Informatics

Quantitative Analysis of MakerDAO's Liquidation System

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Business Informatics

eingereicht von

Martin Kjäer, BSc.
Matrikelnummer 01308533

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ass.Prof. Dr. Monika di Angelo

Wien, 13. August 2021

Martin Kjäer

Monika di Angelo

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Informatics

Quantitative Analysis of MakerDAO's Liquidation System

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Martin Kjäer, BSc.
Registration Number 01308533

to the Faculty of Informatics

at the TU Wien

Advisor: Ass.Prof. Dr. Monika di Angelo

Vienna, 13th August, 2021

Martin Kjäer   Monika di Angelo

Erklärung zur Verfassung der Arbeit

Martin Kjäer, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 13. August 2021

Martin Kjäer

Acknowledgements

I would like to especially thank my supervisors Monika di Angelo and Gernot Salzer, who gave valuable input to this work and also helped to summarize the main findings in a conference paper.

Furthermore I wish to express my gratitude to Georg Strodl, who always supported me during the course of writing this work and who probably is also a blockchain expert by now.

I am indebted to my aunt Birgitta Kjäer for promoting me throughout my Master studies and also for taking the time to proof-read the thesis.

Finally I wish to thank Sarah Bimingstorfer, Johannes Scherrer and Michelle du Plessis for their encouragement and friendship.

Kurzfassung

Die Entstehung von Smart Contract-basierten Blockchains hat in den letzten Jahren eine Revolution ausgelöst, die mit jener des Internets in den 1990ern vergleichbar ist. Verschiedenste neue Produktkategorien wie so genannte Stablecoins wurden auf diesen Blockchains entwickelt. Stablecoins sind Tokens, die preisstabil sein sollen und sind daher vergleichbar mit Fiat Währungen. Das bekannteste dezentralisierte Stablecoin Projekt ist MakerDAO, welches auf der Ethereum Blockchain aufbaut und eine Stablecoin namens DAI anbietet. Besonders der neuartige Ansatz, die wichtigsten Geschäftsprozesse in Smart Contracts abzubilden, macht das Projekt auch aus akademischer Perspektive interessant. Im Rahmen dieser Arbeit analysieren wir MakerDAOs Liquidierungsmechanismen, deren Ziel es ist, den Wert der DAI Stablecoin stabil zu halten. Nach einer Analyse von MakerDAOs wichtigsten Contracts definieren wir Anforderungen, die erfüllt werden müssen, um eine sichere Funktionsweise des Protokolls zu garantieren. Um zu beurteilen, ob die Anforderungen in der Praxis erfüllt werden, leiten wir Metriken von diesen Anforderungen ab. Unsere Analyse ergibt, dass Makers Liquidierungssystem während der meisten Zeit gut funktioniert, unter Stress aber auch an seine Grenzen stößt. Besonders während des so genannten Black Thursdays im März 2020 war das Protokoll nicht imstande, den laufenden Betrieb aufrecht zu erhalten und musste mehrere Millionen ungedeckter Schulden verbuchen. Es scheint, dass MakerDAOs Community die meisten Gründe, die zu diesem Zwischenfall geführt haben, mittlerweile gelöst hat. Nichtsdestotrotz ist noch weitere Forschungsarbeit notwendig um zu analysieren, ob eine erst kürzlich eingeführte überarbeitete Version des Liquidierungssystem in der Praxis hält, was sie verspricht.

Abstract

With the growth and development of smart contract-based blockchains a completely new technology entered the stage and initiated a revolution comparable to the one that begun in the 1990s after the invention of the Internet. During the last years a variety of new product categories have unfolded on top of these blockchains, with stablecoins being one of them. Stablecoins are tokens that aim to be stable in price and are thus comparable to fiat currencies. The most well-known decentralized stablecoin project is MakerDAO, which lives on Ethereum and offers a stablecoin named DAI. Especially due to the project's novel approach of having its core business logic implemented in smart contracts, it is worth to investigate it from an academic perspective. Within the scope of this thesis, we focus on MakerDAO's liquidation mechanisms that aim to secure the value of its stablecoin. After analyzing MakerDAO's core contracts we develop requirements that need to be fulfilled in order to guarantee a secure operation of the protocol. We derive metrics from these requirements that help us assess if the requirements are met in practice. MakerDAO's liquidation mechanism turns out to work well during most times, however, it also faces difficulties when under stress. Especially during the so-called Black Thursday event in March 2020, the protocol was not able to continue normal operation and found itself with several millions of uncovered debt. It seems that the community has fixed most of the issues that led to this incident, yet further research is needed to evaluate how a recently deployed remake of the collateral auctioning mechanism proves itself in practice.

Contents

CHAPTER 1

Introduction

A lot of hype is happening around smart contract platforms, especially Ethereum. While the Ethereum blockchain itself is still under active development and several issues reaching from technical to legal aspects still need to be resolved, many startups and initiatives already presented white papers, launched Initial Coin Offerings (ICOs) and launched their products on different blockchains.

Especially base layer technologies are currently under development as they are needed for many dapps and a lot of progress has been made in this area within the last years. One well known example are stablecoins, which are said to be a cornerstone for many dapps. As the name suggests, stablecoins are tokens that have a much lower price volatility compared to other blockchain tokens like Bitcoin or Ether and thus can be utilized as a reliable medium of value exchange on the blockchain.

The stablecoin project that issued the highest number of stablecoins at the time of writing is Tether. It is issued by Tether Limited, which operates since many years in a centralized manner by managing tokens on different blockchains while funds that are backing the tokens are stored outside the blockchain.

During the last years, however, several decentralized stablecoin projects have gained traction, the most well-known being MakerDAO, which operates on the Ethereum blockchain. Maker[1] is especially interesting as it is one of the most trusted stablecoin projects. It has proven to be able to peg its stablecoin DAI close to the United States Dollar (USD) during most times, while still managing the entire project in a decentralized manner. This is achieved through a set of sophisticated incentive mechanisms enabled by means of smart contracts as well as through decentralized governance mechanisms.

[1]MakerDAO uses both 'Maker' and 'MakerDAO' when it refers to itself and for better readability we do the same from here onwards. Both terms can be used interchangeably.

1.1 Problem Statement

Although stablecoins are an interesting subfield of blockchain technology, not much academic research has been published analyzing them. Stablecoins are especially interesting to investigate, as they enable numerous other applications in the newly emerging field of dapp. Moreover, they can tell us more about current and future developments of decentralized business models.

Even less has been published about MakerDAO, probably because it started offering its full protocol implementation, the Multi Collateral DAI (MCD) not even two years ago. Nevertheless, the initiative has quietly become the largest decentralized stablecoin project so far and manages currently more than 5 billion in issued DAI[2] via its smart contracts.

We argue that it is especially worth to evaluate Maker's liquidation system from a quantitative perspective. The liquidation system is responsible to keep the DAI overcollateralized and is thus one of the most important elements of the protocol. Furthermore, research in this area would complement existing empirical research about Maker's oracles and governance mechanisms as well as analysis about stochastic models that assess Maker from a theoretical viewpoint.

1.1.1 Research Questions

Within the scope of this thesis, we conduct a quantitative analysis of Maker's liquidation system. More specifically, we utilize publicly available Ethereum transactional data to investigate Maker's liquidation system.

First, we aim at identifying requirements that have to be met so that the protocol's liquidation system runs smoothly. After that we derive suitable metrics, which help us to measure if the protocol fulfills the requirements. We define research question one to be:

Which requirements have to be met to allow a stable operation of Maker's liquidation system and which metrics are appropriate to measure if the requirements are fulfilled?

After we have answered research question one we continue our analysis and define a subsequent question as research question two:

Does Maker's liquidation system comply with the requirements in practice?

[2]Which amounts to approximately 5 billion USD

1.2 Methodology

In this section we discuss our methodological approach. Due to the fact that our research area covers multiple disciplines (social science as it is about organizational theory/Decentralized Autonomous Organizations (DAOs)), computer science as it is about distributed systems, and economics, as it is about financial systems), we considered different methods employed by those disciplines. Therefore we briefly discuss some of the visited methods and their respective applicability for our research topic, before we outlay our methodological approach in detail.

Wohlin et al. [60] argue that for empirical research about real projects in software engineering with low level of control a case study methodology can be applied. Yin [63] points out that case studies are methodologically very similar to histories or archival analysis but have the advantage that they can use also other sources of evidence like interviews or direct observations. This indicates that a history or archival analysis might be the better methodology in our case, as we mainly analyze the Ethereum mainchain as well as the publicly available code of Maker's smart contracts.

However, it has to be mentioned that archival analysis is mainly used for social sciences and is therefore also methodologically not the best fit for our research area [57]. Yin [63], however, also points out that different research methods are not mutually exclusive.

Hence, we apply a case study methodology suggested by Yin [63], due to the openness of his approach, which can be applied by many different disciplines. This is important to us, as case studies were and still are mainly applied by social sciences and therefore also most case study methodologies focus on problems related to social sciences (e.g. mainly describe how to collect and process qualitative data). Furthermore Yin has especially a focus on case study design [12], which makes it an ideal starting point for planing the case study.

In the next section, we define our main design choices.

1.2.1 Study Design

Yin [63] describes five components that define a case study design:

1. A case study's question
2. It's propositions, if any
3. Its case(s)
4. The logic linking the data to the propositions, and
5. The criteria for interpreting the findings

Within the following lines we briefly describe, which considerations we have made for each of these components.

Case Study's Questions

The research questions have already been defined in section 1.1.1.

Propositions

We analyze Maker's main smart contracts to get an understanding about the core mechanics of the protocol. This helps us to define requirements, which need to be fulfilled to guarantee a stable operation of Maker's liquidation system.

Case

The case that we are analyzing is Maker itself. More specifically this includes smart contract bytecode and publicly available and verified source code from sources like [13] as well as a set of transactions that interacted with the contracts on the Ethereum mainchain. We limit our investigations to MCD, as it is far more advanced than Single-Collateral DAI (SCD) and also the first *full* implementation of the protocol.

We limit our analysis to the first operative year of Maker's MCD system. MCD went live on November 18th, 2019, so our study period starts on that day and includes data up until November 17th, 2020. We argue that this period is sufficiently long to collect enough data about the system's regular operation as well as stress situations like the one happening on Black Thursday.

Logic Linking the Data to the Propositions

Publicly available data about Maker's smart contracts as well as Ethereum transactional data that is linked to the smart contracts of the protocol are used to run different types of calculations.

Criteria for Interpreting the Findings

We derive a set of metrics from the requirements that we define and use these metrics to assess if the requirements can be fulfilled.

1.2.2 Systematic Literature Review

For the related literature section of this thesis (section 1.3) we make use of a systematic literature review (SLR) methodology to obtain a clearer picture about current research done in the area of stablecoins, especially MakerDAO. We have chosen to apply a systematic literature review proposed by Kitchenham [20].

Arguments for using a SLR are reproducibility as well as lower chance of biased results. Furthermore the approach suggested by [20] was especially developed for software engineering projects and is thus a more applicable methodology for the research questions.

However, as the main focus of this thesis is not a SLR but the case study described above, it has to be mentioned that we only applied core principles of the SLR approach.

1.2.3 Limitations

Within the scope of this thesis we focus on an empirical analysis of Maker's liquidation system and do not include mathematical models that might lead to other insights that can't be identified by this work.

Furthermore some authors argue that a case study methodology with just one case (MakerDAO) is not generalizable to other projects in the space. However, most stablecoin projects make use of very different mechanisms compared to Maker and are thus not easily comparable with each other.

Some parts and the main insights of this thesis have already been condensed and will be published [21][3]. The parts of the thesis that have been used include primarily the requirements and metrics in section 4.1 as well as the summarized results in form of figures and tables. The thesis partly extends the work published in the paper but includes also more details about our findings in general.

1.3 Related Literature

An overview about different stablecoin projects and their stabilization techniques has been created by the authors of [40]. They discuss different attack vectors on stablecoin projects and outline that it is difficult especially for decentralized stablecoin projects like Maker to reach full decentralization due to the fact that completely permissionless oracles still do not exist.

The authors of [23] develop a model of a general noncustodial stablecoin, which is backed by crypto assets. One of the main outcomes of simulations using the model is that deleveraging spirals can create major problems like high volatility for the USD nominated value of both the stablecoin as well as the underlying asset.

Another paper published by the same authors tries to explain what happened to the Maker protocol during Black Thursday by employing a stochastic model [22]. By experimenting with the model, Klages-Mundt et al. formally explained what led to Maker's crisis during Black Thursday and how it could be prevented.

[17] et al. do an empirical case study analysis about Maker's oracles system. They focus on the quality of Maker's price feeds and also studied the impact of wrong pricing information to the calculation of median values. Furthermore the authors analyze Maker's governance structure and discuss flaws and potential solutions to these flaws.

Park et al. [39] present a formal verification tool for EVM[4] bytecode that has been tested to verify some high profile smart contracts on the Ethereum, including those of Maker. The paper includes a description of the tool, however, does not state more details about the outcome of the formal verification of Maker's contracts.

[3]The paper has been accepted to a conference and will get published, but the conference takes place after this thesis gets handed in.

[4]Ethereum Virtual Machine

1.4 Structure of the Work

The rest of the work is organized as follows: In chapter 2 we briefly explain the core idea behind Bitcoin to make the reader familiar with the concepts of the first existing blockchain, before we explain what differentiates Ethereum from Bitcoin.

Chapter 3 aims to explain, how the Maker protocol is designed and how Maker is organized behind the scenes.

This information helps us to then formalize requirements on the liquidation system in chapter 4 were we include also explanations about calculations and our findings.

What follows is the Discussion (chapter 5) where we analyze our hypothesis and summarize the quantitative analysis before we draw our conclusions in chapter 6.

CHAPTER 2

Fundamentals

The concept of transactions aggregated in blocks that get chained together by a hash-based consensus algorithm called proof-of-work (PoW) was first described in 2008 by the pseudonymous Satoshi Nakamoto in his Bitcoin white paper [50]. Even if the term *blockchain* itself is not mentioned in the paper, Nakamoto explains his idea of creating a Peer-to-Peer (P2P) network that does not require a central entity to keep its state secure and where anyone can participate while trust to other participants is not required.

Since the genesis[1] of the world's first blockchain called Bitcoin on January 3rd, 2009 and the following success in the years after, the concept of blockchains has become widely discussed and researched both in academia and the private sector.

Also many new blockchains have evolved, especially in the last 5 years. While some of them differentiate from Bitcoin in just minor details, others have completely different algorithms and principles in place.

In the following section we briefly summarize the main concepts of Bitcoin and the ideas behind blockchains in general before we give a short overview on Ethereum and how it differentiates.

2.1 Bitcoin

Different to the more general concept of blockchains, Bitcoin has received attention from media and more recently even banks mainly due to the value of its coin. While the value that people are willing to pay for one Bitcoin does not say much about the technology directly, it does so indirectly: with a market cap of >600 billion U.S. dollar[2], the most

[1]Genesis is the time, when the first block of a blockchain gets published. This block is therefore also called the genesis block.

[2]In August 2021

famous blockchain has proven, that it can securely store data (i.e. Bitcoin values that are associated with addresses) and also perform transactions on this data in a discrete manner.

To achieve this, Bitcoin employs a number of techniques like public-key cryptography, hashing and most importantly a PoW based P2P consensus algorithm that guarantees, that double spendings[3] do not take place.

In the years before the publication of the Bitcoin whitepaper, also other protocols like b-money [10], Hashcash [3], or Bit Gold [52] have proposed the implementation of PoW algorithms to create something like an electronic money. However, none of them had been really pushed forward or developed further in a way that makes it as usable as Bitcoin. So how has Bitcoin achieved, what others could not do before? In the following lines we want to describe, which technologies Bitcoin employs and how.

First of all, Bitcoin makes use of public-key cryptography to manage something like accounts, as well as an access right system to these accounts.[4] To put it simply, a part of a hashed public key represents an account as an address and each Bitcoin (or a fraction of a Bitcoin) is associated to such an address. With this mechanism, Bitcoin addresses can be created arbitrarily by generating a public-private key pair and then calculating the address by using the public key.

The private key always stays with the owner and is kept secure. This is important as it provides the possibility to sign transactions, e.g. to send Bitcoins to another address. Besides the signature that proves that the sender is the entitled owner of the Bitcoin amount that is to be spent, a transaction includes among other fields also the address of the sender, the address of the receiver and the public key of the sender (as the address is only a representation of the public key).[5] After propagating a newly signed transaction to the network, other participants prove that the sender of the transaction is also the owner of the private key by verifying the signature with the provided public key. If a transaction is proven to be correct, it is added to the mempool and propagated further to other Bitcoin nodes in the network.

Now we explained, how permissionless (everyone can create a new Bitcoin address without restrictions) and trustless (we don't need a 3rd party who provides us with an account and the right to manipulate a balance, rather our private key stays always with us) ownership management works in Bitcoin. Every participant can create a public-private key pair, calculate a Bitcoin address from the public key, and publish this address to

[3]Double spending takes place, when a user tries to spend some of her assets in a transaction, even if these assets have been spent already before in an earlier transaction

[4]On a technical level, Bitcoin does not use accounts, but keeps a list of so called *UTXO* [54] to identify, how much Bitcoin each address owns. The term explains itself: Every Bitcoin transaction (TX) has one or more inputs as well as one or more outputs (O), which receive the Bitcoin value that gets transfered in the transaction. Furthermore, to prevent double spending, an input of a transaction can only be an unspent (U) output of another transaction that has been recorded before.

[5]It has to be mentioned that this is a simplification to explain the general concept. [55] details a more accurate description of how Bitcoin transactions work.

others so that they can send Bitcoins to it. After sending the Bitcoins, a Bitcoin balance is associated to this address and can then only be moved by the new owner, as she is the one who holds the private key.

What remains is the problem that we still need someone who keeps book of all Bitcoin addresses as well as associated Bitcoin balances. If we trust one entity that keeps book of this data, the entity could fraudulently change Bitcoin balances at will, e.g. to reward more Bitcoin to itself. On the other hand, if we don't trust any central authority and just create many copies of the *true* version of the data, distributed across the planet, how can we establish a reliable mechanism that minimizes the necessary trust to other network participants to the absolutely needed minimum?

The first usable and implemented solution to this problem was the real novelty of Bitcoin and it was first described in Satoshi Nakamoto's Bitcoin white paper [50]: the blockchain. Nakamoto describes a network, where new transactions[6] get aggregated in well-defined objects called *blocks*. These blocks contain besides the new transactions also other information like a timestamp, a reference to the previous block in the blockchain as well as a PoW that conduces as a witness that the one that produced the block has invested some amount of resources (hardware, power, time) to produce it. In many blockchains and in Bitcoin as well the PoW process is also called *mining* and the ones running the software are named *miners*, which is an analogy to gold miners.

Similarly to gold miners, Bitcoin miners are also extrinsically motivated to fulfill their job, as the first transaction in every Bitcoin block always compensates the miner who found[7] the block with the so-called *block reward*.

The block reward is one of the core pillars of the network, as it helps the network to stay secure. It is a freshly minted amount of Bitcoin that gets sent to an address decided by the miner. When the Bitcoin blockchain's first block was mined on January 3rd, 2009, the block reward was set to 50 Bitcoin. Since then, it is cut by half (in Bitcoin, this is also called *halving*) every 210 000 blocks. Since 2009, this happened on November 28th, 2012 to 25 Bitcoin, on July 9th, 2016 to 12.5 Bitcoin and since May 11th, 2020, the Bitcoin block reward is set to 6.25 Bitcoin for each new block. If the consensus rules do not get changed later, the halving will continue until Bitcoin reaches its maximum supply of 21 million Bitcoin.

So how does the blockchain looks like? Figure 2.1 shows a simplified abstraction of the Bitcoin blockchain. Each Bitcoin block has a block header and a body. The body contains new transactions that the miner can choose at will from the transactions that are currently waiting in the mempool to get included in a block. The header contains the Merkle tree root of these transaction (which is just a representation of the transactions), a timestamp that states, when the block was mined, the hash of the previous block

[6]The ones that have been collected in the mempool before

[7]In PoW blockchains, the term 'finding' a new block is also used as a way to describe that a new block has been mined.

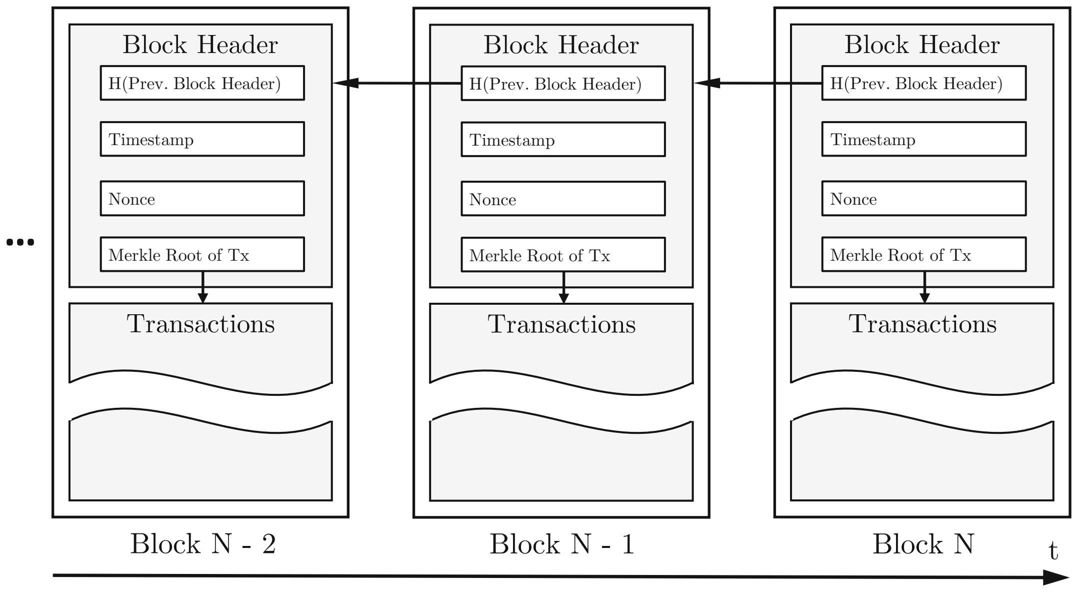

Figure 2.1: Simplified version of the Bitcoin blockchain. *H(x)* represents a hashfunction applied to value x. The latest mined block is block N, *t* stands for time.

