Springer Proceedings in Business and Economics

Panos Pardalos
Ilias Kotsireas
Yike Guo
William Knottenbelt *Editors*

Mathematical Research for Blockchain Economy

2nd International Conference MARBLE 2020, Vilamoura, Portugal

Springer

Springer Proceedings in Business and Economics

Springer Proceedings in Business and Economics brings the most current research presented at conferences and workshops to a global readership. The series features volumes (in electronic and print formats) of selected contributions from conferences in all areas of economics, business, management, and finance. In addition to an overall evaluation by the publisher of the topical interest, scientific quality, and timeliness of each volume, each contribution is refereed to standards comparable to those of leading journals, resulting in authoritative contributions to the respective fields. Springer's production and distribution infrastructure ensures rapid publication and wide circulation of the latest developments in the most compelling and promising areas of research today.

The editorial development of volumes may be managed using Springer's innovative Online Conference Service (OCS), a proven online manuscript management and review system. This system is designed to ensure an efficient timeline for your publication, making Springer Proceedings in Business and Economics the premier series to publish your workshop or conference volume.

More information about this series at http://www.springer.com/series/11960

Panos Pardalos · Ilias Kotsireas ·
Yike Guo · William Knottenbelt
Editors

Mathematical Research for Blockchain Economy

2nd International Conference MARBLE 2020, Vilamoura, Portugal

Springer

Editors
Panos Pardalos
Department of Industrial and Systems Engineering
University of Florida
Gainesville, FL, USA

Yike Guo
Data Science Institute
Imperial College London
London, UK

Ilias Kotsireas
Wilfrid Laurier University
Waterloo, ON, Canada

William Knottenbelt
Department of Computing
Imperial College London
London, UK

ISSN 2198-7246 ISSN 2198-7254 (electronic)
Springer Proceedings in Business and Economics
ISBN 978-3-030-53355-7 ISBN 978-3-030-53356-4 (eBook)
https://doi.org/10.1007/978-3-030-53356-4

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume presents the proceedings of the 2nd International Conference on Mathematical Research for Blockchain Economy (MARBLE 2020), originally scheduled to be held in Vilamoura, Portugal, from June 8 to 10, 2020. The advent of COVID-19 pandemic has necessitated, in common with many other conferences, a postponement to a later date which will be fixed once the evolution of the pandemic around the globe becomes clearer and allows for safe travel.

Thankfully, the pandemic has not prevented us from being able to put together an exciting programme of research papers, keynote talks and a tutorial, in line with MARBLE's goal to provide a high-profile, cutting-edge platform for mathematicians, computer scientists and economists to present the latest advances and innovations related to the quantitative and economic aspects of blockchain technology. In this context, the Technical Programme Committee has accepted 10 research papers for publication and presentation on themes including incentives, game theory, the analysis of portfolios containing cryptoassets, carbon trading and quantum computing. The technical programme also features keynotes by the following distinguished speakers: Garrick Hileman (Blockchain.com), Dawn Manley (Splunk), Silvio Micali (Algorand), Annika Monari (Aventus) and Alan Vey (Aventus), as well as a tutorial on Decentralised Finance presented by Lewis Gudgeon and Dominik Harz (Imperial College London).

We thank all authors who submitted their innovative work to MARBLE 2020. In addition, we thank all members of the Technical Programme Committee and other reviewers, everyone who submitted a paper for consideration, the General Chairs, Profs. Yike Guo and Panos Pardalos, the Organisation Chair, Jas Gill, the Web Chair, Kai Sun, the Publication Chair, Ilias Kotsireas, the Finance Chair, Diana OMalley, the Publicity Chair, Sam Werner, and other members of the Centre for

Cryptocurrency Research and Engineering who have contributed in many different ways to the organisation effort, particularly Katerina Koutsouri. Finally, we are grateful to our primary sponsor, the Brevan Howard Centre for Financial Analysis, for their generous and ongoing support.

Vilamoura, Portugal
May 2020

Panos Pardalos
Ilias Kotsireas
Yike Guo
William Knottenbelt

Contents

Smart Contract Derivatives

Kostis Karantias, Aggelos Kiayias, and Dionysis Zindros

Abstract The abilities of smart contracts today are confined to reading from their own state. It is useful for a smart contract to be able to react to events and read the state of other smart contracts. In this paper, we devise a mechanism by which a *derivative* smart contract can read data, observe the state evolution, and react to events that take place in one or more *underlying* smart contracts of its choice. Our mechanism works even if the underlying smart contract is not designed to operate with the derivative smart contract. Like in traditional finance, derivatives derive their value (and more generally state) through potentially complex dependencies. We show how derivative smart contracts can be deployed in practice on the Ethereum blockchain without any forks or additional assumptions. We leverage any NIPoPoWs mechanism (such as *FlyClient* or *superblocks*) to obtain succinct proofs for arbitrary events, making proving them inexpensive for users. The latter construction is of particular interest, as it forms the first *introspective* SPV client: an SPV client for Ethereum in Ethereum. Last, we describe applications of smart contract derivatives which were not possible prior to our work, in particular the ability to create decentralized *insurance* smart contracts which insure an underlying on-chain security such as an ICO, as well as futures and options.

K. Karantias · A. Kiayias · D. Zindros (✉)
IOHK, Athens, Greece
e-mail: dionyziz@gmail.com

K. Karantias
e-mail: kkarantias@gmail.com

D. Zindros
University of Athens, Athens, Greece

A. Kiayias
University of Edinburgh, Edinburgh, UK
e-mail: akiayias@inf.ed.ac.uk

P. Pardalos et al. (eds.), *Mathematical Research for Blockchain Economy*, Springer Proceedings in Business and Economics,
https://doi.org/10.1007/978-3-030-53356-4_1

1 Introduction

Smart contracts [4, 16] on blockchain [11] platforms have limited capabilities even when developed in Turing Complete languages such as Solidity. They are executed in their own isolated environment, with full access to their own state, but limited access to what is happening in the rest of the blockchain system. This inherently limits them to performing isolated tasks, unless they interoperate with smart contracts designed explicitly to work together with them.

In this work, we put forth a mechanism which allows so-called *derivative* smart contracts to read the (potentially private) state of other, so-called *underlying*, smart contracts, inspect any events they have fired and when, and more generally react arbitrarily to any changes in the execution of other contracts. Notably, unlike any previous mechanism, the underlying contract may not be designed (or willing) to work with the derivative contract and hence our mechanism allows it to remain *agnostic* to the interaction. Like financial derivatives, smart contract *derivatives* can derive their value from the performance of, potentially multiple, underlying contracts. The dependence between the contracts can be arbitrary.

We develop our solution in the form of a Solidity contract which can be used as an *oracle* to create a derivative contract. We give three options for the instantiation of the oracle contract. The first is based on a special Solidity feature and is the cheapest to implement and use. The second is based on the BTCRelay [5] design and requires helpful users to submit every block to this oracle contract. Finally the third draws from the design in [8] and harnesses the power of Non-Interactive Proofs of Proof-of-Work (NIPoPoWs) [7] for efficiency. The oracle smart contract may be of independent interest, as it functions as an Ethereum SPV client running within Ethereum itself and is the first such *introspective* SPV client of its kind.

Previous work. Granting smart contracts access to data external to their blockchain environment has been studied in the context of oracles in which additional trust assumptions are made by the introduction of a trusted third party or committee [17] or in the form of an oracle [18]. The generic transfer of information *between* blockchains without the introduction of additional assumptions has been studied in the context of sidechains [5, 8, 9, 12].

Contributions. Our contributions are summarized as follows:

1. We posit the problem of smart contract derivatives and put forth a construction solving it without additional assumptions.
2. We propose three instantiations of our oracle; the first relying on features of Solidity, the second inspired from the BTCRelay design and the third utilizing NIPoPoWs.
3. We introduce the first *introspective* SPV client, a client for a blockchain system running within that same blockchain system.
4. We discuss how our scheme can be used to instantiate some standard financial derivatives: insurance, futures and options.

2 Introspective SPV

Notation. We use $x \xrightarrow{?} a$ to denote the Merkle Tree [10] inclusion proof for tree root a and element x. We use $x \xrightarrow{?} a[k]$ to denote the Merkle-Patricia Trie proof for the assignment of key k to x in the MPT rooted at a. The verification result of a concrete proof π is denoted $(\cdot \xrightarrow{\pi} \cdot) \in \{\mathsf{true}, \mathsf{false}\}$.

For our construction, we define an *oracle* smart contract that can answer queries regarding other smart contracts. Notably, the oracle contract is decentralized, i.e., it does not make additional trust assumptions and so does not require a trusted third party or committee. The oracle contract can answer the following queries about arbitrary underlying contracts:

1. Retrieve the value of a private state variable of the underlying contract at any point in the past.
2. Recall whether a particular event was fired by the underlying contract at any point and retrieve the values of the event parameters.
3. Report on any past calls made to the underlying contract's methods, whether these were done by other contracts or by normal accounts, including the values given to the parameters and money paid during the call.

Solidity already has some provisions for smart contract interoperability. For example, a token contract usually follows the ERC-20 interface [15], which allows any other contract to inspect its state and perform actions on it.

Even the most helpful smart contracts currently deployed however would come with limitations. Specifically, reading incoming and outgoing transactions and events emitted is currently impossible. While we could manage to partially work around those limitations with a smart contract framework for helpful contracts that records all relevant transactions and events and makes them easily accessible, it is important to remember that smart contracts are immutable and cannot be changed once deployed. Thus, this solution would not work for existing smart contracts that we may be interested in.

Additionally, this solution comes with extra storage requirements. Storage on Ethereum costs disproportionately more that any other operation, and this cost would have to be paid by the unlucky downstream users of the smart contract. Naturally this presents the need for a solution that does not incur such costs on the downstream users, which we are going to present shortly.

Private variable lookup. Assume a legacy smart contract has a variable of interest that happens to be private. This means that with regular Solidity methods this variable cannot be accessed. We show how this can be worked around with the help of an untrusted third party who provides some proof. Provided one knows an actual block hash b on the best chain, one only has to check two MPT proofs to ensure what the value of the private variable px is, namely $px \xrightarrow{?} \mathsf{storageRoot}[loc(px)]$ and $(\_, \_, \mathsf{storageRoot}, \_) \xrightarrow{?} b.\mathsf{stateRoot}[addr]$ where $loc(px)$ refers to the persistent storage location of the variable px and $addr$ refers to the smart contract address.

Detecting transactions. Recall that Ethereum stores an MPT root of all transactions in its header. Thus the MPT proof $tx \xrightarrow{?} b.\mathsf{transactionsRoot}[H(tx)]$ suffices as proof that $tx \in b$. These proofs are already used in practice to prevent front-running [1].

The above operations can be performed as long as our smart contract can verify that a block header b is part of the current chain. We propose several mechanisms of doing so.

`BLOCKHASH` opcode. Ethereum offers the `BLOCKHASH` opcode that allows a smart contract to obtain previous block hashes. This functionality makes ensuring that a provided block b is in the best chain trivial: the contract extracts $b.\mathsf{height}$, invokes `BLOCKHASH` for that height number and compares $H(b)$ with the result of the `BLOCKHASH` invocation. If those match, the block is valid. Unfortunately this functionality is limited to the past 256 blocks [16]. There is a proposal to remove this artificial limit which is expected to be implemented in a future hard fork of Ethereum [3]. For Ethereum, this is the ideal solution to the block verification problem, resulting in the least possible costs for proving events.

BTCRelay-style SPV. BTCRelay [5] rose to prominence in 2016 as a way to provide Bitcoin SPV client capabilities to any Ethereum smart contract. Every block is submitted to the contract by helpful but untrusted participants and a header-chain is formed and confirmed. A convenient mapping is kept so that it can be decided if any block is in the current best header-chain. BTCRelay also offers incentives for submitters of blocks, where the submitters get rewarded whenever the blocks they have posted are used to prove some event. This scheme can be used for block verification of the Ethereum chain on Ethereum — an "ETCRelay".

NIPoPoWs. NIPoPoWs [2, 7] are succinct strings that prove statements about some chain. Their succinctness makes them perfect candidates to use as proofs for block inclusion on an Ethereum smart contract. Details on their use for this scenario are presented in [8]. Note that this use comes with a host of incentives via collateralization that should be implemented for use in our Introspective SPV client.

Implementation. We summarize all these functionalities in the complete *Introspective SPV* contract shown in Algorithm 1. This is, to our knowledge, the first contract that is an SPV client for its host chain.

We remark that using any storage is not necessary and it is only used for illustrative purposes. All functions can be made to operate based on only arguments they receive, without compromising their security.

Algorithm 1 Introspective SPV contract for Ethereum, on Ethereum.

```
1: contract introspective-spv
2:     function submit-block(b, π)
3:         if ¬verify(b, π) then
4:             return ⊥
5:         end if
6:         valid-blocks ∪= {b}
7:     end function
8:     ▷ verify-* functions return false if b ∉ valid-blocks
9:     function verify-tx(tx, b, π)
10:        return tx ?→ πb.transactionsRoot[H(tx)]
11:    end function
12:    function verify-storage(val, loc, addr, b, π)
13:        return ∃σ : val ?→ π[0]σ[loc] ∧ (_, _, σ, _) ?→ π[1]b.stateRoot[addr]
14:    end function
15:    function verify-event(evt, addr, b, π)
16:        return evt.src = addr ∧ ∃i, r_l : evt ∈ r_l ∧ (_, _, r_l, _) ?→ πb.receiptsRoot[i]
17:    end function
18: end contract
```

3 Concrete Instances

We now move to some notable applications that can be accomplished by contracts which build on the Introspective SPV functionality.

Insurance. A quite useful application of a smart contract derivative is the ability to provide *insurance* for an underlying smart contract. This is a contract between an *insurer* and a *policyholder* account. The contract works as follows. Initially, the *insurer* creates the insurance contract, depositing a large amount of money to it to be used as liabilities in case of claims. Subsequently, after checking that the deposited amount secured against liabilities is sufficient, the future *policyholder* account deposits the *premium* as a payment to the insurance contract, which signs them up for the insurance. The premium can also be paid in installments if desired. Once the premium has been paid, the policy is activated for the particular policyholder.

The derivative smart contract insures against a covered loss event which pertains to an underlying smart contract. Unlike traditional insurance contracts, assessing whether a claim is valid or not is not left up to the insurer or courts of law, but is determined by the smart contract in a predetermined and decentralized manner. As such, there can be no disputes on whether a coverage claim is valid.

One such example constitutes insuring an underlying *ICO* smart contract [13] against a specified loss condition. The condition describes what the policyholder considers to be a failed outcome of the ICO. For instance, the policyholder can specify that the ICO's success requires that there are at least 5 whale investors,

defined as investors each of which has deposited more than $1,000,000 in a single transaction over the course of the ICO's fundraising period.

Insurance claims in this example are made as follows. If the insured determines that there has been a loss event (i.e., there have been fewer than 5 whale investors), then at the end of the ICO's fundraising period, they submit a claim. This claim does not include any proof. The opening of a claim initiates a *contestation* period during which the insurer can submit a counter-claim illustrating that the claim was fraudulent. This counter-claim *does* include a proof, which consists of 5 transactions made to the ICO smart contract each of which pertains to a different investor and is valued more than $1,000,000. This counter-claim proof can be checked for validity by using the means described previously. If there are no counter-claims within the contestation period, then the claimant is compensated. In case the policyholder acts adversarially, making a fraudulent claim, the insurer can always present this counter-claim and avoid paying compensation. In case the insurer acts adversarially, electing not to pay compensation when required to do so, the policyholder will make a claim against which the adversarial insurer will not be able to provide a counter-claim.

It is noteworthy that the contract should be resistant to attacks where a malicious policyholder continuously makes false claims that the honest insurer has to defend against causing them monetary loss. To prevent such attacks, the smart contract may request some collateral from the policyholder when claiming, that is taken from them if they are making a false claim and returned to them when they are making a truthful claim. Such incentive mechanisms have been the subject of extensive study in previous work [14].

Options. Traditionally an option is a contract between a *holder* and a *writer*. It allows the holder to either buy (*call option*) or sell (*put option*) an underlying asset to the writer at a specified *strike price* and until a specified *expiry date* [6]. If the holder elects to exercise the option, the writer is obligated to complete the trade. An option can be traded on the open market like any other asset. Buying an option ensues that the existing holder forfeits their contractual rights and transfers them to the buyer, making them effectively the new holder.

Centralized exchanges have a plethora of ways of enforcing the legal obligations of writers. Specifically, if a writer does not fulfill the valid request of a holder, their account may be frozen, their funds may be seized and further action may be taken against them through the traditional legal system.

To implement options in a decentralized manner we assume the existence of a clearing house, which can be implemented as its own smart contract, that will insure the holder against the event that the writer does not fulfill her contractual obligations. A responsible options buyer only buys an option that comes with a guaranteed insurance similar to the one outlined in the previous section. If the writer fails to fulfill her contractual obligations, a claim with the clearing house is started. A successful claim would be of the form: "On some block that occurred after the start of the option contract and before its expiry, I (the holder) requested to exercise my option. By the block of expiry, no event was fired indicating that my writer acted to fulfill my request for the requested price and amount". After the contestation period, the holder is refunded by the clearing house smart contract.

Futures. Similar to an option, a future is a contract between two parties. The defining difference from an option is that the holder is *obligated* to perform a specified action (either buy or sell) on the underlying asset at the strike price and on the specified expiry date [6]. For simplification the expiry date can be described as a block height inside the smart contract. It is easy to see how this system can also be implemented with the help of a clearing house, similarly to an option. Plainly, in case of fraud, the policyholder could claim that "Between the start of the agreement and the expiry, no Solidity event was fired indicating that the writer bought or sold from me the agreed amount at the agreed price". In case of a fraudulent policyholder, all the clearing house has to do is provide proof that this event was fired in some block in the period of interest.

On the availability of insurers. Exchanges implicitly offer insurance for their users by keeping track of how much money they store with them and making sure they are not over-exposed to risk. Banks implicitly offer insurance for their customers based on their credit-worthiness. The same out-of-band criteria can apply for any institution wishing to insure on-chain. An insurer can create an off-chain agreement with the party who can cause a loss and claim, or rely on some on-chain collateral, potentially denominated in multiple tokens/currencies, to be automatically compensated in case of misbehavior.

References

1. Breidenbach, L., Daian, P., Tramèr, F. & Juels, A. (018). Enter the hydra: Towards principled bug bounties and exploit-resistant smart contracts. In *27th USENIX Security Symposium (USENIX Security 18)* (pp. 1335–1352).
2. Bünz, B., Kiffer, L., Luu, L., & Zamani, M. (2019). Flyclient: Super-light clients for cryptocurrencies. *IACR Cryptology ePrint Archive*, *2019*, 226.
3. Buterin, V. EIP 210: Blockhash refactoring. Available at: https://eips.ethereum.org/EIPS/eip-210.
4. Buterin, V. et al. (2014). A next-generation smart contract and decentralized application platform. *white paper*.
5. Chow, J. BTC Relay. Available at: https://github.com/ethereum/btcrelay.
6. Hull, J. (2017). *Options*. Pearson: Futures and Other Derivatives.
7. Kiayias, A., Miller, A. & Zindros, D. (2020). Non-Interactive Proofs of Proof-of-Work. In *International Conference on Financial Cryptography and Data Security*. Springer.
8. Kiayias, A. & Zindros, D. (2019). Proof-of-Work Sidechains. In *International Conference on Financial Cryptography and Data Security Workshop on Trusted Smart Contracts*. Springer.
9. Lerner, S. D. (2016). Drivechains, sidechains and hybrid 2-way peg designs.
10. Merkle, R. C. (1987). A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques* (pp. 369–378). Springer.
11. Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Available at: https://bitcoin.org/bitcoin.pdf.
12. Sztorc, P. (2015). Drivechain - the simple two way peg. http://www.truthcoin.info/blog/drivechain/.
13. Teutsch, J., Buterin, V. & Brown, C. (2017). Interactive coin offerings. Available at: https://people.cs.uchicago.edu/~teutsch/papers/ico.pdf.
14. Teutsch, J. & Reitwießner, C. (2018). Truebit: a scalable verification solution for blockchains.

15. Vogelsteller, F. & Buterin, V. (2015). ERC-20 token standard. Available at: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md.
16. Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, *151*, 1–32.
17. Zhang, F., Cecchetti, E., Croman, K., Juels, A. & Shi, E. (2016). Town crier: An authenticated data feed for smart contracts. In *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security* (pp. 270–282).
18. Zhang, F., Maram, S. K. D., Malvai, H., Goldfeder, S. & Juels, A. (2019). DECO: Liberating Web Data Using Decentralized Oracles for TLS. *arXiv preprint* arXiv:1909.00938.

Bitcoin Crypto–Bounties for Quantum Capable Adversaries

Dragos I. Ilie, Kostis Karantias, and William J. Knottenbelt

Abstract With the advances in quantum computing taking place over the last few years, researchers have started considering the implications on cryptocurrencies. As most digital signature schemes would be impacted, it is somewhat reassuring that transition schemes to quantum resistant signatures are already being considered for Bitcoin. In this work, we stress the danger of public key reuse, as it prevents users from recovering their funds in the presence of a quantum enabled adversary despite any transition scheme the developers decide to implement. We emphasize this threat by quantifying the damage a functional quantum computer could inflict on Bitcoin (and Bitcoin Cash) by breaking exposed public keys.

1 Introduction

The theory behind quantum computers (QC) was first introduced about 40 years ago. Research in the space has produced outstanding results, that have the potential to undermine the most popular cryptographic protocols in use today. One notable theoretical result is Peter Shor's quantum algorithm [25] which can be used to break digital signature schemes such as RSA or ECDSA. The engineering advancements needed to physically implement such a complex machine have only recently started to appear, but a sudden improvement in the approach towards scaling might lead to a powerful QC appearing virtually overnight.

D. I. Ilie (✉) · W. J. Knottenbelt
Centre for Cryptocurrency Research and Engineering, Imperial College London, London, UK
e-mail: dii14@imperial.ac.uk

W. J. Knottenbelt
e-mail: wjk@imperial.ac.uk

K. Karantias
IOHK, Athens, Greece
e-mail: kostis.karantias@iohk.io

P. Pardalos et al. (eds.), *Mathematical Research for Blockchain Economy*, Springer Proceedings in Business and Economics,
https://doi.org/10.1007/978-3-030-53356-4_2

The Bitcoin community is also affected by these developments, as the mechanism for ensuring ownership of funds relies on ECDSA. Bitcoin's cryptography must be updated; in fact there are plenty post-quantum cryptographic schemes to choose from if one is willing to sacrifice speed and storage. Such a scheme will be implemented in Bitcoin at some point and the majority of users will be able to safely lock their funds with quantum resistant signatures. However, in the extreme scenario of a Quantum Computer appearing without notice, not all users would be able to benefit from this upgrade. Interestingly, the recommended practices in Bitcoin would offer some level of quantum resistance that allows recovering funds safely, but unfortunately, many users do not follow these. In this paper we analyse Bitcoin (BTC) and Bitcoin Cash (BCH) for the amount of funds secured by exposed public keys; or, from the perspective of a quantum research group, the "crypto–bounty" for engineering a powerful quantum computer.

1.1 Contributions

This paper builds upon previous work of the authors [27] and brings the following contributions to the research space:

1. In Sect. 4.1 we describe the setting in which a quantum enabled adversary would operate if it were to start attacking the Bitcoin network considering developers and users take immediate measures to protect their funds and recover the network.
2. In Sect. 4.3 we present two models of attackers: one that can run Shor's algorithm virtually instantly and a slower one that might be more realistic for the first generations of attackers.
3. In Sect. 4.4 we describe attack vectors for maximizing the crypto–bounty, i.e. the amount of funds that are impossible to recover by legitimate users in the presence of the attacker.
4. In Sect. 5 we present a study of the evolution of the crypto–bounty in Bitcoin and its most significant fork, Bitcoin Cash. Furthermore, we describe our methodology for obtaining these results and discuss what can be deduced from them.

2 Related Work

Previous work [27] of some of the authors of this paper has looked into revealed public keys to motivate the introduction of a protocol for transitioning to quantum resistance. They have found that approximately 33% of Bitcoin is secured by public keys that are exposed either trivially in an output, or in an input due to address reuse. In this study, we would like to consolidate that data and perform a more in-depth analysis looking not only at the current UTXO set, but also at the history of public key exposure and its source.

Other researchers have carried out a similar analysis [6], but their estimates represent a much lower bound than what we are providing and does not consider P2SH and P2WSH type addresses which are very popular in recent years. Furthermore, their study only looks at reused addresses, while we also inspect reused public keys between different addresses.

We have also become aware of other similar studies created by members of the cryptocurrency space, but we could not find descriptions of their methods or details of the results. One such analysis was done by one of the developers of BlockSci [12], who summarized his result that 5.8 million Bitcoins are secured by exposed public keys in an online discussion [11].

3 Background

In this section we briefly cover the basic concepts necessary for understanding the motivation behind the analysis we are conducting and the methods used to perform it. We present some of the structures that are used in Bitcoin to secure and move funds across the network and offer insight into the workings of Shor's quantum algorithm.

3.1 Bitcoin Fundamentals

Bitcoin transactions are data structures that encode the transfer of ownership of funds using inputs and outputs. Transactions are created, shared and stored by network participants following a protocol which dictates their validity and ordering. Funds reside in unspent transaction outputs (aka. UTXOs), each having an associated value and a locking script. For usability, some standard types of locking scripts are encoded in base 58 to produce addresses, which can easily be shared between users. To spend funds, a transaction input references an output and includes an unlocking script, which is used in combination with the locking script for validation. Ownership is guaranteed by a combination of hash commitments and public key cryptography. In general, each input contains a digital signature over the transaction spending the output. The signature is verified against a public key that is encoded in the output either in plaintext or inside a hash commitment which is revealed by the input. Depending on the type of locking script in the output, the input consuming it needs to provide unlocking data in various formats [4]. We can distinguish the following standard script types:

1. **Pay-To-Public-Key (P2PK)** is the script type used for the first Bitcoin transaction. This is the most simplified form of locking and unlocking scripts: the public key is written in plaintext in the locking script, and the digital signature appears in the input, also in plaintext. Only the owner of the corresponding private key can

create a signature that would be verified against the public key from the locking script.
2. **Pay-To-Multisig (P2MS)** is a script pattern that allows users to lock funds to multiple public keys but require signatures only from a subset of those. Similarly to P2PK the public keys are all listed in plaintext in the locking script together with a minimum threshold number of signatures that need to be provided in the unlocking script. However, public keys are quite large and data on the blockchain costs fees, so this outputs are not very popular.
3. **Pay-To-Public-Key-Hash (P2PKH)** is an improved version of P2PK. The locking script contains a 20 byte hash commitment to the public key and the input contains both the public key and digital signature. This type of script was introduced to minimise the size of the output as hashes are only 20 bytes compared to public key which are 65 bytes uncompressed or 33 bytes compressed.
4. **Pay-To-Script-Hash (P2SH)** outputs are yet another improvement; instead of specifying the locking script in the output, a 20 byte hash commitment to it is stored instead. A valid input contains the unlocking script and the pre–image of the hash commitment from the output. This type of output has the same size as P2PKH outputs, but allows for more complex locking scripts to be encoded without requiring the payer to incur the fees associated with the extra data. This script is most commonly used to nest P2MS, P2WPKH, or P2WSH.
5. **Pay-To-Witness-Public-Key-Hash (P2WPKH)** was deployed in 2017 via a soft-fork in order to address several issues such as signature malleability and throughput limitations. Similarly to P2PKH the locking script contains a 20 byte hash of the public key and the input holds the public key and signature. The difference is that the input data is held in a segregated area called `segwit` which does not contribute towards the hash of the transaction and size limit of a block.
6. **Pay-To-Witness-Script-Hash (P2WSH)** is a script type introduced together with P2WPKH and represents the P2SH version of `segwit`. The output contains a 32 byte hash of the actual locking script, 12 bytes larger than the 20 byte hash from P2SH. This increase is meant to improve security against possible collision attacks as the work needed to find a collision in a 20 byte hash is no longer infeasible. This script type usually nests P2MS.

Digital Signatures in Bitcoin are implemented using the Elliptic Curve Digital Signature Algorithm (ECDSA); an implementation of the Digital Signature Standard (DSS) based on Elliptic Curve Cryptography (ECC) [3]. ECC is an approach to public key cryptography that relies on the mathematics of elliptic curves over finite fields. To this end, the public key is a point on an elliptic curve,[1] and the secret key is the exponent at which a base point[2] is raised in order to obtain the public key. As in any other digital signature scheme, the private key is kept secret and used to sign messages, while the public key is published and used to validate signatures. ECC and therefore ECDSA rely on the assumption that it is intractable to solve

[1] Bitcoin uses parameters defined in secp256k1 [20] due to several mathematical properties which allow computations to be optimized.

[2] See Footnote 1.

the Elliptic Curve Discrete Logarithm Problem (ECDLP) [7], which would allow deducing the private key from the public key, defeating the whole purpose of a digital signature algorithm. Similarly to the more famous integer factorisation problem [8], ECDLP has no known reasonably fast (e.g. polynomial–time) solution on a classical computer [17].

3.2 Quantum Computing

Quantum Computing is experiencing an increase in interest in the last few years as more giants of industry become interested in the possibilities promised by the theory behind quantum algorithms, i.e. they are able to solve certain classes of mathematical problems with drastically improved running time compared to classical algorithms. Although it is outside the scope of this paper to explain the mechanism through which quantum algorithms achieve quadratic or exponential speed-ups over their classical counterparts, we would like to offer some intuition about why they are of interest for ECC, and hence, Bitcoin. Usually, quantum computations encode many values on a single register and then perform computations on it. However, measuring the register would reduce the superposition to a single value, losing all information about the other solutions. Thus, quantum algorithms are designed to manipulate the register in order to extract knowledge about the hidden structure of the values on the register.

Shor's Algorithm [25] is a quantum algorithm that, in generalized form, can solve a class of problems known as the hidden subgroup problem over a finite Abelian field [18]. In fact most public key cryptography in use today relies on the intractability of this problem in certain groups. Indeed, the ECDLP can also be reduced to this problem, meaning that such an algorithm would be able to compute a private key from a public key in relatively short time. The core of the algorithm is the application of the Quantum Fourier Transform (QFT) circuit [15] which runs in polynomial–time. Proos and Zalka have approximated that approximately $6n$ (where n is the security parameter, 256 bits for Bitcoin) qubits would be needed to run the computation [19].

4 Context and Modelling

In this section, we assume the setting under which the quantum capable attacker operates and explain what would constitute the crypto–bounty available to him, even if the community learns of his existence and tries to switch to a quantum resistant signature scheme. As described in previous work in [27, 28], the main attack vectors available to quantum capable adversaries stem from exposed public keys. Such adversaries can use Shor's quantum algorithm with a revealed public key as input, and compute the associated private key, gaining complete control over the original owner's funds.

Live Transaction Hijacking is a type of attack similar to double spending [13, 14, 22, 26]. The attacker can create conflicting transactions that spend the same UTXOs as honest transactions but send the funds to addresses under his control. A quantum enabled adversary can perform this attack only if he is able to compute the private key associated to a public key revealed in the input of a transaction while the transaction is still in the miners' mempool. In the presence of a malicious actor capable of this attack, users would not be able to create any new transactions as this would reveal their public keys and their transactions would be susceptible to hijacking.

4.1 Transitioning to Quantum Resistance

As the community learns of the presence of a quantum attacker approaching the ability to perform live transaction hijacking, Bitcoin users would have to stop all activity and deploy a scheme for transitioning to quantum resistance under the assumption that they are in the presence of an adversary capable of live hijacking. Such a protocol would have to rely on some construction that does not expose the ECDSA public key immediately (or ever), but allows for the linking of transactions to it. Some schemes that achieve this feat, are described in [1, 27, 29], but the community can settle on any viable alternative. However, we note that only UTXOs secured by not-yet-revealed public keys could be safely transitioned to quantum resistance as public keys that are exposed at the time of deploying the transition protocol would still be cracked by the attacker who would be essentially indistinguishable from the actual owner. Therefore, any UTXOs secured by revealed public keys would constitute a bounty for the quantum capable attacker as the legitimate owners would not be able to move the funds to quantum resistant outputs under their control.

4.2 Attack Vectors Considered

Given this setting, quantum enabled adversaries could still consume any outputs associated to public keys revealed before the deployment of the transition scheme. Any attempt from the legitimate owner to transition the funds can be hijacked using the private key computed using Shor's quantum algorithm. Below we enumerate some types of UTXOs attackers can target by obtaining exposed public keys. We only focus on these scenarios as the information is publicly accessible and allows for the retrospective analysis of balances of revealed public keys at a certain block height. Other types of public key unveiling are presented in [27].

1. Outputs of type P2PK and P2MS display the public key in plaintext in the output of the transaction.
2. Outputs of type P2PKH, P2SH, P2WPKH, or P2WSH where the necessary public key has been used as part of P2PK or P2MS, thus exposing it.

3. Outputs of type P2PKH, P2SH, P2WPKH, or P2WSH where the locking script is being used to receive multiple payments which are not consumed at once. This sort of locking script reuse renders the unspent outputs vulnerable as the public key is exposed in the unlocking script of an input. Such behaviour is discouraged due to a number of privacy attacks [5, 9, 16, 24, 30], but many wallet providers and even large exchanges ignore these practices.

4.3 Adversary Model

Firstly, we assume the quantum capable attacker can only affect the blockchain using Shor's algorithm. As other studies [2, 23, 28] have discussed, quantum computers could be used to run Grover's quantum algorithm [10] in order to obtain a quadratic speed-up in mining. It is not clear if this would lead to a real advantage as current ASIC miners are highly optimized and supposedly much cheaper to run than quantum computers. Therefore, our aim is to analyse only the stealing of funds by using Shor's algorithm.

The assumption we are working with is that once a quantum adversary starts acting maliciously, the Bitcoin community detects this behaviour and clients are quickly updated to use a scheme for transitioning to quantum resistance, thus invalidating simple ECDSA signatures.[3] At this point, the attacker can only target the UTXOs secured by already revealed public keys. As legitimate users learn of this development, they race to spend the vulnerable UTXOs in order to transition them to quantum resistant outputs. At the same time, quantum–capable attackers try to spend the same UTXOs using the private keys they managed to break until this point. However, if the attacker cannot run Shor's algorithm fast enough, some users will manage to transition their funds without the attacker having time to break their public key even though it was revealed.

Therefore, in order to maximize the value of the crypto–bounty, an adversary would have to first collect all the revealed public keys, break them, and only then start the attack. This strategy allows him to collect at least all the UTXOs secured by revealed public keys at the time the transitioning protocol is deployed. Although it is impossible to estimate the clock speeds of quantum computers as the technology behind them is still in early stages and the most efficient approach has not been determined yet, we will differentiate between two purely hypothetical types of quantum capable attackers.

Instant attacker is able to deduce private keys from public keys considerably faster than a transaction sits in a miner's memory pool before it is included in a block.

[3] We use the word "simple" as any protocol for transitioning to quantum resistance that aims to be deployed via a soft fork, needs to verify ECDSA signatures for compatibility with non-upgraded clients. However, it must also verify some other form of quantum resistant cryptography such that attackers cannot exploit the scheme.

Our analysis indicates that the median input count per Bitcoin block in the past year is 5350. Thus, an instant quantum attacker should be able to break approximately 5350 public keys every 10 min,[4] meaning he would need to run Shor's algorithm in approximately 100 milliseconds. This speed suffices because the attacker can offer higher fees than honest users, thus incentivizing miners to only select his transactions. Such an attacker could start acting maliciously immediately after he develops a quantum computer as the computation speed he benefits from allows him to scan the mempool and hijack any attempt at transitioning funds to a quantum resistant output.

Slow attacker represents a more realistic although still hypothetical adversary that is able to deduce private keys from revealed public keys but needs a much larger window of time in order to break all the public keys that are revealed. The attacker builds a database of computed private keys, but at the same time, some of these become unusable as users move their funds before the attacker has finished collecting a sufficient amount. There are heuristics the attacker could employ in order to minimize the amount of private keys he attempts to break and maximize profit. For instance, he could analyse patterns of public key usage and target those that are inactive for long periods of time, under the assumption that the original owners have lost the private keys, and therefore, control of the funds. Another effective strategy could be identifying public keys which are used mainly for receiving money, e.g. exchanges use addresses where users deposit their funds but rarely withdraw. However, for the purposes of this study, we assume that at any given moment the attacker has broken only the public keys revealed at least a year in the past.

4.4 Aggregating Vulnerable Outputs

Regardless of the speed of a quantum adversary, in order to identify all the vulnerable UTXOs, an attacker needs to create and maintain two indexes: one from potential locking scripts to revealed public keys and one from public keys to the associated private keys. This allows him to prioritize the breaking of public keys according to the value secured by them. For some output types such as P2PK and P2MS this task does not pose difficulty as the public key is revealed directly in the locking script. In general, an attacker could listen for all transactions and group outputs by their locking script even though the public key that unlocks them is not visible yet. Once at least one of these outputs is spent, the public key used in the unlocking script can be linked to the UTXOs that have not been consumed. However, these strategies only consider reuse of the same locking script in multiple outputs but not the reuse of public keys inside different types of locking scripts. To this end, we can define the concept of equivalent locking scripts:

[4] 10 min is an estimate for the average time a transaction sits in the mempool before being included in a block.

Equivalent locking scripts require the same public key to unlock them. It is possible to detect reuse of public keys across the standard types of outputs presented in Sect. 3. In surplus, to the locking scripts equivalent to a public key, an attacker could also generate equivalent locking scripts for P2MS, i.e. P2MS nested in one of P2SH or P2WSH. We describe the operations to generate each type of equivalent standard locking script below:

1. Any public key, pk, can be hashed using `HASH160(pk)`[5] to obtain the 20 byte hash commitment used inside P2PKH and P2WPKH locking scripts.
2. Any locking script, S, of type: P2PK, P2MS, P2PKH, P2WPKH, or P2WSH can be hashed using `HASH160(S)` to obtain the 20 byte hash commitment used inside P2SH.
3. Any locking script, S, of type: P2PK, P2MS can be hashed using `SHA-256(S)` to obtain the 32 byte hash commitment used inside P2WSH. Notice that `SHA-256(S)` was already computed as part of calculating HASH160.[6]
4. Any combination of up to 15 public keys can be used to generate P2MS locking scripts. Although there is no direct benefit in obtaining P2MS locking scripts as they display the public keys in the output, there might be P2SH or P2WSH outputs which include a hash commitment to a P2MS otherwise not revealed. These can be generated by hashing every possible P2MS obtained from the set of exposed public keys using `HASH160` for P2SH or `SHA-256` for P2WSH. We note that generating all these scripts takes polynomial time (with exponent 15) in the number of unique public keys; in our analysis, we have found approximately 700 million unique public keys. Even if we only try to build combinations of just 3 public keys, it would still take 100 thousand years for the fastest miner[7] on the market today to compute that many hashes. Given the amount of computation required to identify such outputs and the low value of P2SH outputs in general, we do not consider this attack any further.

To use the above data, an adversary needs to keep an index from each possible locking script to the private key that can unlock it. For every public key, the 6 most common locking scripts (i.e. P2PK, P2PKH,[8] P2SH-P2PK, P2SH–P2WPKH, P2WSH–P2PK, P2SH–P2WSH) can be generated and included in the index. Furthermore, any P2MS locking script can be used to generate locking scripts of type P2SH-P2MS and P2WSH-P2MS. With 700 million different public keys and 55 million P2MS locking scripts, the attacker's index would hold approximately 4.3 billion entries. If needed, access to such an index can be optimized using a bloom filter.

[5] `HASH160(x) = RIPEMD-160(SHA-256(x))` [21].

[6] See Footnote 5.

[7] Currently, the fastest ASIC miner we are aware of is the Ebit E10 capable of computing 18Th/s.

[8] Note that P2PKH contains the same hash commitment as P2WPKH, therefore it suffices to store only one of them.

5 Crypto–Bounty Analysis

In this section we describe our approach to estimating the value of cryptocurrency, on the Bitcoin and Bitcoin Cash blockchains, vulnerable to an instant and slow attacker. We have analysed the current state of the Bitcoin and Bitcoin Cash networks, but also the historic evolution of the crypto–bounty, in order to deduce and compare patterns of public key reuse in the communities.

5.1 Methodology

To complete this analysis we have used a combination of open source tools and libraries made available by the Blockchain community. Most notably, BlockSci [12] allowed us to obtain metrics for the number of outputs vulnerable and their value. BlockSci uses a custom parser to generate indexes which are not found in usual implementations of full nodes, such as: index of transactions by address, index of inputs by referenced output, index of addresses by public key, etc. These constructions enable us to analyse various metrics throughout the history of different Bitcoin-based cryptocurrencies.

For the historic analysis we would like to estimate the amount of cryptocurrency that is locked in vulnerable UTXOs at a given point in time, or in domain specific terms: at a given blockchain height, h. Naturally, we only count the outputs which were unspent at that time and associated to a public key which has already been revealed. Notice that most of the outputs that were part of the UTXO set at height h, are actually spent at a later block height $h' > h$, directly revealing the public key associated to them. Therefore, we need to build an index of unique public keys by height of first reveal. We also use an index built by the BlockSci parser to link each output to the input consuming it. Therefore, we can formulate our strategy for computing the evolution of the crypto–bounty available to a quantum enabled adversary.

For each output in every transaction in every block B, compute the height at which the necessary public key is broken. For an instant attacker, the height at which a public key is revealed is also the height at which it is broken. However, for a slow attacker we must consider a delay of one year from the time the public key is revealed. To generalise, we will define `ShorDelay` as the time delay it takes to run Shor's Algorithm for a certain quantum attacker. Depending on the type of output, we can obtain the height H_{reveal}, at which the attacker could have started running Shor's algorithm on the public key associated to the output, in the following ways:

1. **P2PK** → We query the index of public keys to obtain the first height at which the public key in the output was revealed, H_{reveal}.
2. **P2MS** → For this type of output a minimum threshold, m, of public keys need to be broken in order to obtain control over it. We can consider H_{reveal} is the height

at which m of the public keys were revealed. We lookup each of the public keys in the index and sort the heights returned. H_{reveal} is the value on position m (indexed from 1) in the list.

3. **Other spent outputs** → We perform a lookup in the index of inputs by referenced outputs to obtain the unlocking script associated to this output. Consequently, we find the public key needed and lookup the height at which it was revealed, H_{reveal}, using the index of public keys. For nested P2MS unlocking scripts (i.e. P2SH-P2MS, P2WSH-P2MS) there will be multiple public keys. In that case we perform the same operations as for P2MS to compute H_{reveal}.
4. **Other unspent outputs** → The remaining outputs are UTXOs which are not P2PK or P2MS. For these outputs we rely on the BlockSci index of equivalent addresses; this is similar to the index of equivalent locking scripts we described in Sect. 4. Thus, for every UTXO with a locking script which does not directly reveal the public key, we query this index for the equivalent P2PK address. From this we extract the public key and lookup the height at which it was first revealed, H_{reveal}. Due to space and time complexity, BlockSci does not generate every possible P2SH script when implementing the concept of equivalent addresses. Therefore, some P2SH UTXOs will not be identified as equivalent to public keys that are revealed.

Having obtained the height at which the public key of each output was revealed, we can define the height at which an attacker is capable of breaking it as the minimum between H_{reveal} incremented with the delay of running Shor's algorithm and the height at which the output appeared: $B.height$. Therefore, $H_{broken} = min(B.height, \texttt{ShorDelay} + H_{reveal})$. If H_{broken} is smaller than the height at which the output is spent H_{spent}, the adversary would have had a window of attack: (H_{broken}, H_{spent}), during which he could have stolen the value of this output. Finally, we aggregate the values of overlapping windows to produce the results that follow.

5.2 Results and Discussion

All the data presented in our work was collected at blockchain height 605650 for both Bitcoin (BTC) and Bitcoin Cash (BCH). At this block height, there are 18M coins in circulation on each chain.

The current crypto-bounty is presented in Table 1. We notice that BCH users behave more carelessly with their keys. While 31.77% of BTCs are locked in 26.66M vulnerable UTXOs, in BCH 39.42% of the coins are locked in 16.25M UTXOs. Apparently, BCH users lock more of their funds in less outputs than those of Bitcoin. We believe this is a consequence of the `segwit` fork, as BTC blocks do not include the signature data any more, thus allowing for more outputs and inputs than BCH blocks.

Table 1 The current crypto–bounty in Bitcoin and Bitcoin Cash, from the perspective of an instant and slow attacker

		P2PK	P2PKH	P2MS	P2SH	P2WPKH	P2WSH	Total	% of Supply
Instant	BTC	1.75M	1.92M	41.43	2.03M	22.4K	8.77K	5.74M	31.89%
	BCH	1.76M	4.33M	30.6	4.33M	N/A	N/A	7.12M	39.55%
Slow	BTC	1.75M	1.46M	41.34	0.76M	2.9K	3.34K	3.99M	22.16%
	BCH	1.76M	2.28M	30.59	0.21M	N/A	N/A	4.25M	23.61%

The evolution of the crypto–bounty for an instant and slow attacker can be seen in Fig. 1 for BTC and Fig. 2 for BCH. We have left out the plots for P2MS and `segwit` type outputs as their values are insignificant in comparison to the other outputs and would just clutter the image. We did, however, tally their contribution towards the overall crypto–bounty (i.e. the Sum plot).

