pools receive the rewards. They can use CRV instead of their governance token to promote liquidity for a pool. This scarcity of the governance token makes it hard for a potential adversary to acquire sufficient tokens for a hostile take-over of the Curve protocol.

Optimistic Governance and Base Layer Risk Due to the "lazy-voter problem", selected DeFi protocols adopt an optimistic governance approach [Ara21]. Here, a governance decision is assumed to be trusted unless the community vetoes it. Optimistic governance opens up interesting new possibilities: For example, miners might be motivated to censor veto votes to profit from the governance change. This attack vector is possible in a limited way since miners in a majority-based governance system can censor agents from voting because a change would not pass. Optimistic governance increases MEV.

Enactment Delays and Technical Security Enactment delays introduce a trade-off: the initial reason for having zero enactment delay in Maker was to fix security-critical bugs as soon as possible. As the smart contract cannot semantically distinguish between security fixes and other changes, all changes must follow the same process. Hence, giving the users time to examine a governance change by the enactment delay is at the same time a delay in which a critical vulnerability might remain open. Polkadot uses a technical committee for this purpose where elected agents can fast-track governance changes or introduce emergency fixes [Web21].

Forkless Upgrades and Collateralized Governance Governance is the most vulnerable part of the system if the system assets can be accessed using governance functions and users

cannot opt out. When systems hard-fork, like Uniswap, users have to opt-in to the changes by moving their liquidity. While this fragments liquidity across different versions of a protocol, users can switch when they deem the new system safe under the notion of technical risks. On the other hand, forkless systems subject the entire user base to governance risks. Maker has an emergency shutdown mechanism in which MKR tokens are sold to reimburse users in a critical system failure to alleviate this risk. However, the value of the governance token must be higher than all collateral locked in the system. In the long term, the governance token value of a system cannot be higher than its locked liquidity. The value accrual in governance tokens like MKR results from (i) the fees accrued in the protocol and to which MKR holders are entitled and (ii) the excess demand for acquiring MKR as an investment. When the expected value of MKR remains constant, (ii) goes to zero. As the fees cannot be the same as the total liquidity locked in a protocol, MKR or other governance tokens are not valued as much as the total liquidity within its protocol, looking at an infinite time horizon.

Chapter 8

Conclusion

8.1 Summary of Thesis Achievements

We introduce the notion of modeling DeFi protocols as cryptoeconomic systems to reason about agent incentives and strategic behavior. We deduct a definition of economic security based on strategic agent types that have a higher utility from performing desired actions in a DeFi protocol. In particular, we use a worst-case scenario assumption to simplify reasoning about systems without applying probabilistic models. The cryptoeconomic model is enhanced by classifying risks for DeFi, allowing us to build a risk-based model for collateral usage in DeFi. We outline the main risks associated with different types of collateral, i.e., exogenous, endogenous, and implicit collateral, and how they affect DeFi security assumptions. This model generalizes to collateralized DeFi protocols empowering protocol designers to reason about the economic security of their systems.

The model forms our starting point for designing two new protocols to improve the security and efficiency of collateral in DeFi systems: `Promise` and `Balance`. We show that applying `Promise` can lower initial collateral deposits by 1% to 3.5%. In turn, `Balance` can decrease collateral in XCLAIM by 10%. Other systems might save as much as 20% collateral. Moreover, integrating `Balance` leads to increased economic security against rational agents with *the same collateral level* by transforming single-shot games into sequential games. These two protocols,

especially Balance , form the heart of this thesis. If Balance was applied in DeFi, it could free up to USD 4bn in the Aave, Compound, and Maker protocols alone.

Last, we discuss the practical pitfalls of using collateralized systems. We identify governance as the primary target for attacks as it allows to change the rules of a DeFi protocol drastically. In the flash-loan attack scenario we disclosed to Maker in February 2020, an adversary could have stolen around 55,000 ETH, 50,000 MKR, and mint an arbitrary amount of Dai at the time of the disclosure. We informally outline new attack vectors and discuss the trade-offs of using exogenous and endogenous collateral in different cases.

8.2 Future Work

We outline three strands of future work.

Composable Collateral Adjustments Balance (Ch. 5) introduces a collateral adjustment mechanism integrated into a single protocol. In follow-up work, we present Byzantic, a reputation system with *transitive* reputation for agents across multiple protocols [Sav20]. Reputation is exported from a single protocol to several other protocols. Concretely, an agent's position in a layer in $_{1;\text{Balance}}$ is inferred from the position of the same agent in another protocol $_{2;\text{Balance}}$. As DeFi protocols seek to maximize liquidity, lowering collateral based on interactions with several protocols would allow agents to increase their borrow rates. In turn, this leads to higher fees earned.

DeFi protocols can decide to what degree, i.e., a quantifiable number, they trust the reputation reported by other protocols. Concretely, PLFs such as Compound and Aave could integrate Byzantic to give specific agents different minimum collateralization thresholds based on the actions performed in both protocols. The trust level can be encoded via a parameter that governance can change. The token holders of each protocol can decide to which degree they trust and find the other protocol's layering system applicable to their protocol.

In Byzantic, we analyze three lending protocols, Compound, Aave, and Synthetix. In particular, we looked at the depositing of collateral, borrowing of funds, repayment of funds, and liquidations. We found that we can achieve the same amount of liquidations by reducing the collateral by the 10% most active users responsible for 85% of the transactions with the protocol by 30% on average. Moreover, in this work, we introduce dynamic adjustment of the collateral reduction depending on market conditions. While collateral reductions in `Balance` are static, Byzantic integrates a mechanism to adjust them autonomously. Byzantic adopts the collateral reduction level based on the distribution of actions in a protocol and the overall interactions with a protocol. In price-volatile markets, Byzantic reduces the possible collateral reduction. When simulating the Black Thursday events in March 2020, the maximum collateral reduction shrank from 30% to 3% and recovered after the market crash to its previous state.

Composition of Economic Security in DeFi DeFi protocols often lack security proof and rely on the assumption that agents behave economically rationally in the face of uncertainty. Reasoning about the economic security of a single protocol is achievable with the risk model introduced in Ch. 3. Other models, like the agent-based modeling in [Fu+21; Rei+21; Leo+21] and more specific models like , can be used[KM19; Kla+20]. Moreover, other models might be applied with some limitations as they do not capture rational agents [CRR11], assume perfect information [Gar+13], or require complement-free valuations [ST13].