(which is a reference to the block before and the reason why its called Block*chain*) and a nonce that can be randomly chosen by the miner.

The nonce is used for the PoW algorithm. It is the last value that gets added to a block, because it is actually the one that needs to be *found* to make the block valid. The reason for this lies in the details, how Bitcoin's PoW algorithm is constructed: after the transactions have been added to the block, a timestamp gets set and the hash of the previous block gets added. After that, the mining process begins.

Algorithm 2.1: Bitcoin Proof-of-Work algorithm

```
Data: A Merkle root of new transactions $mr(T_{new})$, a timestamp $ts$, a
      Hashvalue of the previous block header $H(BH_{prev})$, difficulty $d$ and seed $s$
1 while no other new and valid block received from other miners do
2     $nonce = rng(s)$
3     $BH_{new} = assembleBlockHeader(mr(T_{new}), ts, H(BH_{prev}), nonce)$
4     if $H(BH_{new}) < d$ then
5         return $BH_{new}$
6     end
7 end
```

Algorithm 2.1 describes how Bitcoin mining (PoW) works: As long as a miner does not receive a valid new block that another miner has mined, she continues the following steps:

A new random nonce gets picked. The nonce, together with the already known other elements of the block header get assembled to a new block header of a new block. Next, the hash of this new block header gets calculated and compared to the so-called difficulty.

The difficulty (as the name suggests) determines, how hard it is to create a new block in Bitcoins blockchain. The hash of the newly created block header gets compared to the difficulty and if it is smaller than the current difficulty, the block as a whole becomes a new valid Bitcoin block that can be propagated to other participants in the Bitcoin network. However, if the hash of the new block is larger than the difficulty, the mining continues until either another miner finds a new valid block, or until we find a nonce that fulfills our criterium.

The difficulty can be viewed as the required number of leading zero bits in the binary denominated format of the hash. Each other leading bit that is set to zero doubles the difficulty, as it is twice as hard to find a hash, which has a lower value than the difficulty. The more people participate in mining (more accurately: the more machines are mining and the better these machines are in terms of possible hashes/second), the faster a new block can be found.

This is also the reason, why the difficulty needs to be adjusted regularly, as there is a target time of 10 minutes between two blocks in Bitcoin [50]. As the chance of finding a block is probabilistic, the effective time between two blocks underlies heavy fluctuations.

Every 2016 blocks[8], the difficulty gets adjusted, depending on the time it took to find those blocks [56]. If the actual average time between two blocks was lower than 10 minutes, the difficulty gets adjusted to a higher level that would bring the block time back to 10 minutes. Similarly, it would be lowered to a level that would bring it back to 10 minutes, if the average time between two blocks would be too high.

So why is the difficulty and the PoW algorithm needed in Bitcoin? The reason lies in the details of what happens, after a miner finds a new block: she propagates the new block immediately to other participants (miners, Ticker Symbol of Bitcoin (BTC) owners, but also other participants that keep a copy of the blockchain) in the network. Each participant proves then, if the newly received block is valid, i.e. if it fulfills all requirements that are defined in the protocol. Some of these requirements are:

- Transactions: Are the signatures of the transactions correct? Is the public key that is associated with each Bitcoin address that has been used in one of the transactions correct? Is the owner of each Bitcoin address really allowed to spend that amount of Bitcoin (you can't spend more than you have)?
- First transaction (miner reward): Is it set to the correct value?
- Is the hash of the previous block set to the correct value?

[8]At a block time of 10 minutes that's ~2 weeks

- Is the timestamp of the new block later than the timestamp of the current last block and not too far in the future (not more than 2 hours)?
- And of course, most importantly for the PoW algorithm: Is the hash value of the block smaller than the current difficulty?

If a block is interpreted valid, the block gets added to the participant's internal record of the blockchain that includes also all previous blocks and the new block gets further propagated to other participants in the network. This way each participant receives the new block within a few seconds after it has been mined. Immediately after a new valid block has been added to the blockchain, miners continue to mine the next block, which references to the block that has just been found. Therefore the cycle repeats itself and block after block gets added to the blockchain.

What makes the blockchain secure is the PoW algorithm, as it makes it very hard for an attacker to manipulate it. If an attacker publishes a malicious block that does not fulfill all rules (e.g. if a block gets published where an address spends more Bitcoin than it has received in earlier transactions, or if a miner assigns herself a larger reward than she is allowed), this block simply gets rejected by other network participants and is therefore worthless.

The only possible attack vector would be to invest a significant amount of resources into mining equipment and power to mine blocks again at a position in the blockchain where blocks already exist. While propagating such a new block to the network would not hurt any consensus rules, it would still be quite hard for a malicious actor to succeed with such a strategy, because there is one more consensus rule in Bitcoin: the longest chain of blocks is always the valid version of the blockchain.

For an attacker to succeed, she would therefore need more mining power than all other miners together so that she can sustainably continue to publish her own (and by others as valid accepted) blocks. She would need to continue her effort of mining and publishing new blocks, until she has published enough blocks that her latest produced block makes the blockchain at least one block longer than the original chain, to make it the new valid chain. This is what is called a 51 % attack, referring to the required portion of all available mining power in the hands of an attacker [61].

Besides a 51 % attack, there are also other attack vectors that would be possible. A 51 % attack would require large amounts of capital to buy mining hardware, space (to setup the mining hardware), a power supply of at least 100MW [59] and a very well hidden procurement process of all those resources so that the public won't find out in advance. Furthermore, it would be very hard for an attacker to sell mined or stolen Bitcoins after such an attack, as trust in the network would immediately shrink dramatically and therefore also Bitcoin's price on exchanges.

Another attack vector would require much less resources, but can succeed only with a certain probability. It would also require an attacker to own a substantial share of

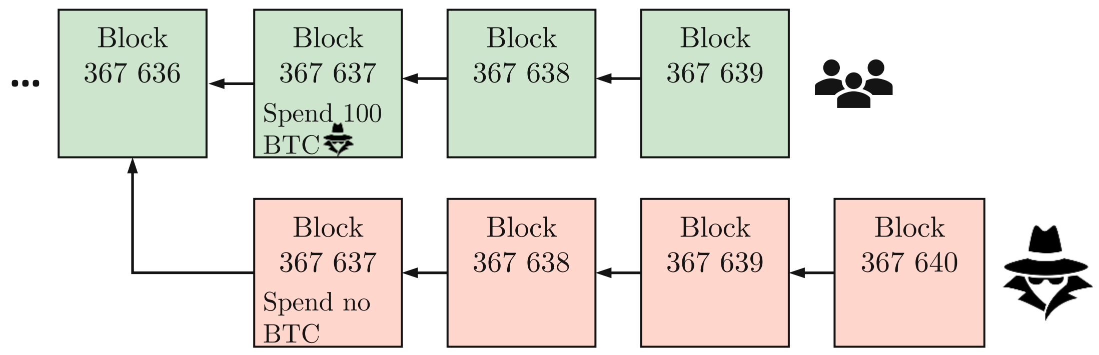

Figure 2.2: Double spend attack: A malicious actor spends 100 BTC in block 367 637 in the original block. Besides that she secretly mines valid blocks but does not publish them until she mines block 367 640 when the secretly mined blockchain becomes longer than the *original*[9] blockchain thus making the secretly mined blockchain the new valid version.

Bitcoin's total hashing power, but not 51 % of it. Figure 2.2 shows such an attack, the so-called double spend attack.

All in all, however, none of the above described attacks have been successfully happened on Bitcoin, which proves that the blockchain is indeed a secure decentralized database, which does not require a trusted third party.

2.2 Ethereum

Bitcoin has successfully proven that it is possible to manage digital assets in a decentralized manner even if none of the participants using the network needs to trust another. The approach of solving this problem was the invention of the blockchain. However, since the introduction of Bitcoin people started to think about how other applications could be implemented using this novel technology.

In this chapter we give a brief overview about Ethereum, which is a general purpose blockchain that offers smart contract functionality and is thus applicable to a much broader range of usecases.

In November 2013 Vitalik Buterin published the first version of the Ethereum whitepaper [58]. In the paper he describes Ethereum's vision the following way:

> "*What Ethereum intends to provide is a blockchain with a built-in fully fledged Turing-complete programming language that can be used to create "contracts" that can be used to encode arbitrary state transition functions, allowing users to create any of the systems [...], simply by writing up the logic in a few lines of code.*" [58]

This high level description of Ethereum already emphasizes the core difference between Bitcoin and Ethereum: While Bitcoin's blockchain builds the foundation for Bitcoin's usecase, namely the transfer of Bitcoins from one user to another, the Ethereum blockchain is not restricted to one application and is even Turing complete.

Buterin further argues that after the invention of Bitcoin a set of other applications evolved. Simplified, these applications have two different approaches:

- Implement the application by designing a new blockchain network. Famous examples include e.g. Namecoin, which offers a decentralized name registration platform [37]
- Build on top of Bitcoin. This includes e.g. simple applications like multi-signature wallets that make use of Bitcoin's scripting language

After the publication of the Ethereum whitepaper, an ICO for Ether tokens was organized during the summer months of 2014 and one year later, on July 30th, 2015 the genesis block of the Ethereum blockchain has been mined after designing and testing several clients.

In the years after Ethereum faced many ups and downs but its community and core developers of Ethereum clients proved that they are willing and capable to improve the blockchain further. Over time the community behind Ethereum was able to iterate from one version to the next, which was and still is one of the core difficulties that the Bitcoin community faces. In table 2.1 we added some of the core differences between the two most prominent blockchains.

Property	Bitcoin	Ethereum
Consensus algorithm	PoW	PoW, Proof-of-Stake planned
Native token	BTC	ETH
Target block time	10 minutes	13 seconds
Scripting capabilities	limited	Turing complete
#Client implementations	4	5
#Successful hardforks	0	11
Last successful hardfork	–	August, 5th, 2021

Table 2.1: Ethereum vs. Bitcoin

As the table highlights, Ethereum produces blocks in much lower intervals when compared to Bitcoin. Regarding the consensus algorithm, Ethereum still makes use of a PoW algorithm but has planned an upgrade to proof-of-stake, which is already planned since many years and will most likely take place in 2022.

Compared to Ethereum, Bitcoin has no history of successful upgrades in the form of hardforks[10]. While Ethereum has undergone 11 successful and planned hardforks since genesis, Bitcoin could only add minor changes that did not need a hardfork.

However, probably the most important differentiator between the two blockchains remains the capabilities of their scripting languages. As it was envisioned by the founders of Ethereum this capability allowed the development of a set of completely new applications which are now natively hosted on the blockchain. Those application can profit from the security of Ethereum, while having the freedom to arbitrarily develop business logic on-chain.

In the last years many projects that build on Ethereum were able to grow their new business models. This includes not only MakerDAO, but also different other applications: decentralized exchanges like Uniswap, lending protocols like Aave, marketplaces that allow the tokenization and trading of art like OpenSea or on-chain games are just a few examples.

[10] A hardfork of a blockchain marks the split of the chain in two parts, which comes from a disagreement about the validity of a block. They usually happen when a planned upgrade of the consensus rules are enforced by its users.

CHAPTER 3

MakerDAO

In this chapter we give an overview about the most important characteristics of the Maker ecosystem, including a brief overview of the main idea behind Maker, a description about the different smart contracts that build the backbone of the DAO and also discuss how governance is structured.

3.1 Protocol

The Maker protocol is designed in such a way that it can run as a decentralized autonomous organization on a public permissionless blockchain. The DAO's rules and behavior are implemented within a set of smart contracts on the Ethereum mainnet. The protocol was developed by the Maker Foundation[1], which was set up to bootstrap the development of the protocol until governance and further development decisions were completely handed over to the Maker community. This way MakerDAO became completely decentralized and thus independent from the Maker Foundation. In the future the success of the DAO lies thus in the hands of the Maker community.

Essentially the Maker protocol uses the instrument of collateralization to secure its system. More specifically, it employs proxy collateralization, which means that the DAI is not backed by USD (this would be direct collateralization with *off-chain* collateral, which cannot be tracked on the blockchain), but rather other assets that can be securely managed in a decentralized manner on the Ethereum blockchain. The reason behind this is the vision of the Maker community to maximally decentralize Maker in order to make it more transparent and trustable for everyone who is interacting with it.

When Maker started end of 2017 with its first implementation of the protocol, some careful design choices have been made in order to test the reliability of the whole system

[1]https://makerdao.com/en/team

while at the same time not putting too much risk on it. This first implementation was called SCD and like the name suggests, it only included one type of collateral, namely Ether [27].

While it was always planned to allow different types of collateral to back the DAI, the SCD implementation was used to test some core functionalities while the time was used to develop the MCD. As the name suggests, MCD is the full version of Maker that includes more types of collateral, ultimately making the whole system more stable against price fluctuations.

It is important to note at this point that the Maker protocol, like the whole Ethereum ecosystem itself was and still is under active development. This leads to the fact that it is per definition not possible to refer to *the* Maker protocol, as it depends heavily, which version of the protocol one is referring to.

Besides technical changes, also naming conventions have been adopted several times within the last years and further changes will most likely happen in the future. Within the scope of this thesis, we focus on MCD, more specifically on MCD that has been used during the first year after the introduction of MCD until November, 2020.

3.1.1 The Two-Token System

Maker uses two different ERC-20 tokens, whereby one token (called DAI) is utilized as the stablecoin and the other token, named Ticker Symbol of the Maker Token (MKR) is employed for governance and recapitalization purposes. DAI is Maker's core product, which is kept stable with a set of mechanisms powered by smart contracts. MKR, on the other hand, grants the holders of the token the right to vote for changes within the Maker ecosystem and therefore can be seen as something comparable to *shares* of the DAO. Besides that MKR also fulfills the purpose of recapitalization in case that a certain amount of system debt accrues (more in section 3.3.3).

3.1.2 Maker Vaults

Vaults are Maker's centerpiece for taking out debt in the form of DAI against collateral. A borrower can deposit different types of collateral into a Vault[2] and the Vault keeps track of how much has been deposited and who owns this collateral.

When a Vault owner wants to lend DAI against the deposited collateral, she has to lock a self-decided amount of the collateral as a security within the system. However, Maker only allows lending DAI, if the current collateralization ratio of this Vault is higher than the liquidation ratio of the respective collateral type.

As each collateral has a different exchange rate, the liquidation ratio is defined separately for each of the collateral types and can be adjusted by Maker governance. E.g. for a

[2]More specifically, for each collateral type and user Maker requires to use a separate Vault.

standard Ether Vault[3] the liquidation ratio is 150 % at the time of writing. This means for each Ticker Symbol of Ether (ETH), a maximum of 2/3 of the value of ETH in USD can be taken out as loan.

Locked collateral serves Maker as a security to guarantee that the DAI is always backed by enough collateral assets that have been deposited into the protocol's smart contracts. Users who take out loans in newly minted DAI can use those DAI for whatever they want, as DAI is an ERC-20 token, which can be transfered like any other ERC-20 token.

This way users can exchange DAI against other tokens, deploy an arbitrage bot that earns fees by trading against the DAI/USD exchange rate or even sell DAI at an exchange for fiat currencies or other off-chain assets. All this can be done without the need to sell the underlying collateral that has been locked in the Vault in order to mint the DAI and this leverage is also the incentive for the supply side of Maker's stablecoin.

To allow MakerDAO to finance the costs of governance and future development of the protocol on the long run, the DAO charges users who take out loans a so called stability fee. This fee accrues continuously on the loan that has been taken out and has to be payed by the owners of Vaults at the time when they want to pay back their debt in order to unlock their collateral. Just like the liquidation ratio, stability fees are defined separately for each collateral type and can be adjusted dynamically by the Maker community (MKR owners).

To make sure that the value of DAI is also guaranteed after the price of a collateral drops, the protocol has a liquidation mechanism in place, which allows that locked collateral can automatically be auctioned off, in case a Vault falls below the liquidation ratio. So-called keepers, which are external actors are incentiviced to participate in collateral auctions to get the chance to buy collateral under market prices. The keeper has to pay the collateral in DAI so that the total supply of the stablecoin can be reduced by the amount that was borrowed by the Vault owner in the first place and and will thus be burned immediately after the auction.

3.1.3 How to keep the price of DAI stable

Like other stablecoin projects as well, one of Maker's core tasks is to keep the value of the stablecoin close to the peg, in the case of Maker the USD. Different techniques to keep the exchange rate of a stablecoin close to the peg exist, but in general they are all based on the elementary economic model of supply and demand [40].

Hence, to adjust the price of the stablecoin, one can in theory adjust both the demand and the supply in such a way that the price of the stablecoin gets closer to the peg. In practice, supply is easier to adjust, although Maker has in its current implementation the possibility to adjust both demand and supply.

[3] Maker introduced several different types of Ether Vaults, which all use Ether as a collateral, however use different collateralization ratios, which also cost different fees. The most common Ether Vault type is still 'ETH-A', which is also the one that we are analyzing within the scope of this thesis. For better readability, we refer to these Vaults just as *Ether Vaults*.

- Supply: Market participants that own tokens that can be used as collateral within the Maker system are incentiviced to mint new DAI in case the price of DAI is above the peg or pay back debt in DAI in case that the price is below the peg
- Supply: Stability fee adjustments incentivize market participants to generate new DAI in case the stability fee gets lowered and to pay back debt in case the stability fee gets increased
- Demand: DAI savings rate adjustments are a means to increase DAI demand (in case of higher DAI Savings Rate (DSR)), or to decrease it (in case of lower DSR)

The DSR is an instrument that allows DAI holders to lock their DAI's without any further restrictions and earn a variable[4] interest rate on their stablecoin holdings. Locked DAI's can be redeemed at any time and accrued stability fees will be paid out. Interest gets financed by the collection of stability fees.

3.1.4 External Actors

The Maker protocol can only function, if external actors with different rights and duties interact with the protocol over the well-defined interfaces. This does not only include Vault owners and MKR holders, but also other actors, as shown in figure 3.1.

- Governors: are responsible for long-term success and stability of MakerDAO. Governors are in charge to decide a strategy that fulfills these goals
- Maintainers: are needed on a daily basis to keep the system running and stable
- Users: are the customers that profit from the services that MakerDAO provides for them

MKR Holders

MKR holders have the responsibility to gather information, make strategic decisions and vote for those decisions within the scope of executive votes[5]. These votes can change many of Maker's core parameters and even exchange whole contracts. If managed right, MKR holders can profit from the long-term growth of Maker's ecosystem indirectly by a falling supply of MKR tokens. This is done through smart contracts that are buying back and burning MKR tokens if the DAO creates enough profits (which are named *system surplus* in the Maker protocol).

[4]MKR holders can change the DSR
[5]`https://community-development.makerdao.com/onboarding/voter-onboarding`

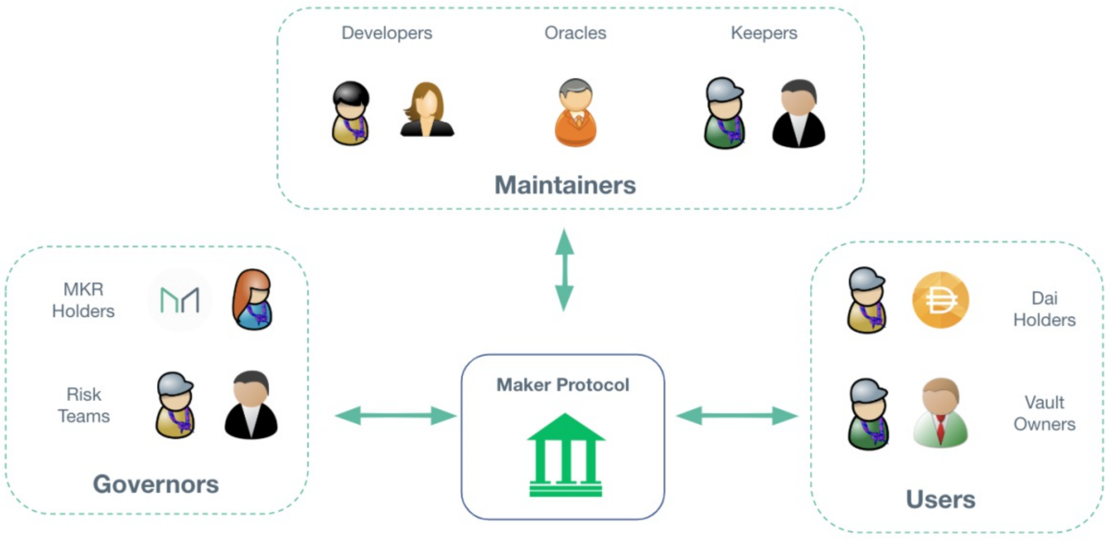

Figure 3.1: Maker's external actors [28]

Risk Teams

Maker's risk teams are a group of experts that were employed by the Maker Foundation and are working for the DAO now. They regularly assess the current state of Maker and suggest risk parameter adoptions if necessary. MKR holders can inform themselves about those proposals and base their decisions on them.