In Fig. 1, notice how the plots for P2PK converge around year 2013 and then remain approximately constant until the present, indicating that there is almost no activity involving P2PK outputs after 2013. This is expected as they are considered legacy locking scripts, but the fact that these outputs have not been moved for years, could signify that their owners have lost control of the private keys. In consequence, no scheme can transition them to quantum resistance and we can consider the 1.75M

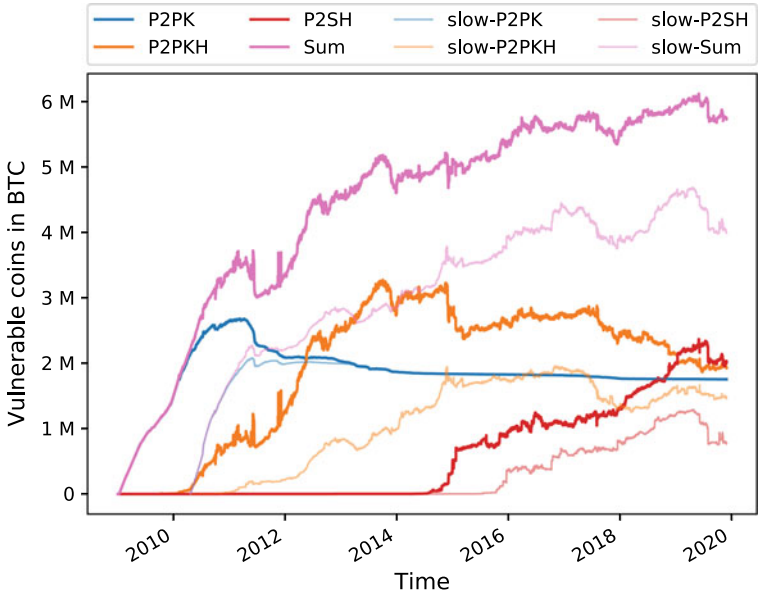


Fig. 1 Evolution of the crypyo-bounty for an instant (opaque) and slow (transparent) attacker, segregated by type of output

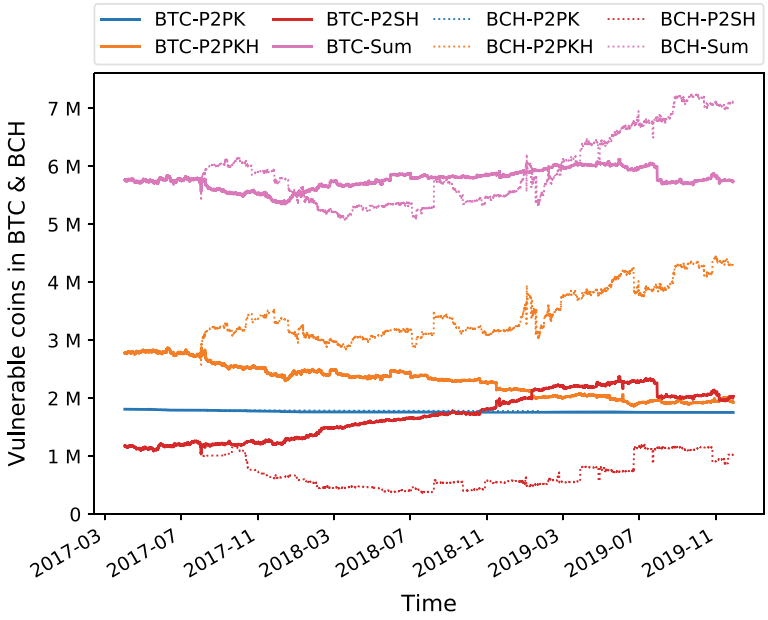


Fig. 2 Comparison between vulnerable coins in BTC (solid) and BCH (dotted)

BTCs (and 1.76M BCHs) locked in these outputs a guaranteed prize for the first quantum computer powerful enough to break a 256 bits ECDSA public key.

Also in Fig. 1, we draw attention to the parallelism, or lack thereof, between two plots for the same type of output. We can use this to describe the frequency and exposure of address reuse. For instance, the P2PKH plots for the slow and instant quantum attacker show almost no correlation from 2013 to 2015. While the reuse of these addresses was considerable as indicated by the instant attacker's plot, the exposure of the outputs was mostly less than a year, as the slow attacker's plot does not follow the same trend.

Comparing BTC with BCH we can ignore the common history up to block 478558 at which BCH forked from BTC. To this end, the plots in Fig. 2 start at block 460000, such that the fork point is clearly visible.

Analysing Fig. 2, we observe that although the total amount of coins vulnerable is relatively the same across BTC and BCH, there is a clear difference in preference for output types in the two communities and this is creating an upward trend for address reuse in BCH. While BTC users move funds away from P2PKH to P2SH and P2WSH, which are harder to identify, the BCH community is increasingly reusing P2PKH addresses, which require only one hash computation to be linked to the public key.

A similar conclusion can be drawn from the charts in Figs. 3 and 4, where we plot the number of outputs that are vulnerable. Observe how the graphs for the slow attacker differ across the two Figures for P2PKH and P2SH. While for BTC (Fig. 3) the plot is fairly distant from that of the instant attacker, in BCH (Fig. 4)

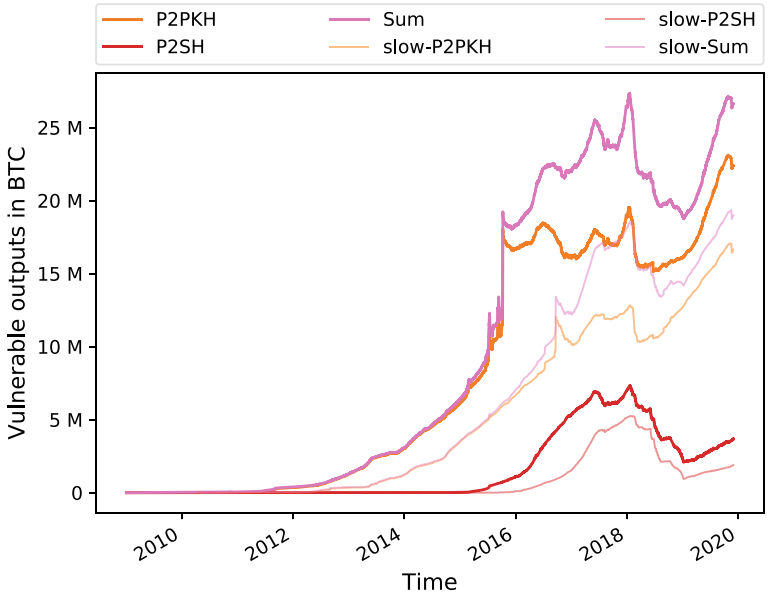


Fig. 3 Historic size of the vulnerable UTXO set from the perspectives of the instant (opaque) and slow (transparent) attackers in BTC

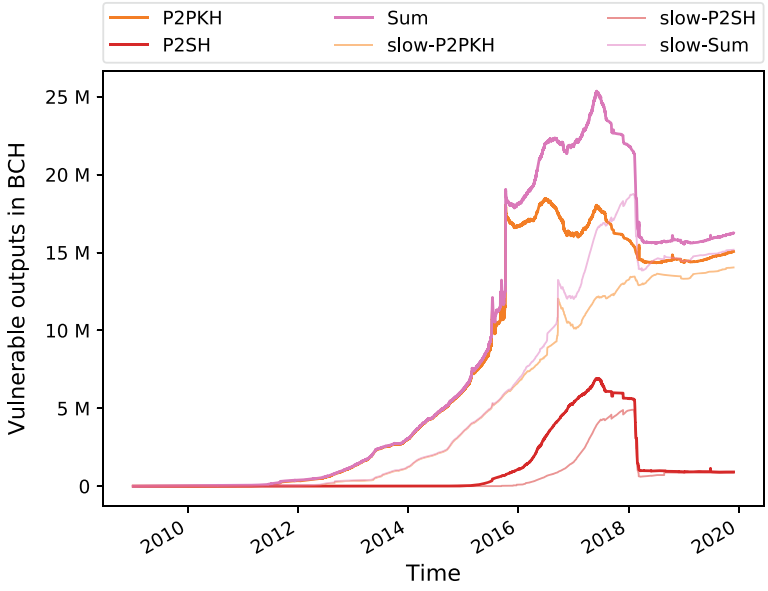


Fig. 4 Historic size of the vulnerable UTXO set from the perspectives of the instant (opaque) and slow (transparent) attackers in BCH

they approach, almost converging. This means that slow attackers in Bitcoin cannot make use of as many public keys as slow attackers in BCH. So even though there are more outputs that reuse addresses in BTC, the exposure of the public keys securing them is generally shorter. Furthermore, the value locked in these outputs is steadily decreasing as we can see from Fig. 2. At the same time, BCH users are reducing the number of outputs, but as indicated by the converging plots for the same type of output from Fig. 4, most of them are associated to public keys revealed long ago, hence the ascending trend of the crypto–bounty for BCH observed in Fig. 2.

6 Conclusion

In this paper, we have formulated the context in which a quantum enabled adversary would have to operate if it were to start attacking the Bitcoin network. We assumed the community would swiftly transition to a quantum resistant signature scheme through the use of a transitioning scheme that allows users to link transactions to their public keys without revealing them in the open. However, it is up to the individual user, to operate in a responsible manner that makes transitioning his funds even possible. Therefore, we emphasize the importance of locking funds in UTXOs associated to public keys which have not been revealed publicly, as otherwise, a quantum enabled adversary would be able to forge any action on behalf of the legitimate user, including transitioning the user's funds to addresses under his control.

We propose two models for the quantum adversary: one that can run Shor's algorithm virtually instantly and a slower one that might be more realistic for the first generations of quantum attackers. Consequently, we outline strategies available to each of them for maximizing profit through denying legitimate transactions to go through by overriding them with higher fee transactions spending the same UTXOs.

Finally, we demonstrate the methodology used to analyse the current and historic state of the vulnerable UTXO set as seen from the perspective of the two types of quantum enabled adversaries. We present the results gathered and analyse patterns of address reuse, noting that a larger percentage of the total supply is stored in considerably fewer outputs in Bitcoin Cash compared to Bitcoin. Furthermore, Bitcoin Cash users reuse addresses for longer periods of time, while in Bitcoin the exposure of public keys with positive balances is somewhat limited. We note that currently there are 1.75M BTCs and 1.76M BCHs which reside in a small number of outputs that have been mostly untouched since 2013, suggesting these may be zombie coins which cannot be recovered by classical means as the private keys have been lost. On the other hand, these coins could be considered a guaranteed reward for the entity that implements the first large scale quantum computer, whenever that would happen.

References

1. Adam, B. (2018). https://twitter.com/adam3us/status/947900422697742337. Accessed: 2018-02-18.
2. Aggarwal, D., Brennen, G. K., Lee, T., Santha, M., & Tomamichel, M. (2017). Quantum attacks on bitcoin, and how to protect against them. arXiv preprint arXiv: 1710.10377.
3. American National Standards Institute. (2005). Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). ANSI X9.62.
4. Antonopoulos, A. M. (2014). *Mastering Bitcoin: unlocking digital cryptocurrencies*. O'Reilly Media, Inc.
5. Biryukov, A., Khovratovich, D., & Pustogarov, I. (2014). Deanonymisation of clients in bitcoin p2p network. In *Proceedings 2014 ACM SIGSAC Conference on Computer and Communications Security* (pp. 15–29). ACM.
6. Bosch, B., & Barmes, I. (2020). Quantum computers and the Bitcoin Blockchain. https://www2.deloitte.com/nl/nl/pages/innovatie/artikelen/quantum-computers-and-the-bitcoin-blockchain.html. Accessed: 2020-01-18.
7. Certicom Research. (2018). Certicom ECC Challenge. https://www.certicom.com/content/dam/certicom/images/pdfs/challenge-2009.pdf. Accessed: 2018-02-17.
8. Crandall, R., & Pomerance, C. B. (2006). *Prime numbers: A computational perspective* (Vol. 182). Springer Science & Business Media.
9. Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H., & Capkun, S. (2016). On the security and performance of proof of work blockchains. In *Proceedings 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 3–16). ACM.
10. Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (pp. 212–219). ACM.
11. Kalodner, H. (2020). https://twitter.com/hkalodner/status/1064700672392773633. Accessed: 2020-01-18.
12. Kalodner, H., Goldfeder, S., Chator, A., Möser, M., & Narayanan, A. (2017). Blocksci: Design and applications of a blockchain analysis platform. arXiv preprint arXiv: 1709.02489.
13. Karame, G. O., Androulaki, E., & Capkun, S. (2012). Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. *IACR Cryptology ePrint Archive* (p. 248).
14. Karame, G. O., Androulaki, E., Roeschlin, M., Gervais, A., & Čapkun, S. (2015). Misbehavior in bitcoin: A study of double-spending and accountability. *ACM Transactions on Information and System Security (TISSEC)*, *18*(1), 2.
15. Lavor, C., Manssur, L., & Portugal, R. (2003). Shor's algorithm for factoring large integers. arXiv preprint arXiv: quant-ph/0303175.
16. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., & Savage, S. (2013). A fistful of bitcoins: Characterizing payments among men with no names. In *Proceedings 2013 Internet Measurement Conference* (pp. 127–140). ACM.
17. Menezes, A. (2001). Evaluation of security level of cryptography: The elliptic curve discrete logarithm problem (ECDLP). Technical Report. University of Waterloo.
18. Mosca, M., & Ekert, A. (1998). The hidden subgroup problem and eigenvalue estimation on a quantum computer. In *NASA International Conference on Quantum Computing and Quantum Communications* (pp. 174–188). Springer.
19. Proos, J., & Zalka, C. (2003). Shor's discrete logarithm quantum algorithm for elliptic curves. arXiv preprint arXiv: quant-ph/0301141.
20. Qu, M. (1999). Sec 2: Recommended elliptic curve domain parameters. Certicom Res., Mississauga, ON, Canada, Tech. Rep. SEC2-Ver-0.6.
21. Rogaway, P., & Shrimpton, T. (2004). Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *International Workshop on Fast Software Encryption* (pp. 371–388). Springer.
22. Rosenfeld, M. (2014). Analysis of hashrate-based double spending. arXiv preprint arXiv: 1402.2009.

23. Sattath, O. (2018). On the insecurity of quantum Bitcoin mining. arXiv preprint arXiv:1804.08118.
24. Schneider, N. (2013). Recovering bitcoin private keys using weak signatures from the blockchain. http://www.nilsschneider.net/2013/01/28/recovering-bitcoin-private-keys.html. Accessed: 2018-02-18.
25. Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, *41*(2), 303–332.
26. Sompolinsky, Y., & Zohar, A. (2016). Bitcoin's security model revisited. arXiv preprint arXiv:1605.09193.
27. Stewart, I., Ilie, D., Zamyatin, A., Werner, S., Torshizi, M., & Knottenbelt, W. J. (2018). Committing to quantum resistance: A slow defence for bitcoin against a fast quantum computing attack. *Royal Society Open Science*, *5*(6), 180410.
28. Tessler, L., & Byrnes, T. (2017). Bitcoin and quantum computing. arXiv preprint arXiv:1711.04235.
29. Tim, R. (2018). https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2018-February/015758.html. Accessed: 2018-02-18.
30. Yarom, Y., & Benger, N. (2014). Recovering OpenSSL ECDSA nonces using the FLUSH+RELOAD cache side-channel attack. *IACR Cryptology ePrint Archive*, *2014*, 140.

An Econophysical Analysis of the Blockchain Ecosystem

Philip Nadler, Rossella Arcucci, and Yike Guo

Abstract We propose a novel modelling approach for the cryptocurrency ecosystem. We model on-chain and off-chain interactions as econophysical systems and employ methods from physical sciences to conduct interpretation of latent parameters describing the cryptocurrency ecosystem as well as to generate predictions. We work with an extracted dataset from the Ethereum blockchain which we combine with off-chain data from exchanges. This allows us to study a large part of the transaction flows related to the cryptocurrency ecosystem. From this aggregate system view we deduct that movements on the blockchain and price and trading action on exchanges are interrelated. The relationship is one directional: On-chain token flows towards exchanges have little effect on prices and trading volume, but changes in price and volume affect the flow of tokens towards the exchange.

Keywords Token economics · Blockchain · Data assimilation · Inference

1 Introduction

Econophysical analysis has been receiving increased attention in recent years and is growing in popularity [11, 18]. The growth of data and ongoing digitization of society is allowing an ever more fine-grained view of economic activity in which transac-

This work is supported by Worley Parsons. We are also grateful for contributions and helpful comments from Barthelemy Duthoit, Kai Sun, Ovidiu Serban and cryptocompare.com.

P. Nadler (✉) · R. Arcucci · Y. Guo
Imperial College London - Data Science Institute, London SW7 2AZ, UK
e-mail: p.nadler@imperial.ac.uk
URL: http://www.imperial.ac.uk/data-science/

R. Arcucci
e-mail: r.arcucci@imperial.ac.uk

Y. Guo
e-mail: y.guo@imperial.ac.uk

P. Pardalos et al. (eds.), *Mathematical Research for Blockchain Economy*,
Springer Proceedings in Business and Economics,
https://doi.org/10.1007/978-3-030-53356-4_3

tions and flows of assets are observable on a very granular level. The emergence of blockchain is a unique phenomenon of this digital economy, in which, in a completely decentralized structure [3], the flow of assets between individuals is automated and publicly traceable. All transactions conducted on Bitcoin and Ethereum are saved and traceable on their respective blockchains [17, 22]. Furthermore, many traded tokens are based on ERC-20 contracts which can be also traced on the Ethereum blockchain, with the price and trading action on cryptocurrency exchanges being an important element of driving public exposure and user interaction with blockchain technology.

The novel and unique nature of this phenomenon, raises the question of how to analyze such a system. Established models in finance or standard data mining techniques from computer science allow only for the study of specific elements of the cryptocurrency ecosystem.

Some work such as [13, 19] have analyzed the graph structure of the Bitcoin blockchain in order to learn about entities and interactions. Similar research has also been conducted for the Ethereum blockchain as in [7]. Additional work was done by [14] to identify entities on the blockchain. These approaches offer a heuristic insight on blockchain interactions, but do not allow for more elaborate analysis of the full cryptocurrency ecosystem and the economic dynamics and incentives related to cryptocurrencies. Research based on empirical finance such as [10, 12] or [16] can yield additional insights by using economic theory, but often work with aggregate data that allow only for general analysis of cryptocurrency pricing dynamics, and overlook individual transactional data and entities.

We thus propose to analyze the cryptocurrency ecosystem in a novel fashion by interpreting it as an econophysical system. We argue that this is a very suitable approach, because the study of aggregate outcomes of individual transactions and particle flows from many sources bears resemble to physical systems [18]. We do so by combining methods of physical sciences and econometrics.

In particular, we employ a data assimilation approach for the analysis of large scale systems, which we combine with the functional form and interpretation of an econometric time varying parameter model with a stochastic volatility element [4]. Data Assimilation is a methodology to update a latent theoretical model with observations of real data, a form of state-space modelling. This is commonly applied in meteorology, hydrology and other physical sciences to study dynamic physical systems [9]. This approach can be applied to many fields where it is necessary to compute and update latent model parameters while ingesting new observations or where uncertainty of a model and forecast need to be quantified [21].

In order to get an as accurate as possible representation of the full cryptocurrency ecosystem, we collect and extract information from the most important sources of cryptocurrency activity. We therefore collect transactional data from the Bitcoin and Ethereum blockchains and supplement the dataset with additional trading data of exchanges.

In an application of our methodology we study the relationship between transaction flows of tokens on the blockchain and how this affects one of the major transactional hubs of cryptocurrencies: exchanges. Specifically, how do properties such

as trading volume and prices of cryptocurrency tokens relate to the on-chain inflow of tokens to the exchange? Some exchanges can be mapped to particular addresses or contracts on the blockchain [14]. Thus, although imperfect because it is difficult to reliably follow changing addresses of exchanges throughout time, this approach enables us to give approximate estimates to questions such as if supply and demand dynamics of exchanges hold, e.g.: Does a large inflow of a particular token at an exchange lead to an oversupply and thus decrease the price of the token? Although it is not possible to identify individual buyers and sellers on exchanges, economic theory postulates that prices are the aggregate outcome of individual transactional behaviour. Thus, linking the price on an exchange with traceable blockchain flows to the exchange is an important dynamic that deserves investigation and is represented in our econophysical modelling approach.

Besides putting forward the idea that the cryptocurrency ecosystem can be interpreted as an econophysical system, we interpret model parameters and find evidence that token flows do not affect price and trading behaviour, but that trading behaviour affects on-chain token flows. We furthermore provide an experiment to show how constantly updating the model dynamics by assimilating observations does not only allow for interpretation of latent parameters, but also provides important information for updating the model to improve model forecasting.

2 Econophysical Modelling

To represent and analyze the cryptocurrency ecosystem as econophysical system we use a data assimilation approach. Consider a possibly non-linear turbulent physical system [11, 21] where the state u is predicted by an approximate system model m which is dynamically updated at discrete timesteps $m(t, u)$ in the analysis step by ingesting new observations.

The state vector $u \in \mathbb{R}^N$ consists of multiple subspaces that evolve dynamically over time, where $u = (u_1, \ldots, u_i)$, with $u_j \in \mathbb{R}^{N_j}$ and $N_1 + \cdots + N_i = N$, modelling the latent parameter dynamics of the system. In order to map parameter space to observation space, there is an observation operator $H(u)$ mapping $\mathbb{R}^N$ to $\mathbb{R}^M$, where M observations d at analysis timestep $m(t, u)$ are given by:

$$d = H(u) + \epsilon_{u,0} \tag{1}$$

the operator $H(u)$ is of dimension M and the Gaussian noise term $\epsilon_{u,0}$ follows $\sim N(0, Q_0)$. We consider assimilation algorithms that filter dynamical systems of a generalizable form. Data assimilation is a technique to incorporate noisy observations into a predictive model [2] to generate predictions and also update and estimate parameter values which have a causal interpretation. Interpreting the system model m as an economic time varying parameter model $m(\Delta t, u, \theta)$ allows for an interpretation of the cryptocurrency system as dynamical system as well as the introduction of additional parameters where $\theta = (\phi, \tilde{\sigma})$ are derived from economic theory.

In this paper, a parametric model for the univariate case is given by:

$$\tilde{y}_t = \tilde{x}_t \phi_t + \tilde{\epsilon}_t \tilde{\sigma}_t \tag{2}$$

$$\phi_t = \phi_{t-1} + \tilde{\nu}_t \tag{3}$$

$$log(\tilde{\sigma}_t^2) = log(\tilde{\sigma}_{t-1}^2) + \tilde{\xi}_t \tag{4}$$

An observation d is represented by scalar value $\tilde{y}_t$ which is a function of multiple other variables given by $\tilde{x}_t \in \mathbb{R}^q$ which is a $1 \times q$ vector. The relationship between both variables is modelled by parameter matrix ϕ_t which is of corresponding dimension. The state vector is represented by ϕ_t describing how economic processes interact over time whereas parameter $\tilde{\sigma}$ describes dynamic volatility of the system. The distributional assumptions of the error terms are: $\tilde{\epsilon}_t \sim N(0, 1)$, $\tilde{\nu}_t \sim N(0, \tilde{Q}_t)$ and $\tilde{\xi}_t \sim N(0, \tilde{R}_t)$. Equation 2 assimilates observations into the model, whereas Eq. 3 and Eq. 4 describe the latent dynamics of the system under analysis. To exemplify, $\tilde{y}_t$ are observations of changes in the transaction volume of an exchange and $\tilde{x}_t$ is the amount of token flowing into the exchanges. When assimilating these observations, the parameters dynamics of Eqs. 3 and 4 are updated and describe the latent relationship between these variables. In order to scale the system up to multiple variables and to include additional lag-coefficients the model is rewritten in a generalized matrix notation:

$$y_t = \Phi_1 y_{t-1} + \cdots + \Phi_l y_{t-l} + \mu_t + \epsilon_t \sigma_t \tag{5}$$

where in this compact notation, $X_t = [y'_{t-1}, \ldots, y'_{t-l}, 1]'$ and $\Phi = [\Phi_1, \ldots, \Phi_l, \mu]'$ with $'$ denoting the transpose. Define $K = (ql + 1)$ as the product of q variables and lag length l including a constant. Thus X_t is of dimension $K \times 1$ and Φ of dimension $K \times q$. Furthermore, y_t is a $q \times 1$ vector of variables under analysis, μ_t is the mean vector and Φ_l is a $q \times q$ matrix of coefficients. σ_t and y_t are of the same dimension whereas ϵ_t is a diagonal matrix.

To include multiple lag-lengths and coefficient matrices, the coefficient matrix Φ is vectorized. This is necessary for policy analysis while preserving markovian properties of the resulting state-space model. Thus define $x_t = I \otimes X_t$ using Kronecker product $\otimes$, where I is the unit matrix and furthermore define $\beta_t = vec(\Phi)$, where $vec()$ indicates the stacking the columns of a matrix. The full model in state space form is then given by:

$$y_t = x'_t \beta_t + \epsilon_t \sigma_t \tag{6}$$

$$\beta_t = F \beta_{t-1} + v_t \tag{7}$$

$$log(\sigma_t^2) = log(\sigma_{t-1}^2) + \xi_t \tag{8}$$

with diagonal matrix F modelling autoregressive dynamics. In order to arrive at a solution for the model and to include new observations optimally in the latent model we obtain the model solution by the loss function including the latent parameters β_t and σ_t. The model solution is then given by the DA optimization:

$$\beta_t, \sigma_t = \underset{\beta,\sigma}{\operatorname{argmin}} \; J(\beta, \sigma) \tag{9}$$

with:

$$J(\beta, \sigma) = ||\beta - \beta_{t-1} + \nu_t||_{Q_t^{-1}} + ||log\left(\frac{\sigma_{t-1}^2}{\sigma^2}\right) + \xi_t||_{R_t^{-1}} \tag{10}$$

The resulting solution leads to a modified Kalman filtering algorithm to assimilate new observations into the model, updating all model parameters [15]. After discussing the assimilated data, we validate the models forecasting performance and discuss the economic interpretation of the parameter estimates in the following sections.

3 Data Collection

To obtain a representation of the cryptocurrency ecosystem we gather data from sources in which users interact most commonly with cryptocurrencies. The data consists of on-chain and off-chain cryptocurrency data. We define on-chain data as data that is available on the blockchain, whereas off-chain data is defined as trading data directly obtained from exchanges. We combine the flows of cryptocurrencies on the Bitcoin blockchain as well as Ethereum by including tokens based on ERC20 tokens. This is complemented with price action of exchanges which are mapped to blockchain addresses.

3.1 Mapping Exchanges to Bitcoin Addresses

Some entities choose to reveal their address voluntarily. If not, some work such as [7, 13, 14, 19] has developed simple heuristics of identifying entities on the Bitcoin blockchain which are defined as *idioms of use*. Using idioms such as "every non-change output is controlled by a single entity" [1] and "an address is used at most once as change" [14] can then be used to track and map entities on the Bitcoin blockchain, by clustering addresses controlled by the same entity. Using this approach it is possible to identify clusters belonging to online wallets, merchants and other service providers such as exchanges by interacting with them. A challenge for these de-anonymization approaches is their fragility, they loose accuracy over time as protocol implementations might change and may also yield false positives [20].

Using similar heuristics as [14], the developer of https://www.walletexplorer.com/ developed a platform for which individual flows to and the balance of clusters belonging to entities is updated overtime. Given the inflow and outflow of Bitcoins to these addresses clusters, the corresponding price and trading volume data of the specific exchanges is mapped to the clusters. Inference models always suffer to some

degree from noise and measurement error. Thus although imperfect, a combination of these datasets gives an approximation of the relationship between on-chain and off-chain activity.

3.2 Mapping Exchanges to ERC-20 Tokens

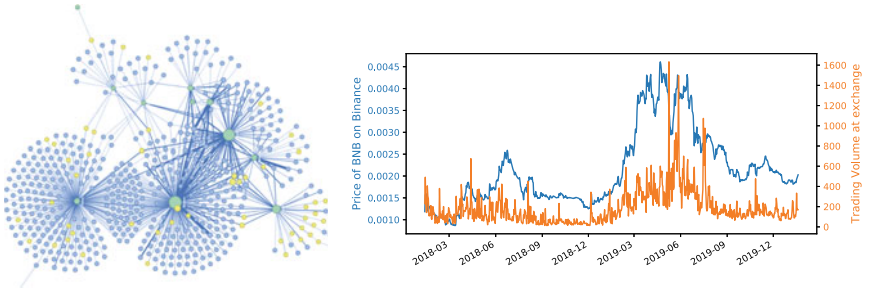
In order to map exchanges trading ERC-20 tokens on the Ethereum blockchain, a list of exchanges and their addresses was constructed by taking the list all self-revealed exchanges via https://etherscan.io/accounts/label/exchange/. Due to the Ethereum networks protocol properties, once an externally owned account is identified, following it over time is less of an issue than in the Bitcoin protocol. Exchanges might use accounts they do not publicly disclose, but the accounts that are disclosed can be regarded as ground truth. In a second step, multiple ERC-20 tokens being related to exchange addresses from the previous step were collected. Using the API of Bloxy (https://bloxy.info/api_methods), data on ERC-20 tokens was gathered. The *List Tokens* API-call allows for the collection of data on ERC-20 and ERC-721 tokens such as the total number of transactions invoked by the *transfer()* and *transferFrom()* methods. An excerpt is given in Table 1. In the next step, the *Token Holders* API-call allows for the retrieval of balance and address of token holders. We use a simple heuristic assuming that when an address belonging to an exchange was actively sending or receiving tokens, it must be a flow related to the exchange to which then the price is mapped. Figure 1a illustrates this, where a green node represents an exchange, and a link represents a token flow. Entities interacting with it are represented by blue and yellow nodes, which represent externally owned accounts and smart contracts respectively. For each exchange node the corresponding trading data is mapped as illustrated in Fig. 1b. We apply this procedure to the 60 most traded Ethereum based tokens, based on the number of transfers.

3.3 Exchanges

The off-chain exchange data of the exchanges was obtained via the APIs of https://min-api.cryptocompare.com/ and https://etherscan.io/tokens where trading volume,

Table 1 Example ERC token list obtained from API call

Symbol	Name	Smart contract address	Transfer count
LPT	Livepeer	0x58b6a8a3302369daec383334672404ee733ab239	5,522,6810
coToken	coToken	0x03cb0021808442ad5efb61197966aef72a1def96	4,824,866
CK	CryptoKitties	0x06012c8cf97bead5deae237070f9587f8e7a266d	4,146,609
EOS	EOS	0x86fa049857e0209aa7d9e616f7eb3b3b78ecfdb0	3,570,224
TRX	Tronix	0xf230b790e05390fc8295f4d3f60332c93bed42e2	2,890,298



(a) On-Chain flow of tokens to exchanges in 48h time period

(b) Price and trading volume of tokens on single exchange over time

Fig. 1 Figure **a** illustrates the on-chain flow of tokens. Green nodes are exchanges which receive and send tokens from accounts and smart contracts, represented by blue and yellow nodes respectively. Figure **b** shows the price and trading volume development for one exchange over time

as well as opening- and closing prices of each cryptocurrency token was obtained. For Bitcoin, the reference price is USD. For ERC20 tokens, the price of tokens is given in Satoshi levels because many tokens are not directly traded in USD. Translating a trading pair into a third currency such as EOS into Bitcoin and then USD obtained from different exchanges will induce noise and biases in the results. Bitcoin is furthermore the de-facto main exchange currency within the cryptocurrency ecosystem, we thus conduct all analysis of tokens in Satoshi levels. The transactional and exchange data used in the analysis is gathered hourly and daily from a timeperiod from 2018/01/01 to 2020/01/14. All series are stationary by analyzing changes in $\Delta ln(x_t)$ with outliers being removed and missing values interpolated.

In order to analyze the on-chain movements and link them to exchange movements of prices over time, we focus on two main features from blockchain movements and exchange action as given in Table 2:

The token i flow to individual exchanges j are mapped and synchronised to exchange trading data for all 60 token and their respective exchanges. Figure 2 shows the amount of tokens flowing and total transactions related to one exchange (a green node in the previous graph figure) which allows us to establish a temporal link and conduct inference to price data as previously shown in Fig. 1b.

4 Model Validation and Forecasting

We discuss how the representation of cryptocurrencies as dynamical system and the resulting assimilation of new observations is used to improve the forecasting performance of the model. The metric we choose to evaluate the model performance is mean squared forecasting errors (MSFE) as given by:

Table 2 Constructed features from token flows

Feature	Description
$\text{Amount}_{i,j}$	Amount of tokens being received and send to addresses associated to exchange
$\text{Transactions}_{i,j}$	Number of times transactions are received and send by addresses associated to exchange, independent of token amount
$\text{Volume}_{i,j}$	The volume from and to the token being traded
$\text{Price}_{i,j}$	The closing price of the token under analysis

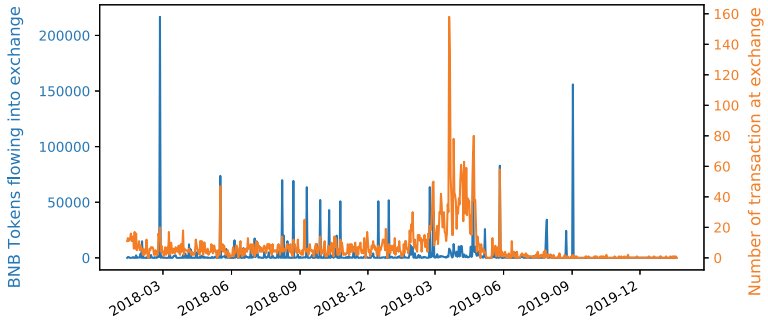


Fig. 2 Flow of BNB tokens to Binance over time. The blue series is the amount of tokens flowing into exchange, the orange series represents absolute amount of transactions conducted on one exchange

$$MSFE = \sum_{n=0}^{N} \left(\frac{\sum_{\tau=\tau_0}^{T-h} (y_{t,n}^{r} - \hat{y}_{t,n})^2}{T - h - \tau_0 + 1} \right) \tag{11}$$

where $\hat{y}_{t,n}$ represents the model prediction, $y_{t,n}^{r}$ the real observation with forecast horizons defined by $h = 1$, and $\tau_0 = 1$ the starting period of the forecast for n variables. Figure 3 depicts the assimilation process of the price series of Binance coin. After initially larger forecasting errors the errors decrease after assimilating the observations and updating the model parameters. The continuous updating of model parameters indicates a convergence towards parameter values that achieve an optimal model fit, with prediction errors decreasing over time. It is also observable how the the errors are a number of magnitude smaller after assimilation of the data, showing the models validity for forecasting purposes. For the full model the aggregate error before assimilation is $MSFE = 32.577$ and $MSFE = 0.204$ afterwards, reducing forecasting errors by several orders of magnitude.

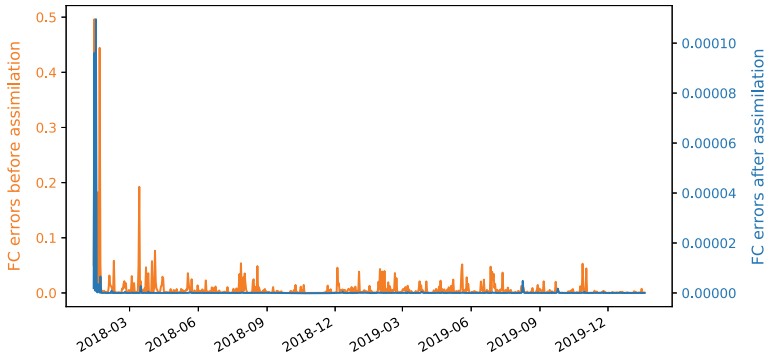


Fig. 3 Assimilating and Forecasting BNB coin. The orange series depicts forecasting errors before assimilation and the blue series the forecast after assimilation

5 Analysis

We proceed discussing and interpreting the results of the assimilation procedure. The model dynamics help to interpret and understand the structure of the cryptocurrency economy. After assimilating the observations, the model state equations Eqs. 7 and 8 are updated, giving an indication of the model dynamics β_t and σ_t. An excerpt of the full state vector β_t illustrating latent parameter dynamics is given in Fig. 4. In the following section we focus on individual entries of the state vector $\beta_{t,i}$ which have an econometric interpretation. Figure 5 shows different volatility regimes $\sigma_{t,i}$ as given in Eq. 8. While price volatility stays on a similarly low level the trading volume fluctuates largely and tappers off towards the end of the sample, which gives context for the other parameter estimates. To analyze the relationship between the economic series we next focus on the dynamics of β_t as in Eq. 7.

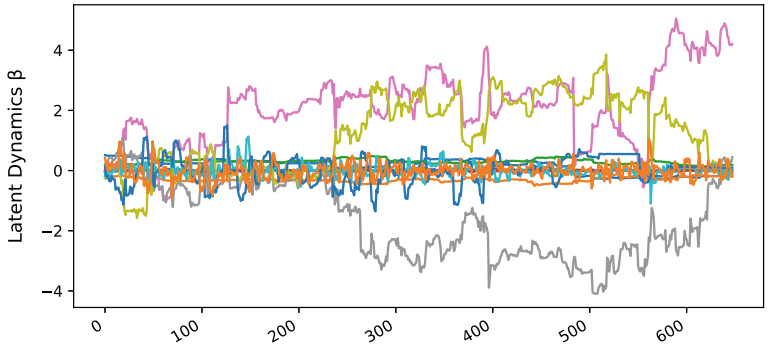


Fig. 4 An overview of multiple time varying latent dynamics as given in Eq. 7

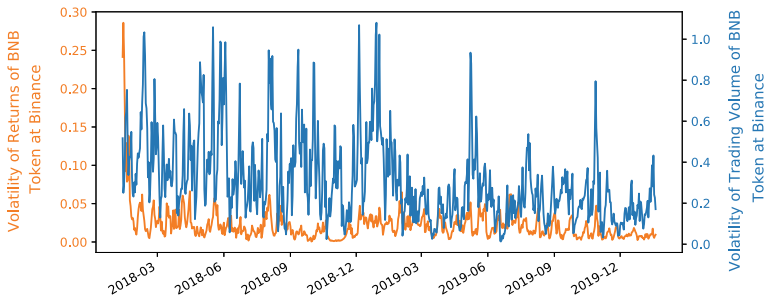


Fig. 5 Stochastic volatility values σ_t of price and volume changes on an exchange

5.1 *The Relationship Between On-Chain and Off-Chain Movements*

We present and discuss results for multiple subsets of the cryptocurrency ecosystem. We first focus and apply our model on the Binance exchange for it's Binance coin (BNB), as well as for other selected currencies and exchanges. We choose to focus on Binance coin as illustrative example because its usage has actual economic benefits for traders, by providing them a discount on trading fees. Thus, if there is a link between trading activity and on-chain token flows, we hypothesize that the link should be very pronounced for BNB. We later conduct robustness checks by presenting results for other exchanges and currencies such as Bancor (BAT).

An overview was given in the previous section by Fig. 1a, b, which describe the on-chain dynamics of prices and trading volumes over time, and the number of transactions as well as amount of tokens flowing towards the exchange respectively.

5.2 *Do On-Chain Movements Affect Prices?*

We first investigate the effect of exchange trading volume and token inflow on prices. Figure 6 shows that as expected, there is a strong relationship between prices and trading volume. Volume changes affect the returns of tokens negatively, establishing a strong link between both variables, a common and established financial link [6], thus variables related to cryptocurrency exchanges exhibit similar patters as regular financial exchanges. In the next step, when studying the relationship between the flow of tokens towards an exchange and prices in Fig. 7, it becomes apparent that on-chain token-inflows are noisy and erratic with an average effect on price changes close to zero.

To extend evidence we furthermore analyze the effect of tokenflows on trading volume. A further studying of Fig. 8 also shows that an inflow of tokens has little effect on the trading volume on an exchange with parameter values being erratic

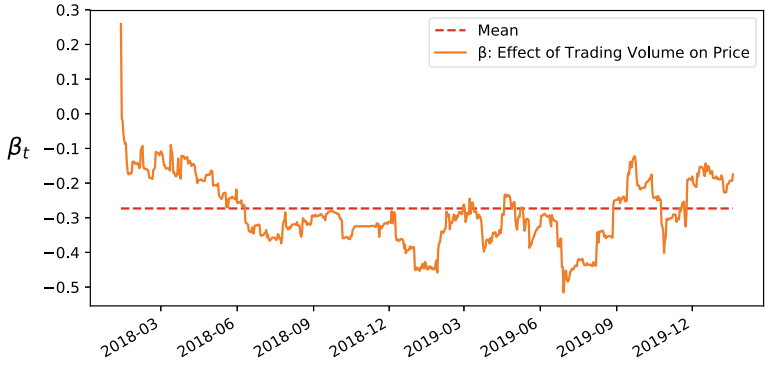


Fig. 6 Time varying latent effect of trading volume on prices for the BNB-token on Binance

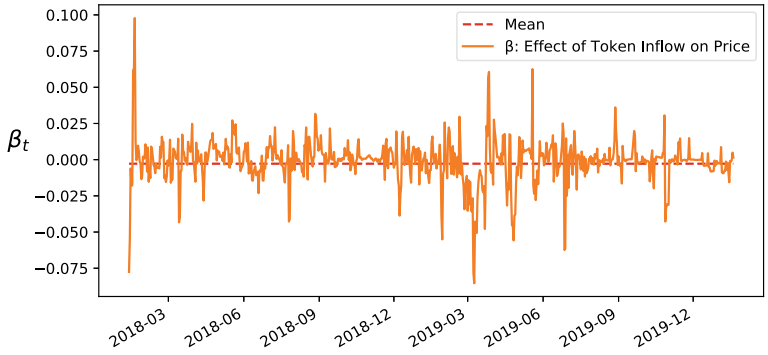


Fig. 7 Time varying latent effect of tokenflow on prices for BNB-token

and fluctuating around zero. This provides evidence that trading variables such as price and volume are interrelated as already established, and that the on-chain flow of tokens has little impact on trading action. Thus simple theories such as that a large inflow of tokens could decrease prices on an exchange are not supported by the model dynamics.

5.3 Do Price and Trading Volume Affect the Flow of Tokens?

When reversing the analysis, a different pattern arises. Observing the parameter in Fig. 9, representing how price and trading volume affect tokenflows, there is evidence that trading volume and prices lead to changes in tokenflows related to the exchange. The effect is initially positive and then negative, with troughs at 2018/10 and 2019/10. The series is more pronounced than the erratic series of tokenflows on price effects (Fig. 8) in the previous section. The signal is also very pronounced

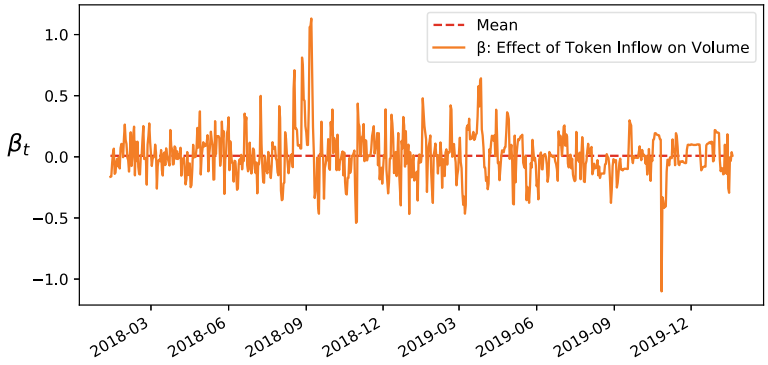


Fig. 8 Time varying latent effect of tokenflows on volume for BNB-token

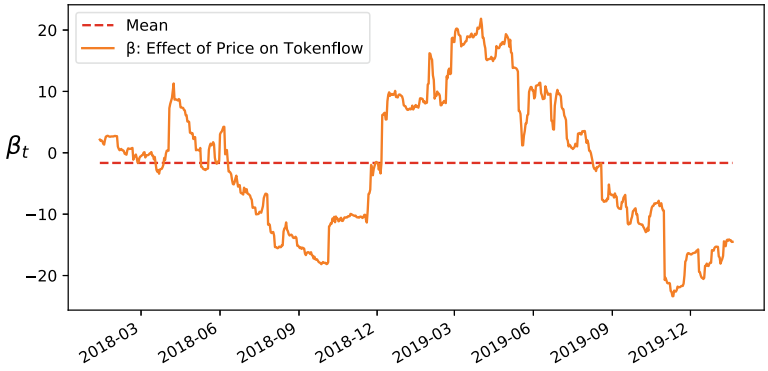


Fig. 9 Time varying latent effect of price changes on tokenflow for the BNB-token on Binance

in Fig. 10, showing that increases in trading volume on an exchange affect the flow of tokens of the exchange. The parameter series is much larger in magnitude and resembles a highly persistent series, whereas parameters representing the effect of tokenflow on exchange variables appear more like noise processes. Comparing Figs. 7 and 8 directly with the other series makes this more apparent, and similar dynamics are found in different model setups. The autocorrelation example in appendix Fig. 13 illustrates the difference.