However, it is already non-trivial to reason about the security of a single protocol considering the often vast parameter space. In [Kla+20], different models are thus suggested to reason about system parts, like considering issuance of stablecoins under considerations of changing parameters. Extending this reasoning across protocols will increase the effort put on a protocol designer.

An exciting avenue for future work is extending the current models to allow reasoning about DeFi as a composition of systems. Similar to how Universal Composability [Can01] is used to "plug-and-play" security proofs, a standardized way to model and reason about composable DeFi protocols is desirable. Applications of this can be (i) reasoning about the economic security

of, e.g., staked assets like stETH [Lid21] as collateral in lending protocols or stablecoins, (ii) impact of liquidations across different lending protocols that might have an impact on each other due to borrowers holding leveraged positions, or (iii) pricing of insurance contracts against composition failures of DeFi protocols.

Applications of Balance Last, Balance can be applied in practice to DeFi protocols. We suggest adjusting the system model to make this a reality. First, Balance assumes perfect competition among agents such that payments are zero. This assumption is too conservative in practice as protocols with zero payouts for the performing agent would not be used. This additional motivating force to participate in the protocol leads to further collateral reduction. Second, we assume that generating a new identity carries no cost for the agent. However, most DeFi protocols require costly transactions to participate in the protocol in the first place. In December 2021, users opening a Maker vault to issue stablecoins have a cost of around USD 60 per transaction. Hence, keeping the same identity is favored by agents. Third and last, the main problem of applying Balance to generic DeFi protocols is that curating agents at the moment depends on agents interacting with the protocol regularly. However, this might not be the case, e.g., for lending protocols where regular interaction is not required unless the collateralization of the agent is close to the liquidation threshold. Hence, Balance could be transformed into an asynchronous protocol. Instead of relying on the actions within a given period, Balance can curate agents based on the sequence of actions without accounting for a time component. Asynchrony allows agents to stay at specific layers without continuously interacting with a protocol. On the other hand, DeFi protocols might need to create certain conditions where agents might be demoted from their reputation. We may also note that the implementation restrictions on Ethereum introduce limitations. These restrictions are lifted on systems like Polkadot, which allow scheduling actions executed regularly by miners. Moreover, Ethereum 2.0 might introduce new programming models that ease the implementation of Balance.

Bibliography

[Aav21] Aave. *Aave - Dashboard*. 2021. url: https://app.aave.com/%7B%5C#%7D/markets (visited on 12/30/2021).

[Aiy+05] Amitanand S Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. "BAR fault tolerance for cooperative services". In: *ACM SIGOPS Operating Systems Review* 39.5 (2005), p. 45.

[AK18] Aditya Asgaonkar and Bhaskar Krishnamachari. "Token Curated Registries - A Game Theoretic Approach". 2018. url: http://arxiv.org/abs/1809.01756.

[AKN19] Guillermo Angeris, Hsien-tang Kao, and Charlie Noyes. "An analysis of Uniswap markets". In: *Cryptoeconomic Systems Journal* (2019), pp. 1–25.

[AKW19] Georgia Avarikioti, Eleftherios Kokoris Kogias, and Roger Wattenhofer. "Brick: Asynchronous State Channels". In: (2019). url: http://arxiv.org/abs/1905.11360.

[Ara21] Aragon. *Optimistic Governance for your DAO*. 2021. url: https://aragon.org/aragon-govern (visited on 01/02/2022).

[ASB18] Mustafa Al-Bassam, Alberto Sonnino, and Vitalik Buterin. "Fraud Proofs: Maximising Light Client Security and Scaling Blockchains with Dishonest Majorities". 2018. url: http://arxiv.org/abs/1809.09044.

[Atz+17] Nicola Atzei, Massimo Bartoletti B, Tiziana Cimoli, Massimo Bartoletti, and Tiziana Cimoli. "A Survey of Attacks on Ethereum Smart Contracts (SoK)". In: *Financial Cryptography and Data Security*. Ed. by Matteo Maffei and Mark

Ryan. Vol. 10204. Lecture Notes in Computer Science July. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 164–186. url: http://link.springer.com/10.1007/978-3-662-54455-6.

[Ava+19] Georgia Avarikioti, Felix Laufenberg, Jakub Sliwinski, Yuyi Wang, and Roger Wattenhofer. "Towards Secure and Efficient Payment Channels." In: *Financial Cryptography and Data Security*. 2019. url: http://arxiv.org/abs/1811.12740.

[AZV17] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. "Hijacking Bitcoin: Routing Attacks on Cryptocurrencies". In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2017, pp. 375–392. url: http://arxiv.org/abs/1605.07524%20http://ieeexplore.ieee.org/document/7958588/.

[Bad+17] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. "Bitcoin as a Transaction Ledger: A Composable Treatment". In: *Advances in Cryptology - CRYPTO 2017*. Vol. 10401 LNCS. 2017, pp. 324–356. url: http://link.springer.com/10.1007/978-3-319-63688-7%7B%5C_%7D11.

[Bad+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 - CCS '18*. 780477. 2018, pp. 913–930.

[Bad21] BadgerDAO. *BadgerDAO Exploit Technical Post Mortem*. 2021. url: https://badger.com/technical-post-mortem (visited on 01/03/2022).

[Bal21] Balancer Labs. *Balancer AMM DeFi Protocol*. 2021. url: https://balancer.fi/ (visited on 12/31/2021).

[Ber18] Paul Razvan Berg. *ERC-1620: Money Streaming*. 2018. url: https://github.com/ethereum/EIPs/issues/1620 (visited on 09/24/2019).

[Ber20a] Dmitriy Berenzon. *Constant Function Market Makers: DeFi's \Zero to One" Innovation*. 2020. url: https://medium.com/bollinger-investment-group/

constant-function-market-makers-defis-zero-to-one-innovation-968f77022159 (visited on 12/31/2021).

[Ber20b] Dmitriy Berenzon. *Liquidity Mining: A User-Centric Token Distribution Strategy*. 2020. url: https://medium.com/bollinger-investment-group/liquidity-mining-a-user-centric-token-distribution-strategy-1d05c5174641 (visited on 12/31/2021).