Developers

Developers are responsible to extend and improve the existing smart contracts of the Maker ecosystem. They have the responsibility to design reliable and safe business processes, which can be used by the DAO. MKR token holders have to vote for the contracts before they come into action.

Oracles

Oracles are crucial for MakerDAO, as they deliver the necessary price information about the different collateral types of the protocol. These price information is then used for automated decisions that are coded into Maker's smart contracts. If the price of a collateral token falls, the system needs to take actions in case that some Vaults fall below the liquidation ratio.

Keepers

Keepers are external actors (humans or bots) that are incenticed to earn profits mainly through auctions carried out by the protocol. This includes collateral auctions from

Vaults that have become undercollateralized but also MKR auctions, in case the protocol buys back or sells MKR tokens to reduce system surplus or system debt, respectively.

Vault Owners

As described in section 3.1.2, Vault owners use the Maker protocol to leverage their (qualified) token holdings by using them as collateral to borrow DAI.

DAI Holders

Holders of DAI tokens make use of DAI as a stable store of value, which is managed in a decentralized manner on the Ethereum blockchain. DAI holders don't have any obligations.

3.2 Decentralized Governance Process

Depending on the scope (political, corporate, NGO, ...), different definitions about what governance is, exist. Maker defines governance as

> "*... a process through which a group of stakeholders come to a decision on a particular change within their system or organization. [...] In the context of MakerDAO, it is done through a voting system where anyone who owns MKR can vote on changes to the Risk Parameters within the Dai Credit System.*"[29]

In general, governance within DAOs is a topic of ongoing research[49], as not only classical governance theory needs to be covered but also the dimension of blockchains. Within the Maker community governance is one of the most discussed topics, as managing governance in a completely decentralized manner comes with a range of problems. Especially participation in governance votes is a serious problem, as low participation can open up potential attack vectors against the protocol.[6]

Maker has organized its core governance process by letting MKR token holders vote for two different types of votes: *executive votes* and *governance polls*. Within the next lines we describe the difference between the two votes and also explain, what the so-called *governance facilitator* has to do with that.

3.2.1 Executive Votes and Governance Polls

Executive Votes

On a technical level executive votes are special purpose contracts that only get deployed to make changes of system parameters or business logic within the Maker protocol. These contracts use templates that are defined in the governance module. Maker employs a

[6] https://forum.makerdao.com/t/why-arent-mkr-holders-voting/301

voting mechanism named *continuous approval voting*[7], which means that votes have to be made regularly and a newly proposed executive vote only comes into effect, if it gains more votes than the previous vote. That means on the other hand that not voting for a new executive vote means automatically that the owners of MKR tokens that do not vote for the newly proposed change do not want the system to change and therefore effectively reject the executive vote.

In an executive vote one MKR token represents one vote and every user can also propose an executive vote that can include arbitrary changes of MakerDAO's smart contracts. MKR token holders can vote by interacting directly with a corresponding executive voting contract but can also make use of Maker's UI called Governance Portal[30], which has been set up to allow a more convenient usage of the voting process. [44] provides an overview of all ever held executive votes and governance polls, including more information about participation (e.g. how many voters participated, how much Ether has be staked per vote over time). Up until August 10th, 2021, 91 MakerDAO executive votes have been placed [45].

Governance Polls

Governance polls are used to get a better understanding of community sentiment without losing the property of just letting MKR token owners vote (which are also eligible to vote on executive votes). This means that governance polls are also placed on-chain and MKR token holders have the possibility to vote for one of several options to signal their opinion about a specific topic.

This tool is used e.g. to find out, if the community is in favor of a change to be made to the protocol or not and this is also the reason, why governance polls are made much more frequently. E.g. for critical changes that have to be carefully designed by the smart contract team of Maker, it makes sense to do a polling before a lot of resources are put into the development of a new mechanism that MKR token holders are anyway not willing to approve. Up until August 10th, 2021, 595 governance polls have been placed [46].

3.2.2 Interim Governance Facilitator and Governance and Risk Meetings

The role of the interim governance facilitator was introduced by the Maker Foundation in August, 2019 [47]. The reason why the role is called *interim* governance facilitator is that this role is hopefully not needed in the future, when the Maker community is completely self-organized and independent. The following role description has been provided by the Maker Foundation:

[7] `https://community-development.makerdao.com/governance/governance#what-is-continuous-approval-voting`

> *"Interim Governance Facilitators (IGFs) are tasked with ensuring the smooth operation of the governance process. This involves a wide range of activities, anything from general administration to signals gathering to governance scheduling."*[47]

As described above, the IGF has many responsibilities. It is important to note that the role description specifically points out that IGFs should not use their role to push the community in any specific direction and they should also not express opinions about governance topics to influence the community. To the contrary, it is the role of the IGF to lead the governance process, take over moderation tasks within Maker's community forum and to set up weekly *governance and risk meetings* where the community can discuss via video conference critical questions of potential protocol change that could be implemented in executive votes.

The governance and risk meetings take place once every week [26][8], except very urgent topics have to be discussed, e.g. after the Black Thursday price crash during March 2020. The meetings are used to discuss topics that are currently discussed in the community and besides the Maker community different guests ranging from Maker Foundation members, to external experts are participating to present their ideas, research or proposals for changes [26]. Also for the governance and risk meetings, it is the responsibility of the IGF to moderate and to try to find a consensus within the community.

The first ever elected IGF is Richard Brown[48]. Before he has already been head of community development at the Maker Foundation and after electing Richard with a government poll held from September 3rd, 2019 till September 10th, 2019 he became officially MakerDAO's first IGF.[9] In May 2020, a second IGF has been officially approved by the Maker community, this time even through an executive vote.[10] The 2nd IGF is a user who is in Maker's community forum known under the user name *LongForWisdom*[24]. He has also been an active member in Maker's forum before and enjoys the trust of the Maker community.

3.2.3 MakerDAO's Off-Chain Entities

Even though the Maker Foundation's goal was always to decentralize the protocol as much as possible, it is (at least currently) not possible to decentralize e.g. the management or creation of a foundation. However, the Maker Foundation, which was responsible for the development of the Maker protocol and which was used to organize Maker's initial fund raising was at the center of setting up the DAO. Besides the Maker Foundation, which at the time of writing already disassembled, there are also other *"off-chain"* entities that are needed for the DAO:

[8]At the time of writing, they happen every Thursdays 5PM UTC
[9]`https://mkrgov.science/poll/29`
[10]`https://forum.makerdao.com/t/mip0c12-sp2-subproposal-for-core-personnel-onboarding-governance-facilitator/2351/6`

Dai Foundation

The Dai Foundation is the legal entity that is the legal owner of all off-chain assets, which the Maker Foundation was holding for the DAO while the protocol was still under development. The Dai Foundation is a foundation under the Danish law, and it can never seek profits. In the foundation's trust deed its mission is defined as to...

> "*...safeguard certain intangible assets that underlie the MakerDAO system, which should not be taken over by or transferred to a single for-profit entity, but rather be used for sustainable growth of the MakerDAO system...*"[35]

End of December, 2019, the Maker Foundation announced that it has officially transfered the Maker and Dai trademarks to the Dai foundation. This marks the first of several steps of the foundation to hand over all assets that it owned and managed for MKR holders. The board of trustees currently comprises of 50 % members that are external and independent and 50 % members that are working at the Maker Foundation.[11]

Maker Ecosystem Growth Holdings, Inc.

The Maker Ecosystem Growth Holdings, Inc. is a corporation, which was founded to help the Maker ecosystem grow by providing services to the Maker community and users that make the protocol better usable.[12] This includes e.g. the development of Oasis[13], an online tool that allows Vault management as well as a trading platform to exchange e.g. DAI or MKR tokens against different other tokens.

3.3 Smart Contracts

Maker has organized its smart contracts in modules, whereby each module is responsible for a set of common objectives. A high-level overview of the different modules and how they are connected to each other is shown in Figure 3.2. Each box represents one module. The name of the module is written in bold and each contract that belongs to a module is listed below the name. Arrows symbolize function calls and show thereby how the different modules are connected to each other.

At this point it should be mentioned that the concise naming of the smart contracts and variables used within smart contracts was done to prevent terminological debates and to decouple financial and technical vocabularies [28]. While it takes some time to get used to the naming conventions, it helps to describe the protocol on a more abstract and formal level.

Next, we describe what each of the modules is responsible for before discussing the contracts in detail.

[11] `https://forum.makerdao.com/t/announcing-the-dai-foundation/1046`
[12] `https://blog.makerdao.com/introducing-the-maker-ecosystem-growth-group/`
[13] `oasis.app`

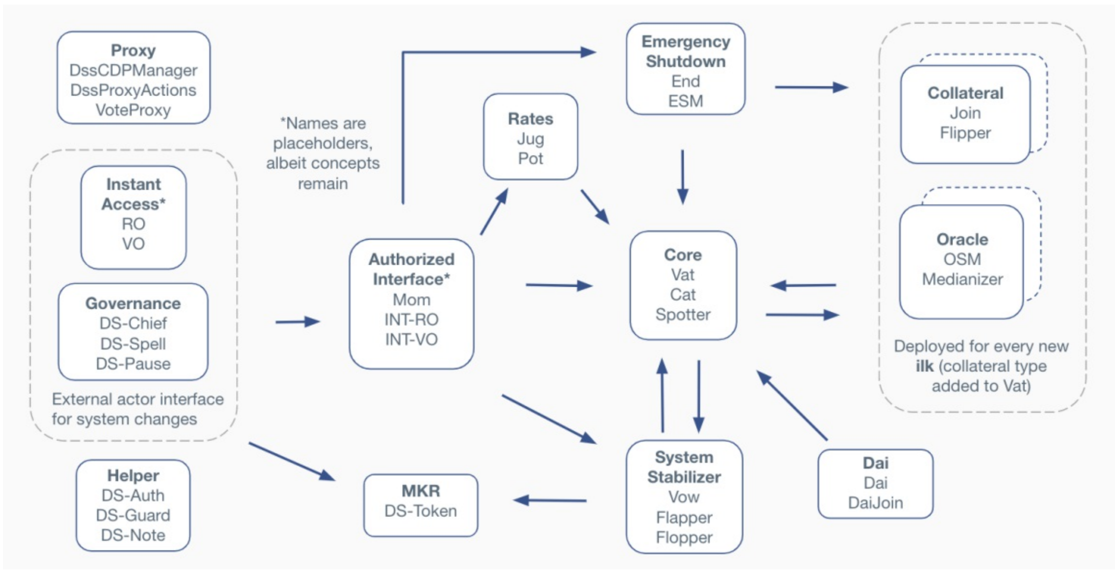

Figure 3.2: Overview of Maker's smart contract Modules [28]

As the name suggests, the Core module contains some of the most important data and logic of the protocol. This includes data about the Vaults, current collateral prices and the logic to identify Vaults that are undercollateralized. Price data comes from the Oracle module, which has functions in place that make sure that this data can not easily be manipulated.

The Collateral module contains logic to connect each of the collateral types to the system. Furthermore, it contains an auction house to auction off collateral from undercollateralized Vaults. Both a Collateral- and an Oracle module are deployed for every collateral type that gets accepted by the DAO.

The System Stabilizer module saves data about current system surplus- and debt. System surplus accrues e.g. through the collection of stability fees and system debt through collateral auctions that have not generated enough DAI to cover the outstanding debt. The Stabilizer module contains an auction house, which is used to decrease system debt through selling newly minted MKR tokens and another auction house to buy back MKR tokens for DAI (if system surplus exceeds a certain amount).

The Dai module contains data about DAI owners with its ERC-20 functions as well as the logic to connect it to the Core module. Furthermore, the minting of new DAI as well as the burning of DAI, which get exchanged against collateral, happens in this module.

Similar to the Dai module, the MKR module keeps track of the MKR token owners. As MKR is also an ERC-20 token, thus the module also contains all functions to allow ERC-20 operations. The module also includes a logic to govern the MKR token contract.

The Rates module contains smart contracts to manage the DSR as well as the logic to update the currently outstanding debt rate in the form of stability fees.

The Emergency Shutdown module provides the protocol with the ability to stop most of the system's normal processes in order to prevent an unforeseeable event like an attack on the system.

The Governance module implements processes that allow MKR token holders to decide on executive votes. It also contains a security contract that delays an executive vote for some time to prevent abuse of the Governance module.

The Helper and Proxy modules help to fulfill certain tasks such as authorization or proxy functions. The Instant Access module and the Authorized Interface module can be neglected for now, as they contain ideas of future development and are not part of the currently deployed protocol.

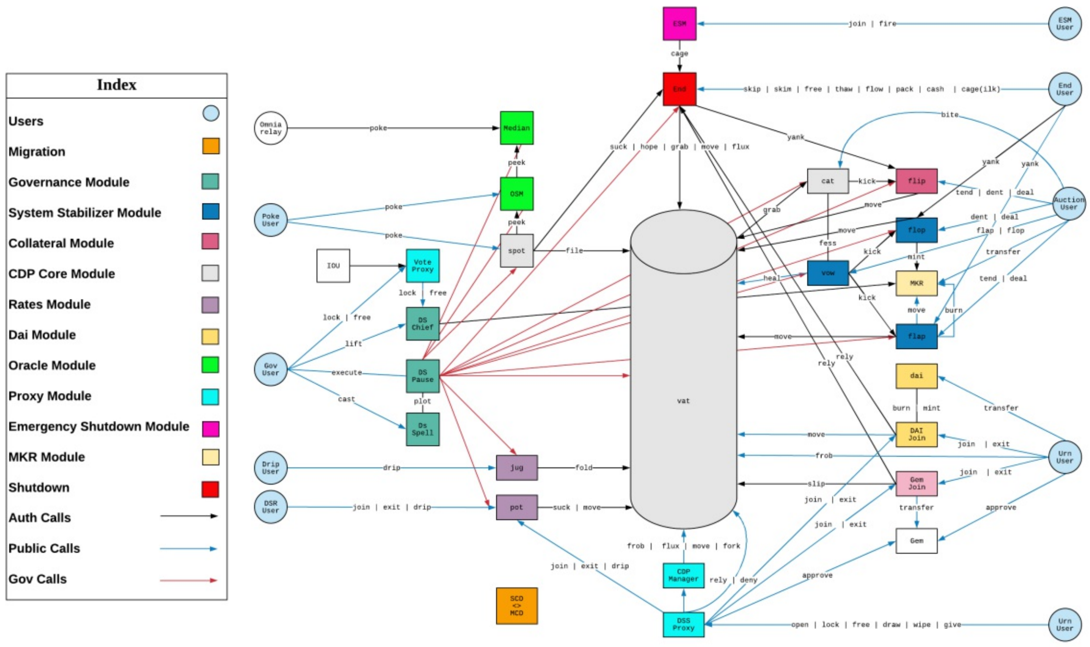

Figure 3.3: Maker protocol system diagram [32]

Figure 3.3 provides a more detailed view on the Maker protocol's smart contracts. While it is not important to understand every detail of the graphic, it should give an idea on how the Maker contracts are interacting with each other. Especially the importance of the `Vat`, a smart contract that is part of the Core module can be seen. It is crucial for the protocol as it stores the most important information of the Maker ecosystem and offers secure interfaces for other contracts that are needed to keep the system stable.

A detailed documentation of all modules and their respective smart contracts can be found in the Maker documentation [34]. Furthermore the source code of all contracts (written in Solidity[14]) can be found on Github [33].

3.3.1 Core Module

Vat

The `Vat`[15] (think of a large tank that holds liquid) keeps track of the most important data of the Maker protocol. This includes data connected to Maker Vaults (locked- and unlocked collateral balances, borrowed DAI balances and Vault ownership management) as well as functions to manipulate Vault states like collateral locking- and unlocking and Vault fungibility. The `Vat` contract was designed to be non-upgradable and only its accepted interfaces (in the form of accepted addresses for certain function calls) can be exchanged.

The `Vat` receives information about deposited collateral from outside adapters and keeps book of current collateral balances, with its variable `gem` of type `mapping (bytes32 => mapping (address => uint))`. This variable stores collateral balances of each collateral type (defined as `bytes32`) and `address` that has been defined when the corresponding join adapter (see section 3.3.2) has been called.

It is important to note that `gem` balances are only unlocked collateral balances. This means that they are not connected to any Vault and can therefore also be withdrawn at any time with the corresponding adapter.

Data about Vaults gets stored in a struct called `Urn`. Each `Urn` has two variables: `ilk`, the balance of locked collateral used as a security against lent out DAI and `art`, the normalized debt. Normalized debt is not debt that has been taken out in DAI, but rather an abstract representation of debt. The reason why debt is represented in a normalized way is to safe gas costs, because for calculating the real outstanding debt of each Vault, just one correction factor (`rate`) needs to be updated when stability fee adjustments happen.

Data about each collateral type get stored in another struct called `Ilk`. This struct includes five variables, the most important ones are `rate`, the just described correction factor used to calculate the real outstanding debt of each Vault and `spot`, which is the liquidation price in DAI. Furthermore, a variable called `line` defines the maximum withdrawable amount of DAI per collateral type, also called debt ceiling. The debt ceiling was introduced to prevent that one collateral type becomes too dominant in terms of total value of DAI lent out against it. This would ultimately lead to higher risk regarding the stability of the whole system.[36]

The `Vat` furthermore stores the two variables `dai` and `sin`, which are both of type `mapping (address => uint256)`. `dai` stores the internal DAI balances of DAI that have been created by lending them against locked collateral in an `Urn`, but that have not been minted with DaiJoin and are hence no ERC-20 tokens yet (see section 3.3.2).

[14]Solidity is the most used a high-level language for the implementation of smart contracts on Ethereum
[15]`https://github.com/makerdao/dss/blob/master/src/vat.sol`

`sin` balances on the contrary are system debt balances of DAI that have been lent out in a Vault in the first place but due to the fact that this Vault became undercollateralized, it is now seen as system debt. System debt can also accrue when DSR interest is paid out. It has to be mentioned though that in the current version of the protocol `sin` balances are only associated with the `Vow`'s address. More about the `Vow` and its responsibilities can be found in section 3.3.3.

The most important functions within the `Vat` are:

- `hope`: this function allows users to give allowance to other addresses to e.g. manipulate the `Vat`'s internal DAI or collateral token balances. Furthermore the `nope` function can remove the allowance again and the `wish` function returns, if an address has the allowance to call certain functions. While this mechanism does not require to use different addresses for Vaults, collateral balances and DAI balances, it allows users to effectively decouple e.g. Vault management (locked collateral, DAI balances) from the management of unlocked collateral balances.
- `frob`: manipulate the state of a Vault. This includes the possibility to change locked collateral balances (increasing or decreasing the amount of locked collateral `Urn.ilk`) as well as DAI balances (increasing or decreasing normalized debt `Urn.art` and the real debt `dai`).
- `grab`: confiscate collateral of an unsafe (undercollateralized) Vault and adjust internal balances accordingly. `grab` requires authentication and will be called by the `Cat` contract.
- `slip`: used to increase or decrease the unlocked collateral balance `gem`. This function requires also authentication and gets called by a `GemJoin` contract (any collateral type that fulfills the ERC-20 standard) or `ETHJoin` when collateral deposits or withdrawals take place.
- `fold`: update the `rate` of an `ilk`.
- `file`: `Vat` has two different `file` functions in place and both get used to update different parameters within the `Vat` like `spot` or `line`. Both file functions require authentication and will be used by Maker governance and `Spotter`.
- `move`: manipulate `Vat`'s `dai` balances by sending DAI from one address to another address. To move internal balances, the sender needs to approve the transaction first by calling the `hope` function.
- `flux`: like `move`, this function is used to manipulate `Vat`'s internal collateral balances `gem` by sending an arbitrary collateral tokens from one address to another address. Also with this function, the sender needs to approve the transaction first by calling the `hope` function, before the balances can be adjusted.
- `heal`: this function allows the `msg.sender` of this function to equate its `dai` and `sin` balances within the `Vat`.

Cat

The `Cat`[16] (think of a cat that bites Vaults that are too risky) contract has functions in place that allow keepers to confiscate collateral of Vaults that are undercollateralized. This means that it can take collateral from a risky Vault (one that is undercollateralized) and sell it via Maker's collateral auction house (Flipper) to cover outstanding debt.

The most important function within `Cat` is `bite`: this function allows keepers to confiscate collateral from an undercollateralized Vault and to kick off a collateral auction, which gets executed within the corresponding collateral auction house (`Flipper`, see section 3.3.2). Any user can call bite, by specifying the owner address of a Vault and the corresponding collateral type.

After a check, if the Vault is really undercollateralized, either all collateral or just a part gets confiscated by calling the `grab` function within the `Vat`. The amount that gets confiscated depends on how much collateral is locked within the Vault. For every collateral type, Maker governance can decide a maximum lot size `lump` of a collateral auction. If the locked collateral amount of the Vault exceeds `lump`, the auction that gets kicked off will just include an amount of `lump` collateral tokens. This is done to prevent that too large amount of collateral are put into auctions, which could lead to the effect that fewer keepers participate in collateral auctions and finally to collateral auctions that end at significantly lower bids than expected.