Well established econometric theory states that highly persistent series are associated with processes in which innovations have long lasting structural effect, whereas innovations in noise processes are negligible and will revert to the mean eventually [5, 8]. Thus, independent of the exact parameter value, highly persistent $\beta_{t,i}$ series imply a structural relationship of exchange trading dynamics leading to on-chain tokenflows, whereas for the reverse this cannot be deduced due to the erratic parameter series.

Given these results, it is deducible that there is a connection between tokenflows on the blockchain as well as trading action on an exchange. Simple supply

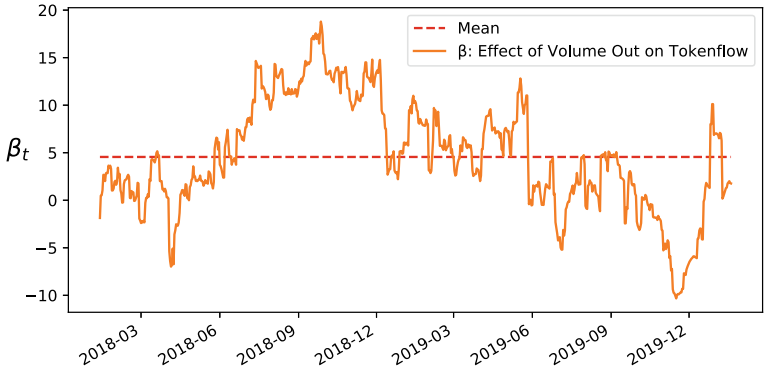


Fig. 10 Time varying latent effect of trading volume on tokenflow for the BNB-token on Binance

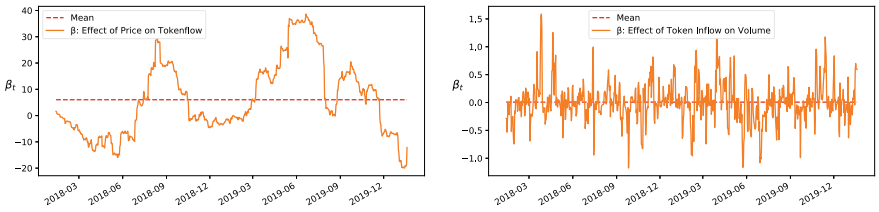
Table 3 Mean Value of Latent Parameters $\beta_{t,i}$

Effect t on $t+1$	$Price_{t+1}$	$Volume_{t+1}$	$Tokenflows_{t+1}$
$Price_t$	–	1.771	1.651
$Volume_t$	0.273	–	4.061
$Tokenflows_t$	0.002	0.008	–

and demand rules do not hold, since the inflow of tokens does not affect prices or volume. Nevertheless, when volume or prices move, they do affect the flow of tokens towards that exchange. A possible interpretation is that strong price and trading movements raise attention of users which increase flows towards that exchange. It can furthermore attract arbitrageurs which send deposits to an exchange to profit from price differences. Table 3 reports the absolute mean of the parameters β_t over time, as discussed above as well as additional parameters not depicted in the previous section. Changes in the column at time t imply future changes at time $t+1$ in the rows. Over the full sample, price changes affect volumes and tokenflows. Volume also affects the other variables, with particularly high values on token flows. In reverse, the average effect of token flows on prices and volume is weak, reinforcing the argument that off-chain and on-chain interaction is one-directional.

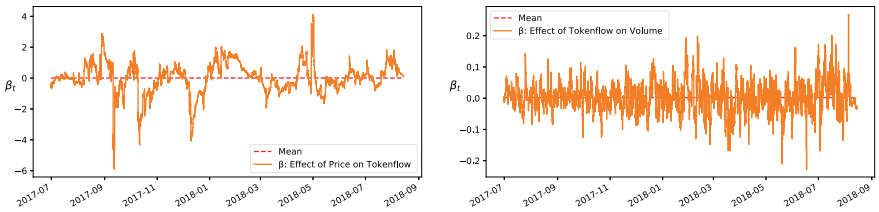
5.4 Robustness Checks

To confirm our results we conduct robustness checks for other currencies. The illustration in Fig. 11a represent the findings of the model for Bancor (BAT). The figure depicts a similar pattern as in the previous analysis with a persistent series depicting the influence of price changes on tokenflow. Figure 11b reveals the same pattern of tokenflows having little effect on exchange trading activity. Results are similar across



(a) β_i price on tokenflows (b) β_i tokenflows on volume

Fig. 11 The effects of price on tokenflows, and tokenflows on volume for BAT coin



(a) β_i price on tokenflows (b) β_i tokenflows on volume

Fig. 12 The effects of price on tokenflows, and tokenflows on volume for Bitcoin flowing to Poloniex

other test setups, with the direction of influence being not from tokenflow to price changes, but from changes in price and trading volume to the flow of tokens.

We show an additional setup modelling the flow of Bitcoins towards the Poloniex exchange and find similar results. The effect of price changes to Bitcoins flowing towards the exchange are given in Fig. 12a. It is more noisy and on average closer to zero, but the dynamics of the process are more persistent and are much larger in magnitude than in the reverse case where the effect of tokenflows on volume resembles much more erratic noise around a zero mean, which is in line with the previous findings.

6 Conclusion

We put forward the idea to treat the emerging cryptocurrency ecosystem as an econophysical system. We analyze the system by deriving and solving a custom loss function for the model. The model is ingesting data obtained from a variety of on-chain and off-chain sources, giving an approximation of the full cryptocurrency ecosystem. We also show that the dynamic system representation is applicable for generating forecasts of potential changes in price as well as flows on the blockchain. This representation enables insights into otherwise unobservable parametric pricing and market dynamics. The model results indicates that there is a weak relationship

between on-chain flows and off-chain action. The evidence shows that the direction of interaction is one-sided and that simple supply and demand dynamics are not observable because the exchange trading action is not affected by token inflows towards a specific exchange. Nevertheless, there is evidence for a reverse effect. The model provides evidence that price action and increases in trading volume drive the inflow of tokens towards a particular exchange. The generalizability of the model allows for further analysis such as investigating the difference between flows stemming from externally owned accounts and smart contracts. Other extensions of our work can include the setup of a real-time assimilation scheme, in which the data is constantly ingested by the model, providing a live activity overview of the cryptocurrency ecosystem. Ongoing work includes the inclusion of high frequency orderbook data which allows to analyze transaction flows on minute and second frequencies. Further work on econophysics modelling and the study of the aggregate outcome of individual transactions will be of use to understand the underlying dynamics of social systems and market activity.

Appendix

See Fig. 13.

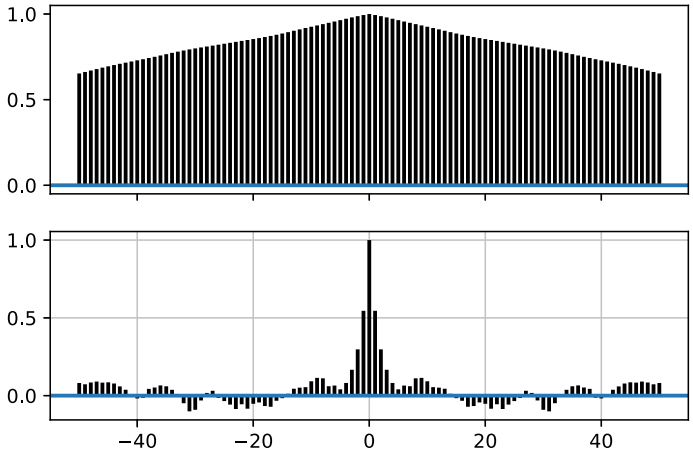


Fig. 13 The autocorrelation functions of the effect of prices on tokenflows (top) and tokenflow on volume (bottom) for BNB coin as given in Figs. 10 and 7 illustrate the difference in persistence

References

1. Androulaki, E., Karame, G. O., Roeschlin, M., Scherer, T., & Capkun, S. (2013). Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security* (pp. 34–51). Springer.
2. Asch, M., Bocquet, M., & Nodet, M. (2016). *Data assimilation: methods, algorithms, and applications* (Vol. 11). SIAM.
3. Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., & Danezis, G. (2019). Sok: Consensus in the age of blockchains. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies* (pp. 183–198).
4. Barndorff-Nielsen, O. E., & Shephard, N. (2002). Econometric analysis of realized volatility and its use in estimating stochastic volatility models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *64*(2), 253–280.
5. Campbell, J. Y., & Mankiw, N. G. (1987). Are output fluctuations transitory? *The Quarterly Journal of Economics*, *102*(4), 857–880.
6. Chen, G.-M., Firth, M., & Rui, O. M. (2001). The dynamic relation between stock returns, trading volume, and volatility. *Financial Review*, *36*(3), 153–174.
7. Chen, T., Zhu, Y., Li, Z., Chen, J., Li, X., Luo, X., Lin, X., & Zhange, X. (2018). Understanding ethereum via graph analysis. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications* (pp. 1484–1492). IEEE.
8. Cochrane, J. H. (1988). How big is the random walk in gnp? *Journal of political economy*, *96*(5), 893–920.
9. Cuomo, S., Galletti, A., Giunta, G., & Marcellino, L. (2017). Numerical effects of the gaussian recursive filters in solving linear systems in the 3dvar case study. *Numerical Mathematics: Theory, Methods and Applications*, *10*(3), 520–540.
10. Fry, J., & Cheah, E.-T. (2016). Negative bubbles and shocks in cryptocurrency markets. *International Review of Financial Analysis*, *47*, 343–352.
11. Guegan, D. (2009). Chaos in economics and finance. *Annual Reviews in Control*, *33*(1), 89–93.
12. Jang, H., & Lee, J. (2017). An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information. *Ieee Access*, *6*, 5427–5437.
13. McGinn, D., Birch, D., Akroyd, D., Molina-Solana, M., Guo, Y., & Knottenbelt, W. J. (2016). Visualizing dynamic bitcoin transaction patterns. *Big Data*, *4*(2), 109–119.
14. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., & Savage, S. (2013). A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference* (pp. 127–140). ACM.
15. Nadler, P., Arcucci, R., & Guo, Y. (2019). A scalable approach to econometric inference. volume 36 of *Advances in Parallel Computing* (pp. 59–68). IOS Press.
16. Nadler, P., & Guo, Y. (2020). The fair value of a token: How do markets price cryptocurrencies? *Research in International Business and Finance*, *52*, 101108.
17. Nakamoto, S., et al. (2008). Bitcoin: A peer-to-peer electronic cash system.
18. Pereira, E. J. d A. L., da Silva, M. F., & Pereira, H. d B. (2017). Econophysics: Past and present. *Physica A: Statistical Mechanics and its Applications*, *473*, 251–261.
19. Ranshous, S., Joslyn, C. A., Kreyling, S., Nowak, K., Samatova, N. F., West, C. L., & Winters, S. (2017). Exchange pattern mining in the bitcoin transaction directed hypergraph. In *International Conference on Financial Cryptography and Data Security* (pp. 248–263). Springer.
20. Seres, I. A., Nagy, D. A., Buckland, C., & Burcsi, P. (2019). Mixeth: efficient, trustless coin mixing service for ethereum. In *International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
21. Smith, P. J., Dance, S. L., Baines, M. J., Nichols, N. K., & Scott, T. R. (2009). Variational data assimilation for parameter estimation: application to a simple morphodynamic model. *Ocean Dynamics*, *59*(5), 697.
22. Wood, G., et al. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, *151*(2014), 1–32.

Stress Testing Diversified Portfolios: The Case of the CoinShares Gold and Cryptoassets Index

Aikaterini Koutsouri, Michael Petch, and William J. Knottenbelt

Abstract Stress testing involves the use of simulation to assess the resilience of investment portfolios to changes in market regimes and extreme events. The quality of stress testing is a function of the realism of the market models employed, as well as the strategy used to determine the set of simulated scenarios. In this paper, we consider both of these parameters in the context of diversified portfolios, with a focus on the emerging class of cryptoasset-containing portfolios. Our analysis begins with univariate modelling of individual risk factors using ARMA and GJR–GARCH processes. Extreme Value Theory is applied to the tails of the standardised residuals distributions in order to account for extreme outcomes accurately. Next, we consider a family of copulas to represent the dependence structure of the individual risk factors. Finally, we combine the former approaches to generate a number of plausibility-constrained scenarios of interest, and simulate them to obtain a risk profile. We apply our methodology to the CoinShares Gold and Cryptoassets Index, a monthly-rebalanced index which comprises two baskets of risk-weighted assets: one containing gold and one containing cryptoassets. We demonstrate a superior risk-return profile as compared to investments in a traditional market-cap-weighted cryptoasset index.

Keywords Cryptoassets · Gold · Index · Risk modelling · Stress testing

A. Koutsouri (✉) · W. J. Knottenbelt
Department of Computing, Imperial College London, London SW7 2AZ, UK
e-mail: k.koutsouri@imperial.ac.uk

W. J. Knottenbelt
e-mail: wjk@doc.ic.ac.uk

M. Petch
CoinShares, Octagon Point, 5 Cheapside, St Paul's, London, England EC2V 6AA, UK
e-mail: mpetch@coinshares.co.uk

P. Pardalos et al. (eds.), *Mathematical Research for Blockchain Economy*,
Springer Proceedings in Business and Economics,
https://doi.org/10.1007/978-3-030-53356-4_4

1 Introduction

Stress testing concepts date back to the early 1990s. During this period, stress testing was mainly used by individual banks for their own risk management plans surrounding their trading activities. The idea of using stress testing was only standardised in 1996 by the Basel Committee on Banking Supervision (BCBS) when an amendment was made to the first Basel Accord (Basel I) [1] that recognised stress testing as an effective way to measure risk. In the second Basel Accord (Basel II) [2], the BCBS asked banks to conduct internal stress testing. However, by the time the financial crisis began in 2007, Basel II was not yet fully implemented internationally.

The financial crisis is an example of a useful stress situation, when banks had to restrict their lending and the limitations of standard value-at-risk methodologies became apparent. At the time, most banks were not properly prepared to accommodate for this situation, which can be linked to the lack of scenario-based risk management planning. Post financial crisis, stress testing has increased widely in implementation across jurisdictions and is used by banks, international organisations, national authorities and academics. Stress tests performed by banks are assessing whether there is enough capital on hand to withstand a potential financial disaster or economic crisis and the results are required to be made publicly available.

In the cryptocurrency industry, while there have been numerous studies of price dynamics [6, 8, 11], there has been a lack of research in risk management for cryptocurrency-related investment products (e.g. cryptocurrency indices, funds, ETPs, etc.). There is, however, a growing awareness of the importance of this issue in the industry. Additionally, the US Federal Reserve is considering adding to their stress testing framework the scenario of a bitcoin market crash [3]. Nonetheless, there is still plenty of work to be done on the already-existing investment products in the cryptocurrency market by the product owners and potentially market supervisors. On that note, our primary analysis target in this paper is the CoinShares Gold and Cryptoassets Index (CGCI). In the case of the CGCI, we aim to isolate its main design principles and propose a framework for scenario-based risk management that is able to unveil vulnerabilities in certain market conditions.

The remainder of this paper is organised as follows. Section 2 presents the essential background on the principles behind the design of the CGCI and stress testing techniques in practice. In Sect. 3, we elaborate on the notion of ARMA–GARCH type filtering and how this can be combined with Extreme Value Theory concepts to produce empirical distribution functions of residuals. We further introduce the concept of copulas as an effective mechanism to model dependency structures. Section 3.4 discusses the importance of introducing plausibility constraints in formulated scenarios. Section 4 demonstrates a number of selected stress scenarios for the CGCI that reflects an improved risk profile compared to a traditional market-cap-weighted cryptoasset index. Section 5 concludes.

2 Background

2.1 Index Methodology

The CoinShares Gold and Cryptoassets Index (CGCI) [12] is a monthly rebalanced index that employs an inverse volatility weighting scheme for two main components: an equally-weighted basket of five cryptoassets, and physical gold. The main purpose of the CGCI is to provide exposure to the alternative asset space while utilising some means of principled risk control leading to lower volatility. The crypto-basket component of CGCI is composed of the top five eligible cryptoassets based on the 6-month rolling mean of the free-float market capitalisation and their composition is reviewed on a monthly basis when the rebalancing occurs. A more detailed presentation of the index is presented in the full methodology document.[1]

The design of the CGCI is based on two basic premises; the high volatility levels of the cryptoasset space, which brings with it a high level of risk, together with their high intraclass correlation, which limits the diversification benefits. The second is the lack of correlation with physical gold [12]. Based on those characteristics, and Shannon's Demon method of diversification and rebalancing [19], the CGCI utilises the concept of volatility harvesting through (a) forming a basket of cryptoassets and (b) combining it with gold using weighted-risk contribution as a rebalancing mechanism.

Given weighting $x = (x_1, x_2)$ for the crypto-basket and gold respectively, implied correlation $\rho = 0$, and supposing that we desire the risk contribution of the crypto-basket to be α times the risk contribution of gold, x_1 is given by:

$$x_1 = \frac{\sqrt{\alpha}\,\sigma_1^{-1}}{\sqrt{\alpha}\,\sigma_1^{-1} + \sigma_2^{-1}} \tag{1}$$

Taking into consideration the former, the index is calculated following a two-stage allocation scheme that involves:

1. Computation of the historical volatility of (a) the equally-weighted crypto-basket, and (b) gold;
2. Asset allocation among the crypto-basket and gold expressed as the bivariate weighted risk contribution problem presented in Eq. 1. The risk contribution ratio is set as $\alpha = 4$, indicating that 80% of the total risk emanates from the crypto-basket.

The CGCI proposes a mechanism for effective risk control that is sufficient because (a) the cryptoassets class and gold are characterised by respectively high and low levels of volatility respectively, and (b) exhibit insignificant correlation. As such, it is important to consider the effect of changes in both factors.

[1] To be made available online in due course by CoinShares (UK) Limited.

2.2 Stress Testing

There are several definitions of stress tests in literature. Studer [21] describes stress tests as the concept of maximum loss over ellipsoidal scenario sets. The stress testing problem is also described by Breuer et al. [4] who propose a number of refinements to Studer's initial approach. According to [4], the quality of the stress test crucially depends on the definition of stress scenarios, which need to meet three requirements: plausibility, severity and suggestiveness of risk-reducing actions. Lopez [14] further refers to stress tests as risk management tools that aim to quantify the impact of unlikely, yet plausible, movements of financial variables on portfolio values.

Stress test design methods vary in practice and are typically divided into two main categories: univariate and multivariate. While univariate stress tests (sensitivity analysis) are easy to perform, they are considered insufficient, as they fail to incorporate the dependence structure of the identified risk factors. Multivariate approaches examine the effects of simultaneous changes in more than one variables and are usually scenario-based. Breuer et al. [4] highlight the importance of an explicit choice of scenarios while Nyström et al. [18] identify the main elements of scenario-based risk management:

1. Recording the market value of the portfolio components
2. Generating scenarios based on a calibrated model of the portfolio behaviour
3. Estimation of the returns distribution
4. Application of risk measures in the obtained distribution

Regarding the model calibrations in step 2, Cherubini et al. [5] consider the bivariate equity portfolio case and suggest a univariate model for each of the two marginal distributions of the portfolio's risk factors (stocks), and one conditional copula approach for the underlying dependence structure.

In the case of CGCI, we identify two main risk factors (the crypto-basket and gold component respectively) and we attempt to model their evolution using stochastic processes. The credibility of results when we apply risk measures to the portfolio distribution, is heavily dependent on the choice of model, and as highlighted by Nyström et al. [18], one should consider a series of stylised facts when simulating the evolution of risk factors. McNeil et al. [15] discuss those stylised facts that characterise the returns of financial time series which can also be observed in the CGCI components.

Stylised facts on autocorrelations and volatility clustering, as proposed by Cont [7], suggest that (i) linear autocorrelations of returns are expected to be very small, (ii) autocorrelation function of absolute returns decays slowly as a function of the time lag and (iii) volatility events are stochastic and appear in clusters. Additionally, we expect (iv) asymmetric and heavy tails in the unconditional distribution of returns that exhibit power-law or Pareto-like behaviors. When examining conditional returns that have been corrected for heteroskedasticity, tails are less heavy.

3 Multivariate Stress Testing

3.1 Univariate Risk Factor Modelling

Following the definitions of McNeil et al. [15], the value of a given portfolio at time t can be denoted as V_t and modelled as a function of time and a random vector $\boldsymbol{X}_t = (X_{t,1}, \ldots, X_{t,d})'$, observable at time t and therefore expressed in the form of $V_t = f(t, \boldsymbol{X}_t)$, which is typically referred to as mapping of risks. The change in the value of the portfolio will be $\Delta V_{t+1} = V_{t+1} - V_t$, the loss is defined as $L_{t+1} := -\Delta V_{t+1}$ and its distribution is referred to as loss distribution. In this study, we will ultimately be concerned with the distribution of ΔV_{t+1}, termed as the Profit and Loss (P&L) distribution.

The evolution of each risk factor can be expressed as an autoregressive moving average (ARMA) process, which accounts for autocorrelation and aims to model the conditional mean. Additionally, conditional homoskedasticity is rejected in financial time series, and volatility is stochastic and forecastable. Generalised autoregressive conditional heteroskedasticity (GARCH) processes can adequately account for this fact. Empirical evidence also suggests that positive innovations to volatility correlate with negative market information (and vice versa). Glosten et al. [10] account for this asymmetry through modelling the positive and negative shocks on the conditional variance asymmetrically (GJR–GARCH).

For a combination of the ARMA and GJR–GARCH approaches applied to a time series X_t, we let $\epsilon_t = X_t - \mu_t = \sigma_t Z_t$ denote the residuals with respect to the mean process (ARMA error) of order p_1, q_1. We assume that σ_t follows a GARCH(p_2, q_2) specification, where p_2 is the order of the squared innovation lag (ϵ_t^2) and q_2 is the order of the variance lag (σ_t^2) and finally obtain the asymmetric ARMA(p_1, q_1) − GARCH(p_2, q_2) equations:

$$\begin{aligned} X_t &= \mu + \sum_{i=1}^{p_1} \phi_i (X_{t-i} - \mu) + \sum_{j=1}^{q_1} \theta_j \epsilon_{t-j} + \epsilon_t \\ \sigma_t^2 &= \omega + \sum_{j=1}^{m} \zeta_j v_{jt} + \sum_{i=1}^{p_2} \left(\alpha_i \epsilon_{t-i}^2 + \gamma_i I_{t-i} \epsilon_{t-i}^2\right) + \sum_{j=1}^{q_2} \beta_j \sigma_{t-j}^2 \end{aligned} \tag{2}$$

where ω is a volatility offset term, γ_j represents the leverage term and m denotes the number of possible external regressors v_j. The indicator function I_t takes on value of 1 for $\epsilon_t \leq 0$ and 0 otherwise.

The main assumption of GARCH models states that standardised residuals are independent and identically distributed (i.i.d.), a key fact to enable us to examine extreme events at the tails of the distributions. In this paper we opt for (a) a Ljung-Box test on standardised residuals to check for evidence of serial autocorrelation, and (b) a Li-Mak test [13] on the standardised residuals to check for remaining ARCH effects.

3.2 Tail Behavior Estimates

Stress scenarios describe how the portfolio would perform under extreme market moves, a fact that yields the modelling process of the risk factors' tails crucial. Extreme Value Theory (EVT) approaches are specifically concerned with the asymptotic behavior of the left and right tails separately. The key assumption of EVT is that it considers i.i.d. sequences. The propagation of asymmetric, heavy-tailed characteristics in the GARCH standardised residuals [22] and their strict white noise behavior, allows for EVT application to the tails. Combined with a non-parametric method for the centre, we can explicitly provide the filtered residual distribution for further simulations.

There are two main approaches that isolate extreme values, block maxima and threshold exceedance models. This is due to the fact that the former study the time series of maxima of consecutive time-series blocks and therefore might disregard large (and often important) portions of the original dataset. In contrast, threshold exceedance methods study all events that exceed a specified high threshold. This study aims to fit a Generalised Pareto Distribution (GPD, the main distributional model for exceedances over threshold) to excess standardised GARCH innovations.

Given a sequence of i.i.d. random values, the Generalised Pareto cumulative distribution function is given by:

$$G_{\xi,\beta}(x) = \begin{cases} 1 - (1 + \frac{\xi x}{\beta})^{-1/\xi}, & \xi \neq 0 \\ 1 - e^{-x/\beta}, & \xi = 0 \end{cases} \tag{3}$$

where ξ and $\beta > 0$ denote the shape and scale parameters, $x \geq 0$ when $\xi \geq 0$ and $0 \leq x \leq -\beta/\xi$ when $\xi < 0$. Given that $\xi < 1$, the GPD mean is $E(X) = \beta/(1-\xi)$. Also, given a random value X with a GPD cumulative distribution function $F = G_{\xi,\beta}$, the cumulative distribution function of the excess distribution over threshold u is given by:

$$F_u(x) = G_{\xi,\beta(u)}(x) = P(X - u \leq x \mid X > u) = \frac{G_{\xi,\beta(u)}(x+u) - G_{\xi,\beta(u)}(u)}{1 - G_{\xi,\beta(u)}(u)},$$

where $\beta(u) = \beta + \xi u$, $0 \leq x < \infty$ if $\xi \geq 0$ and $0 \leq x \leq -(\beta/\xi) - u$ if $\xi < 0$. The excess distribution remains a GPD with the same shape parameter ξ but with a scaling parameter that grows linearly with the threshold parameter u. The mean excess function of the GPD is given by:

$$e(u) = E(X - u \mid X > u) = \frac{\beta(u)}{1-\xi} = \frac{\beta + \xi u}{1-\xi}, \tag{4}$$

where $0 \leq u < \inf$ if $0 \leq u < 1$ and $0 \leq u \leq -(\beta/\xi)$ if $\xi < 0$. It can be observed that the mean excess function is linear in the threshold u, which is a characterising property of the GPD and will assist with the choice of u in later sections.

3.3 Dependence Concepts

In a multivariate approach to risk management one should consider the risk factors' dependence structure. In the case of CGCI, the lack of correlation between the crypto-basket and the gold component is a basic premise that allows for principled risk control through periodic rebalancing. To this end, it is critical to incorporate the dependence structure to the marginal behaviours and investigate the risk sensitivity to different dependence specifications.

In order to quantify the dependence structure, we utilise the concept of copulas, a tool that allows us to decompose a joint probability distribution into its marginals. A d-dimensional copula is a distribution function with standard uniform marginal distributions, that represents a mapping of the unit hypercube into the unit interval in the form of $C : [0, 1]^d \to [0, 1]$. The practicality of copulas is highlighted through Sklar's Theorem that states that any multivariate joint distribution can obtained through (a) the univariate marginal distribution functions and (b) a copula which describes the dependence structure between the variables. Copulas lend themselves particularly useful for this study, where the marginal distributions been defined in detail through the ARMA–GARCH–EVT approach.

For this study we consider two implicit copulas; the Gaussian and t-copula. Given a multivariate normal random vector $\boldsymbol{Y} \sim N_d(\boldsymbol{\mu}, \Sigma)$, its copula is a *Gaussian copula* and, under the property of invariance under monotone increasing transformations, it is the same as the copula of $\boldsymbol{X} \sim N_d(\mathbf{0}, P)$, where P is the correlation matrix of $\boldsymbol{Y}$. In two dimensions, the Gaussian copula is given by:

$$C_\rho^G(u_1, u_2) = \Phi_P(\Phi^{-1}(u_1), \Phi^{-1}(u_2)) = \int_{-\infty}^{\Phi^{-1}(u_1)} \int_{-\infty}^{\Phi^{-1}(u_2)} \frac{1}{2\pi(1-\rho^2)^{1/2}} \exp\left(\frac{-(s_1^2 - 2\rho s_1 s_2 + s_2^2)}{2(1-\rho^2)}\right) ds_1 ds_2$$

where Φ and Φ_P denote the standard univariate normal distribution function and joint distribution function of $\boldsymbol{X}$ respectively, P is the correlation matrix and ρ is the correlation of X_1, X_2 (unique parameter of P in the bivariate case).

We can similarly define a 2-dimensional t-copula of $\boldsymbol{X} \sim t_d(\nu, \mathbf{0}, P)$ by introducing an additional parameter, namely the degrees of freedom:

$$C_{\rho,\nu}^t(u_1, u_2) = t_{P,\nu}(t_\nu^{-1}(u_1), t_\nu^{-1}(u_2))$$

where t_ν and $t_{P,\nu}$ are the standard univariate t distribution function and joint distribution function of $\boldsymbol{X}$ respectively with ν degrees of freedom, expectation 0 and variance $\frac{\nu}{\nu-2}$, and P is the correlation matrix of X_1, X_2. The degrees of freedom in the t-copula allows to adjust the co-movements of marginal extremes, and that makes t-copulas a popular choice for applications that stress the tail dependencies of risk factors.

The estimation of the parameters $\boldsymbol{\theta}$ of a parametric copula C_θ is performed through maximum likelihood. If $\hat{F}_1, \ldots, \hat{F}_d$ denote estimates of the marginal distribution

functions, we can construct a so-called *pseudo-sample* of observations from the copula that consists of the vectors $\hat{\boldsymbol{U}}_1, \ldots, \hat{\boldsymbol{U}}_d$, where

$$\hat{\boldsymbol{U}}_t = (\hat{U}_{t,1}, \ldots, \hat{U}_{t,d})' = (\hat{F}_1(X_{t,1}), \ldots, \hat{F}_d(X_{t,d}))'.$$

The MLE is obtained through maximising:

$$\ln L(\boldsymbol{\theta}; \hat{\boldsymbol{U}}_1, \ldots, \hat{\boldsymbol{U}}_n) = \sum_{t=1}^{n} \ln c_{\boldsymbol{\theta}}(\hat{\boldsymbol{U}}_t)$$

with respect to $\boldsymbol{\theta}$, where $\hat{\boldsymbol{U}}_t$ denotes the pseudo-observation from the copula and $c_{\boldsymbol{\theta}}$ is the copula density. A goodness-of-fit test can be further used in order to evaluate whether the data is appropriately modelled [9, 20].

3.4 Scenario Plausibility

The concepts described in the previous sections are combined towards a scenario generation framework that can be summarised in the following steps:

Model Fitting

1. Define the d-dimensional risk factor vector $\boldsymbol{X}_t = (X_{t,1}, \ldots, X_{t,d})'$, observable at time t, and obtain the logarithmic returns time series $\boldsymbol{r}_t = (r_{t,1}, \ldots, r_{t,d})'$
2. Fit an appropriate asymmetric ARMA–GARCH model to $\boldsymbol{r}_t$, and obtain the standardised residuals $\hat{\boldsymbol{Z}}_t = (\hat{Z}_{t,1}, \ldots, \hat{Z}_{t,d})'$
3. Estimate the marginal distribution functions $F_1(Z_1), \ldots, F_d(Z_d)$ of the i.i.d. standardised residuals, empirically for the body and with a GPD for the tails
4. Transform $\hat{\boldsymbol{Z}}_t$ to uniform variates $\hat{\boldsymbol{U}}_t = (\hat{U}_{t,1}, \ldots, \hat{U}_{t,d})'$ by inversion
5. Estimate parameters $\boldsymbol{\theta}$ of an appropriate copula $C_{\boldsymbol{\theta}}$ with MLE, given the pseudo-observations $\hat{\boldsymbol{U}}_t$

Scenario Generation

1. For a given sample size m, horizon n and parameters $\boldsymbol{\theta}$, simulate $n \times m$ points of the random vector $\boldsymbol{U} = (U_{t,1}, \ldots, U_{t,d})'$, with distribution function $C_{\boldsymbol{\theta}}$
2. Given the margins $F_1, \ldots, F_d$ from step 3 of the fitting process, use quantile transformation to translate to $\boldsymbol{Z}_t = (F_1^{\leftarrow}(U_{t,1}), \ldots, F_d^{\leftarrow}(U_{t,d}))'$
3. Provide the $n \times m$ standardised innovations matrix pairs to the calibrated ARMA–GARCH models and simulate m paths for each risk factor, $\boldsymbol{X}_{t,\ldots,t+n}$
4. Given the risk factor mapping $V_t = f(t, \boldsymbol{X}_t)$ and the simulated returns, construct m portfolio paths and obtain the P&L distribution

When examining the credibility of stress test results, plausibility is an important quality criterion that has been studied by existing literature [4, 16]. The problem set-

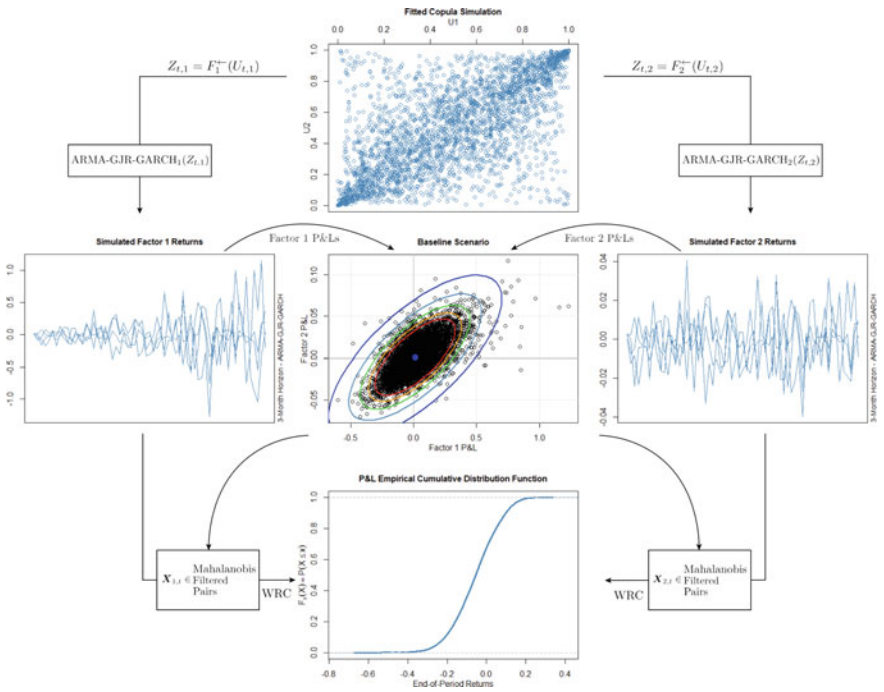


Fig. 1 Scenario generation procedure for 2 risk factors

ting in those studies consists of a set of elliptically distributed risk factors $\boldsymbol{X}_t$, a set of scenarios that represent factor movements, and a linear P&L function. The Mahalanobis distance is introduced a scenario restriction tool that respects the elliptical symmetry and authors utilise it to define the most likely scenarios given a pre-defined quantile of loss.

In this study, the ultimate goal is to observe variations in the P&L distribution; we do not examine the intersection of iso-P&L lines with iso-plausibility ellipses. Nevertheless, we utilise the concept of excluding scenario outliers, as it can bolster the credibility of long-horizon simulations. To this end, preceding step 5 of the described scenario generation procedure, for each set $\boldsymbol{X}_{t,\ldots,t+n}$, we obtain P&L sets $\text{PL}_1, \ldots, \text{PL}_d$; we compute the Mahalanobis distance $\text{MD}(\text{PL}_i)$ and filter out samples that exceed a pre-specified percentile of the underlying Chi-Squared distribution of MD. A schematic diagram of the scenario generation procedure in a 2-dimensional risk factor environment is presented in Fig. 1.

4 Analysis and Results

4.1 Index Replication and Risk Mapping

In the case of CGCI, we isolate two risk factors, namely the crypto-basket and the gold component. The crypto-basket is formulated as an equally-weighted basket of 5 cryptoassets, each with a weight of 0.2. The crypto-basket price base level is set on $\text{EW}_0 = 100$ on July 1st, 2015.

The crypto-basket price from January 2nd, 2016 onwards is given by:

$$\text{EW}_t = \left(1 + \sum_{i \in N_{c,t}} x_{i,R(t)} \times \left(\frac{P_{i,t}}{P_{i,R(t)}} - 1\right)\right) \times \text{EW}_{R(t)} \tag{5}$$

where

- $N_{c,t}$ is the set of the 5 cryptoassets constituents on day t
- $R(t)$ is the most recent CGCI rebalancing date preceding t
- $P_{i,t}$ is the closing price for cryptoasset i on day t, expressed in USD
- $P_{i,R(t)}$ is the closing price for cryptoasset i on the last rebalancing date preceding t, expressed in USD
- $x_{i,R(t)}$ is the weight of cryptoasset i on the last rebalancing date preceding t, equal to 0.2
- $\text{EW}_{R(t)}$ is the crypto-basket price on the last rebalancing date preceding t

The weighting between the crypto-basket and gold in the CGCI is computed through Eq. 1. The Index base level is set on $\text{Index}_0 = 1\,000$ on January 1st, 2016.

The Index level on day t from January 2nd, 2016 onwards is calculated as:

$$\text{Index}_t = \left(1 + \sum_{i \in N_t} x_{i,R(t)} \times \left(\frac{P_{i,t}}{P_{i,R(t)}} - 1\right)\right) \times \text{Index}_{R(t)} \tag{6}$$

where

- N_t represents the 2 CGCI components (crypto-basket and gold) on day t
- $x_{i,R(t)}$ is the weight of constituent i on the last rebalancing date preceding t, equal to the WRC allocation result
- $\text{Index}_{R(t)}$ is the CGCI price level on the last rebalancing date preceding t

We follow Eqs. 5 and 6 and replicate the price time series for the CGCI and transform to logarithmic returns.

4.2 Baseline Scenario Simulation

In this section we simulate a number of scenarios and evaluate the impact of stress in the main assumptions of the CGCI. We differentiate between the baseline, historical and hypothetical scenario. The first simulation describes as realistically as possible a recent index P&L profile that should serve as a benchmark to evaluate the severity of volatility or correlation shocks. The calibration period is chosen to include all daily observations of 2019.

The validity of stylised facts presented in Sect. 2 is verified in Figs. 8, 9, 10, 11, 12 and 13 in Appendix A for the CGCI components; therefore asymmetric ARMA–GARCH processes are likely to explain their evolution. We assume a t-distribution for the residuals for the ML process, iterate through pair-wise values of $p \in [1, 4]$, $q \in [0, 4]$ and choose the combination that yields the minimum AIC. The fitting results are presented in Table 1 of Appendix B.

Next, we estimate the semi-parametric distribution function of the standardised residuals. The linearity of the mean excess function can be used as a diagnostic to assist the selection of appropriate thresholds for the tails. Since the shape and scale parameters in Eq. 4 will be estimated after the threshold is defined, we use an empirical estimator for the mean excess function for positive-valued data $X_1, \ldots, X_n$, given by:

$$e^{\mathrm{EMP}}(u) = \frac{\sum_{i=1}^{n}(X_i - u)I_{\{X_i > u\}}}{\sum_{i=1}^{n} I_{\{X_i > u\}}} \tag{7}$$

We inspect the plot $\left(Z_i, e^{\mathrm{EMP}}(Z_i)\right)$, for the positive (Figs. 14 and 16, Appendix A) and for absolute values of the negative innovations (Figs. 15 and 17, Appendix A). For each tail, we define the threshold $u := Z_i$ for such i, from which the sample becomes approximately linear for higher values and obtain the parametric GPD tails. The baseline parameters can be found in Table 1. Combined with a Gaussian kernel smoothed interior, we obtain the baseline semi-parametric distribution function, displayed in Figs. 2 and 3.

Given the distribution functions F_c, F_g, we perform inverse transform sampling to the fitted ARMA–GJR–GARCH standardised residuals and fit a copula to the pseudo-sample (Table 1). We proceed with the t-copula for the remainder of this paper, as it yields the maximum log-likelihood and provides the means for correlation stress testing through its parameters (ρ, ν).

With the ARMA–GJR–GARCH parameters, cumulative distribution functions, fitted t-copula for the residuals and a 3-month simulation horizon, we produce 10 000 paths for each risk factor, $\boldsymbol{X}_{t,c}, \boldsymbol{X}_{t,q}$. We filter out all path pairs that yield an MD value that exceeds the 99th percentile of the underlying Chi-Squared distribution, and use the remaining to produce $m' < m$ paths for the CGCI. Finally, we compute the P&L at the end of period for each path pair and get the baseline P&L distribution function. The generated scenarios with the MD plausibility bound, and the final P&L distribution for the baseline scenario can be found in Figs. 4 and 5.

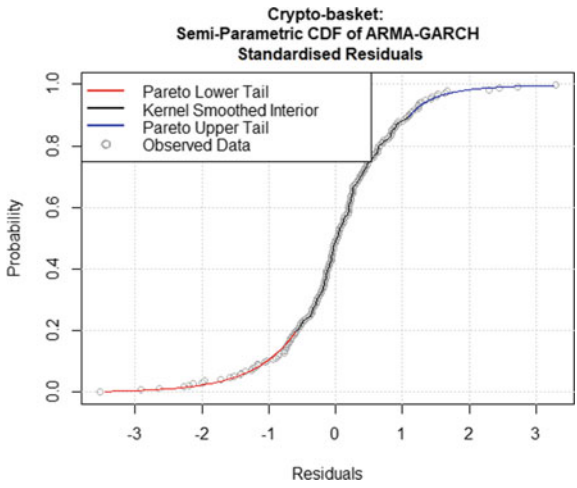


Fig. 2 Semi-parametric CDF Crypto-basket residuals

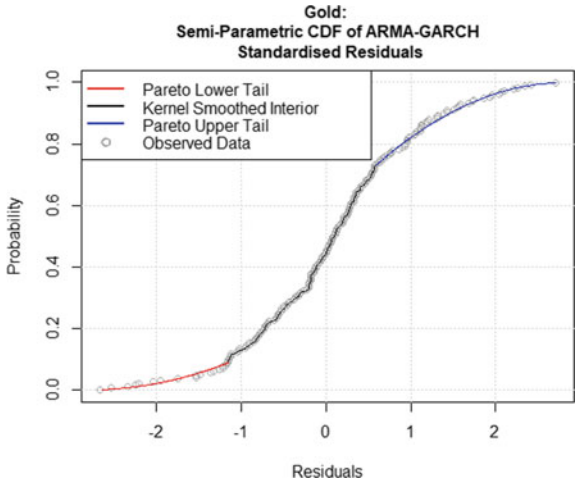


Fig. 3 Semi-parametric CDF Gold residuals

We can also choose to generate a baseline scenario for single risk factor portfolios. In this case, the pseudo-random observations of the residuals can be derived directly from the their cumulative distribution functions. We use this approach to generate 10 000 paths of returns for the MVIS Digital Assets 5 Index [17] (MVDA5)—a market capitalisation-weighted index which tracks the performance of the five largest and most liquid cryptoassets—and compare its P&L distribution with the one derived for the CGCI. The details of the fitting process for the MVDA5 baseline scenario and the plot of the semi-parametric residual distribution can be found in Table 1 of Appendix B and Fig. 18 of Appendix A.

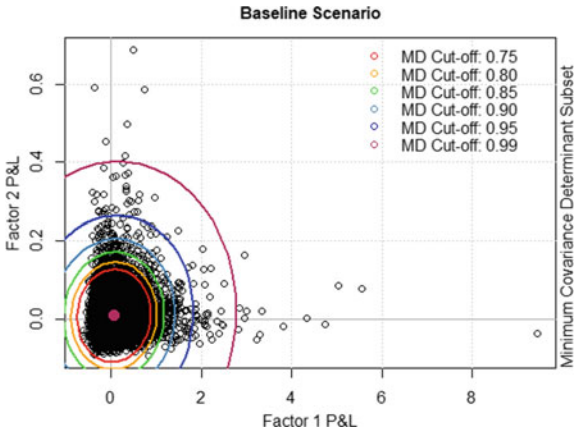


Fig. 4 CGCI baseline scenarios: Mahalanobis plausibility ellipses

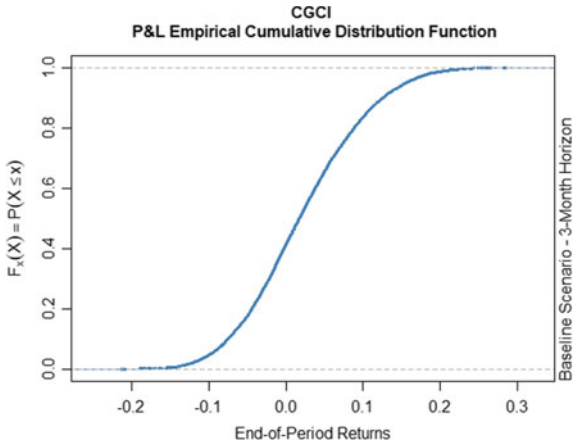


Fig. 5 CGCI baseline scenarios: P&L distribution

4.3 Historical and Hypothetical Scenarios

The model fitting process can be modified to generate stress scenarios that can be compared against the baseline. First, we change the calibration period to reflect stressful market conditions. Those historical scenarios are commonly used in practice because they are based on events that are observed, and therefore likely to reoccur. For CGCI case, we generate the historical scenarios by calibrating the models in the period of 2018-01-01 to 2018-12-31, as it reflects very large downward price movements in the cryptoasset space.

We use this sample to fit the ARMA–GJR–GARCH model for both risk factors and obtain the i.i.d. standardised residuals. Following the same steps as in Sect. 4.2, we model the tails and produce the semi-parametric cumulative distribution functions. The new model parameters can be found in Table 1 of Appendix B. We transform the residuals to uniform variates and fit an appropriate t-copula (Table 1 of Appendix

B). With the new parameters we generate 10 000 paths for each risk factor, filter out implausible pairs, compute the P&L at the end of the period and obtain the new P&L distribution that corresponds to the historical scenario. A single-factor approach is also used to produce the MVDA5 historical P&L distribution. While the historical scenarios can be used to assess the portfolio's performance under realised stressful market conditions, it is biased towards past experience and fails to examine plausible stress events that have not occurred yet.