[Ber21] Dmitriy Berenzon. *Blockchain Bridges: Building Networks of Cryptonetworks*. 2021. url: https://medium.com/1kxnetwork/blockchain-bridges-5db6afac44f8 (visited on 12/31/2021).

[BGM16] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. "Cryptocurrencies Without Proof of Work". In: *Financial Cryptography and Data Security* 9604 (2016).

[Bie21] Jonathan Bier. *The Blocksize War: The battle over who controls Bitcoin's protocol rules*. 2021.

[Böh19] Rainer Böhme. "A Primer on Economics for Cryptocurrencies". In: *School on Security and Privacy for Blockchains and Distributed Ledger Technologies, Vienna*. 2019.

[Bon+15] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. "Research Perspectives and Challenges for Bitcoin and Cryptocurrencies". In: *IEEE Symposium on Security and Privacy* (2015), pp. 104–121. url: https://www.cs.princeton.edu/%7B~%7Dkroll/papers/oakland15%7B%5C_%7Dbitcoin-sok.pdf.

[BP17] Massimo Bartoletti and Livio Pompianu. "An Empirical Analysis of Smart Contracts: Platforms, Applications, and Design Patterns". In: *Lecture Notes in Computer Science*. Vol. 10323. Cham, 2017, pp. 494–509. url: http://link.springer.com/10.1007/978-3-319-70278-0.

[BT94] Arnoud W A Boot and Anjan V Thakor. "Moral hazard and secured lending in an infinitely repeated credit market game". In: *International economic review* (1994), pp. 899–920.

[But13] Vitalik Buterin. *A Next-Generation Smart Contract and Decentralized Application Platform*. 2013. url: https://github.com/ethereum/wiki/wiki/White-Paper (visited on 10/09/2016).

[But16] Vitalik Buterin. *A Proof of Stake Design Philosophy*. 2016. url: https://medium.com/@VitalikButerin/a-proof-of-stake-design-philosophy-506585978d51%7B%5C#%7D.ebrfrh9c4 (visited on 02/20/2017).

[But17] Vitalik Buterin. "Introduction to Cryptoeconomics". In: (2017), pp. 1–47.

[But18] Vitalik Buterin. *Explanation of DAICOs - Better ICOs*. 2018. url: https://ethresear.ch/t/explanation-of-daicos/465 (visited on 01/02/2022).

[BZ18] Massimo Bartoletti and Roberto Zunino. "BitML : A Calculus for Bitcoin Smart Contracts". In: *CCS '18 Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (2018), pp. 83–100.

[Can01] R. Canetti. "Universally composable security: a new paradigm for cryptographic protocols". In: *IEEE International Conference on Cluster Computing, SFCS* (2001), pp. 136–145. url: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=959888.

[Car+16] Miles Carlsten, Harry Kalodner, Arvind Narayanan, and S. Matthew Weinberg. "On the instability of Bitcoin without the block reward". In: *Proceedings of the ACM Conference on Computer and Communications Security*. Vol. 24-28-Octo. 2016, pp. 154–167.

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

[Com21a] Compound. *Compound - Dashboard*. 2021. url: https://app.compound.finance/ (visited on 12/30/2021).

[Com21b] Compound. *Compound - Proposal Detail 64*. 2021. url: https://compound.finance/governance/proposals/64 (visited on 01/03/2022).

[Cro+16] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. "On Scaling Decentralized Blockchains". In: *Financial Cryptography and Data Security*. Ed. by Roger Dingledine and Philippe Golle. Vol. 5628. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, Nov. 2016, pp. 106–125. url: http://www.springerlink.com/index/10.1007/978-3-642-03549-4%20http://arxiv.org/abs/1311.0243%20http://link.springer.com/10.1007/978-3-642-03549-4%20http://link.springer.com/10.1007/978-3-662-53357-4%7B%5C_%7D8.

[CRR11] Ran Canetti, Ben Riva, and Guy N. Rothblum. "Practical delegation of computation using multiple servers". In: *Proceedings of the 18th ACM conference on Computer and communications security - CCS '11*. New York, New York, USA: ACM Press, 2011, p. 445.

[Cur21] Curve. *Curve.* . 2021. url: https://curve.fi/ (visited on 12/30/2021).

[Dai+20] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. "Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability". In: *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2020, pp. 910–927. url: http://arxiv.org/abs/1904.05234%20https://ieeexplore.ieee.org/document/9152675/.

[Dav+18] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. "Ouroboros Praos: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain". In: *Advances in Cryptology - EUROCRYPT 2018*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 46. Cham: Springer International Publishing, 2018, pp. 66–98. url: http://link.springer.com/10.1007/978-3-319-78375-8%7B%5C_%7D3.

[DEF18] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. "FairSwap: How To Fairly Exchange Digital Goods". In: *Proceedings of the 2018 ACM SIGSAC Conference*

on Computer and Communications Security - CCS '18. New York, New York, USA: ACM Press, 2018, pp. 967–984. url: http://dl.acm.org/citation.cfm?doid=3243734.3243857.

[DFH18] Stefan Dziembowski, Sebastian Faust, and Kristina Hostáková. "General state channel networks". In: *Proceedings of the ACM Conference on Computer and Communications Security* 1 (2018), pp. 949–966.

[Dou02] John R. Douceur. "The Sybil Attack". In: *Proceedings of the third international symposium on Information processing in sensor networks - IPSN'04*. New York, New York, USA: ACM Press, 2002, pp. 251–260. url: http://portal.acm.org/citation.cfm?doid=984622.984660%20http://link.springer.com/10.1007/3-540-45748-8%7B%5C_%7D24.

[DP97] Darrell Du and Jun Pan. "An Overview of Value at Risk". In: (1997), pp. 1–85.

[DPS19] Phil Daian, Rafael Pass, and Elaine Shi. "Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proofs of Stake". In: *Financial Cryptography and Data Security* (2019), Proof of Stake. url: https://eprint.iacr.org/2016/919.pdf.

[DW13] Christian Decker and Roger Wattenhofer. "Information propagation in the Bitcoin network". In: *13th IEEE International Conference on Peer-to-Peer Computing, IEEE P2P 2013 - Proceedings* (2013).

[DYd21] DYdX. *dYdX*. 2021. url: https://dydx.exchange/ (visited on 01/03/2022).