Like the locked collateral balance of the Vault also the normalized debt balance of a Vault gets manipulated by `Cat` when `grab` gets called. If the all locked collateral gets auctioned off, also the normalized debt balance will be set to zero. Otherwise, if just `lump` tokens get auctioned off, a proportional amount of normalized debt will be removed from the Vault, allowing the owner of the Vault to still pay back lent out DAI or increase locked collateral, unless another call of `bite` leads to another collateral auction.

After calling `grab`, the `bite` function registers the now removed real debt by calling the `fess` function in the `Vow` (see section 3.3.3). Finally the `Flopper` collateral auction house gets called via its `kick` function to start a new collateral auction. `Cat` specifies the amount of collateral tokens that will be auctioned off as well as the amount of outstanding DAI, including a liquidation penalty `chop` that gets added to every auction. Currently, the liquidation penalty is 13 % for all collateral types.[17]

Spotter

The `Spotter`[18] (to spot someone: *to physically assist them in safely completing a skill*) is the connector between the `Vat` and the `OSM` (*Oracle Security Module*, see section 3.3.4) that stores the current secured price feeds of collaterals.

[16]https://github.com/makerdao/dss/blob/master/src/cal.sol
[17]https://etherscan.io/address/0x78F2c2AF65126834c51822F56Be0d7469D7A523E#readContract
[18]https://github.com/makerdao/dss/blob/master/src/spot.sol

The most important and only function that is not restricted to authenticated access within `Spotter` is `poke`. A sender that calls this function needs to specify the collateral type (`ilk`) via a `bytes32` variable. The `poke` function uses the collateral type to call the `peek` function of the corresponding `OSM` contract that has been defined for the collateral type. After the `peek` function returns the current value of the collateral, it gets adjusted by the liquidation ratio `mat`, which is also defined per collateral type in the `Spotter`. Finally the `file` function of `Vat` gets called to update the price.

3.3.2 Collateral Module

Join

Maker has developed the join contracts `GemJoin` and `ETHJoin`[19] to define the rules for collateral deposits and withdrawals. The adapters (`GemJoin` for each ERC-20 collateral type and `ETHJoin` for Ether) are the *real* owners of deposited collateral. To prevent potential attack vectors, the contracts just have the absolutely necessary logic to receive and send collateral tokens, while the `Vat` takes care of the bookkeeping.

The functions that are not restricted to authenticated access are `join` and `exit`. As the name suggests, the `join` function can be used to move collateral to a join contract. For ERC-20 collateral tokens, the ERC-20 method `transferFrom` gets called within the `join` function in order to move tokens to the join adapter. For Ether transfers, where `ETHJoin` gets used, the `join` function has been defined `payable` in order to allow the contract to receive Ether.

The `exit` function on the other hand has been defined to allow users to exit their unlocked collateral tokens from Maker. Within `GemJoin` the `exit` function makes use of the ERC-20 function `transfer` to move collateral tokens to an user-specified address. Similarly, in the `ETHJoin` contract, a simple Ether transfer is triggered in the `exit` function.

Both `GemJoin` and `ETHJoin` use the `Vat`'s `slip` function to adjust unlocked collateral balances within the `Vat`.

Flipper

The `Flipper`[20] (to flip = *to turn over*) contract is Maker's collateral auction house, which auctions off collateral of Vaults that became undercollateralized. Like the `GemJoin` adapters, there exists one `Flipper` contract for each collateral type. To participate in an auction, bidders have to bid DAI, which gets burned after the auction to guarantee that DAI is always backed by enough collateral. Before keepers can participate in a Flipper auction, it needs to get `kicked` off by `Cat`, which again can only register a new auction, if a Vault falls below the liquidation ratio.

[19] `https://github.com/makerdao/dss/blob/master/src/join.sol`
[20] `https://github.com/makerdao/dss/blob/master/src/flip.sol`

Data about auctions are saved within the `Bid` struct and every time a new auction gets `kicked` off, a new `Bid` gets created and assigned the unique identifier `id`. The `Bid` struct contains the following fields:

- `uint256 bid`: currently highest bid in DAI for the collateral in auction
- `uint256 lot`: amount of collateral that will be auctioned
- `address guy`: address of the last bidder (that placed a bid of `bid` DAI)
- `uint48 tic`: timestamp until another bid must be placed, otherwise the auction ends. Gets rewritten each time a new bid comes in and gets calculated by adding a fixed amount of time (`ttl`, adjustable via Maker governance) to `now`
- `uint48 end`: timestamp, at which the auction ends latest. If `tic` ends earlier and no further bids are placed, the auction can also end. Will be defined, when a new auction gets kicked off and gets calculated by adding a fixed amount of time (`tau`, adjustable via Maker governance) to `now`
- `address usr`: address of the Vault owner that has been liquidated. This is needed to return remaining collateral tokens, if the outstanding DAI debt of the Vault can be covered by the income of the auction.
- `address gal`: receiver of the auction income, currently the `Vow` contract (see section 3.3.3)
- `uint256 tab`: the amount of DAI that should be raised during the auction (debt from the Vault + liquidation penalty)

At the time of writing, both the bid duration `ttl` and the auction length `tau` are set to 6 hours[21]. The parameters have been changed after the Black Thursday Ether price crash which had serious effects also on the Maker ecosystem (more on that see section 4.4). An executive vote that changed the parameters has been placed March 13th, 2020.[22]

A collateral auction is organized in two phases and keepers can participate in each phase by calling the respective functions with the same name: `tend` and `dent`.

The `tend` phase of a collateral auction is the first phase where keepers are bidding with increasing amount of DAI for all collateral (`lot` tokens) of the auction. Keepers need to submit the `id` of the auction as well as the `bid` in DAI when calling the function. If the `bid` is at least 3 % (at the time of writing, see contract variable `beg`) higher than the old bid, or if it is exactly `tab` DAI, the bid gets placed and the `move` function within `Vat` gets called to move `bid` DAI tokens from `msg.sender` to the `gal` address and `move` the amount of DAI that have been bidden in the bid before back to the previous bidder.

[21] `https://etherscan.io/address/0xaa745404d55f88c108a28c86abe7b5a1e7817c07#readContract`

[22] `https://mkrgov.science/executive/0x529b8b4b62b5f32bd47412988a0a66d72f86ba00`

The 2nd phase called `dent` can only start, when the `tend` phase reaches the amount of DAI that should have been reached to cover the outstanding debt `tab`. However, during the whole auction, it is not possible to bid more DAI than `tab`. To the contrary, if during the `tend` phase a bid of `tab` tokens has been placed, the auction changes and new bids now have to be made by calling `dent` and by offering decreasing amounts of collateral tokens that keepers are willing to receive for a fixed price of (`tab`) DAI that they have to pay.

The reason for the complexity of the auction mechanism is that Maker wants to offer Vault owners that became undercollateralized a fair auction process, which takes just as much of their collateral tokens away as is needed to cover the outstanding debt. Similarly as during the `tend` phase, new `lot` bids have to be at least 3 % smaller than the previous bid, in order to be accepted.

When an auction ends (if `tic < now` or `end < now`) and there has been at least one bid, the `deal` function can be called and `lot` amount of collateral tokens will be moved to `guy` by using `Vat`'s `flux` function.

3.3.3 System Stabilizer Module

Vow

The `Vow`[23] (a vow = *a solemn promise*) contract is responsible to manage Maker's on-chain surplus and debt. Maker calls these variables *system surplus* and *system debt* balances. They arise from stability fee and collateral auction earnings (system surplus) as well as through Vault liquidations and DSR payouts (system debt).

Technically, the `Vat` keeps track of the total system surplus by manipulating the `dai(address(Vow))` variable. Likewise, system debt is saved in the `sai(address(Vow))` variable. It is the responsibility of the `Vow` to equate system surplus and system debt.

If there are too big differences between system surplus and system debt that can't be evened out, the `Vow` has to call the `kick` function on the `Flapper` (system surplus auction house) or `Flopper` (system debt auction house), respectively. The auction houses are allowed to buy back and burn or sell and mint new MKR tokens, to decrease system surplus or system debt.

So how does the `Vow` work internally? Every time, `Cat bites` a Vault, which has fallen below the liquidation ratio, it calls `Vow`'s `fess` function to inform the `Vow` about the new system debt. The `Vow` keeps track of this new debt by adding it to its internal variables storing the so-called *debt queue*. The reason that debt is saved in a queue is that Maker wants to give the `Flipper` auction house time, to auction off the seized collateral to cover the outstanding debt before the `Vow` starts a `Flopper` auction.

[23] `https://github.com/makerdao/dss/blob/master/src/flip.sol`

After a (by Maker governance adjustable) time `wait`, the debt can be removed from the debt queue by calling the `flog` function. Currently, the waiting time is set to 6.5 days.[24]

If there is a lot of debt removed from the queue, and if that debt can't be canceled out by system surplus, a `Flopper` auction can be kicked off by calling the `Vow`'s `flop` function. To check if the system surplus has been really canceled out, this function requires that the system surplus is zero at this point, meaning that it has been set off against the system debt. This can be done by calling `Vow`'s `heal` function before, which again calls the `Vat`'s `heal` function.

If the now remaining system debt (`sai(address(Vow))` minus system debt that is still in the queue (`Sin`) minus system debt that already has been put into a `Flopper` auction is larger than a minimum debt `sump`, a `Flopper` auction will be `kicked` off by the `flop` function. Likewise, if there is just debt in the debt queue, or even no debt at all and if the system surplus reaches the system surplus buffer `hump`, a `Flapper` auction will be `kicked` off by the `flap` function.

Flapper

The `Flapper`[25] (a flap (of a bird) = *the move of the wings to fly or start flying*) auction house has the responsibility to buy back MKR tokens in case that a certain level of system surplus has been reached. The `Flapper` contract has similar functions and behavior as `Flipper`, but its internal logic is simpler e.g. because there is just one auction phase (`tend`). `Flapper` buys back MKR tokens for a fixed amount of DAI and burns them after the auction. Keepers can bid increasing amount of MKR that they are willing to sell against the DAI .

The following variables have exactly the same purpose as in the `Flipper` contract: `beg`, `tau` and `ttl`. Also like the `Flipper`, the `Flapper` has a `kick` function as well to allow authenticated addresses to start a new auction. The function gets only called by the `Vow`, who keeps track of system surplus and system debt. Furthermore the `Flapper` also has a `tend` function which allows keepers to place increasing amounts of MKR bids against the DAI that can be exchanged against the MKR tokens after the auction.

The `deal` function allows the keeper to retrieve the bought DAI tokens, once the auction has finished. Also the by the `Flapper` now acquired MKR tokens get directly burned in this function.

Flopper

The `Flopper`[26] (to flop = *to be completely unsuccessful*) auction house is the counterpart of the `Flapper` and has therefore the responsibility to mint and sell new MKR tokens in case that a certain level of system debt has been reached.

[24] `https://etherscan.io/address/0xA950524441892A31ebddF91d3cEEFa04Bf454466#readContract`

[25] `https://github.com/makerdao/dss/blob/master/src/flap.sol`

[26] `https://github.com/makerdao/dss/blob/master/src/flop.sol`

The `Flopper` contract has similar functions and behavior as the `Flipper`, but its internal logic is simpler e.g. because the contract has just one auction phase (`dent`). `Flopper` auction's off newly minted MKR tokens against a fixed amount of DAI. Keepers can bid decreasing amount of MKR tokens that they are willing to receive in exchange against the fixed amount of DAI specified by the contract.

The following variables have the same purpose as the ones in the `Flipper` contract: `beg`, `tau` and `ttl`.

Like the `Flipper`, also the `Flopper` has a `kick` function that allows only authenticated addresses to initiate a new auction and it can also only be called by the `Vow`.

3.3.4 Oracle Module

The oracle module consists of the `Median` and `OSM` contracts and is responsible for the on-chain part of Maker's oracle infrastructure. Figure 3.4 presents a high level overview of both Maker's on-chain, as well as off-chain oracle architecture.

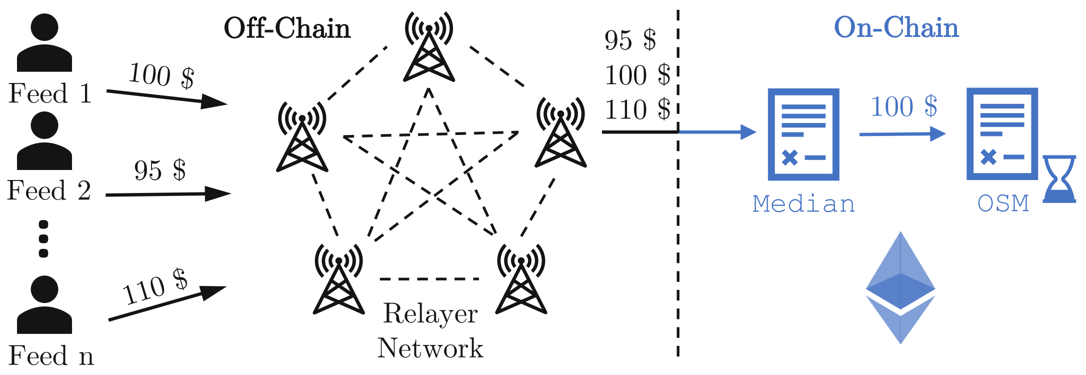

Figure 3.4: Maker's oracle architecture. On the left side (in black) are the involved external actors, on the right side the on-chain architecture (in blue) [21], [28].

The data sources (called oracle feeds) that provide price updates are a set of individuals, DAOs and enterprises, which are chosen by the community. Each data feed has an address assigned to it and only this address can sign messages with price data that is then further used on-chain.

Median

The `Median`[27] contract is, as its name suggests, responsible for identifying the median value from a set of values. For Maker this contract fulfills the duty to identify the median value from a set of collateral price values, provided by ratified oracle feeds.

The reason why a median value of a the set of provided price updates is used instead of the probably more accurate average value has to do with a security consideration. By

[27]`https://github.com/makerdao/median/blob/master/src/median.sol`

using the median value of all feeds, an attacker would need to gain control over half of the oracle feeds to successfully manipulate Maker's oracle infrastructure. This requires a more sophisticated attack compared to just manipulating one or two oracles (which might already be a lucrative attack if the average value would be used instead).

Each collateral type within Maker's ecosystem has its own `Median` contract, which logs the median price of the sorted set of price values each time its external `poke` function gets called. This function proves, if the relayer network that collects the feeds price updates (see figure 3.4) has sent all relevant data and fulfills the following properties:

- The required amount of feeds has provided price data (`bar` variable[28])
- Only whitelisted and thus eligible oracle feeds have submitted price data (this is checked by validating the signatures)
- The timestamp, which has to be submitted by each price feed and price update is greater than the `block.timestamp` of the last successful `poke` function call
- The price values have been provided in the correct order (ascending)
- None of the submitted price updates has been signed by more than one oracle feed

If all tests pass, the function then saves the median of all submitted prices (thus the number of price feeds that is required for a successful function call always needs to be odd) and also emits the `LogMedianPrice` event that logs all price updates of the `Median`.

Other important functions of the `Median` contract include the `peek` function, which allows the `OSM` to read the latest collateral price as well as the `lift` and `drop` functions, which allow MKR governance to add or remove oracle feeds by providing their address.

OSM

The `OSM`[29] (Oracle Security Module) contract's job is to add a delay between the price feed from the `Median` contract and Maker's other contracts. For each collateral type one `OSM` instance has to exist, which reads the collateral price from the corresponding `Median` contract. Delayed collateral prices are then further used by the `Spotter` contract.

The delay (defined in variable `hop`) that is introduced by the `OSM` is a necessary security component for several actors within the Maker ecosystem: It allows Vault owners to react in time and adopt collateral and/or debt balances, before a liquidation can be triggered.

The external function `poke`, which is callable by anyone is the `OSM`'s most important function. It can only be called once every hour and receives the latest collateral price by calling the `Median` contract's `peek` function. The latest collateral price gets stored in a variable called `nxt`, while the collateral price value of the last `poke` call gets written from `nxt` to `cur` just before that.

[28]E.g. for the ETHUSD oracle, the bar variable is set to 13 at the time of writing.
[29]`https://github.com/makerdao/osm/blob/master/src/osm.sol`

CHAPTER 4

Quantitative Analysis

This chapter contains the quantitative analysis of the thesis. This includes mainly an on-chain data analysis of Maker's MCD smart contracts and how they have been used since deployment. Based on the MakerDAO protocol architecture presented in chapter 3, we filter for the right data and identify weaknesses and strengths of the protocol.

First, we briefly describe how and where we gather and interpret the data. After that we have a close look on the Black Thursday incident that took place in March, 2020.

4.1 Protocol Requirements and Derived Metrics

Based on the description of the core mechanics of the protocol (chapter 3), we derive the following requirements that need to be fulfilled in order to guarantee a reliable operation of the protocol regarding a steady overcollateralization of the DAI as well as a healthy and functioning liquidation system.

For the definition of the requirements and metrics we have used and extended our own work [21][1].

1. *Auctions are able to cover the outstanding DAI debt.* With the collateralization required by the protocol, most auctions should actually cover the debt so that it is guaranteed that every DAI is covered by enough collateral at any time.

2. *Auctions achieve adequate prices for the collateral.* Besides the buyback of DAI, which is the most important task of the `Flipper` auctions, a fair price for the auctioned collateral should be achieved so that Vault owners can get back any excess collateral that has not been needed in the auction (Besides the liquidation penalty of 13 %).

[1]See section 1.2.3

3. *Keepers liquidate Vaults quickly after they have fallen below the liquidation ratio.* Falling collateral prices require fast reaction so that the outstanding debt can be covered. This is especially important in this sector, as the past has shown that cryptocurrency prices can decline dramatically in a short amount of time.

4. *Vaults threatened with liquidation adjust their collateralization in time.* The optimal case for Maker would be that Liquidation of a Vault not only carries the risk of ineffective auctions, oracle failures or smart contract bugs, but also incurs a liquidation penalty of 13 %.

5. *Vault owners maintain a collateralization ratio at double the liquidation ratio on average.* Usual price fluctuations of the collateral should not threaten a Vault with liquidation, hence double the collateralization is a safeguard that allows especially owners of not software-managed Vaults to act in time to prevent a liquidation.

Especially the latter two requirements are not strictly necessary, as Vaults that fall below the liquidation ratio get liquidated anyway. Given that the first requirement is fulfilled, the then collected collateral gets auctioned off at a reasonable price which keeps the protocol and the DAI stable and safe. However, mass liquidations can lead at least temporarily to high DAI prices and thus inefficient auctions. Moreover, if keepers can not provide enough liquidity for auctions, the DAI can become undercollateralized which can force the whole protocol in a dangerous situation. Therefore, we include the latter two requirements to this list as well.

We hypothesize that Maker fulfills these requirements for the study period. To test the hypotheses, we define the following metrics.

4.1.1 Metrics

We define the following variables that are used in several of our metrics:

For a time t and a Vault v, we define $\mathrm{ETH}_v(t)$ to be the amount of ETH locked and $\mathrm{DAI}_v(t)$ to be the amount of DAI taken as a loan. To determine $\mathrm{ETH}_v(t)$ and $\mathrm{DAI}_v(t)$, we accumulate all movements of collateral and DAI in and out of the Vault, by observing the calls `frob`, `fork` and `grab` to the `Vat` contract.

Furthermore we define `OSM`(t) to be the ETH value in USD as reported by the event `LogValue` of the `OSM` contract.

M1 *Bite events per week, per day or per hour:* We count the `Bite` events emitted by the `Cat` contract per week (with Monday as start of week), per day or per hour.

M2 *USD price of ETH:* By `Median`(t), we denote the value of ETH in USD as reported by the `LogMedianPrice` events of the `Median` contract. `Median`(t) reflects the external value of ETH at time t.

M3 *Auction result:* The result of an auction a is the ratio of DAI received to ETH sold.

$$\mathrm{AR}_a = \mathrm{DAI}_a/\mathrm{ETH}_a$$

M4 *Fraction of Auctions that reached dent phase:* Auctions that reached the `dent` phase of a `Flipper` auction have successfully covered the outstanding DAI debt of a Vault. We define A to be the set of all auctions and D to be a subset of A which contains only auctions that reached the `dent` phase. The fraction of auctions that reached the `dent` phase is the ratio of the cardinality of set D to the cardinality of set A.

$$\mathrm{FD} = n(D)/n(A)$$

M5 *Collateralization ratio of Vaults:*

$$\mathrm{CR}_v(t) = \mathrm{ETH}_v(t) \times \mathtt{OSM}(t)/\mathrm{DAI}_v(t)$$

M6 *Auction effectiveness:* The effectiveness of an auction a relates the DAI received to the value of the sold ETH at the time t_a, when the auction ends.

$$\begin{aligned}\mathrm{AE}_a &= \mathrm{DAI}_a/(\mathrm{ETH}_a \times \mathtt{Median}(t_a)) \\ &= \mathrm{AR}_a/\mathtt{Median}(t_a)\end{aligned}$$

We determine DAI_a, ETH_a and t_a from the log data of the auction `Flipper` contract, in particular from the calls to the functions `tend` and `dent`.