Additional vulnerabilities are revealed with the generation of hypothetical scenarios. In order to test the resilience to a potential increase in correlation between the CGCI components, we increase the parameter ρ of the t-copula when generating pseudo-samples. We further choose to decrease the degrees of freedom to increase the likelihood of joints extremes. Additionally, we shock the risk factors' volatilities by changing the ARMA–GJR–GARCH volatility constant, ω, and produce volatile risk factor paths. For this study, we modify those parameters such that $\rho = 0.9, \nu = 2$ and $\omega = 10\omega_b$, where ω_b denotes the volatility constant. The P&Ls that are derived through the modified copula is presented in Fig. 19. The same volatility shock is introduced to the MVDA5 volatility constant to obtain its hypothetical scenario risk profile.

Figures 6 and 7 present a comparison of the Baseline, Historical and Hypothetical Scenarios for the case of CGCI and MVDA5. In the case of the CGCI, the slope of the historical P&L distribution is visibly less steep compared to the baseline. The heavier lower tail further confirms the increased risk in the historical scenario. In Fig. 6, the probability of a positive return in the historical scenario lies around 31% compared to 58% for the CGCI baseline. When it comes to the volatility-shocked scenario, it does not differ significantly around the mid-section, with the probability of profit remaining around 58%, but the CDF is slightly heavier on the upper and lower tail. A single correlation shock impacts the risk profile even less, while a simultaneous shock in both volatility and correlation bring the probability of profit down to approximately 53%.

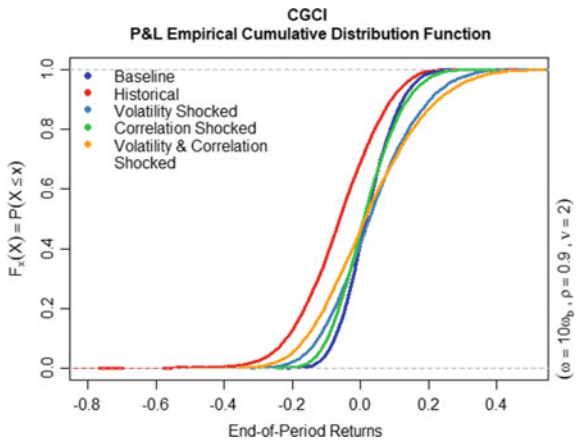


Fig. 6 CGCI P&L Distributions

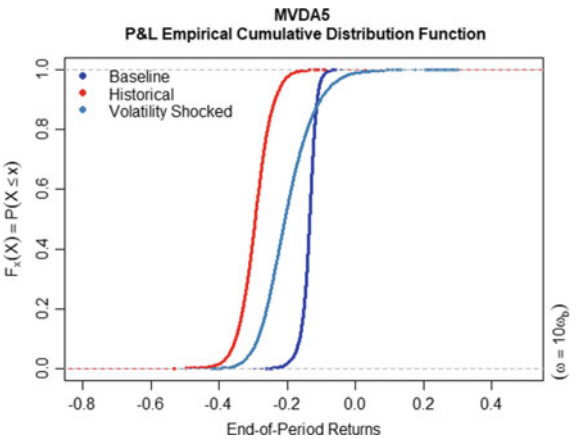


Fig. 7 MVDA5 P&L Distributions

In the case of MVDA5, the baseline P&L distribution reveals significantly higher levels of risk, with no prospect of profit on the 3-month horizon. The historical scenario is more severe, displaying 85% probability of at least a 25% loss on the initial investment. The volatility-shocked MVDA5 is heavier on the upper tail, but the probability of a positive end-of-period return barely exceeds 1%. Overall, CGCI's WRC allocation scheme is characterised by a much more stable risk profile in all the scenarios considered in this study.

5 Conclusion

We have proposed a framework for scenario-based risk management and have described how it can be applied to assess the risk profile of diversified portfolios with cryptoasset components. The joint evolution of the identified risk factors are modelled in a realistic way and the analysed scenarios are severe, yet plausible. By taking into account a variety of plausible future events related to volatility and correlation levels, we demonstrate the superiority of diversified strategies, such as the CGCI, as a means of mitigating risk. While this application focuses in generating basic stress scenarios specifically for the case CGCI, it can be further modified to include arbitrary combinations of risk factor shocks. Additionally, the presented procedure can potentially be used as a forward-looking portfolio optimisation approach.

Acknowledgements Imperial College of London gratefully acknowledges the support given by CoinShares. Their funding provides Imperial with the support needed to conduct the research found within this paper.

Appendix A Plots

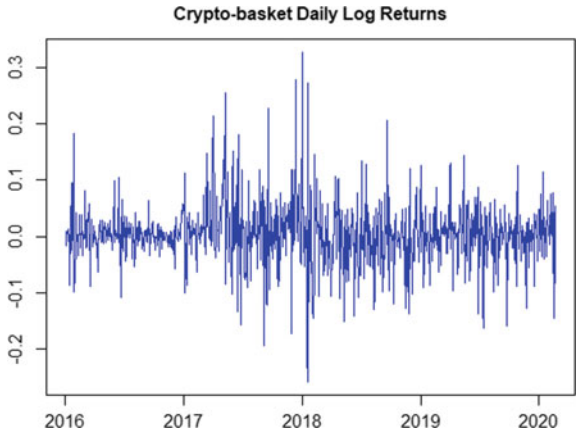


Fig. 8 Crypto-basket log-returns

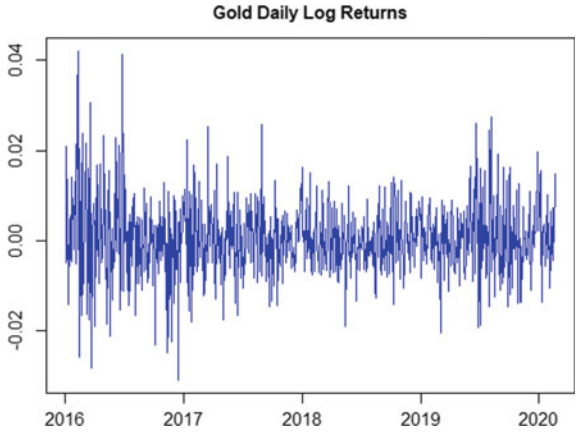


Fig. 9 Gold log-returns

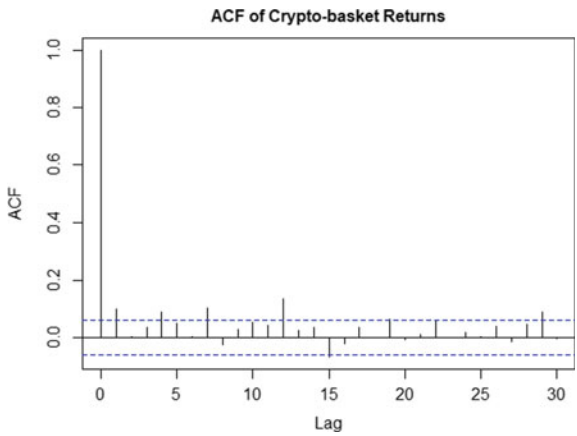


Fig. 10 ACF of Crypto-basket returns

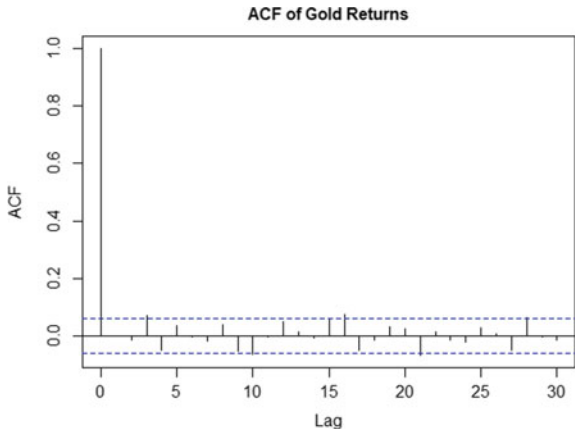


Fig. 11 ACF of Gold returns

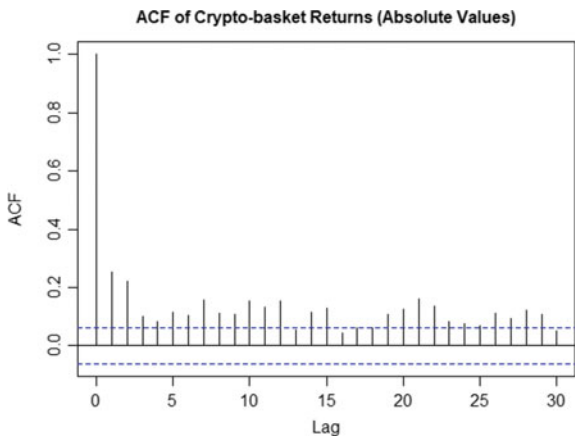


Fig. 12 ACF of Crypto-basket returns (absolute values)

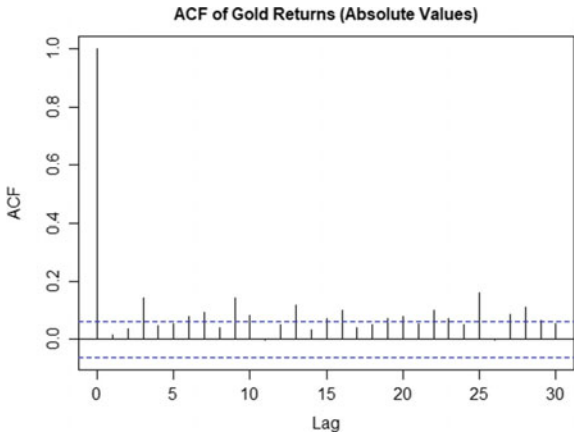


Fig. 13 ACF of Gold returns (absolute values)

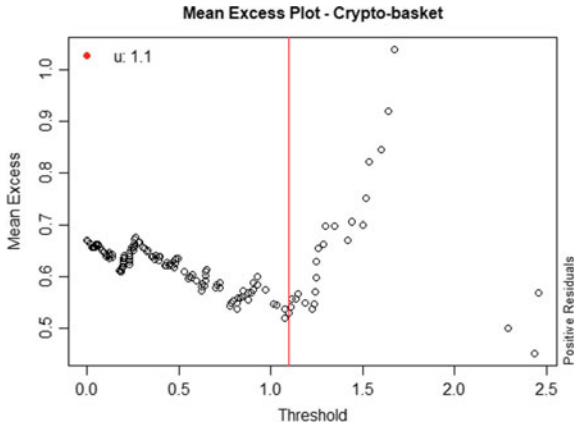


Fig. 14 Crypto-basket mean excess plot Positive residuals

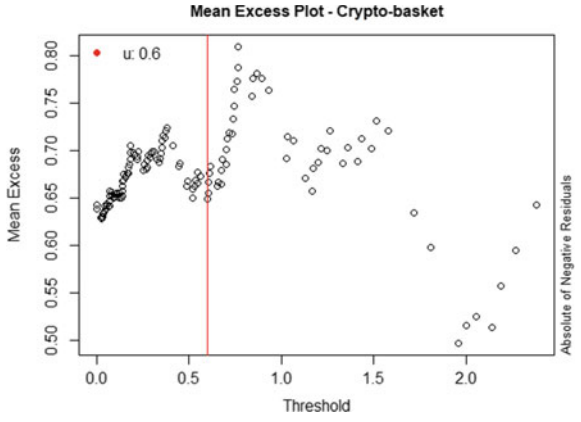


Fig. 15 Crypto-basket mean excess plot Absolute of negative residuals

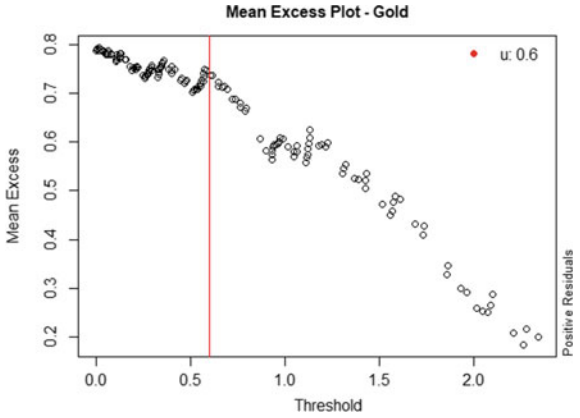


Fig. 16 Gold mean excess plot Positive residuals

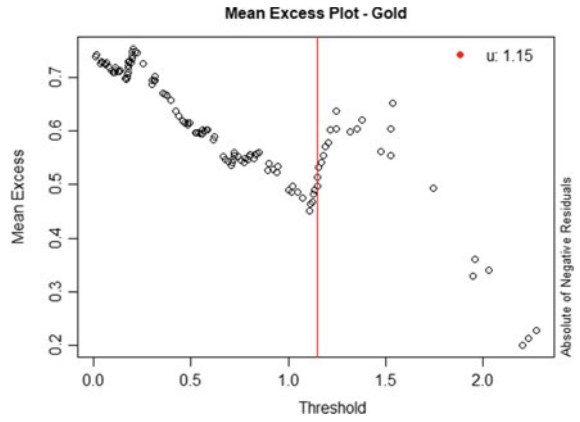


Fig. 17 Gold mean excess plot Absolute of negative residuals

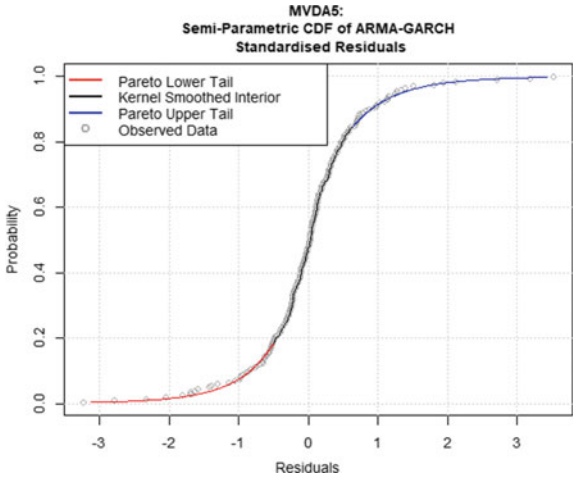


Fig. 18 MVDA5 residuals baseline CDF

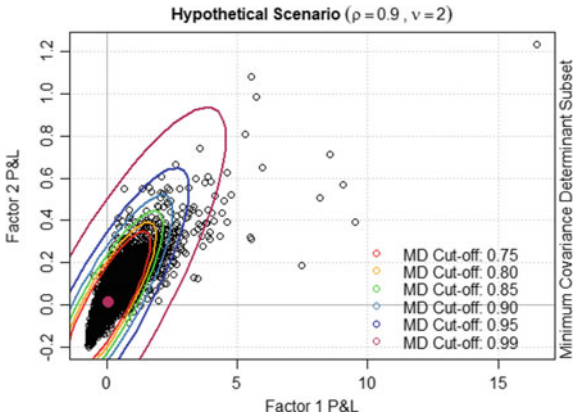


Fig. 19 CGCI hypothetical scenarios

Appendix B Table

Table 1 ARMA-GJR-GARCH, GPD and Copula Parameters and diagnostics

ARMA-GJR-GARCH Fitting

	Crypto-basket Baseline	Crypto-basket Historical	Gold Baseline	Gold Historical	MDVA5 Baseline	MDVA5 Historical
Order (ARMA) (GJR-GARCH)	(4, 4) (1, 1)	(4, 4) (1, 1)	(4, 3) (1, 1)	(4, 4) (1, 1)	(4, 2) (1, 1)	(4, 2) (1, 1)
Parameters (Eq. 2)	ϕ_1: −0.95408 ϕ_2: −0.35581 ϕ_3: −1.07815 ϕ_4: −0.99300 θ_1: 1.03614 θ_2: 0.37295 θ_3: 1.11033 θ_4: 1.06890 ω: 0.000007 α_1: 0.02770 γ_1: −0.06146 β_1: 1.00000	ϕ_1: 0.05108 ϕ_2: 0.45840 ϕ_3: 0.06855 ϕ_4: −0.76304 θ_1: 0.08024 θ_2: −0.42010 θ_3: −0.02407 θ_4: 0.92655 ω: 0.00267 α_1: 0.22360 γ_1: 0.10915 β_1: 0.13794	ϕ_1: −0.90692 ϕ_2: −1.00161 ϕ_3: −0.72425 ϕ_4: 0.01181 θ_1: 0.79031 θ_2: 1.11737 θ_3: 0.74846 ω: 0.000001 α_1: 0.06469 γ_1: −0.13793 β_1: 0.98006	ϕ_1: 0.59656 ϕ_2: 1.08368 ϕ_3: −0.58776 ϕ_4: −0.39986 θ_1: −0.79959 θ_2: −1.00687 θ_3: 0.76978 θ_4: 0.26998 ω: 0.00000 α_1: 0.00000 γ_1: −0.00200 β_1: 1.00000	ϕ_1: −1.21805 ϕ_2: −1.19354 ϕ_3: −0.19727 ϕ_4: 0.02250 θ_1: 1.07132 θ_2: 1.03795 ω: 0.000002 α_1: 0.03245 γ_1: −0.06707 β_1: 1.00000	ϕ_1: −1.93720 ϕ_2: −1.12456 ϕ_3: −0.12869 ϕ_4: −0.02543 θ_1: 1.94084 θ_2: 1.03890 ω: 0.000003 α_1: 0.000001 γ_1: 0.08400 β_1: 0.94997

(continued)

Table 1 (continued)

ARMA-GJR-GARCH Fitting						
	Crypto-basket Baseline	Crypto-basket Historical	Gold Baseline	Gold Historical	MDVA5 Baseline	MDVA5 Historical
AIC	−3.6341	−2.7398	−7.2120	−7.4216	−4.0542	−3.2292
Residual Distribution Shape (KS Test p-value)	3.32702 (0.7003)	3.94689 (0.9604)	19.83051 (0.6673)	5.93231 (0.8674)	2.55280 (0.8684)	4.52857 (0.8335)
Generalised Pareto Distribution Fitting						
Threshold u (Upper) (Lower)	1.10114 −0.60602	0.54891 −0.75895	0.60316 −1.15400	0.75454 −0.61262	0.63667 −0.51043	0.64902 −1.25717
Parameters (Upper) (Lower)	ξ: 0.27178 β: 0.39605 ξ: 0.01501 0.64541	ξ: −0.03047 β: 0.62530 ξ: 0.13459 β: 0.57712	ξ: −0.49452 β: 1.10766 ξ: −0.46581 β: 0.82030	ξ: 0.15642 β: 0.53995 ξ: −0.34770 β: 1.02221	ξ: 0.09272 β: 0.59313 ξ: 0.10358 β: 0.52568	ξ: −0.14684 β: 0.66304 ξ: 0.28828 β: 0.59477
LogLik (Upper) (Lower)	9.66944 30.58602	28.99699 28.07471	44.36422 8.06560	27.00540 40.44401	33.08554 203.4029	25.21121 27.66995
CGCI Copula Fitting						
		Gaussian Copula Loglik	t-Copula Loglik	t-Copula Parameters (ρ, ν)	Goodness-of-fit (p-value)	
Baseline Scenario		0.41348	0.61102	(0.05917, 23.62447)	0.7737	
Historical Scenario		0.24004	0.58172	(−0.03363, 16.62352)	0.3112	

References

1. Basel Committee on Bank Supervision: Basel I. (1988). The Basel Capital Accord.
2. Basel Committee on Bank Supervision: Basel II. (2004). International Convergence of Capital Measurement and Capital Standards: A Revised Framework.
3. Board of Governors of the Federal Reserve System. (2019). Amendments to policy statement on the scenario design framework for stress testing. *Federal Register*, *84*, 6651–6664.
4. Breuer, T., et al. (2009). How to find plausible, severe, and useful stress scenarios. *International Journal of Central Banking*, *5*.
5. Cherubini, U., et al. (2004). *Copula methods in finance*. Wiley.
6. Chu, J., et al. (2017). GARCH modelling of cryptocurrencies. *Journal of Risk and Financial Management*, *10*, 17.
7. Cont, R. (2002). Empirical properties of asset returns: Stylized facts and statistical issues. *Quantitative Finance*, *1*, 223–236.
8. Elendner, H., et al. (2018). The cross-section of crypto-currencies as financial assets: Investing in crypto-currencies beyond bitcoin. In *Handbook of blockchain, digital finance, and inclusion* (pp. 145–173). Elsevier.
9. Genest, C., & Remillard, B. (2008). Validity of the parametric bootstrap for goodness-of-fit testing in semiparametric models. *Annales De L'Institut Henri Poincare – Probabilites et Statistiques*.
10. Glosten, L. R., et al. (1993). On the relation between the expected value and the volatility of the nominal excess return on stocks. *Journal of Finance*, *48*(5), 1779–1801.
11. Kjærland, F., et al. (2018). An analysis of bitcoin's price dynamics. *Journal of Risk and Financial Management*, *11*, 63.

12. Koutsouri, A., Poli, F., Alfieri, E., Petch, M., Distaso, W., & Knottenbelt, W. J. (2019). Balancing cryptoassets and gold: A weighted-risk-contribution index for the alternative asset space. In *Proceedings MARBLE 2019*. Santorini, Greece.
13. Li, W., & Mak, T. (2008). On the squared residual autocorrelations in non-linear time series with conditional heteroscedasticity. *Journal of Time Series Analysis*, 627–636.
14. Lopez, J. (2005). Stress tests: Useful complements to financial risk models. *FRBSF Economic Letter*.
15. McNeil, A., Frey, R., & Embrechts, P. (2005). *Quantitative risk management: Concepts, techniques, and tools* (Vol. 101). Princeton University Press.
16. Mouy, P., et al. (2017). Extremely (un)likely: A plausibility approach to stress testing. *Risk Magazine*, 72–77.
17. MV Index Solutions. (2019). MVIS CryptoCompare Digital Assets 5 Index. Index Factsheet.
18. Nyström, K., & Skoglundy, J. (2002). A framework for scenario-based risk management.
19. Poundstone, W. (2005). *Fortune's formula*. Hill and Wang.
20. Remillard, B., et al. (2009). Goodness-of-fit tests for copulas: A review and a power study. *Insurance: Mathematics and Economics*, 199–213.
21. Studer, G. (1999). Market risk computation for nonlinear portfolios. *The Journal of Risk*, *1*(4), 33–53.
22. Tsay, R. S. (2010). *Analysis of financial time series*. Wiley.

Selfish Mining in Ethereum

Cyril Grunspan and Ricardo Perez-Marco

Abstract We study selfish mining in Ethereum. The problem is combinatorially more complex than in Bitcoin because of major differences in the reward system and a different difficulty adjustment formula. Equivalent strategies in Bitcoin do have different profitabilities in Ethereum. The attacker can either broadcast his fork one block by one, or keep them secret as long as possible and publish them all at once at the end of an attack cycle. The first strategy is damaging for substantial hashrates, and we show that the second strategy is even worse. This confirms what we already proved for Bitcoin: Selfish mining is most of all an attack on the difficulty adjustment formula. We show that the current reward for signaling uncle blocks is a weak incentive for the attacker to signal blocks. We compute the profitabilities of different strategies and find out that for a large parameter space values, strategies that do not signal blocks are the best ones. We compute closed-form formulas for the apparent hashrates for these strategies and compare them. We use a direct combinatorial analysis with Dyck words to find these closed-form formulas.

Keywords Bitcoin · Ethereum · Blockchain · Proof-of-work · Selfish mining · Catalan numbers · Dyck path · Random walk

1991 Mathematics Subject Classification 68M01 · 60G40 · 91A60

C. Grunspan (✉)
Léonard de Vinci, Pôle Univ., Research Center, Paris-La Défense, Labex Réfi, Paris, France
e-mail: cyril.grunspan@devinci.fr

R. Perez-Marco
CNRS, IMJ-PRG, Labex Réfi, Paris, France
e-mail: ricardo.perez.marco@gmail.com

P. Pardalos et al. (eds.), *Mathematical Research for Blockchain Economy*,
Springer Proceedings in Business and Economics,
https://doi.org/10.1007/978-3-030-53356-4_5

1 Introduction

1.1 Selfish Mining Strategies in Ethereum

Research on selfish mining (in short SM) in Ethereum is quite recent. We can mention as recent contributions [7] (numerical study) and [1].

The authors of [1] use a Markov chain model and compute the stationary probability. Then they study what they call the "absolute revenue" of the attacker which corresponds to the apparent hashrate after a difficulty adjustment as explained in our articles on blockwithholding attacks in the Bitcoin network (see [2–4]). Their theoretical analysis seems also confirmed by their numerical simulations. They do not provide closed-form formulas (for example Formulas (8) and (9) in Sect. 3-E involve double infinite sums). But more importantly, their study is limited to the following strategy of the attacker:

(1) The attacker refers to all possible orphan blocks;
(2) When new blocks are validated by the honest miners, the attacker makes public the part of his fork sharing the same height as the "honest" blockchain.

(See Algorithm 1 in [1], Lines 1 and 19 from Sect. 3-C)

We label this strategy as "Strategy 1" or SM1. The procedure of a Bitcoin selfish miner to release his secret fork is irrelevant for the profitability of the classical selfish mining attack. However, this is not so in Ethereum. In particular, the precise algorithm presented in [1] is not the most profitable as we will prove. An alternative strategy for the attacker would be to keep secret all his fork until he is on the edge of being caught-up by the honest miners. Then, and only at this critical moment, he would release his complete fork and override the public blockchain. We label this second strategy as "Strategy 2" or SM2. In Bitcoin, both strategies have the same effect since only matters the number of blocks mined by the attacker and added to the official blockchain. But in Ethereum, this is not so because of the different reward incentives that gives rewards to "nephew" blocks who refer to "uncle" blocks. "Uncle" blocks are orphan blocks with a parent in the official blockchain, and the descendants of this parent in the official blockchain are its "nephew" blocks. Also uncle blocks get rewards when referred by nephews.

1.2 Performance of Ethereum Selfish Mining Strategies

To understand what the best strategy for the attacker is, we need an in-deep knowledge of the nature of the selfish mining attack. In [2] we give a correct economic modeling with a model of repetition game, and we consider the time element that is absent from older Markov chain models. What is important for the attacker is to maximize the number of validated blocks in the official blockchain *per unit of time*, which is

different from the percentage of blocks he validates. With this correct modeling, it becomes then clear that the attack is an exploit on Bitcoin's difficulty adjustment formula, that does include the orphan blocks. Then the attacker lowers artificially the difficulty, at the expense of orphaned honest blocks, and succeeds to validate more blocks per unit of time.

Point (2) in "Strategy 1" creates numerous competitions between the attacker's fork and the honest blockchain. This increases the production of orphan blocks that becomes important for a substantial hashrate of the attacker. Signaling these orphan blocks yields additional rewards to the attacker, but it goes against its main goal to lower the difficulty. Indeed, the difficulty's adjustment formula in Ethereum counts for "uncles", that are the orphan blocks directly attached to the main chain. Therefore, increasing the number of uncles by Point 2 has the following contradictory effects: On one hand, the attacker's revenue increases because of the new "inclusion rewards", but on the other hand, the difficulty is not lowered, so the attacker ends up mining less official blocks per unit of time in Strategy 1 compared to Strategy 2.

On the contrary, if the attacker decides to avoid competitions with honest miners as much as possible, he will earn less inclusion rewards (he can even decide to ignore totally these rewards) but his speed of validation of blocks will increase. So, what is the best strategy will depend very sensitively on the parameters of the reward system.

As explained in [3], the correct benchmark to compare profitabilities of two strategies is the revenue ratio

$$\Gamma = \frac{\mathbb{E}[R]}{\mathbb{E}[T]}$$

where R is the revenue of the miner per attack cycle and T is the duration of an attack cycle. In Bitcoin, after a difficulty adjustment, this quantity becomes in the long run proportional to

$$\tilde{\Gamma} = \frac{\mathbb{E}[R_s]}{\mathbb{E}[L]}$$

where L (resp. R_s) is the number of new blocks (resp. new blocks mined by the attacker) added to the official blockchain per attack cycle. The difficulty adjustment is not continuous in Bitcoin as it is updated every 2016 official new blocks. With the martingale tools introduced in [2], we computed how long it takes for the attack to become profitable (this computation is not possible with the old Markov chain model).

In Ethereum, the difficulty adjustment formula is different. The revenue ratio is proportional to

$$\tilde{\Gamma} = \frac{\mathbb{E}[R]}{\mathbb{E}[L] + \mathbb{E}[U]}$$

where U is the number of referred uncles and R is the total revenue of the attacker in the attack cycle. Moreover, the revenue R per attack cycle has three different contributions:

(1) The revenue R_s coming from "static" blocks.
(2) The revenue R_u coming from "uncles" blocks.
(3) The revenue R_n coming from "nephews" blocks.

In Bitcoin's revenue analysis only R_s is present. Therefore, for Ethereum we have

$$\tilde{\Gamma} = \frac{\mathbb{E}[R]}{\mathbb{E}[L] + \mathbb{E}[U]} = \frac{\mathbb{E}[R_s] + \mathbb{E}[R_u] + \mathbb{E}[R_n]}{\mathbb{E}[L] + \mathbb{E}[U]}$$

The new terms on the numerator $\mathbb{E}[R_u]$ and $\mathbb{E}[R_n]$ increase the revenue of the attacker and are incentives for block withholding attacks. On the other hand, the new term $E[U]$ in the denominator plays against the profitability of the attack and tends to mitigate the attack. Only an exact computation of these terms can show which one is the most profitable strategy. Another particularity of Ethereum is the continuous adjustment of the difficulty. Thus a block-withholding attack is very quickly profitable.

There are other selfish mining strategies in Ethereum. For instance, the attacker can publish his secret blocks slowly, two by two, instead of one by one. In this article we limit our study to Strategy 1 and Strategy 2. The main result are the closed-form formulas for the apparent hashrates in Strategy 1 and 2. The main conclusion is that the effect on the difficulty adjustment is prevalent, so that Strategy 2 outperforms Strategy 1.

2 A Combinatorics Approach

In this section we present a general setup that is common for all strategies. We apply our combinatorics approach to selfish mining as done previously for Bitcoin [4]. Dyck words and Catalan numbers are a powerful tool to compute the revenue ratio of a selfish miner in Bitcoin. In [4] we proved the following Theorem and Corollary:

Theorem 2.1 *Let L be the number of official new blocks added to the official blockchain after an attack cycle. We have*

$$\mathbb{P}[L = 1] = p\ ,$$
$$\mathbb{P}[L = 2] = pq + pq^2\ ,$$

and for $n \geq 3$,

$$\mathbb{P}[L = n] = pq^2(pq)^{n-2}C_{n-2}$$

where $C_n = \frac{(2n)!}{n!(n+1)!}$ is the n-th Catalan number.

Corollary 2.2 *We have* $\mathbb{E}[L] = 1 + \frac{p^2 q}{p-q}$.

We can represent the combinatorics information of an attack cycle ω by the chronological sequence of blocks, S (for Selfish) and H (for Honest). The relation between selfish mining and Dyck words is the following (see [4]),

Proposition 2.3 *Let ω be an attack cycle starting with SS. Then, ω ends with H and the intermediate sequence w defined by $\omega = SSwH$ is a Dyck word.*

Definition 2.4 For $n \geq 0$, we denote by $C_n(x) = \sum_{k=0}^{n} C_k x^k$, the n-th partial sum of the generating series of the Catalan number.

Example 2.5 We have $C_4(x) = 1 + x + 2x^2 + 5x^3 + 14x^4$.

Definition 2.6 We define $\pi_0 = \pi_1 = 0$ and for $k \geq 2$,

$$\pi_k = pq^2(\mathbf{1}_{k=2} + \mathbf{1}_{k\geq 2} \cdot (pq)^{k-2} C_{k-2}).$$

The following lemma results from Theorem 2.1.

Lemma 2.7 *Let ω be an attack cycle.*

- *For $k \geq 0$, the probability that ω is won by the attacker and $L(\omega) = k$ is π_k.*
- *For $k \geq 2$, the probability that ω is won by the attacker and $L(\omega) \leq k$ is $pq^2 + pq^2 C_{k-2}(pq)$.*

Proof We have either $\omega = \text{SHS}$ or ω starts with SS. The result then follows from Lemma 6.2 in the Appendix. □

For Ethereum, the "static" part R_s of the revenue of the selfish miner coming from rewards for validated blocks is the same as for Bitcoin. However, we need to add the new terms R_s and R_n coming from uncle and nephew rewards.

Definition 2.8 If ω is an attack cycle, we denote by $U(\omega)$ (resp. $U_s(\omega)$, $U_h(\omega)$) the random variable counting the number of uncles created during the cycle ω which are referred by nephew blocks (resp. nephew blocks mined by the selfish miner, nephew blocks mined by the honest miners) in the cycle ω or in a later attack cycle.

We denote by $V(\omega)$ the random variable counting the number of uncles created during the cycle ω and are referred by nephew blocks (honest or not) in an attack cycle strictly after ω.

We take from [1] the notation K_u for the uncles reward function, and we denote by π the inclusion reward (see the glossary at the end).

For a general block withholding strategy, the random variables from Definition 2.8 do not contain all the information for the computation of the attacker's revenue. It depends not only on the number of uncles mined by the attacker but also on their distance d to its corresponding nephews.

However, for a miner following a selfish mining strategy, the part of his revenue coming from uncle rewards are easy to compute, as shown in the next Proposition, because only the case $d = 1$ is possible. This observation was already made in [1].

Proposition 2.9 *Let $R_u(\omega)$ be the total amount of uncle rewards of the selfish miner during an attack cycle ω. We have:*

$$\mathbb{E}[R_u] = p^2 q(1-\gamma)K_u(1)\ .$$

Currently on Ethereum we have $K_u(1) = \frac{7}{8}b$.

Proof Let ω be an attack cycle. If $\omega =$ SHH with a second honest block mined on top of another honest block after a competition, the attacker has an uncle which is referred by the second honest block of the honest miners in the cycle ω. Otherwise, if $\omega \neq$ SHH then the attacker has no uncle in the cycle ω (the only uncle blocks are those mined by the honest miners). □

The *apparent hashrate* is the long term apparent hashrate of the attacker after the manipulation of the difficulty by the attacker.

Definition 2.10 We denote by $\widetilde{q}_B$, resp. $\widetilde{q}_E$, the long term apparent hashrate of the selfish miner in Bitcoin, resp. Ethereum, defined by

$$\widetilde{q}_B = \frac{\mathbb{E}[R_s]}{\mathbb{E}[L]}$$
$$\widetilde{q}_E = \frac{\mathbb{E}[R_s] + \mathbb{E}[R_u] + \mathbb{E}[R_n]}{\mathbb{E}[L] + \mathbb{E}[U]}$$

For Bitcoin we have the following formula (see [6] and [2]),

$$\widetilde{q}_B = \frac{[(p-q)(1+pq) + pq]q - (p-q)p^2 q(1-\gamma)}{pq^2 + p - q}$$

For Ethereum only $\mathbb{E}[U]$ and $\mathbb{E}[U_s]$ are relevant for the computation of the apparent hashrate of the selfish miner:

Theorem 2.11 *We have*

$$\widetilde{q}_E = \tilde{q}_B \cdot \frac{\mathbb{E}[L]}{\mathbb{E}[L]+\mathbb{E}[U]} + \frac{p^2 q(1-\gamma)K_u(1)}{\mathbb{E}[L]+\mathbb{E}[U]} + \frac{\mathbb{E}[U_s]}{\mathbb{E}[L]+\mathbb{E}[U]}\pi\,.$$

Currently on Ethereum we have $K_u(1) = \frac{7}{8}$ and $\pi = \frac{1}{32}$.

Proof Using Proposition 2.9, we have:

$$\begin{aligned}\widetilde{q}_E &= \frac{\mathbb{E}[R_s]+\mathbb{E}[R_u]+\mathbb{E}[R_n]}{\mathbb{E}[L]+\mathbb{E}[U]} \\ &= \frac{\mathbb{E}[R_s]}{\mathbb{E}[L]} \cdot \frac{\mathbb{E}[L]}{\mathbb{E}[L]+\mathbb{E}[U]} + \frac{\mathbb{E}[R_u]}{\mathbb{E}[L]+\mathbb{E}[U]} + \frac{\mathbb{E}[U_s]}{\mathbb{E}[L]+\mathbb{E}[U]}\pi \\ &= \tilde{q}_B \cdot \frac{\mathbb{E}[L]}{\mathbb{E}[L]+\mathbb{E}[U]} + \frac{p^2 q(1-\gamma)K_u(1)}{\mathbb{E}[L]+\mathbb{E}[U]} + \frac{\mathbb{E}[U_s]}{\mathbb{E}[L]+\mathbb{E}[U]}\pi\end{aligned}$$

□

In next sections we compute $\mathbb{E}[U_s]$ and $\mathbb{E}[U]$ for different selfish mining strategies.

3 Strategy 1: Maximum Belligerence Signalling All Uncles

We consider here the strategy described in [1] where the attacker engages in competition with the honest miners as often as possible, and signals all possible "uncles".

3.1 General Definitions and Basic Results

Definition 3.1 The relative height of an orphan block $\mathfrak{b}$ validated by the honest miners is the difference between the height of the secret fork of the attacker at the time of creation of $\mathfrak{b}$ and the height of $\mathfrak{b}$. We denote it $h(\mathfrak{b})$.

Example 3.2 For $\omega =$ SSSHSHSHH, the first three "honest" blocks have relative height equal to 2 and the last "honest" block has a relative height equal to 1.

Proposition 3.3 *Let $\mathfrak{b}$ be an uncle block mined by an honest miner and signaled by a nephew block which is at a distance d of $\mathfrak{b}$. Then, we have $h(\mathfrak{b}) < d$.*

Proof Let $\mathfrak{b}'$ be the last block mined by the selfish miner at the date of creation of $\mathfrak{b}$. Notice that $h(\mathfrak{b})$ is also the number of blocks between $\mathfrak{b}$'s parent and $\mathfrak{b}'$. Thus the distance between $\mathfrak{b}$ and a possible nephew is necessarily strictly greater than $h(\mathfrak{b})$. □

Note 3.4 Let $n \geq 0$ and $\omega = \mathrm{SS}w$ be an attack cycle with $w = w_1 \dots w_{2n+1}$, $w_i \in \{S, H\}$ and $w_{2n+1} = H$. Then, w can be identified with a simple finite path $(X_i)_{0 \leqslant i \leqslant 2n+1}$ starting from 0, satisfying: $\forall i \leqslant 2n+1$, $X_i = X_{i-1} + 1$ (resp. $X_i = X_{i-1} - 1$) if $w_i = S$ (resp. $w_i = H$) and ending at $X_{2n+1} = -1$ (see the Appendix). The index i indicates the $(i+2)$-th block validated during ω. It has been mined by the attacker (resp. honest miners) if $X_i = X_{i-1} + 1$ (resp. $X_i = X_{i-1} - 1$).

Proposition 3.5 *Let $\omega = SSw$ an attack cycle starting with two S with $w = w_1 \dots w_{2n+1}$, $w_i \in \{S, H\}$ and $w_{2n+1} = H$. We denote by $X : [0, 2n+1] \longrightarrow [-1, +\infty]$ the path associated with w as in Note 3.4. For $i \leqslant 2n+1$, let $\mathfrak{b}_i$ denote the i-th validated block in w. Then we have:*

$$X_i < X_{i-1} \Longrightarrow h(\mathfrak{b}_i) = X_i + 2$$

Proof By induction on i, we show that $X_i + 2$ represents the advance of the fork created by the attacker over the official blockchain at the time of creation of the i-th block in w. Now, if $X_i < X_{i-1}$ then by Note 3.4, $\mathfrak{b}_i$ is a block validated by the honest miners. So $h(\mathfrak{b}_i)$ is well defined, and we get the result using Definition 3.1. □

Proposition 3.6 *Let $\omega = SSw$ be an attack cycle starting with two S and let $\mathfrak{b}_i$ be the i-th block validated in w. We denote by X the associated path according to Note 3.4. If $\mathfrak{b}_i$ is an uncle then we have:*

(1) $X_i < n_1 - 2$
(2) $X_i < X_{i-1}$

Proof This follows from Propositions 3.3 and 3.5. □

Definition 3.7 If $\omega = \mathrm{SS}w$ is an attack cycle starting with two blocks S, then we denote by $H(\omega)$ the random variable counting the number of blocks in the cycle ω fulfilling (1) and (2) from Proposition 3.6.

If w is an attack cycle, the condition $\omega = \mathrm{SS}\dots$ means that ω starts with two S.

Proposition 3.8 *We have:*

$$\mathbb{E}[H(\omega) | \omega = SS\dots] = \frac{p}{p-q}\left(1 - \left(\frac{q}{p}\right)^{n_1 - 1}\right)$$

Proof See Lemma 6.1 in the Appendix. □

3.2 Expected Number of Referred Uncles by Attack Cycle

We can be more precise in Proposition 3.6.

Lemma 3.9 *Let $\omega = SSw$ and X be the associated path from Note 3.4. We denote by $\mathfrak{b}_i$ the i-th block in w and suppose that conditions (1) and (2) from Proposition 3.6 are satisfied. The probability for $\mathfrak{b}_i$ to be an uncle is equal to γ, except when $\mathfrak{b}_i$ is the first block validated by the honest miners, then this probability is* 1.

Example 3.10 Suppose that $n_1 = 4$ and let $\omega = SSw$ with $w =$ SHSSSHHHH. The blocks validated by the honest miners correspond to an index $i \in E = \{2, 6, 7, 8, 9\}$. We have $X_6 = 2$ and $X_i < 2$ for $i \in E$ and $i \neq 6$. The first block validated by the honest miners is an uncle with probability 1. The second block validated by the honest miners is a stale block which cannot be referred by a nephew block. All other blocks validated by the honest miners in ω can be uncles with probability γ. Note also that the last three blocks of the honest miners are not referred in ω and will be referred by the first future official block of the next attack cycle.

Using these observations, we can now compute $\mathbb{E}[U]$.

Proposition 3.11 *We have:*

$$\mathbb{E}[U] = q + \frac{q^3\gamma}{p-q} - \frac{p^3}{p-q}\left(\frac{q}{p}\right)^{n_1+1}\gamma - q^{n_1+1}(1-\gamma)$$

Proof If $\omega = H$, then $U(\omega) = 0$. If $\omega \in \{\text{SHS}, \text{SHH}\}$, then, $U = 1$. Otherwise, ω starts with two consecutive S. Then, by Proposition 3.8 and Lemma 3.9, we have,

$$\mathbb{E}[U] = (0 \cdot p) + 1 \cdot (pq^2 + p^2q) + (\mathbb{E}[H(\omega)|\omega = \text{SS}\ldots]\gamma + (1-\gamma)(p + pq + \cdots + pq^{n_1-2})) \cdot q^2$$

The last term comes from the following fact: When the first honest block present in ω corresponds to an index i satisfying $X_i < n_1 + 2$, then its contribution to $\mathbb{E}[U]$ is underestimated by $\mathbb{E}[H(\omega)|\omega = \text{SS}\ldots]\gamma$ because it has probability 1 to be an uncle. This only occurs when ω starts with SS...SH with the first k blocks validated by the selfish miner with $k \leqslant n_1$, from where we get the last term. In conclusion we have:

$$\mathbb{E}[U] = pq + \left(\frac{p}{p-q}\left(1 - \left(\frac{q}{p}\right)^{n_1-1}\right)\gamma\right) \cdot q^2 + (1-\gamma)(1-q^{n_1-1}) \cdot q^2$$

and we get the result by rearranging this last equation. □

Note 3.12 In particular, we obtain $\lim_{n_1\to\infty} \mathbb{E}[U] = q + \frac{q^3\gamma}{p-q}$. This limit can also be derived by observing that if $n_1 = \infty$, then $\mathbb{E}[U|L = n] = 1 + \gamma(n-2)$ and using Theorem 2.1.

Now, we compute the expected number of uncles per attack cycle which are referred by nephews (honest or not) belonging to the next attack cycle.

Lemma 3.13 *The probability for an attack cycle to end with exactly k consecutive appearances of "H" with $k \geq 1$, conditional that it starts with SS, is pq^{k-1}.*

Proof Let $k \geq 1$. An attack cycle ω ends with exactly k consecutive appearances of "H" if and only if $\omega = \text{SSw}H$ where w is a Dyck word that ends with exactly $k-1$ "H". The result then follows from Appendix, Lemma 6.4. □

Proposition 3.14 *We have:*

$$\mathbb{E}[V] = \frac{q^2}{p}(1 - q^{n_1-1})\gamma + (1-\gamma)pq^2\,\frac{1-(pq)^{n_1-1}}{1-pq}$$

Proof If an attack cycle ω does not start with two S, then $V(\omega) = 0$. If ω starts with two "S" and ends with exactly k "H" in a row ($k \geq 1$), then only the last $n_1 - 1$ blocks can be uncles signaled by future blocks. This happens with probability γ for each block H in this sequence, except for the first block validated by the honest miners if it belongs to this sequence. In this last case, $\omega = \text{SS}\ldots\text{SH}\ldots\text{HH}$ with at most n_1 letters S and $n_1 - 1$ letters "H". So, by Lemma 3.13, we have

$$\mathbb{E}[V] = q^2 \sum_{k\geq 1} \inf(k, n_1 - 1)\, pq^{k-1}\gamma + (1-\gamma)q \sum_{k=1}^{n_1-1} (pq)^k$$

□

3.3 Expected Revenue of the Selfish Miner from Inclusion Rewards

We compute now $\mathbb{E}[U_h]$.