[Eng96] Gary V Engelhardt. "Consumption, down payments, and liquidity constraints". In: *Journal of money, credit and Banking* 28.2 (1996), pp. 255–271.

[ES14] Ittay Eyal and Emin Gün Sirer. "Majority Is Not Enough: Bitcoin Mining Is Vulnerable". In: *Financial Cryptography and Data Security 2014*. Vol. 8437. Berlin, Heidelberg, 2014, pp. 436–454. url: http://link.springer.com/10.1007/978-3-662-45472-5%7B%5C_%7D28.

[Eth19] Ethereum. *Ethereum 2.0 Speci cations*. 2019. url: https://github.com/ethereum/eth2.0-specs (visited on 03/14/2019).

[Fan+19] Giulia Fanti, Leonid Kogan, Sewoong Oh, Kathleen Ruan, Pramod Viswanath, and Gerui Wang. "Compounding of Wealth in Proof-of-Stake Cryptocurrencies". In: *Financial Cryptography and Data Security 2019*. 2019. url: http://arxiv.org/abs/1809.07468.

[FLM09] Drew Fudenberg, David Levine, and Eric Maskin. "The folk theorem with imperfect public information". In: *A Long-Run Collaboration on Long-Run Games*. World Scientific, 2009, pp. 231–273.

[FLO02] Shane Frederick, George Loewenstein, and Ted O'donoghue. "Time discounting and time preference: A critical review". In: *Journal of economic literature* 40.2 (2002), pp. 351–401.

[FRE22] FRED. *Federal Reserve Economic Data*. 2022. url: https://fred.stlouisfed.org/ (visited on 01/05/2022).

[Fu+21] Watson Fu, Tarun Chitra, Rei Chiang, and John Morrow. "Market Risk Assessment: An analysis of the financial risk to participants in the Aave V1 and V2 protocols." 2021.

[Gaj18] Sebastian Gajek. *Graded Token-Curated Decisions with Up-/Downvoting*. 2018. url: https://medium.com/coinmonks/graded-token-curated-decisions-with-up-downvoting-designing-cryptoeconomic-ranking-and-2ce7c000bb51 (visited on 01/30/2019).

[Gar+13] Juan Garay, Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. "Rational protocol design: Cryptography against incentive-driven adversaries". In: *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS* 200020 (2013), pp. 648–657.

[Ger+15] Arthur Gervais, Hubert Ritzdorf, Ghassan O. Karame, and Srdjan Capkun. "Tampering with the Delivery of Blocks and Transactions in Bitcoin". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15* (2015), pp. 692–705. url: http://dl.acm.org/citation.cfm?doid=2810103.2813655.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. "The Bitcoin Backbone Protocol: Analysis and Applications". In: ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. Lecture Notes in Computer Science June 2017. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 281–310. url: http://link.springer.com/10.1007/978-3-662-46803-6%7B%5C_%7D10.

[Gud+20a] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. "SoK: Layer-Two Blockchain Protocols". In: *Financial Cryptography and Data Security. FC 2020. Lecture Notes in Computer Science.* 12059 (2020), pp. 201–226.

[Gud+20b] Lewis Gudgeon, Daniel Perez, Dominik Harz, Benjamin Livshits, and Arthur Gervais. "The Decentralized Financial Crisis". In: *Proceedings - 2020 Crypto Valley Conference on Blockchain Technology, CVCBT 2020* (2020), pp. 1–15.

[Gud+20c] Lewis Gudgeon, Sam Werner, Daniel Perez, and William J. Knottenbelt. "DeFi Protocols for Loanable Funds: Interest Rates, Liquidity and Market Efficiency". In: *AFT 2020 - Proceedings of the 2nd ACM Conference on Advances in Financial Technologies* (2020), pp. 92–112.

[Har+17] Antoine Harary, David M. Bersoff, Sarah Adkins, Steve Schmidt, Stephanie Lvovich, Gustavo Bonifaz, and Kristin Heume. *2017 Edelman Trust Barometer*. Tech. rep. 17. Edelman, 2017.

[Har+19] 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 (CCS '19)*. New York, NY, USA: ACM, 2019. url: https://eprint.iacr.org/2019/675.pdf.

[Har+20] Dominik Harz, Lewis Gudgeon, Rami Khalil, and Alexei Zamyatin. "Promise: Leveraging Future Gains for Collateral Reduction". In: *Mathematical Research for Blockchain Economy*. Springer, Cham, 2020. Chap. Springer P, pp. 143–160. url: http://link.springer.com/10.1007/978-3-030-53356-4%7B%5C_%7D9.

[Har20] Dominik Harz. *Stealing All of Maker's Collateral*. 2020. url: https://medium.com/@dominik.harz/stealing-all-of-makers-collateral-f940970605b1 (visited on 01/01/2022).

[HB19] Dominik Harz and Magnus Boman. "The Scalability of Trustless Trust". In: *2nd Workshop on Trusted Smart Contracts (Financial Cryptography and Data Security)*. 2019, pp. 279–293. url: http://arxiv.org/abs/1801.09535.

[HBC15] Ferry Hendrikx, Kris Bubendorfer, and Ryan Chard. "Reputation systems: A survey and taxonomy". In: *Journal of Parallel and Distributed Computing* 75 (Jan. 2015), pp. 184–197.

[Hei+15] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. "Eclipse Attacks on Bitcoin's Peer-to-Peer Network". In: *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, 2015, pp. 129–144. url: https://www.usenix.org/node/190891.

[Imm21] Immunefi. *RocketPool and Lido Frontrunning Bug Fix Postmortem*. 2021. url: https://medium.com/immunefi/rocketpool-lido-frontrunning-bug-fix-postmortem-e701f26d7971 (visited on 01/03/2022).

[JIB07] Audun Jøsang, Roslan Ismail, and Colin Boyd. "A survey of trust and reputation systems for online service provision". In: *Decision Support Systems* 43.2 (Mar. 2007), pp. 618–644.

[Jud+19] Aljosha Judmayer, Nicholas Stifter, Alexei Zamyatin, Itay Tsabary, Ittay Eyal, Peter Gaži, Sarah Meiklejohn, and Edgar Weippl. "Pay-To-Win: Incentive Attacks on Proof-of-Work Cryptocurrencies". 2019. url: https://eprint.iacr.org/2019/775.