M7 *Liquidation delay:* The liquidation delay for a Vault is the time between the Vault becoming under-collateralized and the Vault getting liquidated. Technically, we measure the time between the block, where the `OSM` contract emits an event `LogValue` with an ETH price that drives the Vault below the liquidation ratio, and the block, where the `Cat` contract emits a `Bite` event for this Vault, with the Vault remaining undercollateralized in-between.

M8 *Vault management agility:* For every Vault that got liquidated v_{liqu}, let $\mathrm{DAI}_{v_{liqu}}$ be the amount of DAI that was taken out as a loan before liquidation. Furthermore, for every v_{sav} let $\mathrm{DAI}_{v_{sav}}$ the amount of DAI that was taken as a loan at the time when the Vault became undercollateralized.

$$\mathrm{VMA} = \sum \mathrm{DAI}_{v_{liqu}}/(\sum \mathrm{DAI}_{v_{liqu}} + \sum \mathrm{DAI}_{v_{sav}})$$

M9 *Overall collateralization ratio:* For a time t we define the overall collateralization ratio to be the the sum of all ETH locked in Vaults, multiplied by the ETH price as reported by *cOSM* and divided by the sum of all DAI locked in Vaults.

$$\mathrm{OC}(t) = \sum \mathrm{ETH}_v(t) \times \mathtt{OSM}(t)/\sum \mathrm{DAI}_v(t)$$

4.2 Data Source

For the quantitative analysis we make use of the publicly available data of the Ethereum mainchain. Different 3rd party tools like [19], [18] or [16] provide APIs to access data from the Ethereum mainchain in a structured and sometimes even already aggregated manner.

However, there is no guarantee that the data provided by these companies is complete or accurate. While the chances most likely are low that data is missing or incorrect, there is a theoretical risk that cannot be eliminated.

Running an Ethereum 1.0 full node with a client like [2], [15], [25] or [38] is a much better solution, but requires disk space[2] and at least some processing time to fully sync the chain.

Due to other research in the Ethereum ecosystem, we already have an Open Ethereum [25] client that provides data for the research questions of this thesis. Nevertheless, we make use of Anyblock's [16] analytics tool[3] for the thesis, as we aim to test a 3rd party data provider anyway and can thereby also save some computing resources on our local Open Ethereum client.

Before using Anyblock's service for our research we try several example query on both our own machine and the Anyblock API to test, if we receive the same data from both queries. As expected the outputs match after reconciling them.

4.2.1 Analyzed Collateral Type

When Maker's MCD started, only two collateral types where available right from the beginning: ETH and Brave's Basic Attention Token BAT [5].

Within the study period, Maker added step by step several more collateral types, but the dominant one (in terms of minted DAI per collateral type) remained ETH by far. Due to that reason and due to the fact that ETH is beside BAT the only asset where we have data available throughout the whole study period, we decided to limit our analysis to ETH.

4.2.2 ETH/USD Exchange Rates

For some parts of our analysis we need historical ETH/USD exchange rates in a finer-grained resolution than just daily prices (which are freely accessible from many sources).

However, as other sources that we examined require a paid subscription, we decided to make use of Maker's internal oracle price feed `Median`, which publishes prices on a regular basis (if prices change fast even several times per hour).

[2] At the time of writing at least 350Gb[14]

[3] `https://www.anyblockanalytics.com/docs/sql/schema/`

One downside of this approach is that the `Median` contract makes due to security considerations use of a median value of a set of data feeds and thus does not produce a weighted value that might be more accurate. [17] however points out that Maker's oracle price feeds do not deviate much when compared to a volume-weighted average price index.

4.2.3 MakerDAO Contract Versioning

Since MakerDAO's MCD went live, Maker released many upgrades to its protocol to increase security or to adopt to the changing environment (this includes e.g. changes to the stability fee).

Most of Maker's executive votes were set up to enforce just minor protocol changes like risk parameters. Some changes, however, need a redeployment of the contract on the blockchain in order to effectuate them. For those changes, Maker maintains a change log [31] where new versions and the deployment addresses are tracked.

For the data collection procedure of this thesis, we keep track of these versions in order to collect the right data and even more important that there is no data missing. Table 4.1 presents the addresses and versions of Maker contracts that we used in for the quantitative analysis and that were used by the protocol within the study period.

Contract	Version	Address
Vat	≥1.0.0	0x35d1b3f3d7966a1dfe207aa4514c12a259a0492b
Median ETHUSD	≥1.0.0	0x64de91f5a373cd4c28de3600cb34c7c6ce410c85
OSM ETHUSD	≥1.0.0	0x81fe72B5a8d1a857d176c3e7d5bd2679a9b85763
Cat	≥1.1.0	0xa5679c04fc3d9d8b0aab1f0ab83555b301ca70ea
	≥1.0.0	0x78f2c2af65126834c51822f56be0d7469d7a523e
Flip_ETH_A	≥1.1.0	0xf32836b9e1f47a0515c6ec431592d5ebc276407f
	≥1.0.9	0x0f398a2daaa134621e4b687fccfee4ce47599cc1
	≥1.0.0	0xd8a04f5412223f513dc55f839574430f5ec15531

Table 4.1: Maker's mainchain smart contract addresses based on protocol version [31]

4.3 Liquidations - Overview

To get a first understanding how often Vaults[4] get liquidated, we start with an analysis, how often the `Cat`'s `bite` function gets (successfully) called. The `bite` function can be called successfully when it is undercollateralized. In the same transaction some or all

[4]In general Maker Vaults can store any collateral type that has been onboarded by the DAO. However, as we limit our analysis to ETH Vaults, the term 'Vaults' will be used when we refer to 'ETH Vaults' from here onwards.

collateral (depending on how much collateral is stored within the Vault) gets removed from the Vault and a corresponding `Flipper` auction is kicked off.

To analyze liquidations, we extract all `Bite` events that got emitted during the study period (query A.1) and filter by liquidations that effected only ETH Vaults. Figure 4.1 presents this data aggregated on a weekly basis (M1).

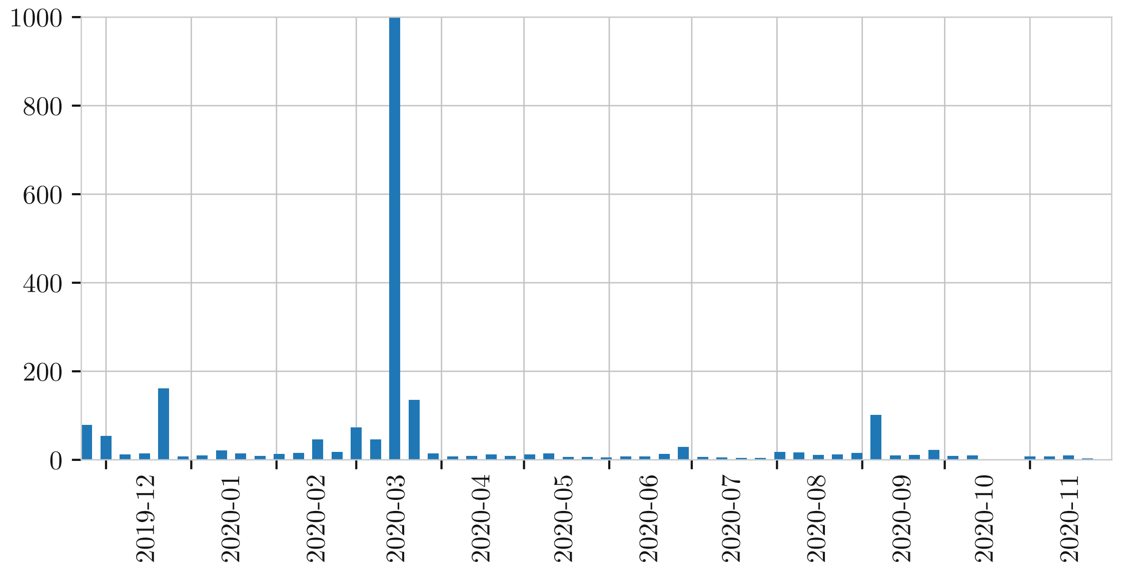

Figure 4.1: Bite events per week. The bar for the week of Black Thursday is clipped, with the actual value being 4574. [21]

As we see in the figure, the number of `Bite` events per week was never higher than 200, except in one week in March 2020, when it spiked far higher to a level of 4574. This week was also the week when the Black Thursday incident happened. The comparison to all other weeks during the study period already gives a hint on how severe this event was.

In a next step, we analyze, how this particular week (March 9th, 2020 - March 15th, 2020) looks like on a daily basis. We plot Ether Vault `Bite` events of this week in figure 4.2. We also add the total number of `Bite` events to this figure to give an idea, how much of all outstanding DAI was backed by Ether compared to other collateral types.

As we can see in figure 4.2, `Bite` events spiked on March 12th and 13th. March 12th was the day that was later called *Black Thursday* and it was also the day where Ether lost most of its value.

In the next section we will describe in more detail what led to the mass liquidations that had a big influence on the users of Maker as well as the protocol itself.

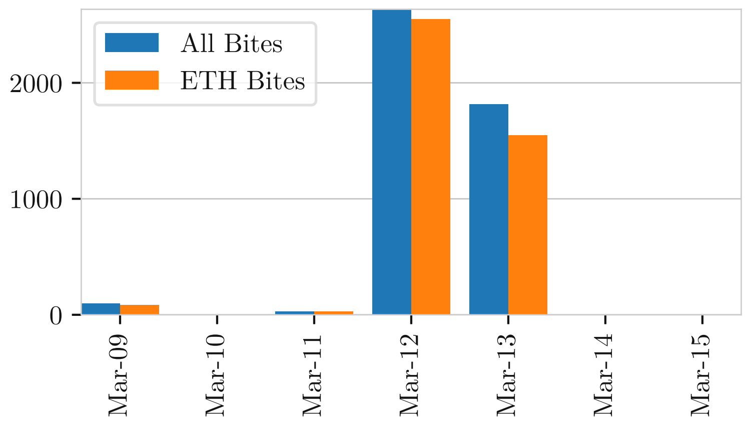

Figure 4.2: Maker Bite events in calendar week 11, 2020

4.4 Black Thursday

On March 12th, 2020 stock markets as well as other assets like cryptocurrencies were depreciating heavily in value from one day to another. Referring to the largest stock sell-off in history which took place during the Great Crash in 1929, newspapers and analysts started to call this day *Black Thursday* of 2020.

The decline was caused by the panic around the COVID-19 pandemic and led for some of the cryptocurrencies to a drop of exchange rates of 50 % and more [22]. Ethereum's native currency Ether was no exception and dropped by more than 30 % during just a few hours [9], which had of course also an influence on Maker.

As we described in Chapter 3, Maker's Vaults have to be overcollateralized at any time with a liquidation ratio, which can be defined for each collateral type. For Ether the liquidation ratio is 150 %, which means that each Vault that uses Ether as collateral can be liquidated by anyone by calling the `bite` function of `Cat` contract, as soon as the Vault falls below this ratio (more details in section 3.3.1). As a prevention, Vault owners can pay back outstanding debt (DAI) or increase the amount of collateral of the Vault, once the Vault becomes at risk to get liquidated due to falling prices of the collateral.

Due to the fact that Maker's price feeds are delayed by one hour, Vault owners generally have enough time to react to falling prices, at least when they are running an automated bot that reacts accordingly and pays back debt or adds some more collateral to the Vault in case the Vault becomes undercollateralized with the next price update.

On March 12th, 2020 however, this was not the case. The sharp decline of Ether's market price forced many Ether-based Vaults below the liquidation ratio, before their owners could react, effectively leading to thousands of Ether collateral auctions that have been kicked off by Keepers.

During this time Ethereum was heavily congested due to the price decline and automated bot transactions, which were triggered by the decline (Ether holders that wanted to liquidate their positions as well as users of other Decentralized Finance (DeFi) protocols), led to higher gas prices, resulting in many transactions not being mined, which prevented Vault owners from acting against the collateral liquidations as they could not pay back DAI debt or increase collateral.

But an even bigger problem were some `Flipper` auctions that ended at a bidding price of almost zero DAI. Due to the congestion of the network and the short auction duration `ttl` of just 10 minutes, defined in the `Flipper` contract, some Keepers got the chance to buy large amounts of Ether for almost no money [9].

In the aftermath of the Black Thursday incident, it turned out that there is a chance that the zero bid auctions where an engineered event, forcefully precipitated through targeted mempool attacks. According to an analysis done by [8], it seems that the attack was made possible through a combination of several factors:

- Due to the rapid Ether price decline, the network was already under stress
- On top of that, attackers additionally flooded mempools with useless transactions, which were below current gas prices of recently mined transactions, but not low enough to get dropped by miners immediately
- By replacing[5] those transactions several times per minute (some addresses sent more than 20 replacement transactions per minute), some nodes became at least partly overloaded and had therefore difficulties to follow other transactions sent by honest users/bots
- Furthermore, attackers slightly mutated each of the replacement transactions and could thus bypass mempool anti-spam protections, which can only detect transactions with same hash values
- Some gas price oracles used by auction Keeper bots got blinded by the large amount of low-priced transactions (of attackers), waiting to get mined in mempools and predicted again gas prices that were too low
- This led to even more transactions with low gas prices and hence reinforced the mempool congestion
- Many auction Keeper bots did not have a feedback mechanism in place that checks, if a sent transaction was mined or not. That means that even if the same bot sent another transaction later with a gas price higher than gas prices of transactions that get included in blocks, some miners cannot add the new transaction to a new block due to a *nonce gap*[6] between the new transaction and the last transaction

[5]Replacement of unmined transactions can be done by sending a new transaction from the same address and the same nonce

[6]In Ethereum each transaction has to have a unique nonce that is exactly greater by one than the last mined transaction sent by the same address.

with a too low gas price. This is due to the fact that the last transaction has already been dropped by the mempool due to its low gas price.

While it would be interesting and important to further analyze the mempool data used by the authors of [8], it is not within the scope of this thesis, as MakerDAO is only indirectly connected to the factors mentioned above. However, the short `ttl` of just 10 minutes made it much easier for the attackers to be successful, and the auction Keeper bot of Maker had also no proper gas price strategy in place[7].

In this section we give a brief overview how the Black Thursday incident impacted the Maker protocol.

4.4.1 Bite Events per Day

To learn more about what happened on March 12th and 13th, we take a closer look on those days.

We again make use of query A.1 for all `Bite` events and furthermore query A.2 to obtain the log data of the `OSM` contract which we use as source for the ETH price. We filter by ETH Vaults only and bucket all `Bite` events into one hour intervals between March 12th and 13th. Figure 4.3 presents this data together with the development of ETH price from the `OSM` contract (M2).

As we can see, most of the liquidations on the two days took place in just 18 hours between March 12th, 12:00 and March 13th, 6:00 and unsurprisingly most `bites` happened just after the largest drop of ETH. The delay of ~1 hour between the decline of ETH and the dramatic rise in the number of liquidations is explainable by the fact that Maker's price oracle `Median` is delayed by one hour by the `OSM`.

While it would be the optimum case that liquidations are never needed (when Vault owners manage their positions e.g. by using bots that react in time and always have enough liquidity at hand), this will probably never be a reality as there will always be users of the protocol that manage their positions by hand and thus will not be able to react quick enough in every situation.

Overall the figure does not show much anomalies besides the fact that liquidations were obviously much higher compared to other days before and after. Liquidations itself are also not a problem as long as the auctions function well. That's why we will dive deeper in the next section where we analyze, how well the auctions were able to cover the outstanding debt.

[7]On March 13th, 2020 a pull request that extends the gas price strategy got merged into the main branch of Maker's auction keeper framework: `https://github.com/makerdao/auction-keeper/pull/40/commits`

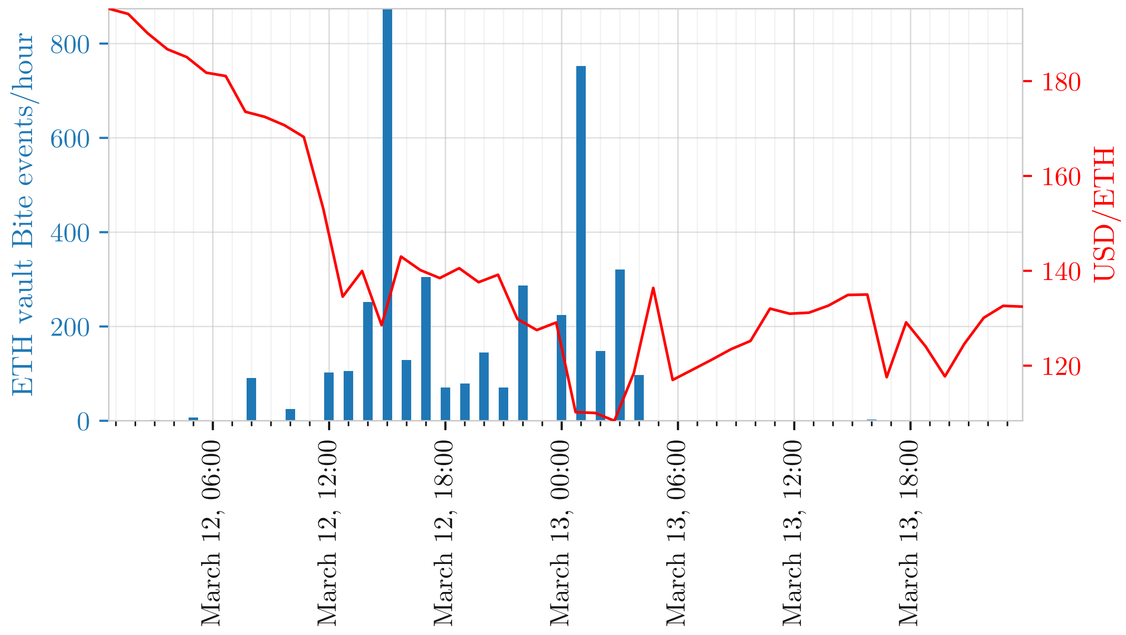

Figure 4.3: Maker Bite events per hour and ETH prices on March 12th and 13th [21]

4.5 Ability to Cover Outstanding DAI Debt in Flipper Auctions

As mentioned in our requirements in section 4.1, one of the core features for Maker to function is its ability to repurchase enough DAI in case a Vault fell below the liquidation ratio. This is done in Flipper auctions and guarantees that all minted DAI always stay overcollateralized.

As the `dent` phase of a `Flipper` auction can only be started, if the `tend` phase has ended, it can easily be obtained, which auctions have covered the outstanding debt and which have not.

We make use of queries A.4 and A.3 to extract all `tend` and `dent` calls to the `Flipper` contracts. After splitting the collected data in three periods, we identify by auction `id` and contract address (as we examine three versions of the `Flipper` auction house during the study period), which auctions where able to reach the `dent` phase and calculate then the absolute number.

It is important to note that the two improved versions of the `Flipper` contract that have been deployed in July and August, respectively, do not have a fundamentally different logic regarding the auction system itself. They rather include optimizations regarding capital efficiency and a changed interaction with the `Cat` contract.

We calculate metric M4 that expresses the fraction of auctions that were able to reach the `dent` phase. The outputs of our calculations are summarized in table 4.2

Period	All	Reached `dent` phase	FD (M4)
Before Black Thursday	626	606	96.8 %
March 12–13	4094	1210	29.6 %
After Black Thursday	458	419	91.5 %
March 14–16	102	63	61.8 %
March 17–Nov 17	356	356	100 %

Table 4.2: Auctions that reached the dent phase [21]

As the table highlights, 97 % of all auctions reached the `dent` phase before the Black Thursday incident. While an even higher number would be preferable, this level is healthy overall, as the 13 % liquidation penalty is already included in the debt, when a Vault gets liquidated. That means every auction that reaches the `dent` phase has successfully collected this fee, effectively covering debt of the few auctions that do not reach the `dent` phase.

As expected, the fraction of auctions that were able to cover the outstanding debt is much lower during the Black Thursday incident. Less than a third of all auctions were able to buy outstanding DAI back, which puts a heavy debt burden on the protocol. In section 4.5.2 we analyze further, which prices the collateral auctions could achieve on those two days.

After March 13, we find the fraction of auctions that were able to cover the debt at a lower rate than before Black Thursday, but if we look closer, we see this metric at a level of 100 % after March 16. Between March 14 and 16 however, just 61.8 % of all auctions were able to buy back the outstanding DAI. We did not investigate further why this is the case but it might be connected to the overall uncertainty after the Black Thursday incident.

It is remarkable that all auctions in the last eight month of our study period were able to buy enough DAI back from the market to cover the outstanding debt.

4.5.1 Influence of Cat Contract Redesign

Newer versions of the `Flipper` contract (version 1.0.9 and 1.1.0) do not have direct influence on the results of our analysis.

However, version 1.1.0 of Maker's contracts introduced a maximum (`box`) on the amount of DAI that can be bought back at any time. If someone wants to liquidate an unsafe Vault (of any collateral type), the `Cat` contract first checks, if the amount of debt that this new liquidation would add to the current debt under auction would exceed the `box` variable.

The reason for this additional parameter was the large amount of liquidations that took place at the same time during the Black Thursday incident. The design of the collateral auctions requires keepers to lock their DAI until someone places a higher bid (then the DAI get paid back) or until the auction ends (in this case the keeper receives the collateral and can refinance himself by selling the collateral).

If a lot of collateral is auctioned off at the same time, there is a real chance, that keepers run out of liquidity due to the fact that a lot of DAI are locked in auctions. This in turn can lead to auctions, which are not able to cover the outstanding debt.