Proposition 3.15 *We have:*

$$\mathbb{E}[U_h] = p^2q + (p + (1-\gamma)p^2q)\left(\frac{q^2}{p}(1 - q^{n_1-1})\gamma + (1-\gamma)pq^2\,\frac{1-(pq)^{n_1-1}}{1-pq}\right)$$

Proof We have $U_h(\omega) = U_h^{(1)}(\omega) + U_h^{(2)}(\omega)$ where $U_h^{(1)}(\omega)$ (resp. $U_h^{(2)}(\omega)$) counts the number of uncles referred by honest nephews only present in ω (resp. in the next attack cycle after ω). It is clear that $U_h^{(1)}(\text{SHH}) = 1$ and $U_h^{(1)}(\omega) = 0$ if $\omega \neq \text{SHH}$. So,

$$\mathbb{E}[U_h^{(1)}] = p^2 q \tag{1}$$

Moreover, given ω, the probability that H is the next official block after ω is $p + (1-\gamma)p^2 q$. This happens if and only if the next attack cycle is either H or SHH. If this event occurs, then the first honest block in the next attack cycle will signal the previous uncles created in ω. Therefore, we have

$$\mathbb{E}[U_h^{(2)}] = (p + (1-\gamma)p^2 q) \cdot \mathbb{E}[V] \tag{2}$$

Hence, we get the result by (1), (2) and Proposition 3.14. □

Corollary 3.16 *We have*

$$\begin{aligned}\mathbb{E}[U_s] =& q + \frac{q^3\gamma}{p-q} - \frac{pq^2}{p-q}\left(\frac{q}{p}\right)^{n_1-1}\gamma - q^{n_1+1}(1-\gamma) \\ &- \left[p^2 q + (p + (1-\gamma)p^2 q)\left(\frac{q^2}{p}(1-q^{n_1-1})\gamma + (1-\gamma)pq^2\frac{1-(pq)^{n_1-1}}{1-pq}\right)\right]\end{aligned}$$

Proof With the same notations as above, we have: $U(\omega) = U_s(\omega) + U_h(\omega)$ and we use Propositions 3.11 and 3.15. □

3.4 Apparent Hashrate of Strategy 1

Using Theorem 2.11, Proposition 3.11 and Corollary 3.16 we can plot the region of $(q, \gamma) \in [0, 0.5] \times [0, 1]$ of dominance of the selfish mining Strategy 1 (SM1) over the honest strategy. This corresponds to $\widetilde{q}_E > q$. We obtain Fig. 1.

We want now to compute the expected revenue of the honest miners by attack cycle. In order to do that, we compute first the expected distance between uncles and nephews by attack cycle.

3.5 Expected Distance Between Uncles and Nephews by Attack Cycle

If $\mathfrak{b}$ is an uncle, we denote by $\delta(\mathfrak{b})$ the distance between $\mathfrak{b}$ and its nephew. We start by a remark.

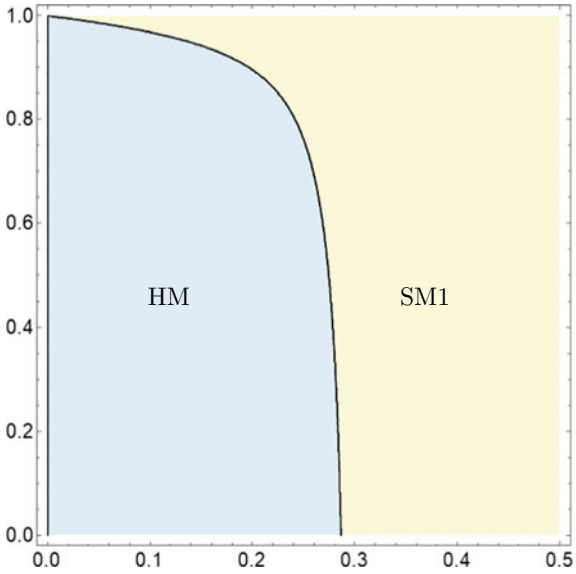


Fig. 1 Comparing HM and SM1 strategies

Remark 3.17 Let $\mathfrak{b}$ be an orphan block validated by the honest miners as in Definition 3.1. If $\mathfrak{b}$ is an uncle then $\delta(\mathfrak{b}) = h(\mathfrak{b}) + 1$.

Definition 3.18 If $\omega = \mathrm{SS}w$ is an attack cycle starting with two blocks S, we set

$$D(\omega) = \sum_{\mathfrak{b}} (h(\mathfrak{b}) + 1)$$

where the sum is taken over all honest blocks $\mathfrak{b}$ in ω fulfilling Conditions (1) and (2) from Proposition 3.6.

Proposition 3.19 *We have:*

$$\mathbb{E}[D(\omega)|\omega = SS\ldots] = \frac{p}{(p-q)^2}\left(2p - q - \big(p + n_1(p-q)\big)\cdot\left(\frac{q}{p}\right)^{n_1-1}\right)$$

Proof Let $\omega = \mathrm{SS}w$ be an attack cycle starting with two S with $w = w_1 \ldots w_\nu$ and let X be the associated path according to Note 3.4. In particular, we have $X_\nu = -1$ and $X_i \geq 0$ for $i < \nu$. By Proposition 3.5 and Lemma 6.1 in the Appendix, we have:

$$\begin{aligned}
\mathbb{E}[D(\omega)|\omega = \mathrm{SS}\ldots] &= \mathbb{E}\left[\sum_{i=1}^{\nu}(X_i+3)\cdot\mathbf{1}_{(X_i<n_1-2)\wedge(X_i<X_{i-1})}\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{\nu} X_i\cdot\mathbf{1}_{(X_i<n_1-2)\wedge(X_i<X_{i-1})}\right] + 3\mathbb{E}\left[\sum_{i=1}^{\nu}\cdot\mathbf{1}_{(X_i<n_1-2)\wedge(X_i<X_{i-1})}\right] \\
&= \frac{p}{(p-q)^2}\left(2q - p - \big(q + (n_1-2)(p-q)\big)\cdot\left(\frac{q}{p}\right)^{n_1-1}\right) + \frac{3p}{p-q}\left(1 - \left(\frac{q}{p}\right)^{n_1-1}\right)
\end{aligned}$$

$$= \frac{p}{(p-q)^2}\left(2q - p + 3(p-q) - \left(q - 2(p-q) + n_1(p-q) + 3(p-q)\right)\cdot\left(\frac{q}{p}\right)^{n_1-1}\right)$$

Hence, we get the result. □

Definition 3.20 Let ω be an attack cycle. We set

$$\Delta(\omega) = \sum_{\mathfrak{b}} \delta(\mathfrak{b})$$

The last sum being taken over all referred uncles in ω.

Proposition 3.21 *We have*

$$\begin{aligned}\mathbb{E}[\Delta] =& pq + \frac{pq^2\gamma}{(p-q)^2}\left(2p - q - \left(p + n_1(p-q)\right)\cdot\left(\frac{q}{p}\right)^{n_1-1}\right)\\ &+ \frac{(1-\gamma)q}{p}\left(q(1+p) - (1+n_1p)q^{n_1}\right)\end{aligned}$$

Proof We proceed as in the proof of Proposition 3.11. If $\omega = H$, then $\Delta(\omega) = 0$. If $\omega \in \{\text{SHS}, \text{SHH}\}$, then, $\Delta(\omega) = 1$. Otherwise, ω starts with two consecutive S. Then, using Lemma 3.9, we get

$$\mathbb{E}[\Delta] = pq^2 + p^2q + (\mathbb{E}[D(\omega)|\omega = \text{SS}\ldots]\gamma + (1-\gamma)(2p + 3pq + \cdots + n_1pq^{n_1-2}))\cdot q^2$$

The last term comes from the following fact: when the first honest block present in ω corresponds to an index i satisfying $X_i < n_1 + 2$, then its contribution to $\mathbb{E}[\Delta]$ is underestimated by $\mathbb{E}[D(\omega)|\omega = \text{SS}\ldots]\gamma$ because it has probability 1 to be an uncle. This only occurs when ω starts with SS...SH with the first k blocks validated by the selfish miner with $k \leqslant n_1$, from where we get the last term. We have:

$$\begin{aligned}2q + 3q^2 + \cdots + n_1q^{n_1-1} &= -1 + \left(\frac{q^{n_1+1}-1}{q-1}\right)' = -1 + \left(\frac{q^{n_1+1}}{q-1}\right)' - \left(\frac{1}{q-1}\right)'\\ &= -1 + \frac{(n_1+1)q^{n_1}}{q-1} - \frac{q^{n_1+1}}{(q-1)^2} + \frac{1}{(q-1)^2}\\ &= -1 + \frac{1}{p^2} + \frac{q^{n_1}}{(q-1)^2}\left((n_1+1)(q-1) - q\right)\\ &= \frac{1-p^2}{p^2} - \frac{q^{n_1}}{p^2}\left(q + (n_1+1)p\right)\\ &= \frac{q(1+p)}{p^2} - \frac{q^{n_1}}{p^2}\left(1 + n_1p\right)\end{aligned}$$

So,

$$\left(2p + 3pq + \cdots + n_1 pq^{n_1-2}\right) q^2 = \frac{q}{p}\left(q(1+p) - \left(1 + n_1 p\right) q^{n_1}\right) \tag{3}$$

Hence we get the result using Proposition 3.19. □

3.6 Deflation

With the new difficulty adjustment formula, the duration time of an attack cycle in Ethereum is $(\mathbb{E}[L] + \mathbb{E}[U])\tau_1$ where τ_1 is the mean interblock time in Ethereum (which is currently 15 seconds). The number of coins created in an attack cycle is $\left(\mathbb{E}[L] + \frac{7}{8}\mathbb{E}[U] - \frac{1}{8}\mathbb{E}[\Delta] + \mathbb{E}[U]\pi\right) b$ where b is the coinbase in Ethereum. Thus, on average, there is a monetary creation of

$$\frac{\mathbb{E}[L] + \left(\frac{7}{8} + \pi\right)\mathbb{E}[U] - \frac{\mathbb{E}[\Delta]}{8}}{\mathbb{E}[L] + \mathbb{E}[U]}\, b$$

for every inter-block time τ_1, whereas without selfish miner, it is only b on average. So, selfish mining leads to a deflation index

$$\iota = \frac{\left(\frac{1}{8} - \pi\right)\mathbb{E}[U] + \frac{\mathbb{E}[\Delta]}{8}}{\mathbb{E}[L] + \mathbb{E}[U]} \tag{4}$$

Currently we have $\pi = \frac{1}{32}$, thus $\iota > 0$.

3.7 Apparent Hashrate of the Honest Miners

Let $\tilde{p}$ be the apparent hashrate of the honest miners in presence of a selfish miner. We have

$$\tilde{p} + \tilde{q} = 1 - \iota \tag{5}$$

where $\tilde{q}$ is the apparent hashrate of the selfish miner. We observe numerically that $\tilde{q} > q - \iota$ for any values of (q, γ). So, even if the attack is not profitable for the selfish miner (case $\tilde{q} < q$) we have $\tilde{p} < p$ which means that the honest miners are impacted by the presence of a selfish miner in the network.

4 Strategy 2A: Brutal Fork Signaling All Uncles

We study now another Selfish Mining Strategy (Strategy 2 or SM2): Brutal fork. In this case, the attacker keeps secret his blocks as long as possible and only releases its fork, all at once, at the end of the attack cycle. We call this strategy "brutal fork" because this leads, periodically, to deep reorganizations of the official blockchain. Strategy 2A (or SM2A) corresponds to the case when also the attacker refers all possible uncles.

Proposition 4.1 *We have* $\mathbb{E}[U] = q - q^{n_1+1}$.

Proof We have $U = 0$ if and only if the attack cycle is H or if it starts with $n_1 + 1$ blocks of type S. Otherwise, we have $U = 1$. So,

$$\mathbb{E}[U] = \mathbb{P}[U > 0] = 1 - (p + q^{n_1+1}) = q - q^{n_1+1}$$

□

We compute now $\mathbb{E}[V]$

Proposition 4.2 *We have* $\mathbb{E}[V] = pq^2 \cdot \frac{1-(pq)^{n_1-1}}{1-pq}$.

Proof We have $V = 1$ if and only if the attack cycle ω is SS..SH..H with $2 \leq k \leq n_1$ S. In that case, the first H is an uncle signaled by the first future official block in the attack cycle after ω. Otherwise, $V = 0$. So, $\mathbb{E}[V] = pq^2 + \cdots + p^{n_1-1}q^{n_1}$, and we get the result. □

Proposition 4.3 *We have* $\mathbb{E}[U_h] = p^2q + \left(p + (1-\gamma)p^2q\right) pq^2 \cdot \frac{1-(pq)^{n_1-1}}{1-pq}$.

Proof The proof is almost identical as the proof of Proposition 3.15. If $U_h^{(1)}(\omega)$ (resp. $U_h^{(2)}(\omega)$) counts for the number of uncles referred by honest nephews only present in ω (resp. in the attack cycle just after ω), then we have $\mathbb{E}[U_h^{(1)}] = p^2q$, $\mathbb{E}[U_h^{(2)}] = \left(p + (1-\gamma)p^2q\right) \cdot \mathbb{E}[V]$ and $U_h = U_h^{(1)} + U_h^{(2)}$. The only difference is the value of $\mathbb{E}[V]$ which this time is given by Proposition 4.2, and we get the result. □

Corollary 4.4 *We have*

$$\mathbb{E}[U_s] = \mathbb{E}[U] - \mathbb{E}[U_h] = q - q^{n_1+1} - \left(p^2q + \left(p + (1-\gamma)p^2q\right) pq^2 \cdot \frac{1-(pq)^{n_1-1}}{1-pq}\right)$$

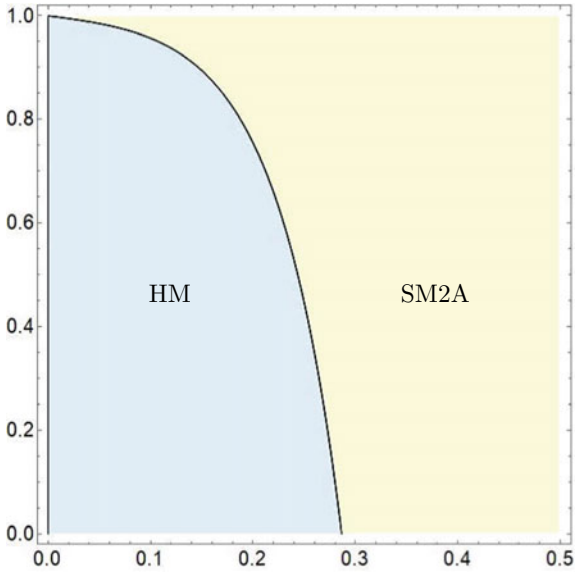


Fig. 2 Comparing HM and SM2A strategies

Note 4.5 When $\gamma = 0$, the two strategies 1 and 2 A are identical: in both cases, the honest miners always build blocks on top of honest blocks. So, $\mathbb{E}[U]$, $\mathbb{E}[U_h]$ and $\mathbb{E}[U_s]$ must coincide for $\gamma = 0$. We can check in the different formulas that this is the case. See Propositions 3.11, 3.15,4.1, 4.3 and Corollaries 3.16, 4.4.

4.1 Apparent Hashrate of Strategy 2A

We use again Theorem 2.11 and we plot in parameter space in Fig. 2 the region of $(q, \gamma) \in [0, 0.5] \times [0, 1]$ comparing Selfish Mining Strategy 2 A to the honest strategy.

We observe that if $\gamma = 0$ then we have SM2A is superior to honest mining when $q > 28.65\%$. Also we have for all values of q and γ that SM2A is superior to SM1. Therefore it is never profitable for the attacker to engage in competitions with the honest miners.

4.2 Apparent Hashrate of the Honest Miners

We compute first the expected distance between an uncle and its nephew. We keep the same notation for Δ as in Definition 3.20.

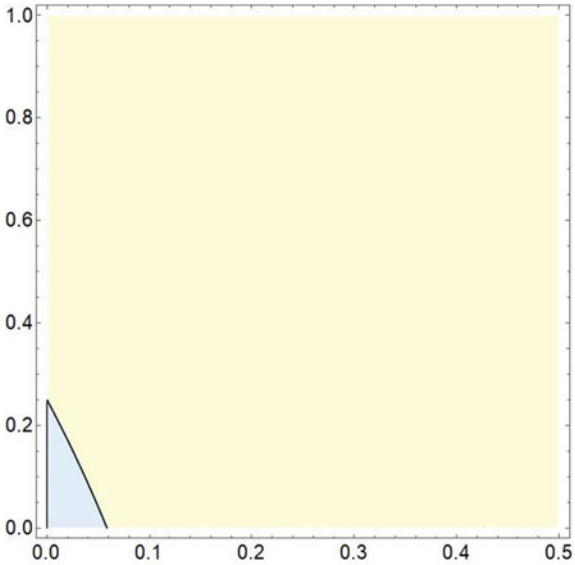


Fig. 3 Comparing $\tilde{p}$ and p: The honest miners loose money even when the attack is not profitable for the selfish miner except for a tiny region around (0, 0) (case SM2A)

Proposition 4.6 *We have* $\mathbb{E}[\Delta] = \frac{q}{p}\left(q(1+p) - \left(1+n_1 p\right)q^{n_1}\right)$

Proof If an attack cycle ω starts with S...SH with k S, $k \leq n_1$, then there is exactly one uncle in ω and its distance to its nephew is k. In any other cases, there is no uncle in ω. Therefore, $\mathbb{E}[\Delta] = \sum_{k=1}^{n_1} kpq^k$ Hence we get the result by (3). □

The apparent hashrate $\tilde{p}$ of the honest miners is $\tilde{p} = 1 - \tilde{q} - \iota$ with ι given by (4). Numerically, we observe that we have always $\tilde{p} < p$ except in a tiny region when q and γ is small ($q < 6\%$ and $\gamma < 22\%$).

5 Strategy 2B: Brutal Fork Without Signaling Uncles

In this strategy, the attacker signals no uncles in order to maximize the impact on the difficulty adjustment formula. In that case we have $U_s = 0$. In our analysis of the profitability of the strategy, we need to consider another important rule of Ethereum's protocol: a nephew can only signal at most two uncles. Instead of computing $\mathbb{E}[U]$, it is simpler to compute $\mathbb{E}[U']$ where $U'(\omega)$ is defined as the number of signaled uncles with nephews in ω. We have,

$$\mathbb{E}[U] = \mathbb{E}[U'] \tag{6}$$

Since the attacker does not signal uncles, we have $U'(\omega) = 0$ if $\omega \notin \{H, \text{SHH}\}$.

To ease notations, we set $U'(H)$ for $U'(\{H\})$.

Lemma 5.1 *We have:*

$$\begin{aligned}&\mathbb{P}[U'(H)=1]=\sum_{i=2}^{n_1-2}\left(1-pq^2-pq^2C_{n_1-2-i}(pq)\right)\pi_i+\pi_{n_1-1}+\pi_{n_1}\\&\mathbb{P}[U'(H)=2]=\sum_{i+j\leq n_1}\pi_i\pi_j\\&\mathbb{P}[U'(H)\geq 3]=0\end{aligned}$$

Proof We have $U'(H)=1$ if and only if the two last attack cycles before H are in the following order from the oldest to the most recent one: ω' and ω such that:

(1) ω won by the attacker with $L(\omega)\leqslant n_1$.
(2) ω' won by the honest miners or by the attacker but with $L(\omega')>n_1-L(\omega)$.

Note that if $L(\omega)\geqslant n_1-1$ then (2) is automatically satisfied. So,

$$\mathbb{P}[U'(H)=1]=\sum_{i=2}^{n_1-2}\left(1-pq^2-pq^2C_{n_1-2-i}(pq)\right)\pi_i+\pi_{n_1-1}+\pi_{n_1}$$

In the same way, we have $U'(H)=2$ if and only if the two last attack cycles before H are ω' and ω such that ω' and ω are both won by the attacker with $L(\omega)+L(\omega')\leq n_1$. Indeed, a block can only refer at most two uncles. Hence, we get the result. □

Example 5.2 For $n_1=6$, we have using Example 2.5:

$$\begin{aligned}\mathbb{P}[U'(H)=1]&=\pi_5+\pi_6+\sum_{i\leqslant 4}(1-pq^2-pq^2C_{4-i}(pq))\pi_i\\&=pq^2\left(14p^4q^4+p^3(5-9q)q^3+2p^2(1-2q)q^2+p\left(q-4q^2\right)+2\right)\\\mathbb{P}[U'(H)=2]&=\pi_2^2+2\pi_2\pi_3+2\pi_2\pi_4+\pi_3^3\\&=p^2q^4\left(5p^2q^2+2pq+4\right)\\\mathbb{P}[U'(H)\geq 3]&=0\end{aligned}$$

Definition 5.3 We define $P_{n_1}(p,q)=\mathbb{E}[U'(H)]$.

Example 5.4 When $n_1=6$, we have by Example 5.2:

$$P_6(p,q)=pq^2\left(14p^4q^4+p^3(q+5)q^3+2p^2q^2+p(4q+1)q+2\right)$$

Lemma 5.5 *We have*

$$\mathbb{E}[U'(\omega)|\omega=SHH]=(P_{n_1}(p,q)+1)\cdot(1-\gamma)+(pq^2+pq^2C_{n_1-3}(pq)+1)\cdot\gamma$$

Proof Suppose that $\omega =$SHH. We have two cases: The second honest block can be built on top of a block validated by the selfish miner or not. If the first official block of ω is honest, then it signals any uncle which is at distance less or equal than n_1, like in the previous situation. Moreover, the first block mined by the selfish miner is an uncle signaled by the second block mined by the honest miners. This gives the first term of the right hand side. If the first official block of ω is a block mined by the attacker, then the first block validated by the honest miners is an uncle signaled by the second block mined by the honest miners. This last block will also signal another uncle which is at distance less than $n_1 - 1$ of the first official block of ω. There is such an uncle if and only if the attack cycle ω' before ω is an attack cycle won by the attacker with $L(\omega') \leq n_1 - 1$. This gives the second term of the right hand side. Hence, we get the result. □

Theorem 5.6 *We have*

$$\mathbb{E}[U] = (p + (1-\gamma)p^2q)P_{n_1}(p,q) + \gamma p^2 q\left(pq^2 + pq^2 C_{n_1-3}(pq)\right) + p^2 q$$

Proof We have $\mathbb{E}[U] = \mathbb{E}[U']$ and

$$\begin{aligned}\mathbb{E}[U'] &= \mathbb{E}[U'(\omega)|\omega = H]\mathbb{P}[\omega = H] + \mathbb{E}[U'(\omega)|\omega = \text{SHH}]\mathbb{P}[\omega = \text{SHH}] \\ &= P_{n_1}(p,q)p + (P_{n_1}(p,q)+1)\cdot(1-\gamma)p^2q + (pq^2 + pq^2 C_{n_1-3}(pq)+1)\cdot\gamma p^2 q\end{aligned}$$

□

5.1 Apparent Hashrate of Strategy 2B

The computation of $\mathbb{E}[U]$ is a polynomial expression in p and q that can be carried out with the help of a computer algebra system. We plot in parameter space in Fig. 3 the region of $(q, \gamma) \in [0, 0.5] \times [0, 1]$ comparing Selfish Mining Strategies 2 A and 2B, and honest mining. We also compare SM1, SM2A and SM2B in Fig. 4.

We observe that if $\gamma = 0$ then we have SM2B is superior to honest mining when $q > 28.80\%$. Also, for $q > 30.13\%$ we have that SM2B is even better than SM2A (whatever γ is). Thus, in this case, the attacker does not even need to bother to signal blocks (Figs. 5).

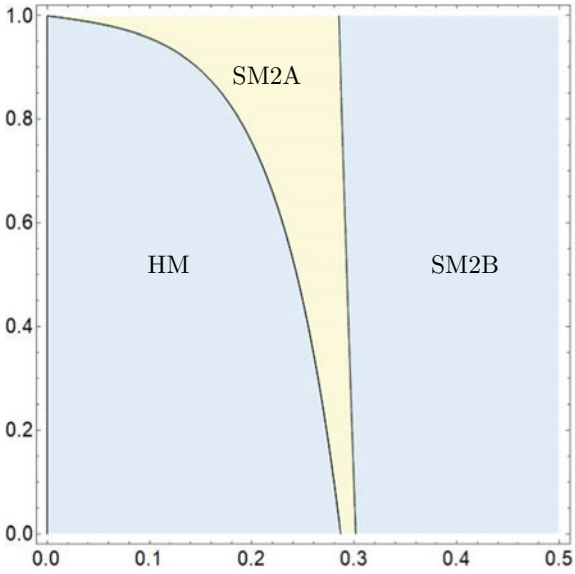


Fig. 4 Comparing the strategies HM, SM2A and SM2B

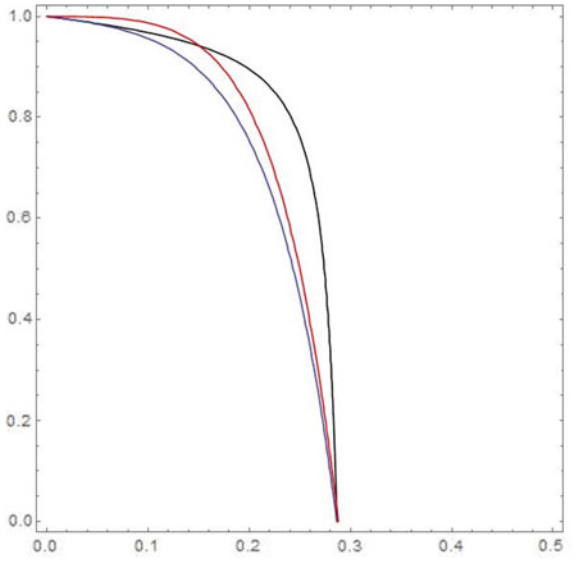


Fig. 5 Comparing the strategies SM1 (black), SM2A (blue) and SM2B (red)

5.2 Apparent Hashrate of the Honest Miners

Definition 5.7 If ω is an attack cycle, we denote by $\Delta'(\omega)$ the average number of the distance between a nephew belonging to ω and an uncle which does not necessarily belong to ω.

In a similar way as before, we prove:

Lemma 5.8 *We have*

$$\mathbb{E}[\Delta'(\omega)|\omega = H] = \sum_{|\mathbf{i}|\leqslant n_1} \left(\sum_j j \cdot i_j \right) \left(1 - pq^2 - pq^2 C_{n_1-2-|\mathbf{i}|}(pq)\right) \prod_j \pi_{i_j}$$

Definition 5.9 We define $Q_{n_1}(p, q) = \mathbb{E}[\Delta'(\omega)|\omega = H]$

The same computations as in the previous section leads to

$$Q_5(p, q) = pq^2 \left(25p^3q^3 + 20p^2q^3 + 8p^2q^2 + 16pq^2 + 3pq + 4\right)$$
$$Q_6(p, q) = pq^2 \left(84p^4q^4 + 54p^3q^4 + 25p^3q^3 + 96p^2q^4 + 20p^2q^3 + 8p^2q^2 + 16pq^2 + 3pq + 4\right)$$

This enables us to compute $\mathbb{E}[\Delta']$ using the following result with $n_1 = 6$.

$$\mathbb{E}[\Delta'] = (p + (1 - \gamma)p^2q)Q_{n_1}(p, q) + \gamma p^2 q Q_{n_1-1}(p, q) + p^2q \ .$$

Finally, we note that $\mathbb{E}[\Delta] = \mathbb{E}[\Delta']$. From here, we get the apparent hashrate of the honest miners using (4) and (5). We observe numerically that we have always $\tilde{p} < p$.

6 Conclusions

We have given closed-form formulas for the long term profitability of different selfish mining strategies in the Ethereum network. This is combinatorially more complex than in Bitcoin network which has a simpler reward system. Precisely, the particular reward system that incentives signaling blocks is an effective counter-measure to Selfish mining but only when the count of uncle blocks are incorporated into the difficulty adjustment formula (this is the case for the current implementation of the difficulty adjustment formula). This analysis provides a good illustration of the fact that selfish mining is an attack on the difficulty adjustment formula. We study, for the first time, selfish mining strategies that do not signal any blocks. We prove that they are the most profitable ones in the long run. It may appear counter-intuitive that refusing the signaling fees is the most profitable strategy with the current reward parameters when q is larger than 30%. But this is explained again because selfish mining is an attack on the difficulty adjustment formula.

Appendix

6.1 Random Walk

We compute the expected numbers of descents in a biased random walk conditional to be bounded by a fixed bound.

Lemma 6.1 *Let (X_k) be a biased random walk starting from $X_0 = 0$ with $\mathbb{P}[X_{k+1} = X_k + 1] = q$ and $\mathbb{P}[X_{k+1} = X_k - 1] = p$ for $k \in \mathbb{N}$, with $p + q = 1$ and $q < p$. Let $\nu(X)$ be the stopping time defined by $\nu(X) = \inf\{i \geq 0;\, X_i = -1\}$, and for $n \geq 0$, let*

$$u_n(X) = \sum_{i=1}^{\nu} \mathbf{1}_{(X_i<n)\wedge(X_i<X_{i-1})}$$
$$v_n(X) = \sum_{i=1}^{\nu} X_i \cdot \mathbf{1}_{(X_i<n)\wedge(X_i<X_{i-1})}$$

Then we have

$$u_n = \mathbb{E}[u_n(X)] = \frac{p}{p-q}\left(1 - \left(\frac{q}{p}\right)^{n+1}\right) \tag{7}$$

$$v_n = \mathbb{E}[v_n(X)] = \frac{p}{(p-q)^2}\left(2q - p - \big(q + n(p-q)\big) \cdot \left(\frac{q}{p}\right)^{n+1}\right) \tag{8}$$

Proof We have $u_0 = 1$ (resp. $v_0 = -1$). If $X_1 = -1$, then we have $u_n(X) = 1$ (resp. $v_n(X) = -1$). If $X_1 = 1$, then

$$\begin{aligned} u_n(X) =& \sum_{i=1}^{\nu'} \mathbf{1}_{(X'_i<n-1)\wedge(X'_i<X'_{i-1})} + \sum_{i=1}^{\nu''} \mathbf{1}_{(X''_i<n)\wedge(X''_i<X''_{i-1})} \\ =& u_{n-1}(X') + u_n(X'') \end{aligned}$$

with

$$\begin{aligned} X'_i =& X_{i+1} - 1 \\ \nu' =& \inf\{i > 0;\, X'_i = -1\} \\ X''_i =& X'_{i+\nu'} - X'_{\nu'} \\ \nu'' =& \inf\{i > 0;\, X''_i = -1\} \end{aligned}$$

By the Markov property, X' and X'' are two independent simple biased random walk with a probability p (resp. q) to move to the left (resp. right). So, taking expectations, we get:

$$u_n = p \cdot 1 + q \cdot (u_{n-1} + u_n)$$

which is equivalent to

$$u_n - \frac{p}{p-q} = \left(\frac{q}{p}\right)\left(u_{n-1} - \frac{p}{p-q}\right)$$

So we get (7) by induction on n. In the same way, we have:

$$\begin{aligned} v_n(X) &= \sum_{i=1}^{\nu'} (X'_i + 1) \cdot \mathbf{1}_{(X'_i < n-1) \wedge (X'_i < X'_{i-1})} + \sum_{i=1}^{\nu''} X''_i \cdot \mathbf{1}_{(X''_i < n) \wedge (X''_i < X''_{i-1})} \\ &= u_{n-1}(X') + v_{n-1}(X') + v_n(X'') \end{aligned}$$

Taking expectations again, we get

$$v_n = p \cdot (-1) + q \cdot (u_{n-1} + v_{n-1} + v_n) \tag{9}$$

Set $c_n = \left(\frac{p}{q}\right)^n v_n$. Then, (9) leads to

$$\begin{aligned} c_n &= c_{n-1} + \left(\frac{p}{q}\right)^{n-1} u_{n-1} - \left(\frac{p}{q}\right)^n \\ &= c_{n-1} + \left(\frac{2q-p}{p-q}\right) \cdot \left(\frac{p}{q}\right)^n - \frac{q}{p-q} \end{aligned}$$

So, by induction, we get

$$c_n = c_0 + \left(\frac{2q-p}{p-q}\right) \cdot \left(\frac{p}{q}\right) \cdot \frac{\left(\frac{p}{q}\right)^n - 1}{\left(\frac{p}{q}\right) - 1} - \frac{nq}{p-q}$$

After rearranging terms, we get (8). □

6.2 *Dyck Words*

Let $\mathcal{D}$ be the space of Dyck words based on the alphabet $\{S, H\}$. If $w = w_1 \dots w_{2k}$ with $k \in \mathbb{N}$, then we define $|w| = k$. We have proved in [4] that we can endow $\mathcal{D}$ with a probability measure $\bar{\mathbb{P}}$ given by $\bar{\mathbb{P}}[w] = p(pq)^{|w|}$ for $w \in \mathcal{D}$. Note that $\bar{\mathbb{P}}[w]$

can be interpreted as the probability that a simple biased random walk X starting from 0 and stopping at -1 follows exactly the path given by w i.e., $X_i = X_{i-1} + 1$ (resp. $X_i = X_{i-1} - 1$) if $w_i = S$ (resp. $w_i = H$) for $i \leqslant 2|w|$ and $X_{2|w|+1} = -1$.

Lemma 6.2 *Let $n \geq 0$ and $\mathcal{D}_n = \{w; |w| \leqslant n\}$. Then, $\bar{\mathbb{P}}[\mathcal{D}_n] = pC_n(pq)$ where $C_n(x)$ is the n-th partial sum of the generating series $C(x)$ of the Catalan numbers.*

Proof We have

$$\bar{\mathbb{P}}[\mathcal{D}_n] = \sum_{w \in \mathcal{D}_n} p(pq)^{|w|} = p \sum_{k=0}^{n} \sum_{|w|=k} (pq)^k = p \sum_{k=0}^{n} C_k (pq)^k = pC_n(pq)$$

□

We can make more precise Proposition 2.3.

Proposition 6.3 *Let $\omega = SSwH$ be an attack cycle starting with SS. Then, $w \in \mathcal{D}$ and $\mathbb{P}[\omega] = q^2 \bar{\mathbb{P}}[w]$*

Lemma 6.4 *The probability that a Dyck word ends with the subsequence SHH..H with n letters H at the end is pq^n.*

Proof Consider the "reversal" map $\sigma : \quad \mathcal{D} \longrightarrow \mathcal{D}$ given by

$$w = w_1 \dots w_{2|w|} \longmapsto \sigma(w) = \tilde{w} = \tilde{w}_{2|w|} \dots \tilde{w}_1$$

with $\tilde{w}_i = S$ (resp. H) if $w_i = H$ (resp. S). Then σ is one to one and preserves $\bar{\mathbb{P}}$ i.e., for $w \in \mathcal{D}$, we have $\bar{\mathbb{P}}[\sigma(w)] = \bar{\mathbb{P}}[w]$. So, the probability that a Dyck word ends exactly with n letter(s) H is the same as the probability that a Dyck word starts with n letter(s) S and then is followed by a letter H. Thus this probability is pq^n. □

For $w \in \mathcal{D}$, we define $f(w) = \inf\{i \geq 0; w_i = H\}$.

Lemma 6.5 *Let $n \geq 0$ and $E = \{w \in \mathcal{D}; f(w) \leqslant \inf\{|w|, n\}\}$. Then we have*

$$\bar{\mathbb{P}}[E] = (1 - q^n) - \frac{p(1 - (pq)^n)}{1 - pq}$$

Proof Let $w \in \mathcal{D}$. To have $f(w) \leqslant |w|$ means that at least one H is followed by an S i.e., w is not of the form SS...SHH...H. For all integer $k \leqslant n$, we have

$$\Sigma_{w;(f(w)=k)\wedge(f(w)\leqslant|w|)}(pq)^{|w|} = pq^{k-1}\cdot\sum_{j=0}^{k-2}qp^j$$

So, if we consider a biased random walk starting from 0 with a probability p to move to the left (resp. right) then both terms represent the probability of the following event: We have $k-1$ first step(s) to the right, then $j+1$ steps to the left with $0 \leqslant j \leqslant k-2$ and then at least one step to the right before reaching 0. So, we have

$$\begin{aligned}\bar{\mathbb{P}}[E] &= \sum_{k=1}^{n} pq^{k-1}\cdot\sum_{j=0}^{k-2}qp^j \\ &= p\sum_{k=1}^{n} q^{k-1}\cdot(1-p^{k-1}) \\ &= p\sum_{k=1}^{n} q^{k-1} - p\sum_{k=1}^{n}(pq)^{k-1} \\ &= (1-q^n) - \frac{p(1-(pq)^n)}{1-pq}\end{aligned}$$

□

6.3 Glossary

6.3.1 Revenue Ratio and Apparent Hashrate

The revenue ratio $\tilde{\Gamma}$ of a miner following a strategy with repetitions of attack cycles like selfish mining is given by $\tilde{\Gamma} = \frac{\mathbb{E}[R]}{\mathbb{E}[T]}$ where R (resp. T) is the revenue of the miner after an attack cycle (resp. the duration time of an attack cycle). The apparent hashrate $\tilde{q}$ is defined by $\tilde{q} = \tilde{\Gamma}\frac{\tau}{b}$ where b (resp. τ) is the coinbase (resp. interblock time).

6.3.2 Terminology

Ethereum has a special terminology that we summarize here.

Uncle	Orphan block whose parent belongs to the official blockchain
Nephew	Regular block that refers to an "uncle" which is at a distance less than n_1
Distance	Number of official blocks between a nephew N and a parent's uncle U

6.3.3 Mining Reward

If an uncle U is referred by a nephew N which is at a distance d, then U earns an "uncle reward" which is worth $K_u(d)b$ and N gets an additional reward of $K_n(d)b$, where b is the coinbase. Also, a nephew can refer at most two uncles. Today, on Ethereum, we have $b = 2$ ETH, $K_u(d) = \frac{8-d}{8} \cdot \mathbf{1}_{d \leqslant n_1}$ with $n_1 = 6$ and $K_n(d) = \pi = \frac{1}{32}$.

Uncle reward	Reward granted to an uncle block referred by a nephew
inclusion reward	Additional reward granted to a nephew that refers an uncle

References

1. Feng, C. & Niu, J. (2019). *Selfish mining in ethereum*, arXiv:1901.04620.
2. Grunspan, C. & Pérez-Marco, R. (2018). *On profitability of selfish mining*, arXiv:1805.08281v2.
3. Grunspan, C. & Pérez-Marco, R. (2018). *On profitability of trailing mining*, arXiv:1811.09322.
4. Grunspan, C. & Pérez-Marco, R. (2019). *Bitcoin selfish mining and Dyck words*, arXiv:1902.01513.
5. Grunspan, C. & Pérez-Marco, R. (2019). Selfish mining and Dyck words in Bitcoin and Ethereum networks. In *Tokenomics Conference Proceedings*, arXiv:1904.07675.
6. Sirer, E. G. & Eyal, I. (2014). Majority is not enough: bitcoin mining is vulnerable. *International Conference on Financial Cryptography and Data Security* (pp. 436–454).
7. Zugenmaier, A., & Ritz, F. (2018). The impact of uncle rewards on selfish mining in ethereum. *IEEE Symposium on Security and Privacy* (pp. 50–57).

The Speculative (In)Efficiency of the CME Bitcoin Futures Market

Toshiko Matsui and Lewis Gudgeon

Abstract The launch of Bitcoin futures on the Chicago Board Options Exchange (CBOE) and the Chicago Mercantile Exchange (CME) in December 2017 marked a notable milestone in the development of cryptoassets. Yet while the speculative efficiency of commodity markets has been extensively investigated, relatively little analysis has been undertaken on the speculative efficiency of Bitcoin markets. In this paper we investigate the speculative efficiency of the Bitcoin market, leveraging an approach based on non-overlapping data samples, which has been previously employed to the same end in the context of the London Metal Exchange (LME). Using non-overlapping data on Bitcoin spot and futures prices as traded on the CME, we find that the 1-month futures price is *not* an unbiased predictor of the spot price, suggesting that the market is inefficient: it may be possible for a speculator to make excess returns. In contrast, with 2-week and 1-week futures we are unable to reject the null hypothesis of market efficiency. Moreover, we find that the futures price becomes a more accurate indicator of the spot price as the futures contract becomes shorter.

Keywords Bitcoin · Cryptoassets · Futures markets · Market efficiency · Speculative efficiency hypothesis (SEH) · Derivatives · Chicago Mercantile Exchange (CME) · London Metal Exchange (LME) · OLS

1 Introduction

Since the creation of Bitcoin [25] thousands of cryptoassets have come to fruition. Bitcoin itself has received considerable attention from policymakers, regulators as

T. Matsui (✉) · L. Gudgeon
Department of Computing, Imperial College London, London, UK
e-mail: t.matsui19@imperial.ac.uk

L. Gudgeon
e-mail: l.gudgeon18@imperial.ac.uk

P. Pardalos et al. (eds.), *Mathematical Research for Blockchain Economy*, Springer Proceedings in Business and Economics,
https://doi.org/10.1007/978-3-030-53356-4_6

well as investors [17]. It has also been suggested that Bitcoin should be regarded as a speculative asset class for investment purposes, rather than a currency [2, 28].

One of the main issues of interest to regulators as well as market participants has been whether the price of bitcoin is predictable, contradicting the efficient market hypothesis (EMH) [13]. The research that does exist on the pricing efficiency of bitcoin has found that it exhibits weak form efficiency [10, 23, 29, 30]—investors cannot use past information to predict future returns—corroborating findings on price behaviour idiosyncratic to bitcoin, such as price clustering [31, 32] and pricing bubbles [12]. Urquhart [30] shows that bitcoin spot price returns are significantly inefficient over the full daily closing price sample from 1st August 2010 to 31st July 2016, but became more efficient between 1st August 2013 and 31st July 2016. As a follow up study of Urquhart [30], Nadaraj and Chu [24] utilize eight different tests and show whether a simple power transformation of the bitcoin returns satisfies the weak form EMH. Bariviera [3] studies the long-range dependence of bitcoin price returns and volatility, by utilizing a Detrended Fluctuation Analysis (DFA) method rather than the commonly used Rescaled Range Series (R/S) method to show variations in informational efficiency in daily price returns and long memory processes in daily volatility. However, by using daily data from 2010 to 2017 and performing a Hurst exponent analysis, Jiang et al. [18] find that while the Bitcoin market exhibits long-term memory (as was suggested by Bariviera [3]), a high degree of inefficiency is observed and the bitcoin market has not become efficient over time. Cobert [10] leverages the work of Brock et al. [8] to investigate whether using a moving average-oscillator in combination with a trading range break-out strategy on high frequency data is profitable for the bitcoin spot market. Al-Yahyaee et al. [1] compares the efficiency of the bitcoin market with other markets including those for gold, stock and currency by using the multifractal detrended fluctuation analysis (MF-DFA) approach. They find evidence of long-memory and multifractality for all four markets and that the bitcoin market is the most inefficient market of the four. Further, Köchling [21] recently demonstrated that the introduction of futures contracts for bitcoin has contributed to the market efficiency of bitcoin markets.

Historically, the EMH has been widely tested on stock, equity, currency markets [16], commodity markets [5] and their derivatives. The commodity futures market is mainly comprised of spot and futures prices with different maturities, and the prices in the two markets exemplify the main measures of market efficiency. Noting that futures prices reflect the expectation of market participants, and hence, assuming the futures price at time t for a contract with maturity length n is an unbiased predictor of the spot price as long as the hypothesis holds in the market at time $t + n$, one way to test the EMH is to examine whether the futures price $F_{t,n}$ is an unbiased estimator of the future spot price S_{t+n} [16]. If the EMH is true, the forecast error $\epsilon_{t,n} = S_{t+n} - F_{t,n}$ has a zero mean and is serially uncorrelated. In the most-cited research, Canarella and Pollard [9] applied an ordinary least squares (OLS) model to London Metal Exchange (LME) non-overlapping data, and an autoregressive moving average (ARMA) process to the error terms in overlapping data to test the speculative efficiency hypothesis (SEH). For both non-overlapping and overlapping data, they concluded that they were unable to reject a null hypothesis that the SEH holds,

suggesting that futures prices are unbiased estimators of future spot prices. Yet there is no consensus: Canarella and Pollard [9], Gross [14], and MacDonald and Taylor [22] concluded the LME to be efficient, while Kenourgios and Samitas [20], Otto [26], Park and Lim [27] concluded that the LME was inefficient.