[Kal+18] Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S Matthew Weinberg, Edward W Felten, Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S Matthew Weinberg, and Edward W Felten. "Arbitrum: Scalable, private smart contracts". In: *USENIX Security 2018*. 2018.

[KB14] Ranjit Kumaresan and Iddo Bentov. "How to Use Bitcoin to Incentivize Correct Computations". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*. New York, New York, USA: ACM Press, 2014, pp. 30–41.

[KC09] Reid Kerr and Robin Cohen. "Smart Cheaters Do Prosper : Defeating Trust and Reputation Systems". In: *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*. Budapest, Hungary, 2009, pp. 993–1000.

[KGF19] Rami Khalil, Arthur Gervais, and Guillaume Felley. "NOCUST - A Securely Scalable Commit-Chain". 2019. url: https://eprint.iacr.org/2018/642.

[Kla+20] Ariah Klages-Mundt, Dominik Harz, Lewis Gudgeon, Jun You Liu, and Andreea Minca. "Stablecoins 2.0: Economic Foundations and Risk-based Models". In: *AFT 2020 - Proceedings of the 2nd ACM Conference on Advances in Financial Technologies* 2 (2020), pp. 59–79.

[KM19] Ariah Klages-Mundt and Andreea Minca. "(In)Stability for the Blockchain: Deleveraging Spirals and Stablecoin Attacks". In: *2019 Crypto Valley Conference on Blockchain Technology (CVCBT)*. 2019, pp. 1–25. url: http://arxiv.org/abs/1906.02152.

[KN12] Sunny King and Scott Nadal. "PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake". 2012.

[Kos+16] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts". In: *2016 IEEE Symposium on Security and Privacy (SP)*. Vol. 2015. IEEE, May 2016, pp. 839–858. url: http://ieeexplore.ieee.org/document/7546538/.

[KSG03] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. "The Eigentrust algorithm for reputation management in P2P networks". In: *Proceedings of*

the twelfth international conference on World Wide Web - WWW '03. New York, New York, USA: ACM Press, 2003, p. 640.

[KW82] David M Kreps and Robert Wilson. "Reputation and imperfect information". In: *Journal of economic theory* 27.2 (1982), pp. 253–279.

[Lau+20] Darren Lau, Daryl Lau, Teh Sze Jin, Kristian Kho, Erina Azmi, TM Lee, and Bobby Ong. *How to DeFi*. 1st. CoinGecko, 2020, p. 200.

[Leo+21] Stefanos Leonardos, Barnabé Monnot, Daniël Reijsbergen, Efstratios Skoulakis, and Georgios Piliouras. "Dynamical analysis of the EIP-1559 Ethereum fee market". In: *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*. New York, NY, USA: ACM, Sept. 2021, pp. 114–126. url: https://dl.acm.org/doi/10.1145/3479722.3480993.

[Lew+15] Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. "Bitcoin Mining Pools: A Cooperative Game Theoretic Analysis Categories and Subject Descriptors". In: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)* (2015), pp. 919–927.

[Lid21] Lido Finance. *Lido Finance - Staking*. 2021. url: https://stake.lido.fi/ (visited on 12/30/2021).

[Luu+15] Loi Luu, Jason Teutsch, Raghav Kulkarni, and Prateek Saxena. "Demystifying Incentives in the Consensus Computer". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*. New York, New York, USA: ACM Press, 2015, pp. 706–719. url: http://dl.acm.org/citation.cfm?doid=2810103.2813659.

[Luu+16] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. "Making Smart Contracts Smarter". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*. New York, New York, USA: ACM Press, 2016, pp. 254–269.

[LZ17] Orfeas Stefanos Thyfronitis Litos and Dionysis Zindros. "Trust Is Risk: A Decentralized Financial Trust Platform." 2017.

[Mak19] Maker. *MKR Tools*. 2019. url: https://mkr.tools/visualizations/historicalcdps (visited on 10/30/2019).

[Mak20] Maker Forum Participants. *Signal Request: Should we have another executive vote regarding the Governance Security Module? - Governance / Signal Archive*. 2020. url: https://forum.makerdao.com/t/signal-request-should-we-have-another-executive-vote-regarding-the-governance-security-module/1209 (visited on 01/03/2022).

[Mak21a] MakerDAO. *MakerDAO*. 2021. url: https://makerdao.com/en/ (visited on 12/30/2021).

[Mak21b] MakerDAO Community Portal. *Emergency Shutdown*. 2021. url: https://makerdao.world/en/learn/governance/emergency-shutdown/ (visited on 01/03/2022).

[McC+18] P McCorry, S Bakshi, I Bentov, Andrew Miller, and ... "Pisa: Arbitration Outsourcing for State Channels". 2018. url: https://www.cs.cornell.edu/%7B~%7Diddo/pisa.pdf.

[McC18] Trent McConaghy. *The Layered TCR*. 2018. url: https://blog.oceanprotocol.com/the-layered-tcr-56cc5b4cdc45 (visited on 01/30/2019).

[MHM18] Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. "Smart Contracts for Bribing Miners". In: *Financial Cryptography and Data Security. FC 2018.* Vol. 10958. Springer Berlin Heidelberg, 2018, pp. 3–18. url: http://link.springer.com/10.1007/978-3-662-58820-8%7B%5C_%7D1.

[Mik17] Mike Goldin. "Token-Curated Registries 1.0". 2017. url: https://medium.com/@ilovebagels/token-curated-registries-1-0-61a232f8dac7.

[Mil+16] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. "The Honey Badger of BFT Protocols". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16* Section 3 (2016), pp. 31–42. url: http://dl.acm.org/citation.cfm?doid=2976749.2978399.

[MR82] Paul Milgrom and John Roberts. “Predation, reputation, and entry deterrence”. In: *Journal of economic theory* 27.2 (1982), pp. 280–312.

[Nak08] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system”. 2008.

[Nex21] Nexus Mutual. *Nexus Mutual Overview*. 2021. url: https://app.nexusmutual.io/cover (visited on 12/31/2021).

[Nis+07] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Ed. by Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. Vol. 1. Cambridge: Cambridge University Press, July 2007, pp. 1–754.

[Nis14] Noam Nisan. “Algorithmic mechanism design: Through the lens of Multi-unit auctions”. 2014.