This problem became even bigger when the auction duration was raised to 6 hours in a response to the zero bid auctions that happened during the Black Thursday incident.

The newly introduced `box` variable has an indirect influence on the outcome of our analysis. It might be the case that there were times after the activation of Makers v1.1.0 contracts, where liquidations did not take place due to the fact that the maximum amount of DAI under auction `box` was reached.

To find out if this was the case, we make use of `Bite` events (query A.1) that have been emitted by the v.1.1.0 `Cat` contract after its activation end of August, 2020. The `Bite` events provide us with a timestamp as well as other information like `tab`, which is the amount of debt that needs to be collected.

Furthermore we use information that gets logged by `claw` calls (query A.5). This function can only be called by the `Flipper` contract of each collateral type and gets called once an auction has finished. The function is used inside `Cat` for bookkeeping of the current amount of debt under auction. Specifically, an internal variable (`litter`) is used to keep track of the current amount of debt under auction. Each time, a new auction gets kicked off, `litter` gets increased and each time `claw` gets called, `litter` gets decreased.

After putting all queried data together and sorting it by block number and log index, we iterate through all entries and calculate the amount of debt that is under auction at any given time. We find the maximum amount of debt under auction to be 391,928 DAI on Septmeber, 21st.

The maximum amount of debt `box` has been set to 20 million DAI during deployment of the v1.1.0 `Cat` contract and has not been changed since, which means that this limit has not been reached by far. Therefore, we argue that our analysis above is not influenced by this mechanism that has been introduced in version 1.1.0 of Maker's contracts.

4.5.2 Auctions During Black Thursday

Due to high gas prices, mempool congestion and an unexpected high amount of collateral debt auctions that `kicked` off on Black Thursday, the `Flipper` auction contract who is designed to recapitalise Maker's DAI debt did also not work as intended.

A measure about how well this mechanism did work is the achieved price for the collateral auctions, i.e. how much DAI have been bidden for one Ether of collateral (M3). To calculate these values we use `dent` and `tend` function calls of the `Flipper` contract (queries A.4 and A.3). We only include auctions on March 12th and 13th, thus we filter the queried data accordingly.

As the `tend` and `dent` calls reflect the bidding process that takes place for each collateral auction, it is easy to calculate the DAI price that was effectively payed for each ETH: We concatenate `tend` and `dent` function calls and sort by auction `id`, `Flipper` contract address and block/log index. Then we use the latest function call that took place for each auction `id` and `Flipper` contract address. The `bid` and `lot` fields of the function calls tell us, how much DAI have been bidden (`bid`) on the amount of ETH (`lot`) and those variables are used to calculate M3.

Table 4.2 shows that more than 4000 auctions take place on the two days, therefore a meaningful representation of the collected data is needed. We decided to present this data in a boxplot diagram that buckets auctions in time intervals of one hour. The result of our analysis is shown in figure 4.4. Besides the distribution of DAI paid for one Ether, the figure also shows how many auctions ended during each period to highlight how severe the situation was.

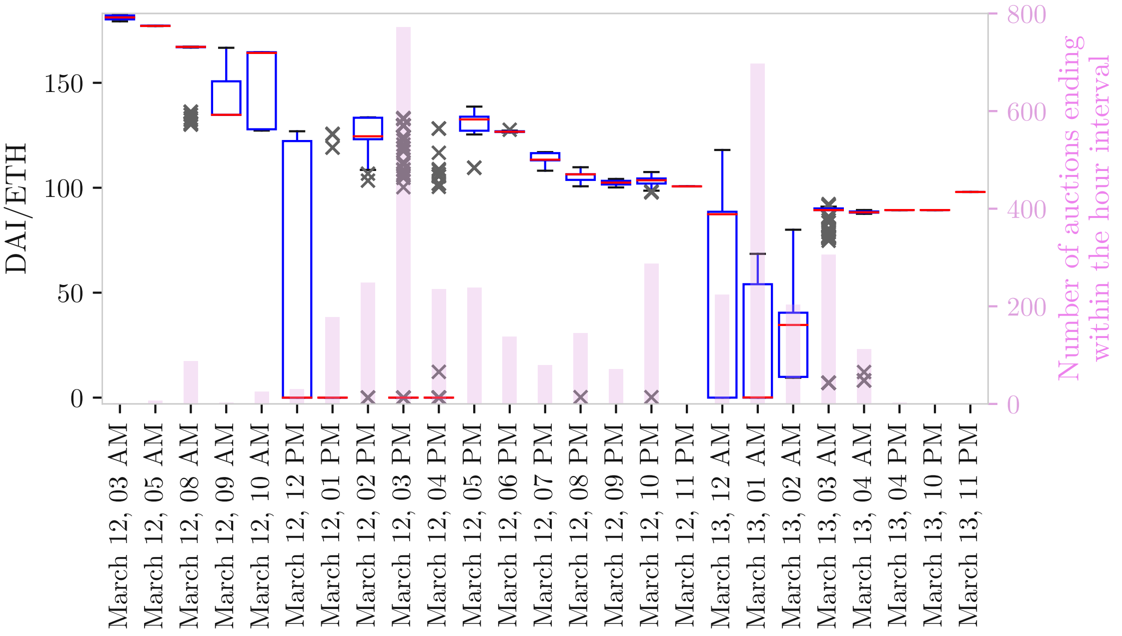

Figure 4.4: Maker DAI/ETH bids achieved on Flipper auctions on March 12th and 13th. Red lines represent the median- and blue boxes the range between upper and lower quartiles of achieved DAI prices per ETH during each hour. Black crosses mark outliers. In pale lilac the number of auctions that ended during each one hour interval are shown. [21]

The figure needs to be interpreted carefully, as there have not been bids during each hour of the day (e.g. there have not been bids between 4 AM and 5 AM or 6 AM and 8 AM on March 12th).

As we can see, the DAI prices paid for one Ether fluctuated heavily during the two days. Especially during the afternoon on March 12th, the situation was dramatic, as even median values touched almost zero values during 4 hours.

The figure does not include the development of the ETH price in USD as this would overload the graphic. However, the development of real ETH price can be seen in figure 4.3 and lies after the drop around noon on March 12th approximately between 110$ and 140$.

Instead of the ETH price development we include the number of auctions that end in each time interval (each hour). By reaching a value of almost 800, the time interval where the highest number of auctions ended is between 3 PM and 4 PM on March 12th. The large number of outliers in this time interval shows that there have still been auctions that ended with a reasonable price, but the median at a level of almost zero and not even existing quartiles tell us that at least three quarters of all auctions ended almost at a level of zero DAI/ETH.

Also in the early hours of March 13th the situation was bad. While relatively large price differences between the lower and the upper quartile indicate that there has been at least a considerable amount of auctions that has achieved more or less fair values in the `Flipper` auctions, the median value of the time interval between 1 AM and 2 AM highlights that at least half of the auctions again became victims of zero bids.

Nevertheless we also have to mention that the figure indicates that the bidding process worked quite well during some times (e.g. between 5 PM and 11 PM on March 12th). During that time, variance was much lower, compared to the hours before and after and also median values where on a much more reasonable level.

Beside the capability to cover the outstanding debt of Vaults, collateral auctions should in general achieve the best possible prices, as Vault owners get collateral that has not been used to cover the outstanding debt and liquidation penalty back. The closer the achieved auction prices are to the current market price, the higher the trust of Vault owners to the protocol in general. That's why we analyze the effectiveness of collateral auctions in the following section.

4.6 Flipper Auction Effectiveness

In section 4.1 we defined that auctions should be able to achieve adequate prices in `Flipper` auctions. We included this requirement as Vault owners should be able to trust that in the unwanted case of a liquidation at least the subsequent auction ends close to the current market price so that Vault owners get the maximum possible amount of collateral back.

To measure how close to market prices the `Flipper` auction house sells the collateral, we defined metric M6 (auction effectiveness). To calculate the auction effectiveness, we make use of `Flipper` `dent` and `tend` function calls (queries A.3 and A.4) as well as query A.2 for the ETH market prices collected from the `OSM` contract.

Like described in section 4.5.2 we first calculate the auction result for each auction `id` and `Flipper` contract by sorting all `dent` and `tend` function calls and using always the last and therefore highest bid for each auction. Then, we sort the `OSM` LogValue events by block number and log index and assign the latest ETH price that has been logged prior to each auction end. Even if the data that we can extract from the `LogValue` events is only an estimation of the real market price, the quality of the data should be good enough for a first estimation of the auction effectiveness. The fact that the values are also delayed by one hour by the `OSM` should not influence the calculation too much, as the chance that the real market price falls after a new `OSM` value gets published should over the large amount of values be approximately be equal with the chance that the real market price rises.

After calculating the auction effectiveness of each auction with the gathered data we generate a boxplot diagram to visualize the outcome of our calculation (figure 4.5)[8].

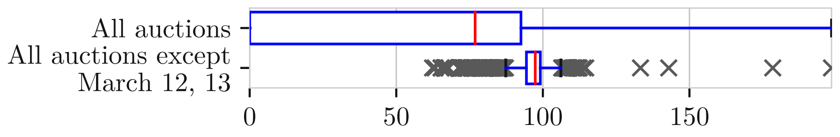

Figure 4.5: Auction effectiveness of Flipper ETH auctions in Percent. Red lines represent the median- and blue boxes the range between upper and lower quartiles of achieved DAI prices per ETH during each hour. Black crosses mark outliers [21].

As the figure reveals with the boxplot's relatively large interquartile range of more than 92 %, the effectiveness of auctions is widely distributed. A 75 % upper quartile at an auction effectiveness of 92.5 % is definitely not a satisfying value, neither the median at 76.9 %. Even worse is the lower quartile at a level of almost zero.

However, these values don't come unexpected. Table 4.2 highlights that 4094 auctions took place during the Black Thursday incident, which is almost 80 % of all auctions in our study period. This, together with the fact that only 29.5 % of the auctions on March 12th and 13th were able to cover their outstanding debt explains why there were so many auctions that have an auction effectiveness of almost zero.

Due to the heavy influence of auctions happening during the Black Thursday incident on the outcome of the auction effectiveness distribution we added a second boxplot to figure 4.5, which excludes the data from the two days to show how Maker's `Flipper` auctions performed on other days during our study period. In this dataset we find the median with 97.41 % at a much higher level and also the interquartile range of just 4.7 % is on a much healthier level.

[8]To generate a meaningful diagram we exclude two outliers that have a calculated effectiveness of more than 200 %. (These outliers can probably be traced back to wrongly modified keeper bots or bids that have been placed by hand with incorrect numbers)

The auction effectiveness on days other than March 12th and 13th seems almost even too high, if we consider that auction keepers also have costs and want to make profits. Some of the costs are:

- Slippage and costs for using an exchange: Most auction keepers will likely exchange their collateral immediately back to DAI so that they can continue to participate in new auctions. Even if the collateral is exchanged on on a DEX[9] like Uniswap, some costs arise.
- Cost of capital: To be able to participate in auctions, an auction keeper has to have capital available, preferable in DAI so that a participation in an auction is possible when an auction is kicked off
- Transaction fees: under normal conditions already the auction has three phases: `tend`, `dent` and `deal`, the latter one is used to claim the collateral after an auction has ended. If the Vault where the collateral was stored before the auction was liquidated by the same keeper that participated in the auction, the transaction fee for the `bite` call needs to be added as well. After the auction the keeper furthermore needs to pay for transaction fees to change the collateral back to DAI.
- Working hours to setup, monitor and update the keeper bot
- Hardware and power/networking costs to run the keeper bot (even if those costs should not be too high, depending how the service is hosted

Summarizing the various cost positions above we can conclude that the relatively high auction effectiveness before and after March 12th and 13 leaves the impression that there might be other external factors involved that we can't identify here. The authors of [11] did a deeper analysis specifically on this topic and come to a similar conclusion.

All in all the auction effectiveness in the full dataset is far from where it should be, which is mostly influenced by the auctions that took place between March 12th and 13th. However, the auction effectiveness is on a very high level on all other days, which in the first place is great for Vault owners and therefore the protocol itself. On the other hand this effect might also prevent interested participants to set up an auction keeper bot as returns are not very high and most auctions end up at relatively high prices. This effect in turn can lead to less capital that participates in auctions which is much-needed especially during times where mass liquidations take place like in March, 2020.

4.7 Liquidation Delay

In this section we analyze the delay that takes place between the time when a Vault can be liquidated (when it becomes undercollateralized) until the `Cat` contract's `bite` function gets successfully called. We define this metric as *liquidation delay* (M7).

[9]Decentralized Exchange

To analyze the liquidation delay we first need to define which circumstances can force a Vault to the point where it becomes undercollateralized. The collateralization ratio M5, which has to fall below 150 % before the `bite` function can be called, is influenced by two factors:

- The movement of the ETH price in the Maker ecosystem made available by the `OSM` contract
- The Vault owner, who can readjust the collateralization ratio by lending out more or less DAI or by locking more or less collateral to secure the loan

However, any manipulation of collateral- or DAI balances that is done by the Vault owner requires that the Vault is overcollateralized at the end of the operation. This means that Vaults can only fall below the collateralization ratio, if a new ETH price is made available by the Maker `OSM` contract.

To calculate the liquidation delay we have to iterate therefore through all Vault manipulations triggered by Vault owners (`Vat`'s `frob` and `fork` functions), as well as Vault liquidations triggered by auction Keepers (`Vat`'s `grab` function) to get a list of active Vault balances at any point in time during our study period. We make use of queries A.6, A.7and A.8 to collect data about the respective function calls.

We use query A.2 to get a list of all ETH price data updates from the `OSM` to be able to calculate the collateralization ratio of each Vault, based on its collateral and DAI balances. Furthermore we use query A.10 to obtain a full list of `rate` updates, so that we can calculate real DAI debts of Vaults, as Vault debt balances are stored normalized (see chapter 3.3.1).

After concatenating all data from the queries, we sort by block number and log index so that we have all data in the correct order. We then iterate through all entries in the list and depending on the type of entry, we adopt Vault balances (in case of `frob` or `fork` function calls), check if there are any undercollateralized Vaults (in case of `OSM LogValue` events, or update the `rate` variable (in case of `Vat fold` function calls)).

Each time we identify an undercollateralized Vault we look for a `grab` function call further down in the list that indicates a liquidation of this Vault and record the block timestamp difference as well as the block number difference between the time, when the `LogValue` event has been emitted and the liquidation. In the unlikely event that there has been collateral added or DAI debt paid back to the Vault before the Vault gets liquidated, we of course do not record anything.

Table 4.3 shows the aggregated data of our findings. As done before, we split our data and show liquidation delays of March 12th and 13th separately from the rest of the data as we found the largest deviations in our dataset on these days. Overall we can conclude that the liquidation delay seems not to be a big issue in the Maker collateral auction system.

Period	Median time	Median blocks	Average time	Average blocks	Maximum time	Maximum blocks
March 12–13	2.6	14	9.3	43	57.1	265
Before and after March 12–13	0.2	1	1.7	8	55.0	228

Table 4.3: Aggregated liquidation delays, time is stated in minutes [21]

The average liquidation delay of 1.7 minutes during times were the system is not under severe stress is not the best possible case, considering it should be as close to zero as possible. However, the median value is already much closer to zero and if we consider that the auction duration `ttl` was anyway extended to 6 hours after the Black Thursday incident, the liquidation delay does not add much additional time to this anyway already extremely long auctioning process. The maximum liquidation delay of 55 minutes is of course by far not optimal, nevertheless considering the much lower average value indicates that this is outlier.

As expected the liquidation delay was considerably higher during the Black Thursday period, which is most likely the result of congested mempools and high gas prices. The maximum liquidation delay is just slightly higher than on all other days during our study period but the average and median times are considerably higher.

All in all, however, even during this stressful days the liquidation delay was by far not the biggest issue of the protocol. Considering that the liquidation delay would only become a problem if it would become a considerably large fraction of the overall time between the moment when a Vault falls below the liquidation ratio until the collateral auction ends, these values are not concerning us.

4.8 Vault Management Agility

Owners of Maker Vaults have to manage their debt positions and balance their Vault collateralization ratio in such a way that they do not get liquidated and at the same time use their collateral in a cost efficient way. Active and security focused Vault management however also influences the overall stability of the protocol itself, as less Vault liquidations are necessary and DAI owners can trust that the DAI will reliably always stay overcollateralized.

We measure Vault management that is security focused and prevents Vault liquidation with metric M8. Even if the metric only includes extreme cases where Vaults already come very close to a liquidation, which are only prevented by the one hour delay of the `OSM`, this delay only exists to leave Vault owners the chance to act accordingly. We therefore believe that this metric can tell us more about how users manage their Vaults and how this evolves over time.

For our analysis we make use of queries A.6, A.7 and A.8 for data about Vault management and queries A.9 and A.2 for data about ETH price logging events emitted by the `Median` and `OSM` contracts, respectively. We also use query A.10 again to update the `rate` variable. After concatenating all collected data and sorting by block number and log index, we iterate over the whole dataset and calculate Vault balances in the same way as described in section 4.7.

Different to the calculation in section 4.7 however, we mark each Vault to be at risk already when a `Median LogMedianPrice` event gets emitted that would later allow an auction keeper to liquidate the Vault, once the `OSM` contract passes this new value. During our study period we log for each Vault when these incidents take place and also how much DAI are at risk.

For each incident that we detect we prove, if the corresponding Vault gets liquidated once the `OSM` takes over the new value or if the owner of the Vault reacts before that happens by pushing the collateralization ratio back to a healthy level. There is also a third possibility, namely, if another (higher) ETH price gets logged at the `Median` contract. This happens, if ETH prices are fluctuating heavily and the need to update the price more often is given. In this case, a liquidation can anyway not take place, as the `OSM` always takes the latest published value while older values are dropped.

We summarize the results of our analysis in table 4.4. As expected we find a concentration of Vaults at risk around Black Thursday, therefore we split our findings in three periods: *before Black Thursday* (November 18th, 2019 – March 11th, 2020), Black Thursday and *after Black Thursday* (March 14th – November 17th, 2020). For comparison we also added a forth period to our table: September 2–6. In these days the 2nd largest drop within our study period took place, the price of ETH dropped by more than 30 %.

Period	Saved	Liquidated	Vault management agility (M8)
Before Black Thursday	1.9	1.33	58.8 %
March 12–13	21.53	18.14	54.3 %
After Black Thursday	37.35	4.61	89 %
September 2–6	3.7	0.76	82.3 %

Table 4.4: Cumulative DAI debts saved vs. liquidated [million DAI] [21]

The table shows cumulative DAI balances that were at risk to get liquidated but Vault owners reacted in time (column *Saved*) as well as cumulative DAI balances that became at risk but got liquidated (column *Liquidqted*). Furthermore we calculate the fraction of DAI that have been saved in each period (column *Vault management agility*).

Our analysis reveals that the overall Vault management seems to have improved a lot during the study period. While from the start of the MCD in November 2019 till March 12th, 2020 on average approximately 6 out of 10 DAI that became at risk to get liquidated could be saved, this number increased to almost 9 out of 10 DAI in the time after Black Thursday. On March 12th and 13th, 2020 the fraction of DAI that could be saved was (as expected) lower, however not much lower than in the time before Black Thursday.

There could be several reasons why this is the case. One is that in the beginning of Maker's MCD system much smaller debt positions have been created and also the sum of all debt that has been borrowed from Maker was still comparatively low: In the days before the Black Thursday incident, the DAI supply just crossed a circulating supply of 120 million DAI while it was more than a billion DAI at the end of the study period [53].

Probably Vault management has therefore increased also due to much larger amounts of debt that is now at risk. One example for this is the yearn.finance [62] yETH Vault that has been introduced in September, 2020 and quickly became one of Maker's biggest Vaults. Yearn's yETH Vault makes use of a Maker ETH Vault and manages its position in an active way. Therefore, a rather low collateralization ratio can be chosen and the real collateralization ratio of the Vault quickly adopted by reading event data from the `Median` contract [1].

Our analysis also shows that the 2nd biggest ETH price drop within our study period that happened beginning of September led also to much better results in terms of the fraction of DAI that could be saved from liquidation. Even if the price decline of approximately 30 % in 4 days is not comparable to the dramatic decline that took place during the Black Thursday incident, it is still interesting that it had so little effect on the protocol.

4.9 Collateralization

A healthy collateralization ratio of Makers Vaults can act as a safety buffer and prevent mass liquidations in times of falling collateral prices and thereby also forestall resulting negative consequences that are triggered or enforced through liquidations (deleveraging spirals [23], locked capital during auctions and thus limited capital that is available for bidding at other auctions, etc.).

This is especially the case for Vaults that are not managed in an automated manner. To measure how well Maker's Vaults are protected against this risk we make use of metric M9 (overall collateralization ratio). Even if this metric represents just an artificial method to measure how well Maker's Vaults are overcollateralized (as collateralization ratios are individual for each Vault and therefore not comparable), it still gives us an idea how the system wide collateralization developed over time. This metric furthermore reflects tendencies of Vault owners towards more or less risk willingness and gives us also an idea, how well the DAI is really secured by collateral assets.

For the data we use queries A.6, A.7, A.8, A.9 and A.10. We calculate the development of Vault balances as described in section 4.7 and 4.8 and calculate M9 at each ETH price update logged by the `Median`.

The results of our analysis are presented in figure 4.6. Besides the development of the overall collateralization ratio we added also added the development of the ETH price for reference. The ETH price development is especially interesting when compared to the overall collateralization ratio as the two lines move in parallel as long as nothing gets changed in the Vaults (either by Vault owners or through liquidations).