While the speculative efficiency of the bitcoin market has been investigated by a number of approaches, an approach which utilizes spot and futures prices to test the speculative efficiency has received relatively little attention. Of the research that does exist in this vein, [11] suggests that the introduction of bitcoin futures has increased the spot volatility of bitcoin, the one which tested the information share methodology such as Gonzalo and Granger to find the spot price leads the futures price [4], while Kapar and Olmo [19] concludes the opposite. In this paper we investigate the speculative efficiency of the bitcoin market, leveraging an approach based on non-overlapping data samples, which has been previously employed to the same end in the context of the LME [9, 26, 27]. We focus on the relatively new bitcoin futures markets as provided on the Chicago Board Options Exchange (CBOE) and the Chicago Mercantile Exchange (CME) from December 2017. Using non-overlapping bitcoin spot and futures price data from December 2017 to April 2020 we reject the null hypothesis that 1-month futures prices are unbiased predictors of future bitcoin spot prices, whereas for 2-week and 1-week futures prices, we are unable to reject the null hypothesis. Additionally, we confirm the futures price becomes a more accurate indicator of the spot price as the futures contract becomes shorter. This suggests current 1-month bitcoin future price is not the perfect predictor of the future bitcoin spot price, and hence the room for excess returns, but the market becomes more efficient as the contract length shortens.

This paper is structured as follows. Section 2 covers preliminaries, including the EMH, the SEH and bitcoin futures markets. We subsequently presents the empirical models and data in Sect. 3, and then results in Sect. 4. We conclude in Sect. 5.

2 Preliminaries

In this section we provide statements of the EMH and the SEH, and then introduce the bitcoin futures markets.

2.1 The EMH

A financial market can be considered efficient if market prices fully reflect all available information [13].

Definition 1 *The Efficient Market Hypothesis (EMH)*: Asset prices reflect all available information. There exist three forms of the EMH—weak, semi-strong, and strong—as follows [7].

Weak form EMH. Future asset prices are random and are not influenced by past information such as past prices. Therefore investors cannot use technical or fundamental analysis to achieve excess returns.
Semi-strong form EMH. Current prices reflect all public information, with prices quickly adjusting to reflect new public information. Therefore only private information allows investors to earn excess returns.
Strong form EMH. All information, both public and private, is already reflected in prices. No investor can earn excess returns.

Among the three forms of the EMH, the most commonly used is the weak form EMH, as it represents the inability of the investors to take advantage of information about past quotes of assets in order to predict the future values of the assets. The efficiency of a market has been of immense interest in finance, as in efficient markets, market participants can trade without any information research. The hypothesis has also been tested by using bitcoin spot prices, as explained in Sect. 1.

2.2 The SEH

Commodity markets are mainly comprised of spot and futures contracts of different maturities. A futures contract obliges the buyer and the seller to fulfil their contractual commitment at a stipulated future date, whereas a spot contract requires the parties to execute their commitments on the spot. The futures price reflects the expectation of market participants, and at the expiry date, a futures contract that calls for immediate settlement should have a futures price equal to the spot price.

One way to examine the EMH for futures markets has been to test the hypothesis that the futures price at time t with the maturity length n, is an unbiased predictor of the spot price in time $t + n$ [16]. This hypothesis has been extensively analyzed in commodity markets. However, testing this hypothesis involves both theoretical and econometric concerns.

On the theoretical level, although the unbiased predictor hypothesis has been associated with the EMH and rational expectations hypothesis [9], Bilson [6] showed that the unbiased predictor hypothesis using the forward price is not a necessary condition of either rational expectations or the EMH to hold. This is established since (i) it is possible to construct frameworks in which market expectations are rational, but in which futures prices differ from the future spot price because of transaction cost and risk aversion and (ii) markets are efficient, in the sense that they remove any opportunity for risk-free excess returns, feature a predictable bias in the futures price forecast. Considering this discussion and to distinguish the unbiased predictor hypothesis from the EMH, Bilson's interpretation [6] in redefining the unbiased predictor hypothesis as the SEH is followed in empirical verification of the speculative efficiency analysis [9, 26, 27].

Definition 2 *The Speculative Efficiency Hypothesis (SEH)*: The futures price at time t with the maturity length n is an unbiased predictor of the spot price in time $t + n$,

assuming the information available to market participants at time t (under Bilson's interpretation [6]).

From the econometric viewpoint, SEH testing poses a serious sampling dilemma: we often need to work with data where the contract length is longer than the frequency of observations. There have been two methods applied to confront this dilemma. The first is to utilize non-overlapping observations and estimate using OLS. This ensures the residuals $u_{t,k}$ are serially uncorrelated[1] [16], however, it involves discarding a lot of observations, which leads to reducing the samples to a featureless level. The second uses overlapping data and applies an autoregressive moving average (ARMA) process for the forecast error term. This one is not as simple as the first approach, but incorporates more information than the first method.

Existing literature on the SEH has mainly focused on commodity markets such as the LME market,[2] but few on bitcoin futures markets so far. This is likely because the bitcoin futures market has emerged very recently (December 2017), whereas the LME has been in existence for over 100 years.

2.3 Bitcoin Futures Markets

The trading of futures contracts for bitcoin commenced on the Chicago Board Options Exchange (CBOE) on 10th December 2017, and the Chicago Mercantile Exchange (CME) on 18th December 2017. However the CBOE announced in March 2019, that it would stop listing additional XBT futures contracts for trading, on account of the fact that trading volumes for CBOE's bitcoin futures were significantly lower than those of the CME.

Although CBOE and CME had been the two largest and best-known bitcoin futures exchanges until the CBOE settled the last bitcoin futures contract on 19th June 2019, there exist dozens of cryptocurrency futures trading platforms. One such platform is Kraken, which offers futures trading in Ether (ETH), Litecoin (LTC), Bitcoin Cash (BCH), and Ripple (XRP). Examples of cryptocurrency futures contract pairs on such platforms include BTC/USD, ETH/USD, LTC/USD, BCH/USD, XRP/USD, and XRP/BTC. Today, more heavily regulated markets, including the CME, still continue to focus on Bitcoin and USD futures contracts, whereas unregulated marketplaces, such as OKEx and BitMEX, have become one of the largest cryptocurrency futures exchanges, launching various cryptocurrency futures trading platforms.

[1] See further discussion in Sect. 3.

[2] There are a number of studies that focus on the market efficiency of the LME, but there is no consensus in the academic literature regarding the efficiency of the LME [26].

2.3.1 Chicago Mercantile Exchange (CME)

The CME launched bitcoin futures "BTC" trading on 18th December 2017, with the volume on the exchange increasing since then, reaching $2.1 billion in December 2019. Figure 1 charts the bitcoin futures price.

The CME's Bitcoin futures contract, ticker BTC, is a USD cash-settled contract, based on Bitcoin Reference Rate (BRR), a daily reference rate of the U.S. dollar price of one bitcoin. The BRR aggregates the spot bitcoin trading activities across four bitcoin exchanges, itBit, Kraken, BitStamp, and GDAX, during a one-hour calculation window between 3pm and 4pm GMT into the U.S. dollar price of one bitcoin as of 4 p.m. GMT. Contract information for CME bitcoin futures contract is summarized in Table 1.

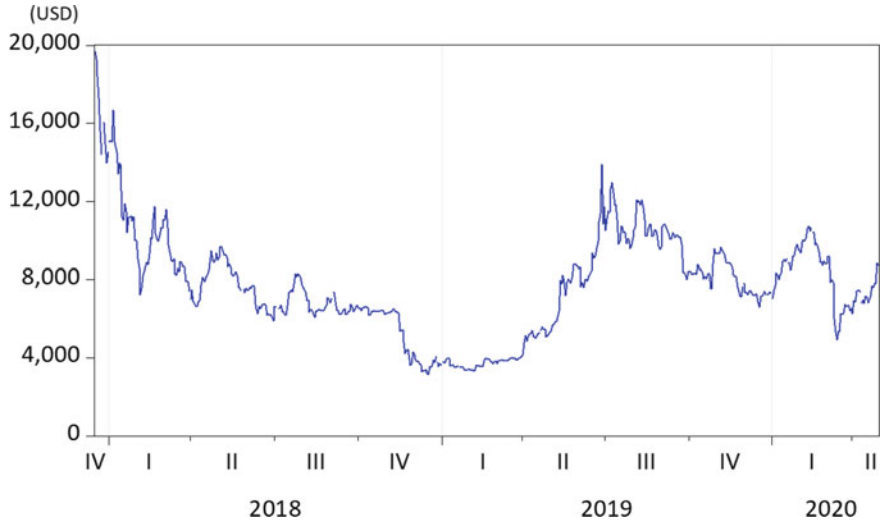


Fig. 1 CME bitcoin futures price (BTC1)

Table 1 Stylized facts on CME bitcoin futures

Variable	Description
Ticker	BTC
First traded	18th December 2017
Contract unit	5 bitcoins
Tick size	$5 per bitcoin
Underlying spot price	Bitcoin Reference Rate (BRR) index
Position limits	2,000 contracts per spot month
Trading hours	Sunday to Friday, 5pm to 4pm CT (one-hour break at 5pm)
Expiry date	Last Friday of each month

BTC futures expire on the last Friday of each month, with the settlement date the next Monday, and are listed on the nearest two months in the March quarterly cycle (Mar, Jun, Sep, Dec) plus the nearest two months if not in the quarterly cycle. Since each contract is for five bitcoins, a trader's maximum exposure is 10,000 bitcoin.[3] Bitcoin futures on the CME are regulated by the Commodity Futures Trading Commission (CFTC), the regulatory body with exclusive jurisdiction over US bitcoin futures markets, whereas other bitcoin futures markets are not necessarily regulated by the authority.

3 Data and Methodology

We apply the methodology used to test the SEH on the LME [9, 26, 27] to investigate the speculative efficiency of the CME bitcoin market. In order to examine whether the futures price on the CME is an unbiased predictor of the future spot price, we used non-overlapping bitcoin spot price and 1-month, 2-week and 1-week bitcoin futures prices.

3.1 Methodology

As discussed in [26, 27], the SEH for commodity markets has been tested by examining if the futures price at time t with the maturity length n, is an unbiased estimator of the spot price in time $t + n$ [16]. Under the condition of risk neutrality and zero transaction cost, the SEH implies the following:

$$S_{t+n} = F_{t,n} + \epsilon_{t,n} \tag{1}$$

Equation (1) shows that the futures price $F_{t,n}$ with maturity in n periods quoted at time t is the best unbiased predictor of the future spot price S_{t+n}, when the futures contract reaches maturity, given the all information are available at time t and given that the forecast error term $\epsilon_{t,n}$ has zero mean and is serially uncorrelated. As discussed in Sect. 2, we follow Bilson [6] in clarification of the unbiased estimator hypothesis as the SEH, as was carried out in [9, 26, 27].

As Canarella and Pollard [9] indicated, one empirical way to test the SEH with non-overlapping data is to estimate the following regression with ordinary least squares (OLS):

$$s_t = a_0 + a_1 f_{t-1} + \epsilon_t \tag{2}$$

[3]The position limit used to be 1,000 contracts per spot month for any single investor, ceiling the position limit to 5,000 bitcoin. The change took effect on 30th September 2019 for the October 2019 contract.

where s_t is the natural logarithm of the spot price s_t and f_t is the natural logarithm of the 3-month futures price F_t. In our method, we use the 1-month futures price instead, as the settlement day is last Friday of each month in the CME, and a 1-month lag suffices to attain non-overlapping data. We also test 2-week and 1-week futures prices to examine if there exists a difference in predictability of the market within different contract length. If the futures price is an unbiased predictor of the spot price at time $t + n$, coefficient for the constant a_0 in the regression for Eq. (2) should be insignificantly different from zero and the coefficient for the futures price a_1 should differ insignificantly from 1.

We employ a Wald test for the joint hypothesis that $a_0 = 0$ and $a_1 = 1$. Wald test examines whether the explanatory variables in a model are significant. In our case, if we are unable to reject the joint hypothesis that $a_0 = 0$ and $a_1 = 1$, we cannot conclude that the futures price is not an unbiased estimator of the future spot price.

3.2 Data

Following Canarella and Pollard [9], we collected bitcoin spot and futures closing prices data at the CME from 19th December 2017 to 30th April 2020 from *Bloomberg* to construct the non-overlapping data set. We used 1-month, 2-week and 1-week expiry futures price and each resulted in N = 28 non-overlapping observations.[4] The prices are listed in U.S. dollars.

Table 2 presents the descriptive statistics of prices and returns of bitcoin spot and futures markets used in the experiment, while Fig. 2 displays the spot prices and the 1-month future prices between December 2017 and April 2020. We can see similarity in price movement between the bitcoin spot and futures prices, suggesting that the futures price predicts the spot price to some extent and vice versa. However, key differences are observed regarding the standard deviation, skewness and kurtosis. The standard deviation is larger for the futures both in terms of prices and returns, indicating that they are more volatile.

We also find large differences between spot and futures prices regarding skewness. In terms of prices, futures exhibit more positive skew than spot prices, with futures prices skewed further to the right than spot prices. In terms of returns, futures are asymmetric, with a slightly right-skewed distribution, while spot returns are left-skewed.

Regarding kurtosis,[5] spot prices and futures prices are relatively leptokurtic when compared with the normal distribution (with heavier tails). In terms of returns, also both series have heavier tails than the normal distribution. Overall, the leptokurtosis of futures suggests that an investment in futures is risky.

[4]The observations for the spot and the 1-month/2-week/1-week futures price were constructed as a closing price of the last Friday of each month.

[5]The kurtosis of a normal distribution is 3.

Table 2 Descriptive Statistics for bitcoin spot and futures prices

	Spot		Futures	
	Price	Return	Price	Return
Mean	7,657.3	0.000025	7,714.0	−0.000136
Median	7,541.6	−0.000292	7,450.0	−0.000523
Maximum	18,674.5	0.250035	19,700.0	0.249214
Minimum	3,156.9	−0.271874	3,145.0	0.234882
Std Dev.	2,604.2	0.048443	2,690.8	0.049260
Skewness	0.63593	−0.137478	0.71924	0.034729
Kurtosis	4.16252	7.587694	4.52716	7.610152

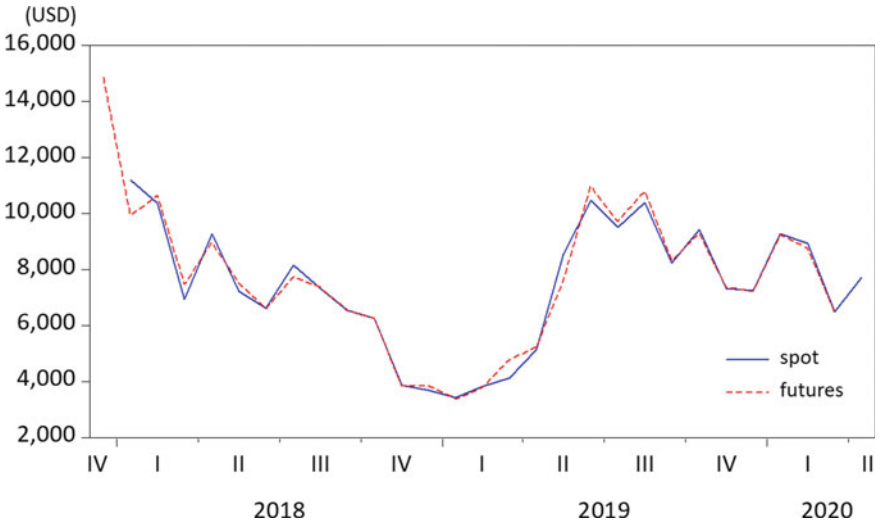


Fig. 2 Comparison of bitcoin spot and futures prices

Non-Overlapping Data

To understand the implication of applying non- overlapping data as sample data, consider the forecast error $u_{t,k} = s_{t+k} - f_{t,k}$, where s_{t+k} denotes the natural logarithm of the spot price at time $t + k$, and $f_{t,k}$ is the natural logarithm of the futures price at time t for delivery (settlement) at time $t + k$. When observations are non-overlapping, namely, $k = 1$, it can easily be shown that the forecast error $u_{t,k}$ is serially uncorrelated, whereas $E(u_{t,k}u_{t+h,k})$ equals to zero only for all $h > k$ when observations are overlapping. We therefore utilized non-overlapping data to yield a simple, tractable model. From an empirical perspective, we check the correlogram as

well as Durbin–Watson (DW) statistic to confirm if the non-overlapping futures price data set which we used for empirical analysis in fact does not include information to be found in first-order autocorrelation, as was discussed in Canarella and Pollard [9].

4 Results

We estimate Eq. (2) in Sect. 3.1 by OLS and present the results in Table 3. Each column refers to a different (non-overlapping) data sample, with *1-month* denoting the futures price as taken 1 month before delivery date, *2-week* denoting the futures price 2 weeks before the delivery date and similarly for the *1-week* futures.

For the futures price to be an unbiased predictor of future spot prices, the null hypothesis (H_0) is that a_0 is statistically insignificant from 0 and that a_1 is statistically insignificant from 1 [9, 15, 27].

Regarding the 1-month data sample, we find that both the constant and the lag of futures price is significant at the 1% level. Moreover, an F-test for joint significance yields a probability of 0.026, enabling us to reject the null hypothesis at the 5% significance level. This means that we can reject the hypothesis that the futures price is an unbiased predictor of future spot prices, or in other words, the *1-month* futures price is not an unbiased predictor of the spot price. In turn, this suggests that the market for *1-month* bitcoin futures is inefficient, suggesting inefficiency in the pricing of the futures price. A market is efficient if prevailing prices in the market reflect all currently available information.

In contrast, regarding both the *2-week* and *1-week* data samples, we are unable to reject the null hypothesis of market efficiency: in these cases we are unable to reject the null hypothesis that the futures price is an unbiased predictor of the spot price. Moreover, we note that as we move from the *2-week* to the *1-week* futures

Table 3 OLS parameter estimates and related statistics for Eq. (2)

Variable	1-month	2-week	1-week
a_0	2.097**	0.568	0.205
	(0.800)	(0.474)	(0.304)
a_1	0.761***	0.937***	0.975***
	(0.089)	(0.055)	(0.035)
N	28	28	28
R^2	0.634	0.889	0.962
F-statistic (H_0: $Constant = 0, a_1 = 1$)	4.228	1.544	0.808
(F-statistic probability)	0.026	0.233	0.457

$^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$. Robust standard errors in parentheses

contract, the estimate of a_1 increases slightly, from 0.937 to 0.975, indicating that as the futures contract becomes shorter the futures price becomes a more accurate indicator of the spot price.

Given the *prima facie* plausibility of a unit root in both the futures price and the spot price, by way of robustness checks we consider the Durbin–Watson statistic, which tests for serial correlation in the residuals of the regressions and suggests there is no evidence of auto-correlation in the residuals.[6] This is confirmed by visual inspection of the lags of the residuals, which do not show statistically significant serial correlation at the 1% level. Therefore, as econometric theory suggests, we consider that the two series cointegrate such that a linear combination of the series is stationary, yielding valid OLS results.

One possible explanation for the apparent market inefficiency arising with the *1-month* futures is that the bitcoin markets are insufficiently liquid, preventing prices from reflecting all available information. The apparent market inefficiency, that the current future price is not the best predictor of the future spot price, suggests speculative inefficiency and therefore the possibility of generating excess returns.

5 Conclusion

In this paper, using non-overlapping CME data from the period December 2017 to April 2020, we analyzed bitcoin spot and futures price data to estimate the speculative efficiency of the Bitcoin market. While several papers have examined the efficiency of the bitcoin market with reference to the bitcoin spot price, here we test for market efficiency by leveraging futures prices. In doing to we adapt a methodology previously applied to the LME to the relatively new bitcoin futures markets as provided on the CBOE and the CME from late 2017 [11].

We find that 1-month bitcoin futures prices are not unbiased estimators of future bitcoin spot prices. This suggests that the market for 1-month bitcoin futures is inefficient, implying the possibility for generating excess returns. On the other hand, for 2-week and 1-week futures prices, we are unable to reject the null hypothesis that futures prices are unbiased predictors of future bitcoin spot prices. Additionally, we found the bitcoin futures price becomes a more accurate indicator of the spot price as the futures contract becomes shorter. This leads to the conclusion that the current 1-month Bitcoin futures price is not an unbiased estimator of the future bitcoin spot price, and hence making room for excess returns, but that the market becomes more efficient as the contract length shortens.

Future work includes the application of the methodology proposed to larger data sets or to test different bitcoin futures markets data. This is because the features of

[6]The Durbin Watson (DW) statistic tests for autocorrelation in the residuals of a regression analysis. The DW statistic takes a value between 0 and 4. A value of 2.0 shows that there is no autocorrelation detected in the sample. Values from 0 to less than 2 indicate positive autocorrelation and values from 2 to 4 suggest negative autocorrelation. In our sample, DW statistic for *1-month* futures regression results is 1.956.

the CME bitcoin futures contract prohibited us from obtaining high frequency non-overlapping data enough to test with daily observations. Testing with data from these markets would guarantee robustness of the empirical results. Additionally, larger data sets would enable us introduce autoregressive moving average (ARMA) model,[7] as tested in [9, 26]. Utilizing the ARMA approach would make it possible for us to empirically test the SEH with overlapping data, which avoids reducing the sample size.

References

1. Al-Yahyaee, K. H., Mensi, W., & Yoon, S. M. (2018). Efficiency, multifractality, and the long-memory property of the bitcoin market: A comparative analysis with stock, currency, and gold markets. *Finance Research Letters*, *27*, 228–234. https://doi.org/10.1016/j.frl.2018.03.017.
2. Baek, C., & Elbeck, M. (2015). Bitcoins as an investment or speculative vehicle? a first look. *Applied Economics Letters*, *22*(1), 30–34. https://doi.org/10.1080/13504851.2014.916379.
3. Bariviera, A. F. (2017). The inefficiency of bitcoin revisited: A dynamic approach. *Economics Letters*, *161*, 1–4. https://doi.org/10.1016/j.econlet.2017.09.013.
4. Baur, D. G., & Dimpfl, T. (2019). Price discovery in bitcoin spot or futures? *Journal of Futures Markets*, *39*(7), 803–817. https://doi.org/10.1002/fut.22004.
5. Beck, S. E. (1994). Cointegration and market efficiency in commodities futures markets. *Applied Economics*, *26*(3), 249–257. https://doi.org/10.1080/00036849400000006.
6. Bilson, J. (1981). The "speculative efficiency" hypothesis. *The Journal of Business*, *54*(3), 435–51. https://EconPapers.repec.org/RePEc:ucp:jnlbus:v:54:y:1981:i:3:p:435-51
7. Bodie, Z., Kane, A. & Marcus, A. J. (2013). Investments.
8. Brock, W., Lakonishok, J. & LeBaron, B. (1992). Simple technical trading rules and the stochastic properties of stock returns. *The Journal of Finance*, *47*(5), 1731–1764. http://www.jstor.org/stable/2328994.
9. Canarella, G., & Pollard, S. K. (1986). The 'eefficiency' of the london metal exchange: A test with overlapping and non-overlapping data. *Journal of Banking & Finance*, *10*(4), 575–593. https://doi.org/10.1016/S0378-4266(86)80006-1.
10. Corbet, S., Eraslan, V., Lucey, B., & Sensoy, A. (2019). The effectiveness of technical trading rules in cryptocurrency markets. *Finance Research Letters*, *31*, 32–37. https://doi.org/10.1016/j.frl.2019.04.027.
11. Corbet, S., Lucey, B., Peat, M., & Vigne, S. (2018). Bitcoin futures - what use are they? *Economics Letters*, *172*, 23–27. https://doi.org/10.1016/j.econlet.2018.07.031.
12. Corbet, S., Lucey, B., & Yarovaya, L. (2018). Datestamping the bitcoin and ethereum bubbles. *Finance Research Letters*, *26*, 81–88. https://doi.org/10.1016/j.frl.2017.12.006.
13. Fama, E. (1970). Efficient capital markets: A review of theory and empirical work. *Journal of Finance*, *25*(2), 383–417. https://EconPapers.repec.org/RePEc:bla:jfinan:v:25:y:1970:i:2:p:383-417.
14. Goss, B. A. (1981). The forward pricing function of the london metal exchange. *Applied Economics*, *13*(2), 133–150. https://doi.org/10.1080/00036848100000020.

[7] ARMA models provide a description of a weak stationary stochastic process in terms of two processes: Autoregression (AR(p)) model and Moving average (MA(q)) model. The model is usually referred to as ARMA(p, q) model where p is the number of AR terms and q is the number of MA terms. The AR part denotes regressing the variable on its own lagged (i.e., past) series values. The MA part models the error term as a linear combination of residual terms of various times in the past.

15. Gross, M. (1988). A semi-strong test of the efficiency of the aluminum and copper markets at the lme. *The Journal of Futures Markets (1986-1998)*, *8*(1), 67.
16. Hansen, L. & Hodrick, R. (1980). Forward exchange rates as optimal predictors of future spot rates: An econometric analysis. *Journal of Political Economy*, *88*(5), 829–53. https://EconPapers.repec.org/RePEc:ucp:jpolec:v:88:y:1980:i:5:p:829-53.
17. Houben, R. & Snyers, A. (2020). Crypto-assets: Key developments,regulatory concerns and responses (april 2020). Policy Department for Economic, Scientific and Quality of Life Policies
18. Jiang, Y., Nie, H., & Ruan, W. (2018). Time-varying long-term memory in bitcoin market. *Finance Research Letters*, *25*, 280–284. https://doi.org/10.1016/j.frl.2017.12.009.
19. Kapar, B., & Olmo, J. (2019). An analysis of price discovery between bitcoin futures and spot markets. *Economics Letters*, *174*, 62–64. https://doi.org/10.1016/j.econlet.2018.10.031.
20. Kenourgios, D. & Samitas, A. (2005). Testing efficiency of the copper futures market: New evidence from london metal exchange. Finance, University Library of Munich, Germany. https://EconPapers.repec.org/RePEc:wpa:wuwpfi:0512010.
21. Köchling, G., Müller, J., & Posch, P. N. (2019). Does the introduction of futures improve the efficiency of bitcoin? *Finance Research Letters*, *30*, 367–370. https://doi.org/10.1016/j.frl.2018.11.006.
22. MacDonald, R., & Taylor, M. (1988). Metal prices, efficiency and cointegration: Some evidence from the london metal exchange. *Bulletin of Economic Research*, *40*(3), 235–240. https://doi.org/10.1111/j.1467-8586.1988.tb00268.x.
23. Mensi, W., Lee, Y. J., Al-Yahyaee, K. H., Ahmet, S., & Seong-Min, Y. (2019). Intraday downward/upward multifractality and long memory in bitcoin and ethereum markets: An asymmetric multifractal detrended fluctuation analysis. *Finance Research Letters*, *31*, 19–25. https://doi.org/10.1016/j.frl.2019.03.029.
24. Nadarajah, S., & Chu, J. (2017). On the inefficiency of bitcoin. *Economics Letters*, *150*, 6–9. https://doi.org/10.1016/j.econlet.2016.10.033.
25. Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf.
26. Otto, S. (2011). A speculative efficiency analysis of the london metal exchange in a multi-contract framework. *International Journal of Economics and Finance*, *1*. https://doi.org/10.5539/ijef.v3n1p3.
27. Park, J. & Lim, B. (2018). Testing efficiency of the london metal exchange: New evidence. *International Journal of Financial Studies*, *6*(1), 1–10. https://ideas.repec.org/a/gam/jijfss/v6y2018i1p32-d136280.html.
28. Selgin, G. (2015). Synthetic commodity money. *Journal of Financial Stability*, *17*(C), 92–99. https://EconPapers.repec.org/RePEc:eee:finsta:v:17:y:2015:i:c:p:92-99.
29. Sensoy, A. (2019). The inefficiency of bitcoin revisited: A high-frequency analysis with alternative currencies. *Finance Research Letters*, *28*, 68–73. https://doi.org/10.1016/j.frl.2018.04.002.
30. Urquhart, A. (2016). The inefficiency of bitcoin. *Economics Letters*, *148*, 80–82. https://doi.org/10.1016/j.econlet.2016.09.019.
31. Urquhart, A. (2017). Price clustering in bitcoin. *Economics Letters*, *159*, 145–148. https://doi.org/10.1016/j.econlet.2017.07.035.
32. Zhang, W., Wang, P., Li, X., & Shen, D. (2018). Some stylized facts of the cryptocurrency market. *Applied Economics*, *50*(55), 5950–5965. https://doi.org/10.1080/00036846.2018.1488076.

Carbon Trading with Blockchain

Andreas Richardson and Jiahua Xu

Abstract Blockchain has the potential to accelerate the worldwide deployment of an emissions trading system (ETS) and improve the efficiency of existing systems. In this paper, we present a model for a permissioned blockchain implementation based on the successful European Union (EU) ETS and discuss its potential advantages over existing technology. The proposed ETS model is both backward compatible and future-proof, characterised by interconnectedness, transparency, tamper-resistance and continuous liquidity. Further, we identify key challenges to implementation of blockchain in ETS, as well as areas of future work required to enable a fully decentralised blockchain-based ETS.

Keywords Blockchain · Carbon trading · ETS · Sustainability · ESG

1 Introduction

Carbon trading systems, such as the European Union Emissions Trading System (EU ETS), provide a market mechanism to incentivise emissions reduction on the basis of *cap and trade*. An overall *cap* on emissions in tonnes of CO_2-equivalent[1] (tCO_2e)

[1] Scaling factors known as Global Warming Potentials (GWPs) are used to normalise the impact of various Greenhouse Gases (GHGs) emitted against CO_2 (which, by definition, has a GWP of 1).

A. Richardson (✉)
Imperial College London, London, UK
e-mail: andreas.richardson17@imperial.ac.uk

J. Xu
University College London, Centre for Blockchain Technologies, London, UK
e-mail: jiahua.xu@ucl.ac.uk

A. Richardson · J. Xu
École polytechnique fédérale de Lausanne, Lausanne, Switzerland

P. Pardalos et al. (eds.), *Mathematical Research for Blockchain Economy*, Springer Proceedings in Business and Economics,
https://doi.org/10.1007/978-3-030-53356-4_7

is imposed by a central authority, which is translated into allowances that are issued to companies. These allowances are surrendered and retired at the end of a reporting period to offset the company's emissions during the period, with the company free to *trade* any surplus allowances on the market [28]. Importantly, should a company have insufficient allowances to cover their (expected) emissions, they are obliged to either purchase surplus allowances from other market participants, or take measures to reduce their emissions; penalties are imposed for non-compliance [28]. Naturally, a high price for allowance units incentivises participants to choose the latter option.

On first inspection, this system appears to be suited to an application of blockchain technology, as it involves multiple distributed parties transacting using common currencies and requires transactions to be recorded in an immutable ledger. Indeed, multiple organisations and startups are actively exploring this approach [8]. However, on closer inspection, the centralised nature of ETSs in their current form and the immaturity of the blockchain industry pose some critical challenges to the adoption of the technology. One of the frequently-cited advantages of blockchain is the "disintermediation of trust" [1, 9, 10], meaning a central trusted authority is not required for the network to reach consensus. Yet current ETS designs make heavy use of trusted authorities: a central (governmental) authority is responsible for the distribution of allowances under the cap, whether by direct allocation or through an auction process; further, companies must report their emissions to the central authority and seek verification of this figure from a third-party [19]. More generally, security loopholes and unethical activities permeating the blockchain space continue to act as a barrier against immediate adoption of this still evolving technology [11, 37, 45].

As a result, a clear and compelling case must be made to justify the advantages of blockchain over existing technologies. A number of frameworks have been proposed for assessing potential blockchain implementations, considering technical, organisational and legal factors [9, 10, 31], whilst a series of strategic questions have been raised for business leaders evaluating blockchain's potential [11, 23]. The extreme interest shown in blockchain technology over recent years and the resulting disillusionment with its failure to meet over-hyped promises means the technology is treated with caution; its pros and cons must be carefully weighed [22, 33, 34].

In this paper, we describe the advantages and challenges of implementing a blockchain-based ETS, and sketch out a hybrid model that is both backward compatible and future-proof.

2 Background

We first present the EU ETS as a prime example of a contemporary ETS, using it to introduce a discussion of the weaknesses in current ETSs and highlight areas where blockchain technology has strong potential. We additionally present a review of selected literature in this space.

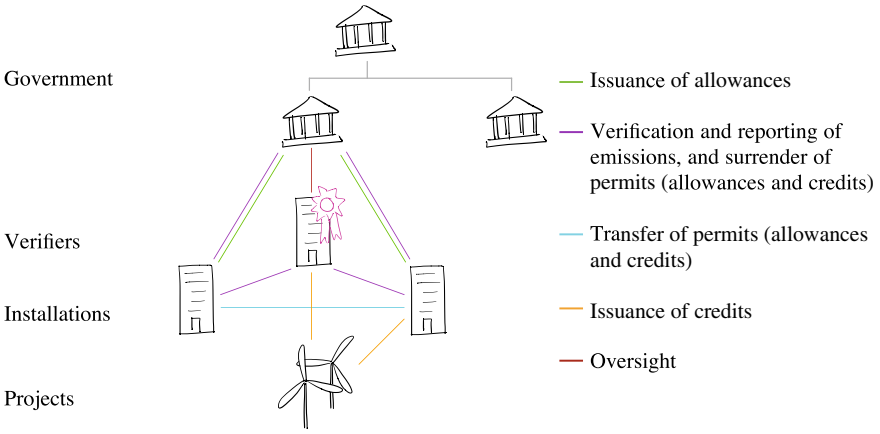


Fig. 1 Overview of the EU ETS. Lines represent transactions between parties; the two layers of government represent the European Commission and member states.

2.1 EU ETS

The EU ETS was launched in 2005 and has become the largest ETS to date, representing the majority of international emissions trading [12, 19]. Its coverage extends to over 11,000 installations (power stations and industrial plants) with significant energy usage as well as airlines operating in the EU, together representing about half of the EU's greenhouse gas (GHG) emissions[2] [18]. A representative schematic of the different players and transactions in the EU ETS is presented in Fig. 1.

Tradeable instruments The EU ETS introduces a new tradeable instrument alongside the allowance unit: credits. Whilst allowances are issued by governments of member states through allocation or auction, credits are generated through emissions-reduction projects in other countries under Kyoto Protocol mechanisms. Any allowances or credits surplus to an installation's requirement to offset its emissions may be freely traded for profit [19].

Impact Relative to a 2005 baseline, the EU ETS is expected to have reduced emissions by 21% in 2020 and by 43% in 2030, indicating that the underlying market mechanism is functioning as expected [18].

[2]The GHGs covered by the EU ETS are carbon dioxide (CO_2), nitrous oxide (N_2O) and perfluorocarbons (PFCs). [19]

2.2 Potential and Suitability of Blockchain

Despite their successes, there still exist issues with both the EU ETS and ETS more broadly, which this paper seeks to address. Specifically, we argue that blockchain technology shows great potential to advance the state of the art in a number of key areas of ETS development.

Coverage Existing ETSs are restricted in terms of geographical coverage, with large portions of the world currently lacking plans to implement ETS [9]. A distributed scalable blockchain-based ETS solution could rapidly support new carbon markets by allowing nodes to join the network with ease. Article 6 of the 2018 Paris Agreement already provides a foundation for decentralised cooperative climate action; blockchain is expected to be a key technology to deliver these ambitions, particularly through future carbon markets [6, 15].

Linkage Accounting, auditing and mutual monitoring of emissions between entities in disconnected ETSs are deemed challenging. For example, it is believed that for the UK, an exit from the EU—and consequently the EU ETS—may hinder attempts to meet future carbon budgets [24]. Although some ETSs have previously implemented links, the process is complex and lengthy, as evidenced by the near decade-long process to link the Swiss and EU ETS [13, 20]. In this context, an easily extensible linked ETS solution that can be rapidly deployed in new areas would be highly desirable.

An interlinked web of ETSs would increase market liquidity and size [16, 29, 36, 40], and reduce opacity inherent in siloed systems. Transparently linking multiple ETSs would increase the likelihood of discovery, and hence lower the chance, of fraudulently claiming credits from the same project in multiple systems ("double-counting") [9, 10, 16].

Cost A (semi-)automated decentralised system embedding smart contracts can be used to reduce overall transaction cost. For individual enterprises, fixed costs can be further cut especially when spread across a large network. Lower transaction costs reduce barriers to entry, allowing coverage to be extended to smaller enterprises and less-developed geographies.

Trust As codified protocols in immutable smart contracts are tamperproof, blockchain-based ETSs are expected to improve trust relative to existing systems [7]. This could help maintain market confidence and integrity with linked ETSs, for example if one ETS operates in a jurisdiction with an increased risk of corruption [16].

Transparency The shared, distributed nature of a blockchain system underpins transparency. Address anonymity (or pseudonymity) with blockchain would allow transaction data to be made available in much greater detail, without compromising privacy or confidentiality concerning e.g. ETS players' trading positions. Compared to the

EU ETS transaction log (EUTL), from which relatively little data is made available, increased scrutiny of public data could strengthen systems and reduce the risk of government corruption [9, 10, 19, 21].

Consensus and fault tolerance Well-designed consensus mechanisms provide a degree of fault tolerance that allows the system to operate normally even with the presence of misbehaving actors or malfunctioning nodes in the network [2, 5, 42]. This is relevant particularly in the context of linking ETSs, where heterogeneous players with various levels of credibility and reliability are connected to the system.

2.3 Existing Work

Existing attempts to bring the benefits of blockchain to carbon trading have generally had a limited impact and a short lifespan [21], being largely predicated on the small voluntary carbon market. As such, any blockchain solution will require significant support from existing ETS regulators to ensure sufficient impetus for further growth and development. Once "critical mass" of usership is achieved however, a solution can be expected to become self-sustaining; the contribution of regulators to facilitate access to the regulatory compliance market is likely to be a significant factor for success. Also of note is that despite favourable press coverage over the past years, blockchain is not a panacea and still suffers from crucial limitations [43] (see discussion in Sect. 4).

In Table 1, we layout selected existing works related to the present discussion; whilst the concept of a blockchain-based ETS has already been broadly discussed, this study considers the practical implementation of such an ETS.

3 Proposal

Given the challenges involved in the development of a completely decentralised blockchain-based ETS, a more pragmatic approach might be to progressively improve upon existing ETS frameworks. Thus, we sketch out a hybrid model combining some degree of decentralisation whilst maintaining a role for trusted authorities, as illustrated in Fig. 2. This does not preclude a future switch to a fully decentralised model, but it is expected to provide an easier transition to blockchain technology for existing ETS players, increasing the practical feasibility of the proposal.

Table 1 Overview of selected existing works

Source	Description
Discussion papers	
[4]	Presents a systems engineering approach to a decentralised emissions trading infrastructure. Reviews architecture (covering e.g. database type, credit issuance, existence of central authority etc.) of other carbon trading schemes
[9]	Outlines blockchain potential for ETS, climate mitigation and climate finance applications in the specific context of Mexico, with discussion of potential implementation (technologies, costs, roadmap etc.)
[10]	Provides an overview of blockchain potential, suitability and challenges for applications including ETS, MRV and climate finance
[16]	Describes current climate markets from a technological perspective and discusses improvements. Presents potential and suitability of blockchain
[21]	Argues for the suitability and potential of blockchain in achieving the commitments of the Paris Agreement and for climate action in general, with discussion of areas of required future work
Implementation work	
[17]	Discusses general suitability of blockchain for ETS applications, and presents a proof-of-concept implementation for a transportation-specific ETS using Hyperledger Iroha
[27]	Proof-of-concept blockchain for green certificates (proof of electricity generation from renewable sources) in a microgrid electricity trading environment using the Corda platform
[30]	Proof-of-concept ETS implementation using "reputation points" to determine market access priority, thus aiming to tackle security issues identified with EU ETS. Includes detailed quantitative analysis of improvement relative to conventional systems
[32]	Develops detailed proof-of-concept blockchain for EU ETS using smart contracts on Ethereum. Discusses software development process (requirements, use cases) and system architecture (implementation) in detail

3.1 Taxonomy

In this proposal the following definitions are used:

Organisation The simplest type of entity in the network, upon which other roles are built. An organisation provides a framework to manage common metadata required to interact with the blockchain (e.g. public/private key management).

Authority Governmental or supranational body, with legislative power over other authorities or enterprises within a certain jurisdiction.

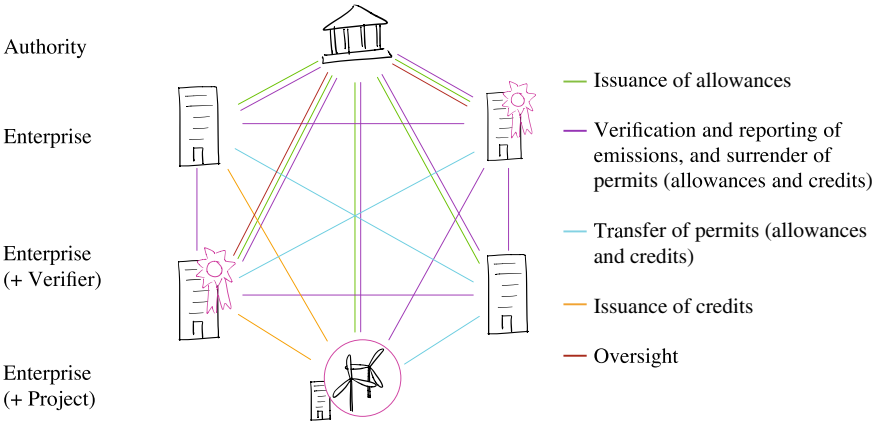


Fig. 2 Schematic showing potential interactions in the outlined blockchain-based ETS

Enterprise	"Legal person" consisting of one or more installations or projects that uses the network to report and/or offset emissions. An enterprise may be mandated to participate in the network by an authority or access it voluntarily, and may carry verifier status.
Installation	Physical source of GHG emissions—such as factories, manufacturing plants, offices—owned by one or more enterprises.
Project	Emissions reduction scheme generating carbon credits under Kyoto Protocol mechanisms, owned by one or more enterprises.
Verifier	Status awarded to an enterprise by an authority, allowing it to perform verification functions within the network.

3.2 Tokens

Two types of token are envisaged for network, closely mirroring current ETS bookkeeping [9, 19] (see Sect. 3.5).

Emission	1 tCO_2e verified GHG emissions.
Permit	Permit to emit 1 tCO_2e. Permits can represent both allowances issued by an authority, or credits granted by a verifier.

3.3 Processes

In this section, we present a basic outline of processes performed on the network, illustrated with smart contract pseudocode required for their execution.[3] At the scale required of an international ETS, a custom-developed blockchain may be more appropriate; nevertheless, our approach provides a general framework for presenting and debating a proof of concept.

Role change An authority can change the role of an enterprise, including promotion of an enterprise to verifier status or removal of an existing status.

Algorithm 1 Role change

```
1 Function setRole(address sender, address target, string newRole)
2     require sender.role = authority and target.role ≠ newRole   // Sender must be authorised and request is a change
3     target.role ← newRole
```

Issuance of permit Emission permits can represent either allowances issued by an authority, or credits issued by a verifier.

An authority can mint permit tokens typically in an amount corresponding to the desired cap level. The tokens may be issued through direct allocation or auction.

Algorithm 2 Mint permit token

```
1 Function mintPermit(address signer, address target, uint amount)
2     require signer.role = authority   // Only authorities can mint permits
3     target.balance[permit] += amount
4     market.balance[permit] += amount
```

Permit tokens can also represent credits granted by a verifier to an enterprise owning emission-reducing projects.

Algorithm 3 Grant permit token

```
1 Function grantPermit(address signer, address target, uint amount)
2     require hasProject(target) and signer.role = verifier   // Verifier ensures that the enterprise has a carbon-reducing project
3     target.balance[permit] += amount
4     market.balance[permit] += amount
```

Both `mintPermit` and `grantPermit` allow permit tokens to be issued "out of thin air", thus increasing the total circulating supply of the token in the market.

[3] Our pseudocode is inspired by the Solidity language used to implement smart contracts on the Ethereum blockchain.

In Sect. 3.4 we quantify the effect of this mechanism on the market price of permit tokens.

Issuance of emissions Emission tokens may be minted by any enterprise if a verifier co-signs the transaction as a true reflection of the enterprise's emissions.

Algorithm 4 Mint emission token

```
1 Function mintEmission(address sender, address signer, uint amount)
2     require signer.role = verifier        // Must be signed by a verifier
3     sender.balance[emission] += amount
4     market.balance[emission] += amount
```

Transfer tokens Permit tokens which represent emission allowances and credits may be freely transferred among network participants, who may choose to create derivative products such as swaps and options (as in the EU ETS [19, p. 71]) or to send tokens to an exchange.

Algorithm 5 Transfer permit tokens

```
1 Function transferPermit(address sender, address target, uint amount)
2     require amount ≤ sender.balance[permit]
                                  // Must have enough token for request
3     sender.balance[permit] –= amount
4     target.balance[permit] += amount
```

Burn tokens Emission tokens are burnt alongside an equal or greater number of permit tokens in a single transaction. This process also allows enterprises to voluntarily surrender excess permit tokens if they so choose (as is possible in the EU ETS [19, p. 131]). Enterprises cannot transact with emission tokens in any other way.

Algorithm 6 Burn tokens

```
1 Function burnToken(address sender, uint amount)
2     require amount ≤ sender.balance[permit]  // Must have enough token
3     if sender.balance[emission] ≥ amount then
                                  // Only burning part of emission balance
4         sender.balance[emission] –= amount
5     else if sender.balance[emission] < amount then
          // Burning beyond emission balance (voluntary surrender)
6         sender.balance[emission] = 0
7     sender.balance[permit] –= amount
```

Token exchange Organisations can freely trade their permit tokens with the authority, which also acts as a liquidity provider. To ensure liquidity in the market and hence enhance the tradability of tokens, we can implement the Bancor protocol [25,

39] which automates price determination according to the dynamics of supply and demand (see Algorithms 7 and 8, and Appendix "Bancor Algorithm for Token Exchange").