[NR01] Noam Nisan and Amir Ronen. “Algorithmic Mechanism Design”. In: *Games and Economic Behavior* 35.1-2 (Apr. 2001), pp. 166–196.

[Oly21] OlympusDAO. *OlympusDAO*. 2021. url: https://app.olympusdao.finance/%7B%5C#%7D/dashboard (visited on 12/30/2021).

[PD16] Joseph Poon and Thaddeus Dryja. *The Bitcoin Lightning Network: Scalable O - Chain Instant Payments*. Tech. rep. 2016, p. 59. url: https://lightning.network/lightning-network-paper.pdf.

[Pec20] PeckShield. *bZx Hack Full Disclosure (With Detailed Pro t Analysis)*. 2020. url: https://peckshield.medium.com/bzx-hack-full-disclosure-with-detailed-profit-analysis-e6b1fa9b18fc (visited on 01/03/2022).

[PS13] Isaac Pinyol and Jordi Sabater-Mir. “Computational trust and reputation models for open multi-agent systems: A review”. In: *Arti cial Intelligence Review* 40.1 (2013), pp. 1–25.

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. “Analysis of the Blockchain Protocol in Asynchronous Networks”. In: *Advances in Cryptology - EUROCRYPT 2017*. Vol. 10211. 2017, pp. 643–673. url: http://link.springer.com/10.1007/978-3-319-56614-6%7B%5C_%7D22.

[Rei+21] Daniël Reijsbergen, Shyam Sridhar, Barnabé Monnot, Stefanos Leonardos, Stratis Skoulakis, and Georgios Piliouras. "Transaction Fees on a Honeymoon: Ethereum's EIP-1559 One Month Later". Oct. 2021. url: http://arxiv.org/abs/2110.04753.

[Rek20] Rekt. *Rekt - Harvest Finance*. 2020. url: https://rekt.eth.link/harvest-finance-rekt/ (visited on 01/03/2022).

[Rib21] Ribbon Finance. *Ribbon Finance: Crypto structured products on Ethereum*. 2021. url: https://app.ribbon.finance/ (visited on 12/30/2021).

[ROH16] Wessel Reijers, Fiachra O'Brolcháin, and Paul Haynes. "Governance in Blockchain Technologies and Social Contract Theories". In: *Ledger* 1 (2016), pp. 134–151.

[Rou15] Tim Roughgarden. "The Price of Anarchy in Games of Incomplete Information". In: *ACM Transactions on Economics and Computation* 3.1 (2015), pp. 1–20.

[Rou17] Simon de la Rouviere. *Introducing Curation Markets: Trade Popularity of Memes and Information (with code)*. 2017. url: https://medium.com/@simondlr/introducing-curation-markets-trade-popularity-of-memes-information-with-code-70bf6fed9881 (visited on 01/30/2019).

[Rud18] Achill Rudolph. *Ranking Token Curated Registries*. 2018. url: https://medium.com/coinmonks/ranking-token-curated-registries-e9a92dc85b31 (visited on 01/30/2019).

[Sav20] Ioan-Daniel Savu. "Byzantic: Privacy-Preserving Collateral Reduction for DeFi Protocols". MSc. Imperial College London, 2020. url: https://www.imperial.ac.uk/media/imperial-college/faculty-of-engineering/computing/public/1920-pg-projects/Byzantic--Privacy-Preserving-Collateral-Reduction-for-DeFi-Protocols.pdf.

[Sco77] James H Scott. "Bankruptcy, secured debt, and optimal capital structure". In: *The journal of nance* 32.1 (1977), pp. 1–19.

[SJ85] RenéM Stulz and Herb Johnson. "An analysis of secured debt". In: *Journal of nancial Economics* 14.4 (1985), pp. 501–521.

[SL01] Tuomas W. Sandholm and Victor R. Lesser. "Leveled Commitment Contracts and Strategic Breach". In: *Games and Economic Behavior* 35.1-2 (2001), pp. 212–270.

[Slo21] Slowmist. *Blockchain Hacks*. 2021. url: https://hacked.slowmist.io/en/.

[ST13] Vasilis Syrgkanis and Eva Tardos. "Composable and efficient mechanisms". In: *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing - STOC '13* (2013), p. 211. url: http://dl.acm.org/citation.cfm?doid=2488608.2488635.

[Syn20] Synthetix. "Synthetix Litepaper". 2020. url: https://docs.synthetix.io/litepaper/.

[Ten16] Tendermint. *Introduction to Tendermint - Tendermint*. 2016. url: https://tendermint.com/intro (visited on 02/02/2017).

[TMB19] Jason Teutsch, Sami Mäkelä, and Surya Bakshi. "Bootstrapping a stable computation token". 2019. url: http://arxiv.org/abs/1908.02946.

[TR17] Jason Teutsch and Christian Reitwießner. "A scalable verification solution for blockchains". 2017.

[Tri92] George G Triantis. "Secured debt under conditions of imperfect information". In: *The Journal of Legal Studies* 21.1 (1992), pp. 225–258.

[Tsa+18] Petar Tsankov, Andrei Dan, Dana Drachsler-Cohen, Arthur Gervais, Florian Bünzli, and Martin Vechev. "Securify". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. August. New York, NY, USA: ACM, Oct. 2018, pp. 67–82. url: http://arxiv.org/abs/1806.01143%20https://dl.acm.org/doi/10.1145/3243734.3243780.

[Uni21] Uniswap. *Uniswap App*. 2021. url: https://app.uniswap.org/%7B%5C#%7D/swap (visited on 12/30/2021).

[WB17] Will Warren and Amir Bandeali. "0x : An open protocol for decentralized exchange on the Ethereum blockchain". 2017.

[Web21] Web3 Foundation. *Governance - Polkadot Wiki*. 2021. url: https://wiki.polkadot.network/docs/learn-governance/ (visited on 01/02/2022).

[Wer+21] Sam M. Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J. Knottenbelt. "SoK: Decentralized Finance (DeFi)". In: (Jan. 2021). url: http://arxiv.org/abs/2101.08778.

[WK18] Yi Lucy Wang and Bhaskar Krishnamachari. "Enhancing Engagement in Token-Curated Registries via an Inflationary Mechanism". 2018. url: http://arxiv.org/abs/1811.09680.