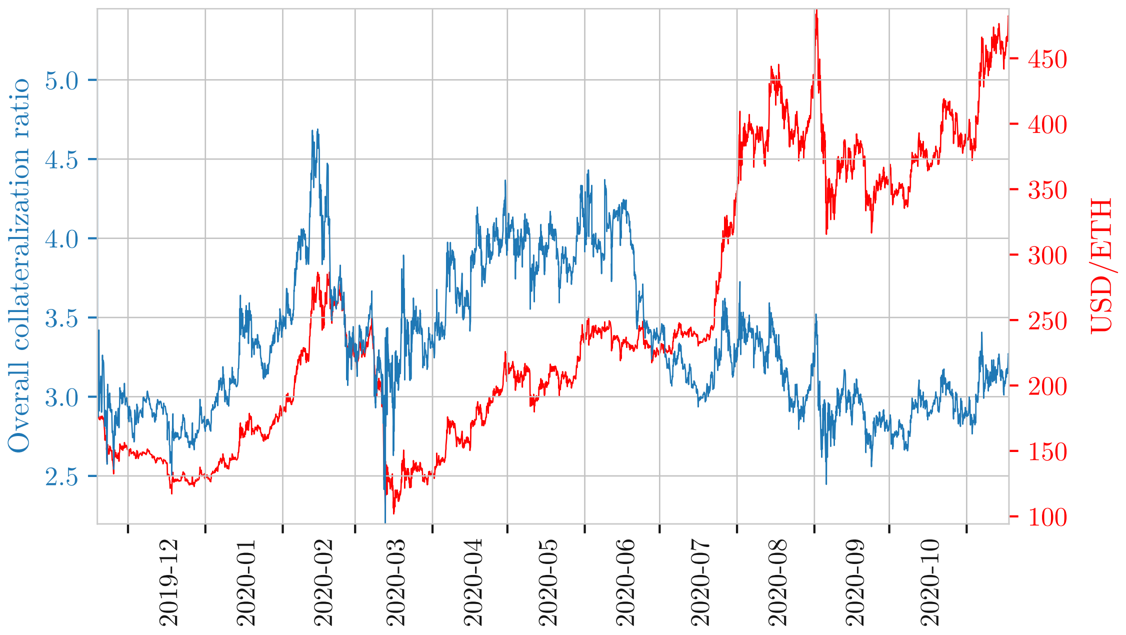

Figure 4.6: Development of the overall collateralization ratio. The development of the ETH price is shown in red [21].

During the first months of the study period this behavior is clearly apparent, when the overall collateralization ratio climbs from a relatively low level of approximately 3 to more than 4.5 in almost less than a month while the price also rises heavily.

The Black Thursday incident is clearly identifiable in the middle of March, when the overall collateralization ratio drops within hours from a level of ~3 to the lowest level within the whole study period of 2.20 in the morning hours of March 13th. The ratio however recovered quickly afterwards and reached safe levels of 3 and more in the days after. The relatively low (compared to the months before) level of 3 just before the Black Thursday incident once more highlights, how unexpected the sudden price drop came for most Vault owners.

Thereafter the ratio climbs again to a healthy level of more than 4 before it declines sharply in the middle of June, 2020. Within a month the ratio even falls back to a level of 3 where it stays until the end of the study period (not without oscillating +/− ~0.25 around the level of 3). This development is even more surprising as the ETH price almost doubles in this time from levels of less than 250 USD to almost 500 USD end of August.

There might be reasons that can explain this development that are connected to the DeFi summer[10], where many protocols experienced high growth and an increase professionalization of usage due to the development of yield farming programs like Yearn's [62]

[10]https://www.coindesk.com/comp-below-100-defi-summer-over

and others. However, exploring these reasons are a topic outside of the scope of this thesis and therefore won't be further investigated by us.

All in all it seems that the overall collateralization ratio of ETH Vaults seems to stabilize at a level of 3 on the long run. While an even higher level would be always better for the protocol itself, having one DAI overcollateralized by a factor of ~3 is at least a very good security for the value and stability of the DAI.

For the stability of the liquidation system itself, a minimum required level of the overall collateralization ratio to guarantee a safe operation depends heavily on how Vault management is carried out by the owners of the Vaults. The higher the amount of debt that is not managed in an automated way, the higher should the overall collateralization ratio be. Looking at the outcomes of our analysis in section 4.8 we can conclude that a overall collateralization ratio of 3 is much less of a problem at the end of the study period than it was in spring of 2020, as Vault owners are managing their debt positions more actively after the Black Thursday incident.

4.9.1 Development of Collateralization Ratios

As the overall collateralization ratio fluctuates a lot over the study period we aim to get a better understanding of how the different Vaults are managed. Hence we repeat our calculations once more but instead of calculating the overall collateralization ratio of all Vaults over all debt and collateral values, we bucket Vaults in 11 collateral ratio ranges from less than 150 % (which should be always almost zero, as Vaults should be liquidated at this point) to greater or equal 600 %.

For each update of the ETH price from the `Median` contract we first calculate the collateralization ratio of each Vault, based on the currently lent out amount of DAI and ETH. After that we bucket every Vault in the 11 predefined ranges and finally log the total amount of DAI debt that is stored in all Vaults of one bucket.

After iterating over the whole study period we create a stacked area chart, which is presented in figure 4.7. The figure highlights how much risk Vault owners were willing to take relative to others and how that developed over time.

Spikes that are scattered throughout the figure represent sudden changes of the collateralization ratio of some amount of DAI. These changes most commonly happen due to collateral price changes, but can of course also come from Vault rebalancings, either initiated by users or keepers.

Also in this figure we can identify a clear trend towards more risk friendly vault management, as e.g. the amount of DAI that was managed between a collateralization ratio of 150 % to 200 % reached new maximums towards the end of the study period.

Fortunately, the amount of DAI that fell below 150 % collateralization ratio can only be spotted once, during Black Thursday in March. This supports our findings documented in section 4.7, which found the liquidation delay to be low throughout the study period, with one exception:

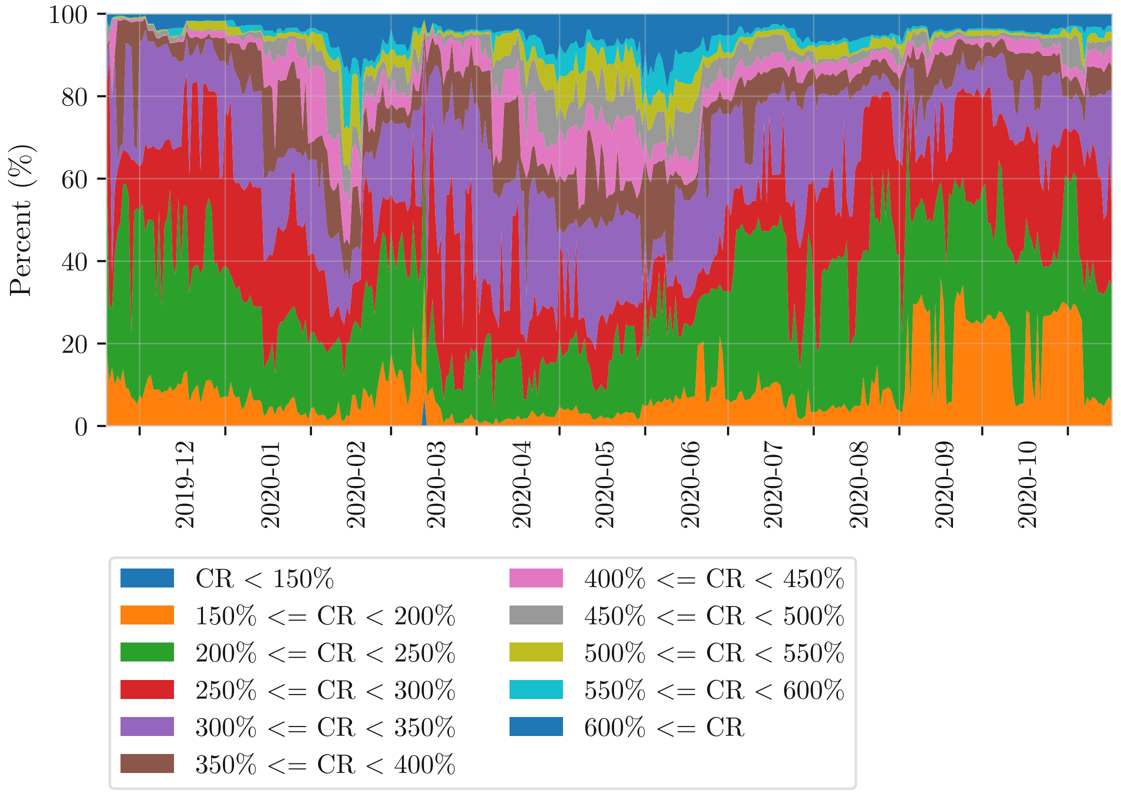

Figure 4.7: Development of collateralization ratios

CHAPTER 5

Discussion

In this chapter, we summarize our findings and evaluate Maker's liquidation system. Furthermore we briefly describe how the community reacted to the Black Thursday incident and which consequences were drawn on the protocol level.

5.1 Assessment of Maker's Liquidation System

In Chapter 4 we defined 5 requirements that need to be fulfilled in order to guarantee a safe operation of the protocol in regards of a steady overcollateralization and thus a functioning operation of Maker's own liquidation system.

As we hypothesized that the protocol fulfills these requirements we will evaluate them in this section by assessing a set of metrics that we have defined in section 4.1.1.

H1: Auctions are able to cover the outstanding DAI debt

Test: We determine the fraction of auctions that were able to cover their debt by calculating the number of auctions that reached the `dent` phase of a `Flipper` auction during a specified period, divided by the number of all auctions during this period (metric M4).

Rationale: In the first phase of the auction bidders bid increasing amounts of DAI for the collateral that gets auctioned off, the second phase of the auction starts only when the outstanding target debt is met. Hence only auctions that have logged at least one bid in the second phase of a collateral auction have been able to cover their debt.

Assessment: During most of our study period collateral auctions were able to cover their debt. Only during the Black Thursday incident this was by far not the case. Table 4.2 highlights that during this time not even 30 % of the auctions were able to buy back enough DAI to cover the debt that was lent out by the Vault owner in the first place.

However, before and after this two day period with 97 % (before B.T.) and 91.5 % (after B.T.) this metric could be found at a much healthier level. In the time after March 16th until the end of the study period the fraction of auctions that were able to cover their debt even reaches 100 %. Considering that the liquidation penalty of 13 % also collects some fees, the auctions that collected enough DAI should cover the costs that arise from auctions that did not reach the target.

Summarizing the above, the hypothesis has to be rejected for Black Thursday, while it can be retained for the remainder of the study period.

H2: Auctions achieve adequate prices for the collateral

Test: We check whether the auction effectiveness (metric M6) is close to one.

Rationale: The optimal effectiveness (when collateral is sold at market price) would be one, but this does not include the costs that auction keepers have. However, as Vault owners that got liquidated get back collateral that was not needed to cover the outstanding debt, an average effectiveness of close to one increases the trust in the overall system.

Assessment: The results from our analysis reveal that the auction effectiveness fluctuates heavily over all collateral auctions within the study period. For better interpretability of our findings we generated a boxplot diagram that highlights, how widely the data is distributed (figure 4.5).

With a median auction efficiency of 76.9 % over all auctions we have to reject the hypothesis. Even when we consider all costs (slippage, cost of capital, transaction fees, server costs, etc.) that auction keepers need to cover, this value is too low. The large interquartile range with the 25 % quartile that is at a level of almost zero makes this even worse.

However, if we just analyze the results from all days without March 12th and 13th, the hypothesis can be retained. When excluding these two days the median auction effectiveness even reaches a level of more than 97 % and a low interquartile range of just 4.7 %.

H3: Keepers liquidate Vaults quickly after they have fallen below the liquidation ratio

Test: We determine the liquidation delay (metric M7) and examine if it stays at a reasonable low level.

Rationale: The liquidation delay is one of the factors that influences the total time it takes from the moment, collateral prices start to decline to the point when the collateral auction ends. As steeply falling collateral prices can lead to the fact that collateral auctions are not able to cover outstanding debt, the total time should be as low as possible and thus also the liquidation delay should be very low.

Assessment: The results of our analysis that are summarized in table 4.3 are promising during almost all days of the study period, except Black Thursday. On March 12th and 13th the liquidation delay was on average 9.3 minutes or 43 blocks and still at a median level of 2.6 minutes or 14 blocks. Even if these numbers alone do not explain the catastrophic auction results during these two days, they are at least not optimal. This fact is even worse if we consider that the liquidation delay should especially during times of drastically declining collateral prices be as low as possible to allow for a quick recapitalization of the system. We therefore reject the hypothesis for March 12th and 13th.

During all other days the liquidation delay was at a satisfying low average level of 1.7 minutes or 8 blocks and at a median of even 0.2 minutes or 1 block. The hypothesis can thus be retained for all other days during the study period.

H4: Vaults threatened with liquidation adjust their collateralization in time

Test: We determine the Vault management agility (metric M8).

Rationale: The Vault management agility tells us more about how Vault owners manage their debt positions. The higher the ratio, the more collateral has been saved in time before getting auctioned off. A ratio of 100 % would mean that every Vault that gets close to liquidation gets saved in time, which would mean that all Vaults are managed in an automated way.

Assessment: When compared to the metrics and their outcomes of the previous hypothesis, the Vault management agility is not much worse during Black Thursday, when compared to the days before. Table 4.4 demonstrates that the Vault management agility was at a level of 58.8 % in the time before Black Thursday, whereas it was with 54.3 % not even 5 % lower during Black Thursday.

In the rest of the study period, however, the Vault management agility rose to a much healthier level of 89 % and even in the beginning of September, when the 2nd steepest decline of the ETH price in the study period took place the metric was with 82.3 % just slightly lower.

Overall, these numbers leave the impression that larger Vaults and the growth of the protocol itself (in terms of total issued DAI) leads to a professionalization of Vault management. This is good for the protocol as automatic Vault management is a mechanism that prevents stress during times of falling collateral prices already in the first place. We therefore reject the hypothesis before and during the Black Thursday incident but it holds in the rest of the study period.

H5: Vault owners maintain a collateralization ratio at double the liquidation ratio on average

Test: We analyze how the overall collateralization ratio (metric M9) develops over the course of the study period.

Rationale: The higher the overall collateralization ratio is, the more the whole protocol is resilient to stress situations. In times of falling collateral prices this is a further safety measure, which can prevent mass liquidations already in the first place. While a ratio of 150 % would mean that all Vaults are exactly positioned at the liquidation ratio, this would also mean that the maximum possible risk is put on the protocol as every Vault owner that does not use an automated Vault management mechanism is forced into liquidation at some point.

Assessment: Figure 4.6 reveals that the overall collateralization ratio fluctuates from a minimum of 2.2 on Black Thursday and a maximum of more than double that value at 4.69 on February, 14th. Especially in the first months after the launch of Maker's MCD It seems that Vault management is done primarily passively, as the overall collateralization ratio moves very much in parallel with the ETH price.

Even after the dramatic drop during Black Thursday in March the overall collateralization ratio moves in parallel to the ETH price again before the metric starts to decouple from the ETH price in June. Several possible explanations for this behavior exist, but the general trend towards more professionalized Vault management and the rise of DeFi and yield farming during the summer months of 2020 might be the reasons that explain this best.

In the last three months of the study period the overall collateralization ratio stabilizes at a level of approximately 3 and on average, we found the collateralization ratio to be at 3.32, thus the hypothesis holds during the whole study period.

5.2 Maker's Response to the Black Thursday Incident

Due to the zero bid `Flipper` auctions, the protocol accrued several millions in bad debt (`sin`), which had to be covered by selling MKR tokens in `Flopper` auctions. Also Vault owners whose collateral ended up a zero bid auction or whose auction did not reach an adequate level did not get back any collateral tokens, as the `dent` phase was never reached, effectively leaving them with an even bigger loss than expected.

As both Vault owners and keepers were buying large amounts of DAI to pay back Vault debt and to place bids for `Flipper` auctions respectively, the demand for DAI increased during this phase. Due to this fact it was not possible for DAI to keep the price peg to the USD during this time, which led to DAI prices as high as 1.11 USD for each DAI on March 14th, 2020.[1]

The Maker community and governance facilitators were forced to act quickly and a set of extraordinary governance and risk calls as well as executive votes took place. Already on March 13th an executive vote [41] to adjust several of Makers risk parameters like the extension of `ttl` from the `Flipper` contract from 10 minutes to 6 hours took place.

[1]Data from Coinmarketcap: https://coinmarketcap.com/currencies/multi-collateral-dai/

On March 15th another executive vote [42] was placed to lower the stability fee from 4 % to 0.5 % in hope to incentivize Vault owners to generate more DAI so that the DAI price could be stabilized again. Beside other changes, also the governance security module delay was set down from 24 hours to 4 hours, so that MKR owners have the possibility to react quicker to unforeseeable circumstances and to push necessary executive votes fast to the protocol.

On March 17th the next executive vote took place [43]. This vote marked the introduction of USDC [7] as a new collateral type. Even if the addition of a centralized asset to Maker's set of collateral types was heavily discussed in the Maker community, it was the quickest possibility to bring DAI back to its peg, as large amounts of DAI could be generated by basically minting them with USDC. This new collateral provided keepers, Vault owners and arbitrageurs an easy possibility to generate enough DAI liquidity and thus to stabilize the system, including the peg of the DAI to the USD.

In the months after the community started discussing a complete redesign of Maker's liquidation system by changing the auction mechanism from an English auction model to a Dutch auction model [6]. Different to the English auction, the Dutch auction starts with a high price, which decreases over time until a bidder accepts the price and buys some or all collateral under auction.

The advantage for Maker to make use of such a system are that keepers do not need to keep capital at the sidelines, which is only used, once a Maker auction kicks off. Furthermore, there are also no locking periods during which a keeper has to wait if higher bids are placed (which is the case in the English auction system).

This feature allows the use of flash loans, which are loans that can be borrowed without the need of a collateral, but have to be paid back in the same transaction. At the end of the transaction the flash loan provider proves, if all borrowed capital was paid back. If this condition can not be fulfilled, the whole transaction gets reverted, which thus does not represent a risk for the lenders of the loan.

In 2021 Maker finished the process of designing and testing the new auction contracts and started to replace step by step old auction contracts with new ones [51]. Since then almost all auctions were able to cover their outstanding debt [4], which gives hope that the new collateral auction house also performs well on the long run. However, a deep evaluation of the mechanics of Makers liquidation system 2.0 is necessary, especially under stress conditions.

Overall, the Black Thursday price crash and the discussions and executive votes that took place afterwards have proven that Maker's community can act fast and reach consensus in extreme events. Nevertheless, it also showed Maker's vulnerability regarding network congestion and how much impact already one bad system design choice can have on the whole protocol when put under stress.

CHAPTER 6

Conclusion

In this work we analyzed the liquidation system of MakerDAO in the first year of the full protocol implementation, the multi collateral DAI (MCD). Ensuing from Maker's business processes, which are organized in a set of smart contracts hosted on the Ethereum blockchain we defined several requirements that need to be fulfilled to guarantee a smooth operation of Maker's liquidation system. After deriving metrics that aim to measure if our requirements are fulfilled we made use of Ethereum transactional data to analyze if the protocol meets our expectations.

Our results show that overall Maker's liquidation mechanics work well during most times. However, the protocol also faced a difficult time within the study period: on March 12th, 2020, the DAO plummeted into its biggest crisis so far. In the course of this day, which is commonly also known as Black Thursday, stock markets as well as crypto markets experienced a dramatic price decline amidst the fear around the fast spreading Covid-19 pandemic. Falling crypto asset prices have a direct impact on Maker's collateralization mechanism and led on that day to an unprecedented wave of Vault liquidations that did not meet expectations.

During Black Thursday, none of our requirements could be fulfilled by the protocol. Nevertheless, in the months before, Maker was able to meet four of our five stipulations and after the incident even all of our requirements could be fulfilled. Summarizing our findings, we can conclude that during times of low price volatility of collateral assets, the protocol's liquidation system was able to fulfill its purpose. However, when collateral prices, especially of large capital assets like Ethereum's own token Ether, were falling steeply, the pressure rises drastically. In the time after the incident, the community developed a set of improvements, the largest of which comprises even a complete redesign of the auction mechanism from an English auction system to a Dutch auction system. While the just rolled out new auction system seems to work well so far, it remains to be seen if it can prove to be also more resilient to stress situations like the one that took place on Black Thursday.

6.1 Future Work

It would be worth to carry out a similar analysis on Maker's new liquidation 2.0 system. While this system was just rolled out in Spring 2021, it would be interesting to reuse some of our metrics, adopt them to the new reverse auction system and carry out an analysis of the first year of the liquidations 2.0 system in Spring 2022. While we believe that the new auction system solves many of the problems that the old system had (e.g. long locking periods for bids), every auction system has its own drawdowns and it would be worth to compare the two systems to each other.

It has to be mentioned though that the level of stress on a time critical element like a collateral auction mechanism is dependent on the intensity of the price decline of the used collateral. Thus, it might be that the performance of the new auction system will only be proven, once a comparable price shock occurs.