Algorithm 7 Trade tokens

```
1  Function tradeToken(address sender, uint amount)
2      supply ← market.balance[permit]
3      cashAmount ← reserve * ((1 + amount/supply)^(1/fraction) – 1)
                                          // Based on eq. (7) in Appendix
4      if amount > 0 then
5          require cashAmount ≤ sender.cash        // Must have cash to spend
6          sender.balance[permit] += amount
7          market.balance[permit] += amount
8          sender.cash –= cashAmount
9          reserve += cashAmount
10     else if amount <= 0 then
11         require–amount ≤ sender.balance[permit]   // Must have token to sell
12         sender.balance[permit] += amount
13         market.balance[permit] += amount
14         sender.cash –= cashAmount
15         reserve += cashAmount
```

Algorithm 8 Convert cash

```
1  Function convertCash(address sender, unit amount)
2      supply ← market.balance[permit]
3      tokenAmount ← supply * ((amount/reserve + 1)^fraction −1)
                                          // Based on eq. (8) in Appendix
4      if amount > 0 then
5          require amount ≤ sender.cash            // Must have cash to spend
6          sender.balance[permit] += tokenAmount
7          market.balance[permit] += tokenAmount
8          sender.cash –= amount
9          reserve += amount
10     else if amount <= 0 then
11         require–tokenAmount ≤ sender.balance[permit]    // Must have token to
            sell
12         sender.balance[permit] += tokenAmount
13         market.balance[permit] += tokenAmount
14         sender.cash –= amount
15         reserve += amount
```

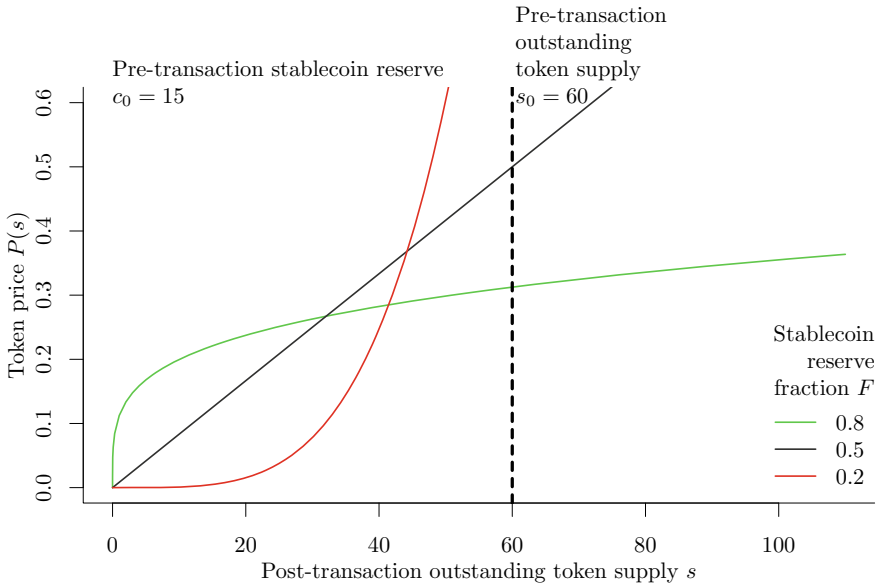


Fig. 3 Token price as a function of supply as specified in (4)

3.4 Market Adjustment

With the development of technology, the cost of emissions reduction will decrease over time. As a result, the supply of surplus allowances and credits will increase, whilst demand for them decreases, driving the price of tokens down. Thus it may become cheaper for firms to use credits to offset their emissions rather than reducing the emissions directly.

With the emissions cap being a moving target, the authority may steer the market in order to continuously motivate emission reduction. In addition to having the power to change the cap level and so restrict the supply of allowances the authority can also adjust, the price of tokens through the exchange. According to (1), the tokens' market price can be raised in two ways:

- Reducing reserve fraction F, allowing for stablecoin (digital cash) to be spent from the exchange and thus enhancing its purchasing power;
- Increasing the total stablecoin reserve c_0, thus enhancing the purchasing power of the exchange;

Naturally, in order to reduce the token price, the authority can simply do the opposite.

3.5 Carbon Bookkeeping on and off Blockchain

We demonstrate the compatibility of our proposed blockchain-based model with the existing ETS frameworks through an illustrative example. In principle, blockchain is underpinned by time-honoured bookkeeping mechanisms including TEA (Triple Entry Accounting) and REA (Resources-Events-Agents) [26]. Therefore, the compatibility between the conventional and the newly proposed ETS record-keeping framework is expected to an extent.

In our illustrative example, the following series of accounting events take place. For simplification purposes, we ignore the slippage effect as the transacted emissions in our example is assumed to account for an insignificant portion of the total market volume.

1. On January 1, 2020, the market value for of one permit token was 20 €.
   - Authority $\mathscr{A}$ allocated Enterprise $\mathscr{E}$ with allowances for 100 tCO_2e by issuing $\mathscr{E}$ 100 permit tokens.
   - Verifier $\mathscr{V}$ approved Enterprise $\mathscr{E}$'s carbon-reducing project in a developing country and granted $\mathscr{E}$ with credits for 40 tCO_2e by issuing $\mathscr{E}$ 40 permit tokens.
   - Enterprise $\mathscr{E}$ transferred 10 permit tokens to Enterprise $\mathscr{F}$.
2. On June 30, 2020, the market value of one permit token increased to 24 €.
   - Enterprise $\mathscr{E}$ recorded 55 tCO_2e emissions from January to June.
   - Enterprise $\mathscr{E}$ cashed out 240 € by selling 10 tokens to the market.
3. On December 31, 2020, the market value of one permit token decreased to 22 €.
   - Enterprise $\mathscr{E}$ recorded 70 tCO_2e emissions from June to December.
   - Enterprise $\mathscr{E}$ bought 5 permit tokens from the market to cover the emissions.
   - Enterprise $\mathscr{E}$ surrenders 125 permit tokens to offset the total emissions of 125 tCO_2e during year 2020.

In Table 2, we juxtapose smart contract execution with double-entry journalisation from the perspective of Enterprtise $\mathscr{E}$ to show the correspondence of the two systems. We use the fair value method [35, 38] to record.

Conventionally, revaluation gains or losses are only recorded on prescribed financial accounting dates (June 30 and December 31 in our example). However, with a blockchain-based network that connects account books of participating enterprises, equity change due to market movement can be recorded automatically and continuously. This would enable a constantly up-to-date valuation of an enterprise with no additional labour cost on accounting, auditing and reporting.

Table 2 Carbon accounting with smart contract execution and journalisation

Smart contract execution	Journalisation		Debit	Credit
`mintPermit` ($\mathscr{A}$, $\mathscr{E}$, 100)	**Asset**	Emission permit—Allowances	2,000	
	Liability	Deferred income		2,000
`grantPermit` ($\mathscr{V}$, $\mathscr{E}$, 40)	**Asset**	Emission permit—Credits	800	
	Liability	Deferred income		800
`transferPermit` ($\mathscr{E}$, $\mathscr{F}$, 10)	**Liability**	Deferred income	200	
	Asset	Emission permit		200
	Asset	Emission permit	520	
	Equity	Gain on revaluation		520
`mintEmission` ($\mathscr{V}$, $\mathscr{E}$, 55)	**Liability**	Deferred income	1,100	
	Equity	Income		1,100
	Equity	Expenses—Emissions	1,320	
	Liability	Permit surrenderable		1,320
`convertCash` ($\mathscr{E}$, −240)	**Asset**	Cash	240	
	Asset	Emission permit		240
	Liability	Deferred income	200	
	Equity	Income		200
	Equity	Loss on revaluation	240	
	Asset	Emission permit		240
`mintEmission` ($\mathscr{V}$, $\mathscr{E}$, 70)	**Liability**	Deferred income	1,300	
	Equity	Income		1,300
	Equity	Expenses—Emissions	1,430	
	Liability	Permit surrenderable		1,430
`tradeToken`($\mathscr{E}$, 5)	**Asset**	Emission permit	110	
	Asset	Cash		110
`burnToken`($\mathscr{E}$, 125)	**Liability**	Permit surrenderable	2,750	
	Asset	Emission permit		2,750

4 Further Challenges and Considerations

Implementation The specific platform chosen to host a blockchain-based ETS is a central consideration. In [10], Bitcoin, Ethereum, Hyperledger Fabric and EOS are evaluated for climate policy applications, considering programmability, operating cost, security and usability, with the finding that Ethereum and Hyperledger Fabric are the most promising platforms to date. Similarly, [32] considers Ethereum and Hyperledger Fabric as strong candidate frameworks for ETS implementation, noting important distinctions between the two: Ethereum is by default public and

permissionless; Hyperledger Fabric is private and permissioned. A more complete discussion of many other platforms can be found in [2].

As discussed previously, both developing a derivative blockchain solution (e.g. using Hyperledger Fabric) and developing an entirely custom implementation should be considered in finding an approach to host an ETS at large scale. The relative benefit of building upon an established system (e.g. pre-existing audited and/or open-source code) must be weighed against the degree of customisation desired. Further, should differing implementations be developed by governments or organisations, standardisation could still enable interoperability [14].

Governance and trust Since the "allocation of allowances, the opening and closing of ETS registry accounts or the recognition of offset credits "still fall under sovereign tasks of the government", a comprehensive carbon network would by default involve governments as central authorities [9]. The initial delegation of authority in a permissioned blockchain-based ETS requires participants to trust the authority establishing the network and thus the integrity of the tokens issued. With ETS linkage that connects different states and regions, participants may not trust all authorities equally, imposing the need for an on-chain governance design that ensures the integrity of authorities. Importantly, whilst smart contracts may be ideally suited for the rigorous application of defined rules, these rules must first be developed in collaboration with stakeholders [16].

Further, a potential future shift towards a decentralised blockchain without explicit governmental oversight presents a significant complication: should there exist trust asymmetries between players, the fungibility of tokens issued by different entities will be challenged and could lead to fragmentation of the network. One solution could be standardisation [14].

Enforcement Current ETSs utilise legislation to compel enterprises to participate. Whilst a voluntary carbon market does also exist, it is significantly smaller than the regulatory compliance market [41]. In a completely decentralised international ETS, it is less clear what would encourage participation (or discourage non-participation). Additionally, defining how criminal activity on the network would be deterred is challenging, potentially requiring a supranational enforcement body to maintain network integrity. Indeed, "new governance systems will be needed to ensure market and environmental integrity in a peer-to-peer environment" [16].

Measurement, reporting and verification (MRV) A critical issue with any blockchain solution is its interface with the real world [43]; the maxim "garbage in, garbage out" aptly illustrates the consequences of poor input data. The verification and accreditation processes in the EU ETS are complex and potentially burdensome [19]. Moving beyond the model of trusted verifiers to a truly decentralised approach will require significant effort to develop alternative MRV methodologies.

The internet of things (IoT) will enable a universally trusted mechanism for MRV of real-world data, by automating data flows and processes [16, 21]. IoT technology is expected to reduce the cost and time requirement of MRV, whilst enhancing trust

through increased reliability and the accessibility of audited code. Real-time sensing will enable a faster compliance cycle than the current yearly process in the EU ETS [9, 21], whilst increased trading activity through more frequent reporting and compliance will enhance market liquidity. Additionally, diverse data sources such as earth observation satellites will enable stronger verification of reported emissions or emissions reductions.

Efficiency Represented by Bitcoin, existing blockchain networks commonly employ a computationally expensive consensus mechanism, "proof-of-work", and are rather inefficient compared to centralised database systems, especially in terms of electricity usage [3, 16]. One study has estimated that the global Bitcoin network consumes approximately as much power as the country of Ireland, and forecasts this consumption increasing more than three-fold in the future [44]. Consequently, various alternative consensus mechanisms have been proposed to improve efficiency and reduce the environmental impact of the infrastructure itself [5, 37].

5 Conclusion

In this paper, we investigate the applicability and demonstrate the technical feasibility of blockchain technology to carbon trading on ETS. We conclude that despite the potential for blockchain to enhance the impact and reach of current ETSs in numerous ways, significant barriers remain, limiting the applicability of the technology today. A basic outline of a permissioned blockchain solution largely mirroring today's EU ETS has been presented as a viable transitional first step towards the development of a fully-decentralised blockchain-based ETS, which could significantly accelerate the deployment of this important emissions reduction tool worldwide. We maintain that significant legislative and legal barriers remain to be overcome for sensible and effective implementation of a decentralised blockchain-based ETS.

Acknowledgements The authors thanks Chris N. Bayer, Juan Ignacio Ibañez, and Vincent Piscaer for their comments and suggestions.

Appendix

Bancor Algorithm for Token Exchange

As demonstrated with Algorithms 7 and 8, the Bancor exchange protocol [25, 39] ensures constant tradability of a token, as it prices a token algorithmically, as opposed through matching a buyer and a seller. We use the notation listed in Table 3 to explain the protocol.

Table 3 Mathematical notation for token exchange

Notation	Definition	Unit
Preset hyperparameters, occasionally adjusted		
F	Constant fraction of stablecoin reserve	—
Input variables		
s_0	Pre-transaction outstanding token supply	tokens
c_0	Pre-transaction stablecoin reserve	€/token
e	Tokens to be bought (negative when sold)	tokens
t	Stablecoins to be spent (negative when received)	€
Output variables		
s	Post-transaction outstanding token supply	tokens
$P(.)$	Post-transaction token price, dependent on token supply s	€/token
$C(.)$	Post-transaction stablecoin reserve, dependent on token supply s	€

For demonstration purposes, we assume that the medium of exchange is a stablecoin, measured in €, that circulates on the same blockchain as the permit tokens.

It holds that, the stablecoin reserve C (in €), always equals a fraction, preset as $F \in (0, 1)$, of the product of token price P (in €/token) and outstanding token supply s (in tokens). That is, the following equation is always true:

$$C(s) \equiv F\, s\, P(s) \tag{1}$$

Taking the derivative with respect to s on both sides:

$$\frac{\mathrm{d}C(s)}{\mathrm{d}s} \equiv F\left[P(s) + s\frac{\mathrm{d}P(s)}{\mathrm{d}s}\right] \tag{2}$$

There exists another relationship between C, P and s: if one buys from the exchange an infinitesimal amount of tokens, $\mathrm{d}s$, when the outstanding token supply is s, then the unit token price at purchase would be $P(s)$. The exchange receives stablecoins and thus its reserve increases according to:

$$\mathrm{d}C(s) = P(s)\,\mathrm{d}s$$

Rearranging:

$$P(s) = \frac{\mathrm{d}C(s)}{\mathrm{d}s} \tag{3}$$

Combining (2) and (3),

$$P(s) = F\left[P(s) + s\frac{\mathrm{d}P(s)}{\mathrm{d}s}\right]$$
$$\frac{\mathrm{d}P(s)}{P(s)} = \left(\frac{1}{F} - 1\right)\frac{\mathrm{d}s}{s}$$

Integrating over $s \in (s_0, s)$:

$$\int_{x=P(s_0)}^{P(s)} \frac{\mathrm{d}x}{x} = \left(\frac{1}{F} - 1\right)\int_{y=s_0}^{s} \frac{\mathrm{d}y}{y}$$
$$\ln P(s) - \ln \frac{c_0}{F\, s_0} = \left(\frac{1}{F} - 1\right)(\ln s - \ln s_0)$$

Now we can express token price $P(.)$ as a function of s:

$$P(s) = \frac{c_0}{F\, s}\sqrt[F]{\frac{s}{s_0}} \tag{4}$$

Plugging (4) into (1), we can derive the exchange's stablecoin reserve $C(.)$ as a function of s:

$$C(s) = F\, s\, \frac{c_0}{F\, s}\sqrt[F]{\frac{s}{s_0}} = c_0\sqrt[F]{\frac{s}{s_0}} \tag{5}$$

Assume one spends t amount of stablecoins in exchange for e amount of tokens when the outstanding token supply equal s_0. After the purchase, the outstanding token supply becomes $s_0 + e$, while the stablecoin reserve increases by t, i.e.,

$$t + c_0 = C(s_0 + e) \overset{\text{according to (5)}}{=} c_0\sqrt[F]{\frac{s_0 + e}{s_0}} = c_0\sqrt[F]{1 + \frac{e}{s_0}} \tag{6}$$

Rearranging (6), we get:

- the amount of stablecoins to be paid (or received when negative), t, based on the amount of tokens to be bought (or sold when negative), e, and the outstanding token supply s_0 (Algorithm 7),

$$t = c_0\left(\sqrt[F]{1 + \frac{e}{s_0}} - 1\right) \tag{7}$$

- the amount of tokens to be bought (or sold when negative), e, based on the amount of stablecoins to be paid (or received when negative), t, and the outstanding token

supply s_0 (Algorithm 8).

$$e = s_0 \left[\left(\frac{t}{c_0} + 1 \right)^F - 1 \right] \tag{8}$$

References

1. Adams, R., Parry, G., Godsiff, P., & Ward, P. (2017). The future of money and further applications of the blockchain. *Strategic Change*, *26*(5), 417–422. https://doi.org/10.1002/jsc.2141.
2. Aggarwal, S., Chaudhary, R., Aujla, G. S., Kumar, N., Choo, K. K. R., & Zomaya, A. Y. (2019). Blockchain for smart communities: Applications, challenges and opportunities. *Journal of Network and Computer Applications*, *144*(February), 13–48. https://doi.org/10.1016/j.jnca.2019.06.018.
3. Aitzhan, N. Z., & Svetinovic, D. (2018). Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Transactions on Dependable and Secure Computing*, *15*(5), 840–852. https://doi.org/10.1109/TDSC.2016.2616861.
4. Al Kawasmi, E., Arnautovic, E., & Svetinovic, D. (2015). Bitcoin-based decentralized carbon emissions trading infrastructure model. *Systems Engineering*, *18*(2), 115–130. https://doi.org/10.1002/sys.21291.
5. Andoni, M., Robu, V., Flynn, D., Abram, S., Geach, D., Jenkins, D., McCallum, P., & Peacock, A. (2019). Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews*, *100*(November 2018), 143–174. https://doi.org/10.1016/j.rser.2018.10.014.
6. Asian Development Bank. (2018). Decoding Article 6 of the Paris Agreement. Tech. rep., Asian Development Bank, Manila, Philippines. https://www.adb.org/publications/decoding-article-6-paris-agreement.
7. Banerjee, A. (2018). *Re-engineering the carbon supply chain with blockchain technology*. Infosys. https://www.infosys.com/Oracle/white-papers/Documents/carbon-supply-chain-blockchain-technology.pdf.
8. Baumann, T. (2017). Using blockchain to achieve climate change policy outcomes. *IETA Insights: Greenhouse Gas Market Report*, *2017*(3), 14–15. http://www.ieta.org/resources/Resources/GHG_Report/2017/Using-Blockchain-to-Achieve-Climate-Change-Policy-Outcomes-Baumann.pdf.
9. Braden, S. (2019). *Blockchain for Mexican climate instruments: Emissions trading and MRV systems*. Bonn: Deutsche Gesellschaft fur Internationale Zusammenarbeit (GIZ). https://www.giz.de/en/downloads/giz2019-en-blockchain-emissions.pdf.
10. Braden, S. (2019). *Blockchain potentials and limitations for selected climate policy instruments*. Bonn: Deutsche Gesellschaft fur Internationale Zusammenarbeit (GIZ). https://www.giz.de/en/downloads/giz2019-en-blockchain-potentials-for-climate.pdf.
11. Braun, A., Cohen, L. H., & Xu, J. (2019). fidentiaX: The tradable insurance marketplace on blockchain. *Harvard Business School Case 219-116*. https://www.hbs.edu/faculty/Pages/item.aspx?num=56189.
12. Braun, A., Utz, S., & Xu, J. (2019). Are insurance balance sheets carbon-neutral? Harnessing asset pricing for climate change policy. *Geneva Papers on Risk and Insurance - Issues and Practice*, *44*(4): 549–68. https://doi.org/10.1057/s41288-019-00142-w.
13. Council of the EU. (2019). Linking of Switzerland to the EU emissions trading system - entry into force on 1 January 2020. http://www.consilium.europa.eu/en/press/press-releases/2019/12/09/linking-of-switzerland-to-the-eu-emissions-trading-system-entry-into-force-on-1-january-2020/.

14. Deshpande, A., Stewart, K., Lepetit, L., & Gunashekar, S. (2017). Understanding the landscape of Distributed Ledger Technologies / Blockchain: Challenges, opportunities, and the prospects for standards. *RAND Corporation*. https://www.rand.org/pubs/research_reports/RR2223.html.
15. Dinakaran, C., Carevic, S., Rogers, S., Srinivasan, S., & Mok, R. (2019). Harnessing blockchain to foster climate markets under the Paris agreement. https://blogs.worldbank.org/climatechange/harnessing-blockchain-foster-climate-markets-under-paris-agreement.
16. Dong, X., Mok, R. C. K., Tabassum, D., Guigon, P., Ferreira, E., Sinha, C. S., et al. (2018). Blockchain and emerging digital technologies for enhancing post-2020 climate markets. *The World Bank*. https://doi.org/10.1596/29499.
17. Eckert, J., López, D., Azevedo, C.L., & Farooq, B. (2019). A blockchain-based user-centric emission monitoring and trading system for multi-modal mobility. http://arxiv.org/abs/1908.05629.
18. European Commission: EU Emissions Trading System (EU ETS). https://ec.europa.eu/clima/policies/ets_en.
19. European Commission. (2015). *EU ETS handbook*. European Union. https://ec.europa.eu/clima/sites/clima/files/docs/ets_handbook_en.pdf.
20. Federal Office for the Environment. (2019). Linking the Swiss and EU Emissions Trading Schemes. https://www.bafu.admin.ch/bafu/en/home/topics/climate/info-specialists/climate-policy/emissions-trading/linking-the-swiss-and-eu-emissions-trading-schemes.html.
21. Fuessler, J., León, F., Mock, R., Hewlett, O., Retamal, C., Thioye, M., Beglinger, N., Braden, S., Hübner, C., Verles, M., & Guyer, M. (2018). Navigating blockchain and climate action. *Climate Ledger Initiative*. https://climateledger.org/resources/CLI_Report-January191.pdf.
22. Furlonger, D., & Kandaswamy, R. (2019). *Hype cycle for blockchain business*. Gartner, Inc. https://www.gartner.com/document/3953756.
23. Harbert, T. (2019). Blockchain for business starts in the supply chain. https://mitsloan.mit.edu/ideas-made-to-matter/blockchain-business-starts-supply-chain.
24. Hepburn, C., & Teytelboym, A. (2017). Climate change policy after Brexit. *Oxford Review of Economic Policy*, *33*, S144–S154. https://doi.org/10.1093/oxrep/grx004.
25. Hertzog, E., Benartzi, G., & Benartzi, G. (2018). Bancor Protocol Continuous Liquidity for Cryptographic Tokens through their Smart Contracts. Tech. rep. https://storage.googleapis.com/website-bancor/2018/04/01ba8253-bancor_protocol_whitepaper_en.pdf.
26. Ibañez, J. I., Bayer, C. N., Tasca, P., & Xu, J. (2020). REA, triple-entry accounting and blockchain: Converging paths to shared ledger systems. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3602207.
27. Imbault, F., Swiatek, M., De Beaufort, R., & Plana, R. (2017). The green blockchain: Managing decentralized energy production and consumption. In *Conference Proceedings - 2017 17th IEEE International Conference on Environment and Electrical Engineering and 2017 1st IEEE Industrial and Commercial Power Systems Europe, EEEIC / I and CPS Europe 2017* (pp. 1–5). https://doi.org/10.1109/EEEIC.2017.7977613.
28. International Carbon Action Partnership. (2019). What is emissions trading. *ETS Brief*, *1*. https://icapcarbonaction.com/en/?option=com_attach&task=download&id=663.
29. International Carbon Action Partnership. (2019) On the way to a global carbon market: Linking emissions trading systems. *ETS Brief, 4*. https://icapcarbonaction.com/en/?option=com_attach&task=download&id=666.
30. Khaqqi, K. N., Sikorski, J. J., Hadinoto, K., & Kraft, M. (2018). Incorporating seller/buyer reputation-based system in blockchain-enabled emission trading application. *Applied Energy*, *209*(July 2017), 8–19. https://doi.org/10.1016/j.apenergy.2017.10.070.
31. Küpper, D., Ströhle, J., Krüger, T., Burchardi, K., & Shepherd, N. (2019). *Blockchain in the factory of the future*. Boston Consulting Group. http://image-src.bcg.com/Images/BCG-Blockchain-in-the-Factory-of-the-Future-July-2019_tcm20-223907.pdf.
32. Liss, F. (2018). Blockchain and the EU ETS: An architecture and a prototype of a decentralized emission trading system based on smart contracts. Ph.D. thesis, Technical University of Munich. https://doi.org/10.13140/RG.2.2.15751.65448.

33. Litan, A., & Leow, A. (2019). *Blockchain primer for 2019*. Gartner, Inc. https://www.gartner.com/document/3920410.
34. Litan, A., & Leow, A. (2019). *Hype cycle for blockchain technologies*. Gartner, Inc. https://www.gartner.com/document/code/383155.
35. Öker, F., & Aduzel, H. (2017). Reporting for carbon trading and international accounting standards. In *Accounting and corporate reporting - Today and tomorrow*. InTech. https://doi.org/10.5772/intechopen.68959.
36. Partnership for Market Readiness, International Carbon Action Partnership: Emissions Trading in Practice: A Handbook on Design and Implementation (2016). https://icapcarbonaction.com/en/?option=com_attach&task=download&id=364.
37. Perez, D., Xu, J., & Livshits, B. (2020). Revisiting Transactional Statistics of High-scalability Blockchains. In *Proceedings of the ACM Internet Measurement Conference (IMC)*.
38. Ratnatunga, J., Jones, S., & Balachandran, K. R. (2011). The valuation and reporting of organizational capability in carbon emissions management. *Accounting Horizons*, *25*(1), 127–147. http://aaajournals.org/doi/10.2308/acch.2011.25.1.127.
39. Rosenfeld, M. (2017). *Formulas for Bancor system*. https://drive.google.com/file/d/0B3HPNP-GDn7aRkVaV3dkVl9NS2M/view.
40. Santikarn, M., Li, L., La Hoz Theuer, S., & Haug, C. (2018). *A guide to linking emissions trading systems*. Berlin: ICAP. https://icapcarbonaction.com/en/?option=com_attach&task=download&id=572.
41. Seeberg-Elverfeldt, C. (2010). 2 carbon markets - Which types exist and how they work. In *Carbon finance possibilities for agriculture, forestry and other land use projects in a smallholder context* (pp. 5–11). Rome: Food and Agriculture Organization of the United Nations. http://www.fao.org/docrep/012/i1632e/i1632e.pdf.
42. Sikorski, J. J., Haughton, J., & Kraft, M. (2017). Blockchain technology in the chemical industry: Machine-to-machine electricity market. *Applied Energy*, *195*, 234–246. https://doi.org/10.1016/j.apenergy.2017.03.039.
43. Tucker, C., & Catalini, C. (2018). What blockchain can do. *Harvard Business Review* (June). https://hbr.org/2018/06/what-blockchain-cant-do.
44. de Vries, A. (2018). Bitcoin's growing energy problem. *Joule*, *2*(5), 801–805. https://doi.org/10.1016/j.joule.2018.04.016.
45. Xu, J., & Livshits, B. (2019). The anatomy of a cryptocurrency pump-and-dump scheme. In *28th USENIX Security Symposium (USENIX Security 19)* (pp. 1609–1625). Santa Clara, CA: USENIX Association. https://www.usenix.org/system/files/sec19-xu-jiahua_0.pdf.

Economic Games as Estimators

Michael Zargham, Krzysztof Paruch, and Jamsheed Shorish

Abstract Discrete event games are discrete time dynamical systems whose state transitions are discrete events caused by actions taken by agents within the game. The agents' objectives and associated decision rules need not be known to the game designer in order to impose structure on a game's reachable states. Mechanism design for discrete event games is accomplished by declaring desirable invariant properties and restricting the state transition functions to conserve these properties at every point in time for all admissible actions and for all agents, using techniques familiar from state-feedback control theory. Building upon these connections to control theory, a framework is developed to equip these games with estimation properties of signals which are private to the agents playing the game. Token bonding curves are presented as discrete event games and numerical experiments are used to investigate their signal processing properties with a focus on input-output response dynamics.

Keywords Estimation · Dynamic games · Cryptoeconomic systems

Supported by the Research Institute for Cryptoeconomics at WU Vienna in collaboration with BlockScience, Inc.

M. Zargham (✉) · K. Paruch · J. Shorish
Vienna University of Economics and Business, Vienna, Austria
e-mail: zargham@block.science; michael.zargham@wu.ac.at

K. Paruch
e-mail: krzysztof.paruch@wu.ac.at

J. Shorish
e-mail: jamsheed.shorish@wu.ac.at

M. Zargham
BlockScience, Inc, Tempe, AZ, USA

P. Pardalos et al. (eds.), *Mathematical Research for Blockchain Economy*,
Springer Proceedings in Business and Economics,
https://doi.org/10.1007/978-3-030-53356-4_8

1 Introduction

Cryptoeconomic systems [25] are digital data-driven multiscale, adaptive and dynamic networks with a system-level state available to all agents. These systems use cryptographic tokens as information carriers, allowing for economic activities to emerge on top of a shared distributed ledger technology (DLT) enabled infrastructure such as blockchain. Formally, these economies can be described using a state space representation [22, 30, 31], which allows the encoding of agents, transactions and mechanisms, as well as state transitions resulting from activities within the network. Additional requirements on reachable system states can be imposed using *configuration spaces* [29] to design possible future system trajectories, without assuming agents' decision rules or specifying agents' preference functions. Intended design of configuration spaces is a standard technique applied in the creation of robotic systems to restrict possible movements of machines, see for example [19].

Understanding the formal structure of cryptoeconomic systems is facilitated using game theory, a mathematical framework that formalizes the dynamics of multi-agent systems within a spectrum of repeated discrete games (see e.g. [6]) on the one side and continuous differential games (e.g. [9]) on the other side. Game theory has been applied to cryptoeconomic systems in a variety of ways. In a DLT protocol layer, the economics of consensus mechanisms [1] and the effects on network security [14] have been studied using game theoretic concepts. Applications of DLT to finance, such as portfolio diversification, have also been studied in [2].

Game-theoretic models are usually based upon a specification of the players, or *agents*, and their preferences, strategy sets and associated payoffs. While standard 'toy' models such as the discrete-time repeated prisoner's dilemma (see e.g. [20]) and the continuous-time conflict models (e.g. [9]) are pedagogically useful, more complicated models are required for greater realism. For example, population games [21] model strategic interactions with a large number of individually negligible agents with an explicitly dynamic model of individual choice defined by the revision protocol of every agent. When stochastic revision opportunities arise, agents are free to change strategies, resulting in a Markov process describing the mean dynamics of the system. Three important classes of population games are potential games [17], supermodular games [24] and stable games [8]. Potential games use a single global potential function to represent all players' incentives to change their strategies, which adds additional structure to the game environment. An equilibrium is guaranteed to exist, and there is a wide array of distributed learning algorithms that guarantee convergence [16].

Learning in games [5, 7, 27] explores how a process might emerge for convergence to e.g. a Nash equilibrium from various initial conditions. Evolutionary games are similar, focusing on the dynamics of strategy changes within a population [21]. Finally, mean field games are sequential games with a continuum of players, in which players affect their opponents in ways that are insignificant at the individual level but significant when aggregated [11], and evolution takes place according to a dynamical relation [16, 26].

Each of the above modeling paradigms possess different 'encodings' of space and time, agent models, system interaction, and payoffs. But there exists a sufficiently general notion of a game that can subsume most or all of these encodings, by interpreting a game as a *system* evolving over time based on the actions of a group of agents. The game is then akin to the plant of a control system, and the agents as a collection of individual controllers with private state, signals, and objectives [3]. The design of the game is akin to the design of the system plant, to be controlled by an *a priori* unknown set of controllers–the agents–and becomes a formal mechanism design problem. We introduce such an interpretation in this work, defining *discrete event games* as discrete time dynamical systems whose state transitions are discrete events caused by actions taken by agents within the game. In this approach there are observable and provable states of the interpreted system plant, regardless of agent objectives, decision rules, etc. which are in general not known to the designer.

Under this interpretation, changes to the system state caused by agent actions act as samples of their private preferences or private information. This allows one to consider a game as an *estimator* that gathers information over time from a multi-agent system [13]. The game design, then, acts to dynamically estimate useful summary statistics of the underlying distribution of agents, even as that distribution changes over time. If, for example, agent decision-making influences the price of an asset (such as a cryptocurrency token), then it is the price which is estimated by the discrete event game. In a similar fashion, the design pattern of combining discrete event models with agent behavior and system parameter estimation also arises in cyber-physical systems [10].

This work contributes to the existing literature in game theory, market theory and estimation theory. It analyzes economic games played by agents on intentionally shaped sub-spaces of the state space, namely on lower-dimensional manifolds. These manifolds are designed to ensure that all economic activity takes place in a space specifically shaped to reflect the conditions under which the game is intended to be played. A configuration space ensures that these lower-dimensional manifolds have designed characteristics and conserve invariant properties of the system. This allows the focus to shift to the conditions and states of the game, rather than to the particular behavior, strategies and private preferences of agents (since some properties of the system will stay true regardless of the choices of its participants). The game outcome allows the inference of system-level properties that are revealed by actions taken by the agents, *without knowing further details about their particular preferences*. By doing so, it is aggregated agent behavior that acts as a signal, estimating specific parameters (such as prices, treated in this work).

The creation of the conditions for a digital economic game with enforced state space restrictions described above (and expressed in further detail in [29]) is ensured by the use of DLT, which maintains a tamper-proof universal state layer. Whether a future state is reachable will heavily depend upon the design of the configuration space, which can be restricted using *token bonding curves* to impose invariant properties upon the system and thereby limit possible system state trajectories. A more comprehensive description of bonding curves is provided in Sect. 3, but in a nutshell they are used as an enforceable mechanism in the market design of token economies,

where they function as continuous liquidity mechanisms allowing agents to influence the state. Bonding curves are most commonly used in corporate finance, financial instrument pricing and as rights management tools. System participants will make their decisions based on the utility gains they perceive from evaluations of expected effects (in accordance with their private, local preferences), but will take into consideration that outcomes will follow global *laws of motion* dictated by the 'rules of the game' encoded in the shape of the space.[1] A market for tokens emerges as a result of economic activity between agents, with the token price as a variable describing one particular global property of the system estimated from individual signals of constituent agents. This observation paves the way for a possible contribution to price theory, treating bonding curves as estimators of market prices.

The paper is structured as follows: Sect. 2 introduces the notation and definitions required to formally represent discrete event games, configuration spaces and the estimation framework. Section 3 reviews the characteristics of bonding curves within the state space representation, via the configuration space and the representation's mechanisms. It continues with a description of the formal process of global price estimation derived from private signals of the agents. Section 4 then presents dynamic price estimation with open loop agents, derived from numerical results for a specific bonding curve parametrization. Finally, Sect. 5 concludes and outlines future work.

2 Notation and Definitions

2.1 Discrete Event Games

Consider a system in which *agents* interact within a *network*, the topology of which is specified as part of a global *system*. Agents are decision-making entities that are completely characterized by their *state*, considered as a finite vector of k elements taken from a field. In what follows it is assumed that the field is the usual real number line $\mathbb{R}$, but this may be generalized. The state characterizes the agent insofar as it specifies 'private' information known only to that agent.[2]

Agents are indexed by an identifier $a \in \{1, 2, 3, \ldots, n < \infty\}$, while time $t \in \mathbb{Z}^{\geq 0}$ is a 'lattice' upon which agent decisions and actions are placed. Time also indexes the flow of information, which impacts the state of the agent. Thus, an agent's state may be summarized by a vector $\hat{x}_{a,t}$. Denote the *agent state space* by $\hat{X}_a \subseteq \mathbb{R}^k$, so that $\forall a,\ \forall t,\ \hat{x}_{a,t} \in \hat{X}_a$.

The *agent-level state* is decentralized but may nonetheless be summarized as

[1] Cf. [4, 15] for statistical mechanical and econophysical approaches that address this micro-meso-macro aggregation.

[2] We adopt notation and conventions from the signal processing literature throughout, opting for a unified exposition at the potential risk of cross-disciplinary "notation collision".

$$(\hat{x}_{1,t}, \hat{x}_{2,t}, \ldots, \hat{x}_{n,t}) \in \prod_{a=1}^{n} \hat{X}_a \subseteq \mathbb{R}^{nk}.$$

The network carries its own internal state, the *system-level state*. As the network is assumed to be a finite (probabilistic or deterministic) state machine, the internal state may be given by a finite vector of m elements, with (as in the agent case) elements taken from a field. For simplicity we again assume that the field is identical for all elements and equals $\mathbb{R}$, but the approach (and future research) accommodates arbitrary fields. The system-level state, denoted $\bar{x}_t$, depends upon the information arrival process summarized by time t. The *system-level state space* is then a set $\bar{X} \subseteq \mathbb{R}^m$, so that $\forall t,\ \bar{x}_t \in \bar{X}$.

The *system state* x_t is the state of all agents and the system-level state, i.e.

$$x_t := (\hat{x}_{1,t}, \hat{x}_{2,t}, \ldots, \hat{x}_{n,t}, \bar{x}_t) \in X := \prod_{a=1}^{n} \hat{X}_a \times \bar{X} \subseteq \mathbb{R}^{nk} \times \mathbb{R}^m.$$

We refer to X as the system or *global state space*.

The role of the system is to provide an 'institutional framework' within which agents interact, both with each other via their network interaction, and with the system itself as the propagator of network interaction. By 'interaction' we mean that, conditional upon a system state x_t at time t, an agent a may select from a menu of *actions*, representing valid (or "legal") actions that are admissible to the network. For simplicity, we suppose that this menu of actions is represented by a mapping $U(x_t, a)$, which is assumed to return a non-empty set at every point in time and for all agents.[3]

If an agent acts at time t, they select an action u_t from $U(x_t, a)$. In Sect. 3 it will be assumed that every agent possesses a *decision rule* incorporating available information and the admissible set of actions $U(x_t, a)$ at time t. The action u_t does not refer to a particular agent because of the way time operates as a 'lattice' for agent decisions.[4] Formally, we require

Assumption 1 Time is sufficiently finely granulated to ensure that action collisions do not occur, i.e. all agent actions are ordered by t. □

Assumption 1 means that for every t there is one and only one corresponding action u_t, selected by an agent a from $U(x_t, a)$. While this is a formal requirement for

[3] It may be the case that, for agent a,

$$U(x_t, a) \equiv U(\hat{x}_{a,t}, a),$$

i.e. the agent's state space is sufficient to define their actions (this would be the case, for example, in a game where every agent has their own action set, or where every agent conditions only upon their own private information). In what follows we allow for full conditioning on x_t.

[4] Naturally there may still be an implicit unique mapping from u_t to a, as is the case with sending Bitcoin, [30].

what follows, in practice this assumption is reasonable because the system is a state machine—hence events within the system arrive in discrete steps.[5]

The agent's selection of an action u_t *changes the system state* x_t. The system thus possesses a *mechanism* that, taking an action, transitions the system state (i.e. agent level state and the system level state) to the next lattice point $t + 1$. More formally, such a mechanism may be viewed as a transition function f, with $x_{t+1} = f(x_t, u_t)$. Note that f takes values over all possible agent actions $u_t \in U(x_t, a)$, for every possible x_t and for every a—this may be a consequence of a conservation law, which is discussed further in Sect. 2.2 below.

Definition 1 By a **discrete event game** is meant the tuple $(\bar{X}, \{\hat{X}_a\}_{a=1}^n; U, f)$, comprised of the agent and system level state sets, the decision mapping for agent actions,[6] and the system state transition function f.

A *trajectory* is a sequence of system states $\{x_t\}$ created by the repeated selections of actions by agents, in response to the system state (or their private agent state, if the system state is not fully visible to every agent). Without further restrictions it is clear that there are infinitely many possible trajectory realizations *ex ante*, depending upon the richness of the sets underlying the discrete event game. In what follows, we demonstrate that it is possible (and usually desired) for the designer of the system to impose additional structure that will restrict possible trajectory realizations to spaces (such as topological manifolds) that reduce the complexity of the game's resulting dynamical evolution.

2.2 Configuration Spaces

In the system design process one or more quantities of interest are usually *conserved*, i.e. are time-invariant over every possible global system state trajectory. A simple example is a discrete event game in which at time t a finite resource, Y_t, must be allocated across agents. If we suppose that an agent a's local state at t is their allocation of this resource, then resource conservation implies

$$\sum_{a=1}^{n} \hat{x}_{a,t} \equiv Y_t \ \forall t. \tag{1}$$

This is a *restriction* on the attainable combinations of individual agent resources that must respect the allocation restriction.

Over time the relative allocation between agents may change, so that for some or all agents, $\hat{x}_{a,t} \neq \hat{x}_{a,t+1}$. But restriction (1) nevertheless holds at every t. In addition,

[5] While ostensibly this model assumes a strict ordering of actions, this is a consequence of the definition of x_t as a global state and f as a global mechanism. Partial orderings may suffice provided a local state transition depends only upon information provided in the local state; see e.g. [13].

[6] The decision mapping defines an agent's *strategy set*, which is a standard primitive defining a game; see e.g. [6].

there may be flows into or out of the system that cause Y_t to change, where the change ΔY_t is allocated to one or more agents. Such flows are common constructs in network routing [28], crypto token allocation (such as within the original Bitcoin protocol, see [18]) and crypto mining games (e.g. [23]), where in the latter case there are conserved flows between agents but injections of new token supply into the system according to predefined monetary policies.

The key implication of resource conservation, such as (1) above, is that it reduces the dimension of allowable system trajectories—generally, there is a reduction of one dimension for each (independent) restriction. The system designer may thus focus upon a smaller space for realizations of the trajectory, called the *configuration space* (see e.g. [29] for an introduction to configuration spaces).

The quantity (or quantities) conserved throughout the dynamical evolution of the system can be expressed by designing *conservation laws*, i.e. real-valued[7] functional (linear or non-linear) relationships $V : X \to \mathbb{R}$ where the quantity to be conserved, such as a global state x_t, satisfies $V(x_t) \equiv \bar{V} \in \mathbb{R}\ \forall t$.

A conservation law so designed may be *enforced* by ensuring that the global state transition mechanism over admissible action sets $U(x_t, a)$, $\forall a$, respects

$$\begin{aligned} V(x_0) &= \bar{V}, \\ V(x_t) &= V(f(x_t; u_t))\ \forall t,\ \forall u_t. \end{aligned}$$

Selecting a pair (V, f) to implement desired conservation laws is the *mechanism design* problem facing the system designer. It depends crucially upon which conserved quantities are present, as well as upon the requirements defining the kinds of actions agents expect to be able to take. If f characterized a set of actions such as sending cryptocurrency, and V encoded a desired invariant such as conserving that cryptocurrency, then one could derive the necessary admissible function U by restricting the domain of f to the preimage of the invariant set. This results in the rule that agent a cannot send tokens exceeding its available balance.

A more general mechanism design problem characterizes the goal of the system using *performance metrics*, which tell the designer—and the participants in the system—which states (or functions of states) are considered desirable. When the game is allocating resources it may do so to the benefit of those agents that move the system state in a direction which improves relative to a performance metric. Framed as such, the network itself may be viewed as an evolutionary optimization algorithm where the agents' local efforts to maximize payouts serve to ascend a potential field characterized by the aforementioned performance metrics. Figure 1 shows a possible construction of such discrete event game.

[7] Although we focus upon real-valued laws here because of the estimation of continuous real-valued signals, in general finite or even infinite state machines may also characterize conservation laws.

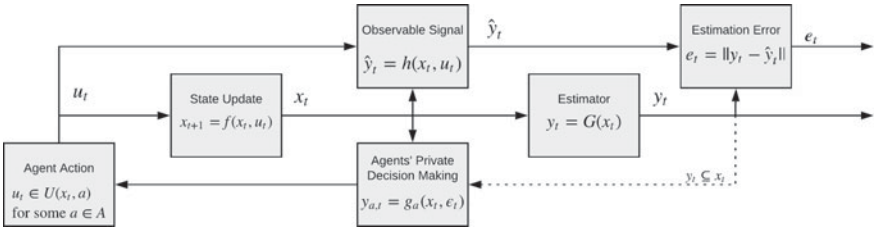


Fig. 1 Block Diagram for a discrete event game with estimation capabilities

2.3 Samples, Signals, and Estimation

In addition to conservation laws, there may be desirable quantities of interest that are generated by the system as a result of agent actions $\{u_t\}$. For example, the buy or sell decisions made according to a token bonding curve will determine a realized price, $\hat{P}_t$. This price is subject to noise from exogenous factors (e.g. market conditions, off-chain supply and demand shocks, etc.), denoted by some process ε_t, and so will be difficult to estimate. In general, we let $\hat{y}_t$ represent the realized value, which is a noisy *signal* of the variable of interest.