[Woo09] Michael Wooldridge. *An Introduction to MultiAgent Systems*. 2nd. Wiley Publishing, 2009.

[Woo14] Gavin Wood. "Ethereum: a secure decentralised generalised transaction ledger". 2014.

[Yea21a] Yearn Finanace. *yearn. nance*. 2021. url: https://yearn.finance/%7B%5C#%7D/home (visited on 12/30/2021).

[Yea21b] Yearn Security. *Vulnerability disclosure 2021-02-04*. 2021. url: https://github.com/yearn/yearn-security/blob/master/disclosures/2021-02-04.md (visited on 01/03/2022).

[Yer17] David Yermack. "Corporate governance and blockchains". In: *Review of Finance* 21.1 (2017), pp. 7–31.

[Yu+19] Jiangshan Yu, David Kozhaya, Jeremie Decouchant, and Paulo Verissimo. "RepuCoin: Your Reputation is Your Power". In: *IEEE Transactions on Computers* (2019), pp. 1–1. url: https://ieeexplore.ieee.org/document/8645706/.

[Zam+19] Alexei Zamyatin, Dominik Harz, Joshua Lind, Panayiotis Panayiotou, Arthur Gervais, and William J. Knottenbelt. "XCLAIM: Trustless, Interoperable, Cryptocurrency-Backed Assets". In: *Proceedings of the IEEE Symposium on Security and Privacy, May 2019.* 2019, pp. 1254–1271. url: https://eprint.iacr.org/2018/643.pdf.

[Zam+21] Alexei Zamyatin, Mustafa Al-Bassam, Dionysis Zindros, Eleftherios Kokoris-Kogias, Pedro Moreno-Sanchez, Aggelos Kiayias, and William J. Knottenbelt. "SoK: Communication Across Distributed Ledgers". In: *Financial Cryptography and Data Security. FC 2021. Lecture Notes in Computer Science.* Vol. 12675. Springer, Berlin, Heidelberg, 2021.

[Zol19] Micah Zoltu. *How to turn $20M into $340M in 15 seconds*. 2019. url: https://medium.com/coinmonks/how-to-turn-20m-into-340m-in-15-seconds-48d161a42311 (visited on 01/01/2022).

Appendix A

Notation and Symbols

Table A.1: Overview of notation and symbols.

Symbol	Definition
	Cryptoeconomic protocol
$P = \{1, \ldots, n\}$	Set of n agents
$\in \{\Sigma_d, \Sigma_u\}$	Desired and undesired action
	Agent type from the space of agents types Θ
$\hat{}$	Agent's reported type
u_i	Utility of agent $i \in P$
V_i	Private valuation agent $i \in P$ attaches to the realisation of a particular outcome
c_i	Costs for agent $i \in P$
r	Interest rate
	Discount factor
$E[\cdot]$	Expected value
t	Time step
	A cryptographically verifiable protocol specification
p	Payment held in escrow and released to the providing agent on fulfillment of the agreement.
C	A monetary collateral.
$A = \langle \ , p, C \rangle$	Agreements with specification , payment p, and set of collateral C
N	USD value of collateral
f	Collateral factor
	Cryptoeconomic protocol with Balance
$A = \langle \ , p, s, C \rangle$	Agreements with score s
$L = \{L_1, \ldots, L_l\}$	An ordered list of l layers
$L_m = \langle l, u, f \rangle$	Layer with a lower bound l, upper bound u, and a collateral factor f for layer $L_m \in L$
C_m	Collateral at a specific layer $L_m \in L$, e.g., C_1 for layer 1
f_m	Collateral factor for layer $L_m \in L$
	Cryptoeconomic protocol with Promise
m	The number of future periods
	The likelihood that the user remains in the protocol
n	The number of times an agent did not deliver the service
	Discount factor for future utility

Appendix B

Balance Incentive Proofs

The rational agent type $_r$ needs to decide which action to perform at the highest layer. From Figure 4.2, we see that at the highest layer, the agent has either the choice to perform the desired action, which results in a utility of $p \quad c_A \quad \mathrm{E}[rC_3]$ or an undesired action which is equivalent to a utility of $v \quad c_A \quad \mathrm{E}[rC_3] \quad C_3$. In this simplified figure, there are three layers. However, we can generalize this by assuming $!$ layers. By Definition 3.1, the rational type cannot decide considering these two utility functions. Reaching a decision requires $_r$ to consider several time-steps. By integrating Balance, we transform the decision process into a *sequential game*.

B.1 Decision Boundary for the Rational Type

The total utility received by an agent performing desired actions in the highest layer, for a sequence that lasts $t = !$ rounds, is expressed by:

$$u_A = \sum_{t=0}^{!\mathsf{X}^1} \frac{}{1+r}^{t} \quad p \quad c_A \quad \mathrm{E}[rC_!] \tag{B.1}$$

Summing over cycles to infinity yields the following expression (swapped for t to express "within" cycle time):

$$u_A = \sum_{=0}^{\infty} (\frac{}{1+r})^{\ !} \quad \sum_{=0}^{!-1} (\frac{}{1+r}) \ (p \quad c_A \quad \mathrm{E}[rC_!]) \tag{B.2}$$

The total utility corresponding to an agent who commits an undesired action in the highest layer $!$, followed by $!$ 1 desired actions to return to that layer, can be expressed as a sequence with $t = !$ rounds as follows. We express the deposit C of layer $L_m \ 2 \ L$ at time t with C_t.

$$u_A = v \quad c_A \quad \mathrm{E}[rC_!] \quad C_! + \sum_{t=1}^{!-1} \frac{}{1+r}^{\ t} \quad p \quad c_A \quad \mathrm{E}[rC_t] \tag{B.3}$$

Summing over cycles to infinity yields the following expression (swapped for t to express 'within' cycle time):

$$u_A = \sum_{=0}^{\infty} (\frac{}{1+r})^{\ !} \quad v \quad c_A \quad C_! \quad \mathrm{E}[rC_!] + \sum_{=1}^{!-1} (\frac{}{1+r}) \ (p \quad c_A \quad \mathrm{E}[rC \ ]) \tag{B.4}$$

The decision boundary for agents of type T_r is found by equating equations (B.2) and (B.4). This is as follows.

$$\begin{aligned} &\sum_{=0}^{\infty} (\frac{}{1+r})^{\ !} \quad v \quad c_A \quad C_! \quad \mathrm{E}[rC_!] + \sum_{=1}^{!-1} (\frac{}{1+r}) \ (p \quad c_A \quad \mathrm{E}[rC \ ]) \\ &= \sum_{=0}^{\infty} (\frac{}{1+r})^{\ !} \quad \sum_{=0}^{!-1} (\frac{}{1+r}) \ (p \quad c_A \quad \mathrm{E}[rC_!]) \end{aligned} \tag{B.5}$$

The following lemma simplifies this boundary condition.