APPENDIX A

SQL Queries

```
/* all Bite events */
    WITH e AS (
          SELECT log_index, transaction_hash,
          CONVERT_FROM(
             DECODE(
                SPLIT_PART(
                   SUBSTRING(
                      CAST(args->0->>'hex' AS TEXT), 3
                   ), '00', 1
                ), 'hex'
             ), 'UTF8'
          ) AS "ilk",
          args->1->>'hex' as "urn",
          CAST(args->2->'num' AS NUMERIC) AS "ink",
          CAST(args->4->'num' AS NUMERIC) / 10^45 AS "tab", --
             denominated in "rad"
          CAST(args->6->'num' AS NUMERIC) AS "auction_id"
          FROM event
          WHERE event.address = '0
             x78F2c2AF65126834c51822F56Be0d7469D7A523E'
          /* MCD_CAT v1.0.0 */
          OR event.address = '0
             xa5679C04fc3d9d8b0AaB1F0ab83555b301cA70Ea'
          /* MCD_CAT v1.1.0 */
          AND event = 'Bite'
          AND event.timestamp < '2020-11-18')
  SELECT block_number, log_index, timestamp,
          ilk, urn, ink, tab, auction_id,
            tx.from, tx.to, value, gas, gas_price, gas_used
    FROM e
            JOIN tx
            ON (tx.hash = e.transaction_hash)
ORDER BY tx.block_number ASC, e.log_index ASC;
```

Query A.1: Query Bite Events from Cat contract

```
/* all LogValue events */
  SELECT block_number, log_index, timestamp, args->0->>'hex' AS
        val
    FROM event
   WHERE address = '0x81FE72B5A8d1A857d176C3E7d5Bd2679A9B85763'
            /* ETHUSD OSM */
            AND event = 'LogValue'
            AND timestamp < '2020-11-18'
ORDER BY block_number ASC, log_index ASC
```

Query A.2: Query LogValue Events from OSM contract

```
/* all dent logs */
    WITH l AS (
            SELECT log.block_number, log_index, transaction_hash,
                address,
            CAST(topics[3] AS TEXT) AS id,
            CAST(topics[4] AS TEXT) AS lot,
            SUBSTRING(data,267,64) AS bid
            FROM log
            WHERE (address = '0
                xd8a04F5412223F513DC55F839574430f5EC15531'
            /* MCD_FLIP_ETH_A v1.0.0 */
            OR address = '0
                x0F398a2DaAa134621e4b687FCcfeE4CE47599Cc1'
            /* MCD_FLIP_ETH_A v1.0.9 */
            OR address = '0
                xF32836B9E1f47a0515c6Ec431592D5EbC276407f')
            /* MCD_FLIP_ETH_A v1.1.0 */
            AND SUBSTRING(topics[1] from 3 for 8) = '5ff3a382'
            /* dent(uint256,uint256,uint256) */
            AND log.timestamp < '2020-11-18')
  SELECT l.block_number, log_index, transaction_hash, l.address
       ,
            tx.from, tx.to, gas_price, tx.timestamp,
            l.id, lot, bid
```

```
     FROM l
     JOIN tx ON (tx.hash = l.transaction_hash)
      AND tx.status = 'true'
ORDER BY l.block_number ASC, l.log_index ASC;
```

Query A.3: Query dent function calls from Flipper contract

```
/* all tend logs */
     WITH l AS (
          SELECT log.block_number, log_index, transaction_hash,
              address,
          CAST(topics[3] AS TEXT) AS id,
          CAST(topics[4] AS TEXT) AS lot,
          SUBSTRING(data,267,64) AS bid
          FROM log
          WHERE (address = '0
              xd8a04F5412223F513DC55F839574430f5EC15531'
          /* MCD_FLIP_ETH_A v1.0.0 */
          OR address = '0
              x0F398a2DaAa134621e4b687FCcfeE4CE47599Cc1'
          /* MCD_FLIP_ETH_A v1.0.9 */
          OR address = '0
              xF32836B9E1f47a0515c6Ec431592D5EbC276407f')
          /* MCD_FLIP_ETH_A v1.1.0 */
          AND SUBSTRING(topics[1] from 3 for 8) = '4b43ed12'
          /* tend(uint256,uint256,uint256) */
          AND log.timestamp < '2020-11-18')
   SELECT l.block_number, log_index, transaction_hash, l.address
       ,
          tx.from, tx.to, gas_price, tx.timestamp,
          l.id, lot, bid
     FROM l
     JOIN tx ON (tx.hash = l.transaction_hash)
      AND tx.status = 'true'
ORDER BY l.block_number ASC, l.log_index ASC;
```

Query A.4: Query tend function calls from Flipper contract

```
/* all claw logs */
   SELECT block_number, log_index, timestamp,
          CAST(topics[3] AS TEXT) AS rad
     FROM log
    WHERE address = '0xa5679C04fc3d9d8b0AaB1F0ab83555b301cA70Ea'
          /* MCD_CAT v1.1.0 */
```

```
        AND SUBSTRING(topics[1] from 3 for 8) = 'e66d279b'
            /* claw(uint256) */
        AND timestamp < '2020-11-18'
ORDER BY block_number ASC, log_index ASC;
```

Query A.5: Query claw function calls from Cat contract

```
/* all frob logs */
SELECT block_number, log_index, timestamp,
       ('0x' || SUBSTRING(topics[3], 27)) AS u,
          SUBSTRING(data, 395, 64) AS "dink",
          SUBSTRING(data, 459, 64) AS "dart"
  FROM log
 WHERE address = '0x35D1b3F3D7966A1DFe207aa4514C12a259A0492B'
       /* MCD_VAT v1.0.0 */
    AND SUBSTRING(topics[1] FROM 3 for 8) = '76088703'
       /* frob(bytes32,address,address,address,int256,int256)
          */
    AND SUBSTRING(topics[2], 3, 10) = '4554482d41' -- ETH-A
    AND timestamp < '2020-11-18'
 ORDER BY log.block_number ASC, log.log_index ASC
```

Query A.6: Query frob function calls from Vat contract

```
/* all fork logs */
   SELECT block_number, log_index, timestamp,
          ('0x' || SUBSTRING(topics[3], 27)) AS u,
                ('0x' || SUBSTRING(topics[4], 27)) AS v,
                SUBSTRING(data, 331, 64) AS "dink",
                SUBSTRING(data, 395, 64) AS "dart"
     FROM log
    WHERE address = '0x35D1b3F3D7966A1DFe207aa4514C12a259A0492B
        '

          /* MCD_VAT v1.0.0 */
       AND SUBSTRING(topics[1] FROM 3 for 8) = '870c616d'
          /* fork(bytes32,address,address,int256,int256) */
       AND SUBSTRING(topics[2], 3, 10) = '4554482d41' -- ETH-A
       AND timestamp < '2020-11-18'
 ORDER BY log.block_number ASC, log.log_index ASC
```

Query A.7: Query fork function calls from Vat contract

```
/* all grab logs */
SELECT block_number, log_index, timestamp,
       ('0x' || SUBSTRING(topics[3], 27)) AS u,
             SUBSTRING(data, 395, 64) AS "dink",
             SUBSTRING(data, 459, 64) AS "dart"
  FROM log
 WHERE address = '0x35D1b3F3D7966A1DFe207aa4514C12a259A0492B'
        /* MCD_VAT v1.0.0 */
   AND SUBSTRING(topics[1] FROM 3 for 8) = '7bab3f40'
        /* grab(bytes32,address,address,address,int256,int256)
          */
   AND SUBSTRING(topics[2], 3, 10) = '4554482d41' -- ETH-A
   AND timestamp < '2020-11-18'
 ORDER BY log.block_number ASC, log.log_index ASC
```

Query A.8: Query grab function calls from Vat contract

```
/* all LogMedianPrice events */
SELECT block_number, log_index, timestamp,
       CAST(args->0->'num' AS NUMERIC) AS "val"
  FROM event
 WHERE address = '0x64DE91F5A373Cd4c28de3600cB34C7C6cE410C85'
        /* ETHUSD Medianizer */
       AND event = 'LogMedianPrice'
       AND timestamp < '2020-11-18'
 ORDER BY block_number ASC, log_index ASC
```

Query A.9: Query LogMedianPrice Events from Median contract

```
/* all fold logs */
SELECT block_number, log_index, timestamp,
        topics[4] AS rate
  FROM log
 WHERE address = '0x35D1b3F3D7966A1DFe207aa4514C12a259A0492B'
        /* MCD_VAT v1.0.0 */
   AND SUBSTRING(topics[1] FROM 3 for 8) = 'b65337df'
        /* fold(bytes32,address,int256) */
   AND SUBSTRING(topics[2], 3, 10) = '4554482d41' -- ETH-A
   AND timestamp < '2020-11-18'
 ORDER BY log.block_number ASC, log.log_index ASC
```

Query A.10: Query fold function calls from Vat contract

List of Figures

List of Tables

Glossary

collateralization ratio U.S. dollar value of the locked collateral of a Vault divided by the outstanding DAI debt of this Vault. 18, 19, 38, 53–56, 58, 63

dapp Decentralized applications are applications that run on a smart contract-based blockchain and offer their services in a decentralized and trustable manner. 1, 2

ERC-20 A token standard with six formally specified methods based on Ethereum request for comment #20. 18, 19, 26, 28, 29, 31

liquidation ratio The ratio at which a Vault becomes undercollateralized. If a Vault's collateralization ration falls below this ratio, a liquidation can be triggered by keepers. 18, 19, 21, 31, 33, 38, 39, 43, 46, 54, 62–64

mempool Short for Memory Pool. A Mempool stores valid transactions that have not been included in a block yet. In general full nodes of a blockchain maintain Mempools and propagate newly added valid transactions also between each other to guarantee that these transactions will be included into a future block. 8, 9, 44, 45, 48

proof-of-work A proof-of-work algorithm is a mathematical puzzle that requires some cost and time (work) by a computer to be solved. However, the solution to the puzzle is verifiable in a very short amount of time by others. 7

yield farming Yield farming is the practice of leaving crypto tokens to a protocol to earn interest from it. The interest can come from different types of yield farming like lending or liquidity mining (e.g. from a Uniswap pool).. 57, 64

Acronyms

BTC	Ticker Symbol of Bitcoin. 11, 13, 14
DAO	Decentralized Autonomous Organization. 3, 17–22, 24–26, 35, 41
DeFi	Decentralized Finance. 44, 57, 64
DSR	DAI Savings Rate. 20, 26, 29, 33
ETH	Ticker Symbol of Ether. 19, 39–42, 45, 46, 49–51, 53–58, 63, 64, 75
ICO	Initial Coin Offering. 1, 14
MCD	Multi Collateral DAI. 2, 4, 18, 37, 40, 41, 55, 56, 64
MKR	Ticker Symbol of the Maker Token. 18–23, 25–27, 33–36, 64, 65
P2P	Peer-to-Peer. 7, 8
PoW	proof-of-work. 7–12, 14
SCD	Single-Collateral DAI. 4, 18
USD	United States Dollar. 1, 2, 5, 17, 19, 40, 50, 57, 64, 65

Bibliography

[1] Anthony Sassano: How the yETH Vault Works - The Daily Gwei #64 (2021), https://thedailygwei.substack.com/p/how-the-yeth-vault-works-the-daily, accessed 2020-06-22

[2] go-ethereum Authors, T.: Go ethereum (2020), https://geth.ethereum.org/, accessed 2020-10-13

[3] Back, A.: Hashcash - a denial of service counter-measure (1997), http://www.hashcash.org/papers/hashcash.pdf, accessed 2020-04-01

[4] Block Analitica: Block analitica - auctions (2021), https://maker.blockanalitica.com/auctions/, accessed 2021-08-09

[5] Brave Software, I.: Brave rewards (2021), https://brave.com/brave-rewards/, accessed 2020-02-28

[6] charlesstlouis: A liquidation system redesign: A Pre-MIP discussion (2020), https://forum.makerdao.com/t/a-liquidation-system-redesign-a-pre-mip-discussion/2790, accessed 2021-03-10

[7] Coinbase: USD Coin (USDC) - Stablecoin by Coinbase (2021), https://www.coinbase.com/usdc, accessed 2020-01-20

[8] Corporation, B.: Evidence of mempool manipulation on black thursday: Hammerbots, mempool compression, and spontaneous stuck transactions (2021), https://www.blocknative.com/blog/mempool-forensics, accessed 2020-11-23

[9] cyrus: Black Thursday Response Thread - Governance Plan and Forum Response (2020), https://forum.makerdao.com/t/black-thursday-response-thread/1433/82, accessed 2021-03-08

[10] Dai, W.: bmoney (1998), http://www.weidai.com/bmoney.txt, accessed 2020-04-02

[11] Darlin, M., Papadis, N., Tassiulas, L.: Optimal bidding strategy for maker auctions (2021)

[12] Eisenhardt, K.M.: Building theories from case study research. The Academy of Management Review **14**(4), 532–550 (Oct 1989). https://doi.org/10.2307/258557

[13] Etherscan: Ethereum (ETH) Blockchain Explorer (2021), `https://etherscan.io/`, accessed 2021-06-30

[14] Etherscan.io: Ethereum full node sync (default) chart | etherscan (2021), `https://etherscan.io/chartsync/chaindefault`, accessed 2020-10-14

[15] Foundation, T.L.: Hyperledger besu (2021), `https://www.hyperledger.org/use/besu`, accessed 2020-10-13

[16] GmbH, A.A.: Anyblock analytics (2021), `https://www.anyblockanalytics.com/`, accessed 2020-10-13

[17] Gu, W., Raghuvanshi, A., Boneh, D.: Empirical measurements on pricing oracles and decentralized governance for stablecoins. SSRN Electronic Journal p. 17 (2020). https://doi.org/10.2139/ssrn.3611231

[18] Inc., G.P.: The graph (2021), `https://thegraph.com/`, accessed 2020-10-13

[19] Inc., I.: Infura (2021), `https://infura.io/`, accessed 2020-10-13

[20] Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. EBSE Technical Report **EBSE 2007-001** (2007)

[21] Kjäer, M., Di Angelo, M., Salzer, G.: Empirical Evaluation of MakerDAO's Resilience. In: 2021 3rd Conference on Blockchain Research Applications for Innovative Networks and Services (BRAINS). IEEE (2021)

[22] Klages-Mundt, A., Minca, A.: While stability lasts: A stochastic model of stablecoins. CoRR **2004.01304** (2020), `https://arxiv.org/abs/2004.01304`

[23] Klages-Mundt, A., Minca, A.: (In)stability for the blockchain: Deleveraging spirals and stablecoin attacks. CoRR **1906.02152** (2021), `https://arxiv.org/abs/1906.02152`

[24] LFW: MakerDAO Forum Profiles: LongForWisdom (2021), `https://forum.makerdao.com/u/LongForWisdom`, accessed 2020-11-24

[25] Limited, P.T.: Parity ethereum client - open ethereum (2021), `https://www.parity.io/ethereum/`, accessed 2020-10-13

[26] Maker Community: Maker Community Development - Governance and Risk Meeting (2021), `https://community-development.makerdao.com/governance/governance-and-risk-meetings`, accessed 2020-11-26

[27] Maker Foundation: Introducing the new whitepaper for the dai stablecoin system (2017), https://blog.makerdao.com/introducing-the-new-whitepaper-for-the-dai-stablecoin-system/, accessed 2021-04-12

[28] Maker Foundation: Maker Protocol 101 (2019), https://docs.makerdao.com/maker-protocol-101, accessed 2021-05-10

[29] Maker Foundation: Governance (2020), https://community-development.makerdao.com/makerdao-scd-faqs/scd-faqs/governance, accessed 2020-10-20

[30] Maker Foundation: Maker Governance Portal (2020), https://vote.makerdao.com/, accessed 2021-10-20

[31] Maker Foundation: Maker Foundation Multi-Collateral DAI Public Releases Page (2021), https://changelog.makerdao.com/, accessed 2021-06-02

[32] Maker Foundation: Maker Multi Collateral DAI Documentation (2021), https://docs.makerdao.com/, accessed 2021-07-09

[33] Maker Foundation: Maker Multi Collateral DAI Smart Contract Source Code (2021), https://github.com/makerdao/dss/tree/master/src, accessed 2021-04-20

[34] Maker Foundation: Maker Protocol Smart Contracts Modules Documentation (2021), https://docs.makerdao.com/smart-contract-modules, accessed 2020-12-21

[35] Maker Foundation: The Maker Foundation Transfers Trademarks and Software IP to Independent Dai Foundation (2021), https://blog.makerdao.com/the-maker-foundation-transfers-trademarks-and-software-ip-to-independent-dai-foundation/, accessed 2020-01-28

[36] Maker Foundation: The Maker Protocol: MakerDAO's Multi-Collateral Dai (MCD) System (2021), https://makerdao.com/en/whitepaper/, accessed 2021-02-10

[37] Namecoin: Namecoin (2021), https://www.namecoin.org, accessed 2021-07-02

[38] Nethermind: Nethermind - ethereum solutions for developers and enterprises (2021), https://nethermind.io/, accessed 2020-10-14

[39] Park, D., Zhang, Y., Saxena, M., Daian, P., Roşu, G.: A formal verification tool for ethereum vm bytecode. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. p. 912–915. ACM (2018)

[40] Pernice, I.G.A., Henningsen, S., Proskalovich, R., Florian, M., Elendner, H., Scheuermann, B.: Monetary Stabilization in Cryptocurrencies – Design Approaches and Open Questions. In: 2019 Crypto Valley Conference on Blockchain Technology (CVCBT). pp. 47–59. IEEE (2019). https://doi.org/10.1109/CVCBT.2019.00011

[41] Protofire: Maker Governance Dashboard – Executive Votes (2020), `https://mkrgov.science/executive/0x529b8b4b62b5f32bd47412988a0a66d72f86ba00`, accessed 2021-04-11

[42] Protofire: Maker Governance Dashboard – Executive Votes (2020), `https://mkrgov.science/executive/0xd77ad957fcf536d13a17f5d1fffa3987f83376cf`, accessed 2021-04-11

[43] Protofire: Maker Governance Dashboard – Executive Votes (2020), `https://mkrgov.science/executive/0x049e4d10c1b7280cfed5b0d990e39f9c54529a32`, accessed 2021-04-11

[44] Protofire: Maker Governance Dashboard (2021), `https://mkrgov.science/`, accessed 2020-11-26

[45] Protofire: Maker Governance Dashboard - Executive Votes (2021), `https://mkrgov.science/executive`, accessed 2020-11-26

[46] Protofire: Maker Governance Dashboard - Governance Polls (2021), `https://mkrgov.science/polls`, accessed 2020-11-26

[47] rich.brown: Mandate: Interim Governance Facilitators (2019), `https://forum.makerdao.com/t/mandate-interim-governance-facilitators/264`, accessed 2020-11-24

[48] rich.brown: MakerDAO Forum Profiles: Richard Brown (2021), `https://forum.makerdao.com/u/rich.brown`, accessed 2020-11-24

[49] Roman, B., Christoph, M.B., John Leslie, K.: Governance in the blockchain economy: A framework and research agenda. Journal of the Association for Information Systems pp. 1020–1034 (2018). https://doi.org/10.17705/1jais.00518

[50] Satoshi Nakamoto: Bitcoin: A Peer-to-Peer Electronic Cash System (2009), `https://bitcoin.org/bitcoin.pdf`, accessed 2021-02-12

[51] Smart Contracts Domain Team: Mip45: Liquidations 2.0 (liq-2.0) - liquidation system redesign (2021), `https://forum.makerdao.com/t/mip45-liquidations-2-0-liq-2-0-liquidation-system-redesign/6352`, accessed 2021-04-11

[52] Szabo, N.: Bit gold (2005), `https://nakamotoinstitute.org/bit-gold/`, accessed 2020-04-01

[53] u/makerburn: makerburn.com (2021), https://docs.makerdao.com/, accessed 2020-02-01

[54] User:Furunodo: Utxo - bitcoin wiki (2021), https://en.bitcoin.it/wiki/UTXO, accessed 2020-04-06

[55] User:Theymos: Transaction - bitcoin wiki (2021), https://en.bitcoin.it/wiki/Transaction, accessed 2020-04-06

[56] User:Tybeet: Transaction - bitcoin wiki (2021), https://en.bitcoin.it/wiki/Talk:Difficulty, accessed 2020-04-06

[57] Ventresca, M.J., Mohr, J.W.: Archival research methods. In: The Blackwell Companion to Organizations, pp. 805–828. Blackwell Publishing (Oct 2017). https://doi.org/10.1002/9781405164061.ch35

[58] Vitalik Buterin: Ethereum whitepaper (2013), https://ethereum.org/en/whitepaper/, accessed 2021-07-02

[59] Vranken, H.: Sustainability of bitcoin and blockchains. Current Opinion in Environmental Sustainability **28**, 1–9 (2017). https://doi.org/https://doi.org/10.1016/j.cosust.2017.04.011

[60] Wohlin, C., Höst, M., Henningsson, K.: Empirical Research Methods in Software Engineering, pp. 7–23. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)

[61] Ye, C., Li, G., Cai, H., Gu, Y., Fukuda, A.: Analysis of security in blockchain: Case study in 51Conference on Dependable Systems and Their Applications (DSA). pp. 15–24. IEEE (2018). https://doi.org/10.1109/DSA.2018.00015

[62] yearn: yearn.finance (2021), https://yearn.finance/, accessed 2020-06-22

[63] Yin, R.K.: Case study research and applications : design and methods. SAGE, Los Angeles, 6th edition edn. (2018)