Lemma B.1. *Comparing payo s of desired as opposed to undesired actions into the in nite horizon, with ! 1 , is the same as comparing the payo s of these actions over a single cycle.*

Proof. Inspection of (B.5) shows that the two sides of the equation are equal when the terms inside the summation operator (summing from = 0 to 1) are equal. □

Therefore the equation becomes

$$\begin{aligned} v \quad c_A \quad C_{!} \quad \mathrm{E}[rC_{!}] + \sum_{t=1}^{!X^1} (\frac{}{1+r})^t (p \quad c_A \quad \mathrm{E}[rC_t]) \\ = \sum_{t=0}^{!X^1} (\frac{}{1+r})^t (p \quad c_A \quad \mathrm{E}[rC_{!}]) \end{aligned} \tag{B.6}$$

Which simplifies to:

$$v = p + C_{!} + \sum_{t=1}^{!X^1} (\frac{}{1+r})^t (\mathrm{E}[rC_t] \quad \mathrm{E}[rC_{!}]) \tag{B.7}$$

If we equate (B.1) and (B.3), we get the decision boundary for an agent to perform a desired or undesired action. We express this by the factor f. By (5.3), f_m determines the deposit of a layer m in relation to the base deposit C_{base}.

Lemma B.2. *Introducing* Balance *creates a sequential game in which factors f determine the deposit at each layer. The factor of the highest layer $f_{!}$ in a sequential game with at least two layers can be set lower than f_1 and still achieves a higher utility for the desired action compared to an undesired action.*

Proof. First, we equate (B.1) and (B.3) characterizing the decision boundary. Next, we set $C_{!} = f_{!} C_{\text{base}}$ and $C_t = f_t C_{\text{base}}$ (where t and m are are substitutes for a single cycle). This leaves us with the following equality.

$$\begin{aligned} & \sum_{t=0}^{!X^1} \quad \overline{1+r} \quad^t \quad p \quad c_A \quad \mathrm{E}[r f_{!} C_{\text{base}}] \\ & = v \quad c_A \quad \mathrm{E}[r f_{!} C_{\text{base}}] \quad (f_{!} C_{\text{base}}) \\ & + \sum_{t=1}^{!X^1} \quad \overline{1+r} \quad^t \quad p \quad c_A \quad \mathrm{E}[r f_t C_{\text{base}}] \end{aligned} \tag{B.8}$$

We assume that the agent can progress to the next layer after every time step t. Also, we apply our initial assumptions that $p \quad c_A$ for performing the desired action is 0, and that c_A for the undesired action is 0. We then simplify (B.8).

$$\begin{aligned} & \sum_{t=1}^{!-1} \frac{1}{1+r}^{t} \mathrm{E}[r f_{!} C_{\text{base}}] \\ = v \quad & f_{!} C_{\text{base}} + \sum_{t=1}^{!-1} \frac{1}{1+r}^{t} \mathrm{E}[r f_t C_{\text{base}}] \end{aligned} \tag{B.9}$$

$$v = f_{!} C_{\text{base}} + \sum_{t=1}^{!-1} \frac{1}{1+r}^{t} \mathrm{E}[r C_{\text{base}}(f_t \quad f_{!})] \tag{B.10}$$

When we equate v and C_{base}, we imply the decision boundary without Balance. We can then express $f_{!}$ in relative terms by setting $v = C_{\text{base}} = 1$, and rearranging for $f_{!}$.

$$f_{!} = \frac{1 \quad \sum_{t=1}^{!-1} \frac{1}{1+r}^{t} \mathrm{E}[r f_t]}{1 \quad \sum_{t=1}^{!-1} \frac{1}{1+r}^{t} \mathrm{E}[r]} \tag{B.11}$$

From (B.11), we observe that if f_t is smaller than 1, then $f_{!}$ would become greater than 1. This would allow the agent to perform undesired actions in the first layer at a positive utility since v would become larger than $C_1 f_1 \quad p$. Therefore, the factor at the lowest layer must be greater than 1. If we set any factor f_t equal to 1, $f_{!}$ becomes also 1. Consequently, allowing a factor f_t to be greater than 1 allows us to set a deposit factor $f_{!}$ smaller than f_1. □

Lemma B.3 (Minimum deposit at the lowest layer)**.** *The deposit factor at the lowest layer f_1 must be greater than 1 to allow the factor at the highest layer $f_{!}$ to be less than 1.*

Proof. The proof follows from the proof of Theorem B.2. □

B.2 Linear Factor Adjustment

We assume a linear relationship between the smallest factor $f_{!}$ and the other factors f_t. Next, we assume that at C_1 the factor f_1 is $>$ 1 and that at $C_{!}$ the factor $f_{!}$ 1 is the lowest factor. From this, we can calculate linear relation between f_1 and $f_{!}$ as follows.

$$f_t = f_1 \quad \frac{f_1 \quad f_{!}}{! \quad 1} \quad (t \quad 1) \tag{B.12}$$

We can then replace f_t in (B.10) by (B.12) to express the term purely in terms of $f_{!}$. Further, we assume that the valuation v is lower than the deposit C_1, i.e., the deposit is higher than the desire for performing an undesired action. At $t = 2$, the agent is at layer 1 when playing the undesired action. At $t = 2$, the agent is at layer 2 by our assumption. Hence, we can replace L with t in the resulting equation by a slight abuse of notation. This leaves us with the following equation:

$$f_{!} = \frac{v! \quad v \quad \sum_{t=1}^{!} \frac{1}{1+r}^{t} \mathrm{E}[r f_1(! \quad t)]}{! \quad 1 \quad \sum_{t=1}^{!} \frac{1}{1+r}^{t} \mathrm{E}[r(! \quad t)]} \tag{B.13}$$