By implementing a pair (V, f), the system designer creates an *estimate* y_t from noisy samples $\hat{y}_t$, by forcing the trajectory of realized states to admit a mapping G carrying x_t to the estimate y_t. For example, a token bonding curve restricts token demand and supply and in turn generates an estimate P_t of a hidden signal, for which realized price $\hat{P}_t$ is a noisy sample. In general, the sequence of samples $\{\hat{y}_t\}$ depends upon individual agent mappings, which attempt to condition upon ε_t and the system state x_t. This allows the *local representation* of y_t, denoted by $y_{a,t}$, to be expressed in the form $g_a(x_t, \varepsilon_t)$. We interpret the mapping $g_a(x_t, \varepsilon_t)$ as the *representation technology* of the agent. The local representation $y_{a,t}$ may then be used by agent a in decision-making, i.e. there may be a further mapping carrying x_t, $y_{a,t}$ and ε_t (if separately influencing decisions apart from g_a) into a decision u_t from $U(x_t, a)$. Such a mapping, although not articulated in detail here, would be a *decision rule* for the agent.

There may be potential feedback between an agent's action, u_t, and the factor influencing the signal $\hat{y}_t$, i.e. it may be that

$$\frac{\partial \varepsilon_t}{\partial u_t} \neq 0. \tag{2}$$

This is the case for the token bonding curve example presented in Sect. 3: the signal is the actual realized token price, $\hat{P}_t$, which is determined by buy and sell decisions of the agents, while the spot price P_t is determined from the conservation law V and the same agent trading decisions.

The first-order impact of the agent's action on the noise process ε_t given in (2) implies that an observation of signal $\hat{y}_t$ at t represents a 'draw' or sample from the

underlying distribution of local agent representations $y_{a,t}$. In essence, the estimate y_t is the system's 'best guess' of the signal $\hat{y}_t$, accounting for variations in both space (agents) and time. A performance metric that naturally suggests itself is that of an *error* e_t such that

$$e_t = e(y_t, \hat{y}_t) := \|y_t - \hat{y}_t\| \, \forall t.$$

The error specification allows an application of estimation theory to the analysis of the system's stochastic convergence, by 'steering' the system to achieve as low an error as possible. Future research will also focus upon boundary conditions for the error for a variety of signal and estimate specifications.

In practice, the more detailed the foundations of the discrete event game, and its designed implementation of (V, f), the easier it will be to arrive at results with formal bounds in a chosen error metric. The cost, naturally, is the risk that violations of these more detailed modeling assumptions have the potential to undermine conclusions drawn in this fashion. Our analysis in this sense is simply a 'launching point' for a richer modeling paradigm crossing dynamic mechanism design theory with estimation theory (see e.g. [12]).

3 Bonding Curves as Price Estimators

The token bonding curve system, in which a *community token* is managed using bonding curve contracts, fits readily within the framework outlined in Sect. 2. A representation of mechanisms and restrictions of such a game is shown in Fig. 2. The system specifies a series of token holdings as outlined in [29], which we summarize here.

Definition 2 The **reserve** $R_t \in \mathbb{R}_{++}$ at time t is the total quantity of reserve currency tokens bonded to the bonding curve contract.

The reserve currency is provided by a contract external to the community deploying the bonding curve. This could be the native cryptocurrency or tokenized fiat, such

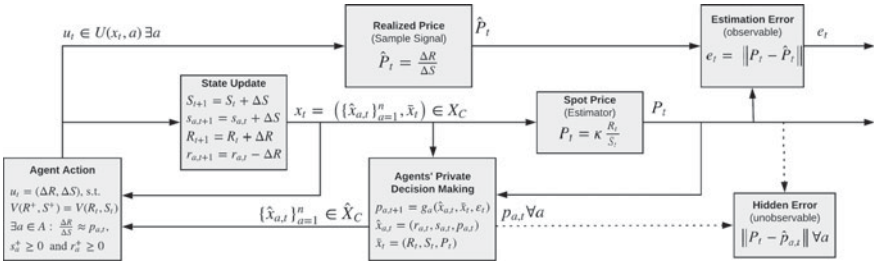


Fig. 2 Block Diagram for a discrete event game, matching an agent-based model on a bonding curve, including observable and unobservable estimation errors

as a stablecoin. At time t each agent a possesses their own holding of the reserve currency, denoted by $r_{a,t} > 0$.

Definition 3 The **supply** $S_t \in \mathbb{R}_{++}$ at time t is the total quantity of community tokens issued by the bonding curve contract.

The supply is the total quantity of the community token held by agents. An individual holding $s_{a,t}$ of the supply is part of the local state of agent a at t.

A bonding curve may be characterized using the mechanism design process (V, f)—in particular, it provides the mechanism f and the mappings U guaranteed to preserve V under f. Agents can adjust their token holdings by depositing the reserve currency to mint new community tokens, or burn all or part of their community token holdings to withdraw the reserve currency. Regardless, the supply S_t and reserve R_t always satisfy $V(R_t, S_t) = \text{constant}$. Furthermore, the system's estimate of the token price $P_t = P(R_t, S_t)$ is part of the state.

Definition 4 The **spot price** $P_t \in \mathbb{R}_{++}$ at time t is the estimate of the value of the community token, in units of R per units of S.

Since agents can freely adjust their community token holdings via the bonding curve, the spot price P_t may be interpreted as a dynamic estimate of the value imbued in the token by agents with representation technology $g_a(x_t, \varepsilon)$. The justification for this claim is further borne out by the characterization of the configuration space in Sect. 3.1. Note that each agent may hold their own (private and potentially exogenous) estimate of the value of the community token, denoted $p_{a,t} = g_a(x_t, \varepsilon)$—this will be discussed shortly.

Definition 5 The **system-level state** is $\bar{x}_t := (R_t, S_t, P_t) \in \bar{X} \subset \mathbb{R}^3_{++}$.

We shall see shortly why $\bar{X}$ is a proper subset of $\mathbb{R}^3_{++}$ once token bonding curve and supply conservation laws are taken into consideration.

Each element of the system-level state has an agent-level state counterpart, based upon the agent community token and reserve holdings.

Definition 6 The **agent-level state** is $\hat{x}_{a,t} := (r_{a,t}, s_{a,t}, p_{a,t}) \in \hat{X}_a \subseteq \mathbb{R}^3_{++}$.

In what follows we suppose (although it is not strictly required) that agents can observe the system-level state, but not each other's agent-level states. The system state x_t exists for all t even though it is not globally observable.

Definition 7 The **system state** is $x_t = (\hat{x}_{1,t}, \ldots \hat{x}_{n,t}, \bar{x}_t) \in X \subset \mathbb{R}^{3(n+1)}_{++}$ and lies in the Cartesian product of the system-level state and the agent-level state.

3.1 The Configuration Space

The system design incorporates both a mechanism f, which is the token bonding curve mechanism above, and a set of conservation laws V, indicating the design

goals that can be formulated as time-invariant quantities of interest. In the case of the bonding curve, a *system level design goal* is to establish diminishing returns for both depositing and withdrawing reserve currency from the bonding curve. This is accomplished by restricting the relationship between R and S:

Definition 8 The bonding curve **conservation function** is given by

$$V(R_t, S_t) := \frac{S_t^\kappa}{R_t} \equiv V_0, \tag{3}$$

where $V_0 = V(R_0, S_0) := \frac{S_0^\kappa}{R_0}$ is a constant defined by initial supply S_0 and initial reserve R_0. Parameter κ is the curvature of the bonding curve.

Definition 8 allows us to assert that the spot price P_t is completely determined by reserve currency and community token supply holdings and the functional form of V[8]:

$$P_t = P(R_t, S_t) = \left. -\frac{\partial V/\partial S}{\partial V/\partial R}\right|_{(R_t, S_t)}. \tag{4}$$

Definition 9 The **system-level configuration space**, $\bar{\mathbf{X}}_\mathbf{C}$ is a 1-manifold, created by applying two one-dimensional restrictions, $V(R_t, S_t) = V_0$ and $P_t = P(R_t, S_t)$ to the three-dimensional state space $\bar{X}$:

$$\bar{\mathbf{X}}_\mathbf{C} := \{\bar{x} = (R_t, S_t, P_t) \in \bar{X} \mid V(R_t, S_t) = V_0,\ P_t = P(R_t, S_t)\} \subset \bar{X}. \tag{5}$$

In addition to the system level design goal, there is also a local conservation restriction. For the community token supply, the total agent holdings at time t cannot exceed the available supply. Letting s_t denote the vector of community tokens held by all agents $(s_{1,t}, \ldots, s_{n,t})$, we have

$$V_S(s_t, S_t) := \sum_{a=1}^{n} s_{a,t} - S_t \equiv 0. \tag{6}$$

Definition 10 The **agent-level configuration space**, $\hat{\mathbf{X}}_\mathbf{C}$ is a $(3n-1)$-manifold, created by enforcing the conservation constraint $V_S(s_t, S_t) = 0$ on the $3n$-dimensional agent-level state space $\prod_a \hat{X}_a$:

$$\hat{\mathbf{X}}_\mathbf{C} := \{\{(r_{a,t}, s_{a_t}, p_{a,t}) \in X_a\}_{a=1}^n \mid \sum_{a=1}^{n} s_{a,t} = S_t\} \subset \hat{X}. \tag{7}$$

Note that there is also an inherent asymmetry between the reserve currency and the community token. Community tokens cannot be introduced or removed without

[8]Cf. Proposition 1 of [29] for a proof of this assertion.

doing so through the bonding curve, meaning that community tokens are *internal* to the system. By contrast, the reserve currency can be introduced to or removed from the system without recourse to an internal mechanism—although the reserve currency is assumed to be globally conserved (when considering its holdings outside of the system), it is not locally conserved and is thus *external* to the system. The bonding curve then takes the role of *interface* between the two value systems, with one broader in scope (where the reserve currency originates) and one narrower in scope (where the specialized community token is used).

Definition 11 The **configuration space**, $\mathbf{X_C}$ is a $3n$-manifold, which is the Cartesian product of the system-level and agent-level configuration spaces.

$$\mathbf{X_C} := \hat{\mathbf{X}}_\mathbf{C} \times \bar{\mathbf{X}}_\mathbf{C} \subset \mathbf{X} = \mathbb{R}_{++}^{3n+3} \tag{8}$$

3.2 Mechanisms

In order to arrive at the laws of motion for the system, it is necessary to characterize the specific bonding curve mechanisms for reserve currency and community token dynamics. In addition, we include a specification of admissible agent actions, to close the feedback mechanism between these actions and the resulting realized signal process. We continue to use [29] as our framework in what follows.

Definition 12 The **Bond-to-Mint** mechanism takes a system-level state $\bar{x}_t = (R_t, S_t, P_t)$ and an agent a's action, given by a bonded quantity $\Delta R_t := r_{a,t} - r_{a,t+1} \geq 0$ such that $r_{a,t+1} \in \mathbb{R}_{++}$. Quantity ΔR_t is reserve currency transferred to the bonding curve, and returns the state x_{t+1} such that

$$(R_{t+1}, S_{t+1}, P_{t+1}) = \left(R_t + \Delta R_t, \sqrt[\kappa]{V_0(R_t + \Delta R_t)}, \frac{\kappa(R_t + \Delta R_t)}{\sqrt[\kappa]{V_0(R_t + \Delta R_t)}} \right)$$

and the associated updates to the agent-level state $\hat{x}_{a,t}$ are given by,

$$(r_{a,t+1}, s_{a,t+1}, p_{a,t+1}) = \left(r_t - \Delta R_t, s_{a,t} + \sqrt[\kappa]{V_0(R_t + \Delta R_t)}, g_a(x_{t+1}, \varepsilon_t) \right)$$

and $(r_{a',t+1}, s_{a',t+1}, p_{a',t+1}) = (r_{a',t}, s_{a',t}, g_{a'}(x_{t+1}, \varepsilon_t)$ for all agents $a' \neq a$ where g_a is a private mapping for agent a and ε_t is an exogenous signal.

Definition 13 The **Burn-to-Withdraw** mechanism takes a system-level state $\bar{x}_t = (R_t, S_t, P_t)$ and an agent a's action, given by a burned quantity $\Delta S_t := s_{a,t+1} - s_{a,t} \leq 0$ such that $s_{a,t+1} \in \mathbb{R}_{++}$. Quantity ΔS_t is token supply removed from the system, and results in the state x_{t+1} such that

$$(R_{t+1}, S_{t+1}, P_{t+1}) = \left(\frac{(S_t + \Delta S_t)^\kappa}{V_0}, S_t + \Delta S_t, \frac{\kappa(S_t + \Delta S_t)^\kappa}{V_0(S_t + \Delta S_t)} \right)$$

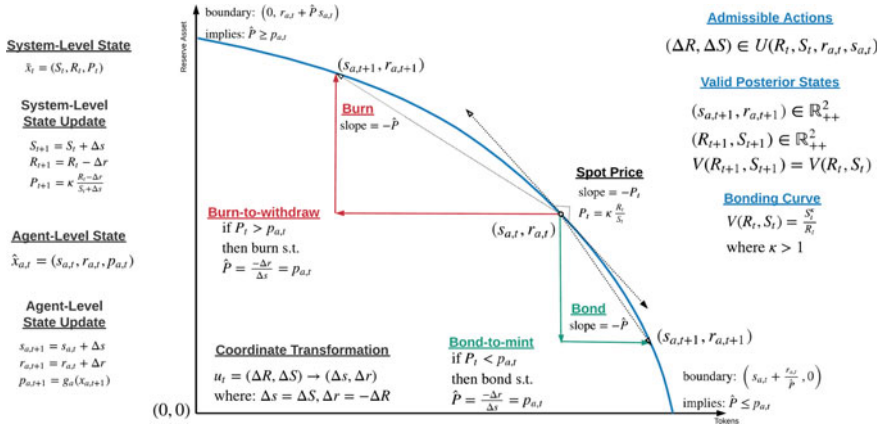


Fig. 3 An agent's view of their admissible actions given by the bonding curve

and the associated updates to the agent-level state $\hat{x}_{a,t}$ are given by,

$$(r_{a,t+1}, s_{a,t+1}, p_{a,t+1}) = \left(r_{a,t} + R_t - \frac{(S_t + \Delta S_t)^\kappa}{V_0}, s_{a,t} + \Delta S_t, g_a(x_{t+1}, \varepsilon_t)\right)$$

and $(r_{a',t+1}, s_{a',t+1}, p_{a',t+1}) = (r_{a',t}, s_{a',t}, g_{a'}(x_{t+1}, \varepsilon_t)$ for all agents $a' \neq a$ where g_a is a private mapping for agent a and ε_t is an exogenous signal.

Given these mechanisms, an agent's action set $U(x_t, a)$ can be fully determined from both agent-level restrictions and system-level conservation laws (see also Fig. 3 for a visualisation of the agent's perspective):

Definition 14 An action $u_t := (\Delta R_t, \Delta S_t)$ is **admissible** if at time t:

$$u_t \in U(x_t, a) := \hat{U}(x_t, a) \cap \bar{U}(x_t) \; \forall t,$$

for agent a, where the agent-level admissibility condition is:

$$\hat{U}(x_t, a) = \{(\Delta R_t, \Delta S_t) \in \mathbb{R}^2 \mid (r_{a,t} - \Delta R_t, s_{a,t} + \Delta S_t) \in \mathbb{R}^2_{++}\}$$

and the the system-level admissibility condition is

$$\bar{U}(x_t) = \{(\Delta R_t, \Delta S_t) \in \mathbb{R}^2 \mid (R_t + \Delta R_t, S_t + \Delta S_t) \in \mathbb{R}^2_{++}, \\ V(R_t + \Delta R_t, S_t + \Delta S_t) = V(R_t, S_t)\}.$$

3.3 Price Estimation

The bonding curve system is presented with a sequence of observations associated with the actions $u_t = (\Delta R, \Delta S)$, from which sample prices $\hat{P}_t = \frac{\Delta R}{\Delta S}$, referred to as *realized prices*, are computed. Due to the restrictions $U(x_t, a)$ on admissible actions, $\hat{P}_t$ is not necessarily a true sample of $p_{a,t} = g_a(x_t, \varepsilon_t)$ for the active agent a. Assuming agent a is acting at time t and that the mapping g_a accounts for any private signals and utility functions (including discounting), the sample price $\hat{P}_t$ may be interpreted as arising from an agent level constrained optimization, for example:

$$u_t = \underset{(\Delta R, \Delta S) \in U(x_t, a)}{\arg\min} \|\Delta S\, g_{a,t}(x_t, \varepsilon_t) - \Delta R\|. \tag{9}$$

Due to the dimensional restrictions in the configuration space, the admissible u_t for agent a lies within an open interval embedded in the plane (r_a, s_a), as shown in Fig. 3. It suffices for agent a to search this interval for their preferred posterior state, and to choose $u_t = (r_{a,t} - r_{a,t+1}, s_{a,t+1} - s_{a,t})$ accordingly. Whether or not u_t is treated as a strategic action, as in (9), the curvature $\kappa > 1$, implies that every point in the open interval $U(x_t, a)$ is uniquely characterized by the price $\hat{P}_t = \frac{\Delta R}{\Delta S}$. Furthermore, the estimator $P_t = G(x_t) = \kappa \frac{R_t}{S_t}$ is a critical point where $\hat{P}_t > P_t$ will always call for burning, and $\hat{P}_t < P_t$ will always call for bonding (see [29], Lemmas 1 and 2). Also from [29], the posterior spot price always decreases for burn actions, and always increases for bond actions. Thus it is guaranteed that the update directions match, that is $(P_{t+1} - P_t)(\hat{P}_t - P_t) \geq 0$ when any agent a takes an action u_t at time t.

Combining this machinery with the assumption that agents act directionally aligned with their preferences $(g_a(x_t, \varepsilon_t) - P_t)(\hat{P}_t - P_t) > 0$, the groundwork is laid for deriving estimation error bounds of the form $\|g_a(x_t, \varepsilon_t) - G(x_t)\| = \|p_{a,t} - P_t\| \leq \xi \|\hat{P}_t - P_t\|\, \forall a \forall t$, with minimal assumptions regarding the agents. As a first step we proceed next to computational experiments, which test the input-out response dynamics of the bonding curve system visualized in Fig. 4; in particular we compare the estimator P_t to sample sequences $\hat{P}_t$ and the associated (unique) actions $u_t \in \bar{U}(x_t)$. Estimation error is given by $e_t = \|\hat{P}_t - P_t\| \forall t$.

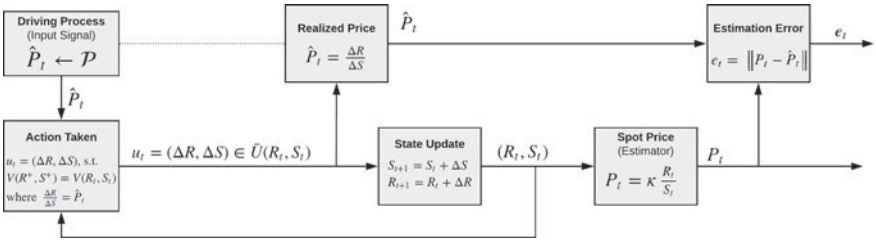


Fig. 4 Block Diagram representation of the input-output response dynamics for the bonding curve, viewed as an price estimator

4 Price Estimator Response Dynamics

Consider a sequence of realized prices $\hat{P}_t$; for any prior system-level state $\bar{x}_t$, there is a unique $u_t = (\Delta R_t, \Delta S_t)$ satisfying $\Delta S_t \hat{P}_t = \Delta R_t$ provided that $\hat{P}_t > R_t/S_t$. The price R_t/S_t, also called the *floor price*, is the ratio of reserve-to-supply, and represents the realized price of liquidating the bonding curve at any time t. It thus acts as a lower bound for all realized prices $\hat{P}_t$. By restricting attention to the sequence $\hat{P}_t$, it is possible to analyze the signal processing properties of the bonding curve using only the system-level trajectory $\bar{x}_t$.

Definition 15 The **driving process** $\mathcal{P}$ generates a sequence of price samples $\hat{P}_t$ satisfying the condition that $\hat{P}_t > \frac{R_t}{S_t}$ for all t.

The input-output response dynamics of the bonding curve system are constructed by comparing the inputs $\hat{P}_t$ to the outputs P_t for deterministic wave-forms as well as non-deterministic input signals $\hat{P}_t$, given a particular characterizations of the bonding curve discrete event game.

4.1 Experimental Apparatus

To illustrate the impact on P_t of different driving processes, three deterministic signals are defined: a Square-Wave, a Triangle-Wave and a Sine-Wave. To capture stochastic effects, a Martingale stochastic process is also introduced. These simple signals most commonly used in basic estimation experiments abstract away from the complexity of possible inputs $\hat{P}_t$, while simultaneously acting as a starting point for the analysis of a broad range of feedback mechanisms caused by a closed-loop game. The deterministic signals can be characterized by wavelength λ, amplitude A and phase ϕ, for time $t \in \{0, \ldots 4000\}$, and are described using the functional forms in Table 1. The apparatus supports testing across a range frequencies λ, magnitudes A and phase shifts ϕ in accordance with best practices [12].

Our experiments use the bonding curvature parameter $\kappa = 2$ and the system is initialized with a community token supply $S_0 = 1000000$ and reserve currency units $R_0 = 50000$, resulting in an initial price $P_0 = 0.10$ reserve units per token and an invariant $\bar{V} = V_0 = 20000000$. The deterministic driving functions are taken with $\phi = 0$, $B = P_0$ and $\lambda = 2000$. Amplitude A takes values $\frac{P_0}{2}$, $\frac{P_0}{100}$, and $\frac{P_0}{2}$ for the Square-Wave, Triangle-Wave and Sine-Wave, respectively. For the Random Walk, an initial condition $\hat{P}_0 = P_0$ is applied, and the percent change in $\hat{P}_0$ is drawn from a Gaussian distribution with mean $\mu = 0$ and variance $\sigma = 0.05$. Additionally, 10×10-run Monte Carlo experiments were executed for the Martingale case, generating 10 runs for each $\sigma \in \{0.1/2^K | K = 1, \ldots 10\}$.[9]

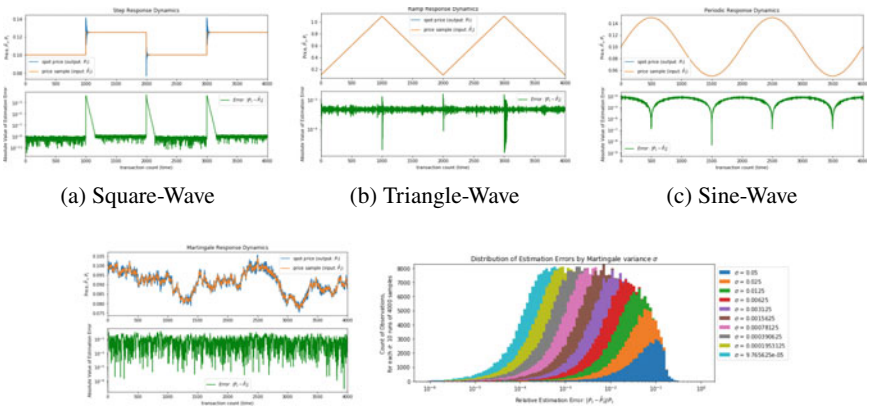
[9]These reflect a sampling of the permissible values of σ—a more detailed analysis is relegated to future research.

Table 1 Driving process functional forms for numerical experiments

Waveform	Driving process $\mathcal{P}$	Restriction
Square-Wave	$\hat{P}_t(t,\lambda,A,\phi)$ $= B + A\,\mathbb{1}_{\{((t-\phi) \mod \lambda)<\lambda/2\}}$	$B > A$
Triangle-Wave	$\hat{P}_t(t,\lambda,A) = B +$ $\frac{2A}{\lambda}\left\lvert((t-\phi) \mod \lambda) - \frac{\lambda}{2}\right\rvert -$ $\frac{2A}{4}$	$B > \frac{2A}{\lambda}$
Sine-Wave	$\hat{P}_t(t,\lambda,A,\phi)$ $= B + A\sin\left(\frac{2\pi t-\phi}{\lambda}\right)$	$B > A$
Random Walk	$\hat{P}_t(t,\mu,\sigma) = (1+\delta_t)\hat{P}_{t-1}$ where $\delta_t \sim N(\mu,\sigma)$	$\mu = 0$

4.2 Numerical Results

The Square-Wave response in Fig. 5a shows that the step response is tightly tuned, resulting in a large overshoot but remaining stable, oscillating and converging quickly. This behavior is characteristic of a high gain proportional controller. The Triangle-Wave (Fig. 5b) and Sine-Wave (Fig. 5c) signals are equally reminiscent of such a controller; the Triangle-Wave exhibits steady state error during the ramp and the Sine-Wave tracks most closely at the peaks and troughs. In the Random Walk case (Fig. 5d) tracking behavior is observed, but the error radius appears large. Despite the high frequency noise, the error does not appear to accumulate, and does appear to remain within a ball roughly on the order of 3σ (Fig. 5e), likely an artifact of the random walk whereby $\frac{\Delta\hat{P}_t}{\hat{P}_t} \sim N(0,\sigma)$ for all t.



(a) Square-Wave (b) Triangle-Wave (c) Sine-Wave

(d) Random Walk (e) Repeated Martingale Param Sweep

Fig. 5 Summary of response dynamics and estimation error for the experiments in Sect. 4.1. Single trajectories are shown for (**a**) to (**d**) while (**e**) shows the distribution of relative estimation errors for various σ-variance Martingales (**d**)

5 Conclusions and Future Work

The novel equipment provided by dynamic games is the system-level state, and the ability to define and enforce a configuration space that is a proper subset of the state space using mechanism design techniques related to state feedback control. Applying estimation and control-theoretic principles to token bonding curves, analytical groundwork was developed to characterize the relationship between realized prices and spot prices, as well as to posit an estimation bound relating the spot price and the agents' hidden preferences. Numerical experiments demonstrated the signal processing characteristics of the bonding curve, and provide further evidence that expanding the discrete event game machinery and its associated estimation capabilities will provide new tools for practical mechanism design, with a focus on cryptoeconomic and cyber-physical systems.

References

1. Abadi, J., & Markus, B. (2019, August). *Blockchain economics*.
2. Caginalp, C., & Caginalp, G. (2018, May). *Cryptocurrency equilibria through game theoretic optimization*.
3. Cortés, J., & Egerstedt, M. (2017). Coordinated control of multi-robot systems: A survey. *SICE Journal of Control, Measurement, and System Integration*, *10*, 495–503.
4. Foley, D. K. (1994). A statistical equilibrium theory of markets. *Journal of Economic Theory*, *62*(2), 321–345.
5. Fudenberg, D., & Levine, D. (1998). *The theory of learning in games*. MIT Press.
6. Fudenberg, D., & Tirole, J. (1991). *Game theory*. Cambridge, MA: MIT Press.
7. Hart, S. (2005, August). *Adaptive heuristics*.
8. Hofbauer, J., & Sandholm, W. (2009, July). Stable games and their dynamics. *Journal of Economic Theory*, *144*, 1665–1693.
9. Isaacs, R. (1999). *Differential games: A mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation.
10. Jahandideh, et al. (2019). Hybrid rebeca: Modeling and analyzing of cyber-physical systems. In R. Chamberlain, W. Taha, & M. Törngren (Eds.), *Cyber physical systems. Model-based design* (pp. 3–27). Springer International Publishing.
11. Jovanovic, B., & Rosenthal, R. (1988). Anonymous sequential games. *Journal of Mathematical Economics*, *17*(1), 77–87.
12. Kay, S. M. (1993). *Fundamentals of statistical signal processing: Estimation theory*. USA: Prentice-Hall Inc.
13. Kia, S. S., Van Scoy, B., Cortes, J., Freeman, R. A., Lynch, K. M., & Martinez, S. (2019). Tutorial on dynamic average consensus: The problem, its applications, and the algorithms. *IEEE Control Systems Magazine*, *39*(3), 40–72.
14. Liu, Z., Luong, N. C., Wang, W., Niyato, D., Wang, P., Liang, Y. C., & Kim, D. I. (2019). *A survey on applications of game theory in blockchain*.
15. Lux, T. (2009). Applications of statistical physics in finance and economics. In *Handbook of research on complexity*. Edward Elgar.
16. Marden, J. R., & Shamma, J. S. (2015). Game theory and distributed control. In *Handbook of game theory with economic applications* (Vol. 4, pp. 861–899). Elsevier.
17. Monderer, D., & Shapley, L. (1996). Potential games. *Games and Economic Behavior*, *14*(1), 124–143.

18. Nakamoto, S. (2008, December). *Bitcoin: A peer-to-peer electronic cash system*.
19. Pan, J., & Manocha, D. (2015). Efficient configuration space construction and optimization for motion planning. *Engineering*, *1*(1), 046–057. https://doi.org/10.15302/J-ENG-2015009, http://www.sciencedirect.com/science/article/pii/S2095809916300443.
20. Rapoport, A., Chammah, A. M., & Orwant, C. J. (1965). *Prisoner's dilemma: A study in conflict and cooperation* (Vol. 165). University of Michigan Press.
21. Sandholm, W. H. (2010). *Population games and evolutionary dynamics*. MIT Press.
22. Shorish, J. (2018, January). Blockchain state machine representation. *SocArXiv*.
23. Singh, R., Dwivedi, A., Srivastava, G., Wiszniewska-Matyszkiel, A., & Cheng, X. (2019, November). *A game theoretic analysis of resource mining in blockchain*.
24. Topkis, D. M. (1998). *Supermodularity and complementarity*. Princeton Press.
25. Voshmgir, S., & Zargham, M. (2019, November). Foundations of cryptoeconomic systems. *Working paper series*, Research Institute for Cryptoeconomics, Vienna.
26. Wang, X., Xiao, N., Xie, L., Frazzoli, E., & Rus, D. (2014, December). Discrete-time mean field games in multi-agent systems. In *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)* (pp. 711–716).
27. Young, H. P. (2005). *Strategic learning and its limits*. Oxford University Press.
28. Zargham, M., Ribeiro, A., Ozdaglar, A., & Jadbabaie, A. (2014). Accelerated dual descent for network flow optimization. *IEEE Transactions on Automatic Control*, *59*(4), 905–920.
29. Zargham, M., Shorish, J., & Paruch, K. (2019, December). From curved bonding to configuration spaces. *Working paper series*, Research Institute for Cryptoeconomics, Vienna.
30. Zargham, M., Zhang, Z., & Preciado, V. (2018). A state-space modeling framework for engineering blockchain-enabled economic systems. In *NECSI ICCS*. ArXiv.org.
31. Zhang, Z., Zargham, M., & Preciado, V. (2020). On modeling blockchain-enabled economic networks as stochastic dynamical systems. *Applied Network Science: Blockchain and Cryptocurrency*, *5*, 19.

Promise: Leveraging Future Gains for Collateral Reduction

Dominik Harz, Lewis Gudgeon, Rami Khalil, and Alexei Zamyatin

Abstract Collateral employed in cryptoeconomic protocols protects against the misbehavior of economically rational agents, compensating honest users for damages and punishing misbehaving parties. The introduction of collateral, however, carries three disadvantages: (i) requiring agents to lock up substantial amount of collateral can be an entry barrier, limiting the set of candidates to wealthy agents; (ii) affected agents incur ongoing opportunity costs as the collateral cannot be utilized elsewhere; and (iii) users wishing to interact with an agent on a frequent basis (e.g., with a service provider to facilitate second-layer payments), have to ensure the correctness of each interaction individually instead of subscribing to a service period in which interactions are secured by the underlying collateral. We present Promise, a subscription mechanism to decrease the initial capital requirements of economically rational service providers in cryptoeconomic protocols. The mechanism leverages future income (such as service fees) prepaid by users to reduce the collateral actively locked up by service providers, while sustaining secure operation of the protocol. Promise is applicable in the context of multiple service providers competing for users. We provide a model for evaluating its effectiveness and argue its security. Demonstrating Promise's applicability, we discuss how Promise can be integrated into a cross-chain interoperability protocol, XCLAIM, and a second-layer scaling protocol, NOCUST. Last, we present an implementation of the protocol on Ethereum showing that all functions of the protocol can be implemented in constant time complexity and Promise only adds USD 0.05 for a setup per user and service provider and USD 0.01 per service delivery during the subscription period.

D. Harz (✉) · L. Gudgeon (✉) · R. Khalil · A. Zamyatin
Department of Computing, Imperial College London, London, UK
e-mail: d.harz@imperial.ac.uk

L. Gudgeon
e-mail: l.gudgeon18@imperial.ac.uk

P. Pardalos et al. (eds.), *Mathematical Research for Blockchain Economy*, Springer Proceedings in Business and Economics,
https://doi.org/10.1007/978-3-030-53356-4_9

1 Introduction

Since their creation, arguably the most significant property of blockchains is their facilitation of trustless exchange between entities with weak identities [8]. Yet the trustless nature of the systems means not only that parties *may* transact without trusting each other, but also that they *should not trust* each other. This creates a design challenge for interactions which would typically involve such trust. In this paper, we focus on blockchain protocols which, at least in part, encode trust by monetary collateral. Here, collateral is value escrowed by a service provider, Alice, to guarantee the user, Bob, that regardless of the behavior of Alice, Bob cannot lose funds. In particular, payment, cross-chain, and generic computation protocols can be designed such that Bob is guaranteed to receive from Alice at least the amount of funds that are at risk in case she misbehaves. Protocols involving collateral include cross-chain communication [21], scalable off-chain payments [15], state channels [11], watchtowers [4, 5, 16], and outsourcing of computation and verification games [20].

Problem. Relying on collateral as trust is itself associated with a set of challenges. Collateralization requires the provision of a substantial amount of funds upon protocol initialization, limiting the set of participants to a selected few. Leaving participation to a small set of agents can lead to phenomena like the "rich are getting richer" through wealth compounding [12]. While it is not possible to grant less wealthy agents proportionally higher rewards due to Sybil identities [9], we can lower the entry barrier for agents to join a protocol. Finally, locked funds result in opportunity costs for the agent who could use their collateral for participating in other protocols [14].

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

Application. We discuss how Promise can be applied to XCLAIM and NOCUST. Both protocols are suitable candidates for Promise, as in both protocols "service providers" are a necessary part. XCLAIM is a cross-chain protocol that allows creation of Cryptocurrency-backed Assets (CbA) on an issuing blockchain enabled by a collateralized third-party called a *vault* [21]. Vaults provide collateral on the issuing blockchain to ensure that it is not economically rational for them to steal the locked cryptocurrency on the backing blockchain. NOCUST is a commit-chain protocol that allows to send cryptocurrency payments off-chain facilitated by so-called *operators* [15]. Operators are service-providing agents that (i) collect fees for operating the off-chain payment network, and (ii) provide a certain amount of collateral to insure finalization of payments. Both, vaults and operators are service providers from the perspective of Promise: (i) any agent can become a service provider by locking a certain amount of collateral, and (ii) agents can earn fees by providing their services. Moreover, we show that the implementation of Promise in an Ethereum Solidity smart contract only adds USD 0.03 to setup Promise between a user and a service provider, USD 0.01 to provide the deposit for the service provider and USD 0.01 to provide the pre-payment. During the subscription period, each delivery of the service adds a cost of USD 0.01. Finally, the withdrawal of both deposit and accumulated payments adds USD 0.01 for the service provider.

Outline. We introduce the system model and assumptions in Sect. 2, followed by a description of Promise in Sect. 3. Next, we discuss the security of Promise and argue in which cases Promise can provide benefits to users and intermediaries in Sect. 4. Also, we present how Promise can be applied to existing systems in Sect. 5. We discuss related work in Sect. 6 and conclude in Sect. 7.

2 System Model

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

Formally, we adopt the definitions of agreements $\mathcal{A}$ in cryptoeconomic protocols from [14]. The service providing agent Alice A and the receiving agent Bob B participate in an agreement encoded by a specification Φ, payments p and a deposit D. In such an agreement, Alice needs to fulfill the specification Φ and provide the collateral D in advance. When Alice fulfills the specification, all future payments p held in escrow are released to Alice.

Table 1 Symbols used in Promise

Symbol	Description
V_i	Total value of a task to agent i
D	A monetary deposit
$\mathcal{A}$	An agreement reached between a service provider and a user
Φ	A protocol specification, specifying the task for the service provider and the required proof that the task has been performed
p	Payment held in escrow and released to the providing agent on fulfillment of the agreement
m	The number of future periods
π	A generic cryptoeconomic protocol
c	The cost of an individual transaction
$\mathrm{E}[r]$	The expected rate of return
$u_i(t)$	The utility of agent i at time t
β	The likelihood that the user remains in the protocol
n	The number of times Alice did not deliver the service
δ	Discount factor for future utility

Promise is a mechanism to reduce initial collateral locking. However, Promise is not meant as a stand-alone protocol, but rather, serves as a "plug-in" to existing cryptoeconomic protocols. Given a generic cryptoeconomic protocol π that satisfies the assumptions of Sect. 2.3, we can apply Promise and write the protocol as π_{P}. We note that the agreement $\mathcal{A}$ is given by the generic protocol π. We assume that Alice and Bob have entered into agreement $\mathcal{A}$ and have agreed on the specification Φ, payments p, and the deposit D.

We give a summary of symbols in Table 1.

2.1 Specifications

The specification Φ describes the *task* that Alice needs to provide and the *pf* that serves as evidence that the task has been provided. There are several approaches to encode the specification. In the BitML calculus [6], a specification consists of (i) a model describing a contract and agent choices symbolically and (ii) a model encoding a sequence of transactions that form a smart contract in Bitcoin computationally. For example, Alice can deliver a digital good to Bob in return for a payment. The contract would then specify that if Bob receives the good, payment to Alice is being made. Specifications are also useful when exchanging digital goods with the FairSwap protocol [10]. In FairSwap, Alice sends a digital good to Bob and provides proof of

sending by providing a witness (hashes of the transferred data) to a smart contract as proof. The specification in FairSwap is encoded as a boolean circuit that evaluates whether the provided witness (the hash) satisfies the specification. In case the circuit evaluates to true, Alice is paid for delivering the data.

Expressing the specification abstractly gives us the freedom to leave the encoding and implementation up to the protocol that integrates with Promise. For the remainder of the model for Promise, we assume the specification can either be fulfilled, i.e., $\Phi = 1$ or not, i.e., $\Phi = 0$.

2.2 Roles

Promise adopts the BAR model of rational agents [2] including private preferences of agents as proposed in [14]. We define the following roles.

- **Alice**, the Intermediary: Alice is economically rational and entrusted with executing a task. She provides a deposit D into the escrow before executing the task and receives m payments p upon successful completion. Alice prefers to adhere to the specification Φ if her utility for doing so is greater than other action choices.
- **Bob**, the User: Bob represents the user requesting execution of a task by Alice. A user provides payments $\{p_1, \ldots, p_m\}$ into the escrow. The user is assumed to be honest and correctly reports behavior of Alice.
- **Escrow**: The escrow is a smart contract responsible for holding deposits by Alice and payments by Bob.
- **Verifier**: The verifier detects malicious behavior of Alice. In practice, this role is fulfilled by a smart contract, a dedicated third party, or the user.

2.3 Assumptions

The verifier in the system is able to detect any faults by Alice and is able to prove that Alice was at fault. This means, that the specification Φ of the protocol π has some "proof". For example, this could be the hash input of a boolean circuit as in FairSwap, a transaction inclusion proof as required by XCLAIM, or fraud-proofs [3].

We further assume that the protocol utilizing Promise implements payments and deposits through a ledger functionality (e.g., as described in [13]). Also, there is a one to one mapping between the collateral and a user, such that the collateral of an intermediary is not split between multiple users. Agents in the system can be identified with their public/private key pair. Finally, time is denoted with t.

2.4 Utilities

In our model, we assume that agents are economically rational and self-interested. An agent will therefore decide on a course of action depending on the utility associated with those actions. We use a simplified model here, were the intermediary Alice can choose between two actions and Bob has no choice once he committed to the agreement $\mathcal{A}$.

Alice can either fulfill the specification or not, with the following payoffs one period ahead. V_A denotes the additional monetary gain that Alice expects to receive if she chooses to deviate from the protocol, where $V_A \geq 0$. We only include a valuation on the malicious side to Alice as Alice could be bribed to violate the specification. This is a worst case assumption: Alice can only be influenced by increasing her incentive to misbehave. While we could also include a positive valuation for honest behavior, this would not strengthen our security assumptions.

V_B denotes the monetary value that Bob attaches to receiving the service. Note that we assume *private information*: Bob does not know Alice's private valuation V_A, and Alice does not know Bob's valuation of the service V_B.

Last, c denotes the cost of an individual transaction. $\mathrm{E}[r]D$ reflects the expected opportunity cost of locking the capital for one period where E[ ] denotes an expected value and r is a rate of return. The rate of return indicates the potential interest an agent could earn by participating in another protocol. For example, instead of locking D in the protocol, Alice could trade D, lock D in staking [12] or lending [1] protocols to earn an interest.

$$u_A = \begin{cases} p - \mathrm{E}[r]D, & \text{if } \Phi = 1 \\ V_A - \mathrm{E}[r]D - D, & \text{if } \Phi = 0 \end{cases} \tag{1}$$

$$u_B = \begin{cases} V_B - p - c, & \text{if } \Phi = 1 \\ D - V_B - c, & \text{if } \Phi = 0 \end{cases} \tag{2}$$

Each round the game resets. Therefore, Alice fulfills the specification iff

$$p > V_A - D \tag{3}$$

Assuming that $V_B - p - c > 0$, otherwise Bob would not seek the service from Alice in the first place, he stands to gain utility if Alice provides the service.

2.5 Security

Following the rational agents assumptions and the utility definitions in Eqs. (1) and (2), we define a secure cryptoeconomic protocol as follows.

Definition 1 (*Security*) Assuming rational service provider A with a private valuation V_A, a cryptoeconomic protocol π implementing a specification Φ is secure if A's utility u_A for fulfilling the specification Φ is higher than her utility for violating the specification Φ, i.e., $p > V_A - D$.

In the following, we introduce Promise in detail. We use Definition 1 to show that integrating Promise into a generic protocol π does not affect its security. The core proof is to show that π and π_P are equivalent with respect to their security.

3 Promise

In Promise we allow Bob to provide multiple payments in advance and delay the receipt of the payments by Alice. In turn, Alice is able to reduce the initially provided collateral from D to D_I such that $D_I < D$. At $t = 0$, Bob is able to lock m payments $\{p_1, \ldots, p_m\}$ in escrow and determine a period τ after which Alice can receive the payments. When $t < \tau$, Alice continues to accumulate collateral as time passes by keeping the cumulative total of her payments p_i in escrow. We provide an intuition in Fig. 1. Promise has the following advantages for Alice and Bob.

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

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

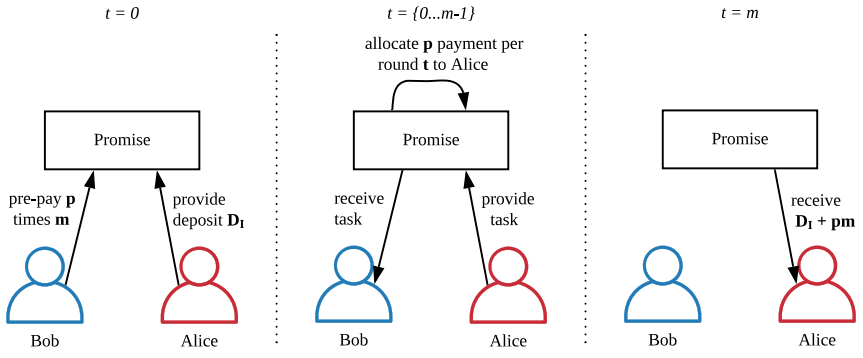


Fig. 1 Promise allows intermediaries (Alice) to lock less initial deposit D_I and use payments p_i provided by users (Bob) as additional deposit. The initial deposit and payments are locked until time m determined by Bob. Only when Alice fulfills the specification Φ until $t = m$ can Alice withdraw her initial deposit D_I and the total payments pm

3.1 Protocol

The Promise protocol consists of three steps. We denote the service provider as A, the user as B, and the smart contract implementing Promise as P. We assume that A and B have agreed the total payment and the period over which the payment is to-be-paid in advance.

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

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

3.2 Sequential Games and Discounting

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

Without Promise, at each round t, Alice decides if she prefers to fulfill the specification based on the utilities denoted by Eqs. (1) and (2).

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

Hence, Alice's decision needs to account for all $p \cdot m$ payments. Furthermore, payments are made in the future. A promised payment in the future is less valuable to Alice today, which we denoted with the parameter δ. $0 < \delta < 1$ denotes the discount factor of an agent's valuation of future utility. We argue that an agent can spend received payments somewhere else or potentially invest the payment for a profit. Hence, the service provider faces an opportunity cost for delayed payments. The payoffs to Alice, if she follows the same course of action over every round, are as follows.

$$u_A(t) = \begin{cases} \sum_{t=0}^{m-1} \left(\frac{\delta}{1+r}\right)^t (p - (t+1)\mathrm{E}[r]p - \mathrm{E}[r]D_I), & \text{if } \Phi = 1 \\ \sum_{t=0}^{m-1} \left(\frac{\delta}{1+r}\right)^t (V_A - \mathrm{E}[r]D_I - D_I), & \text{if } \Phi = 0 \end{cases} \tag{4}$$

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

$$u_B(t) = \begin{cases} \sum_{t=0}^{m-1} \left(\frac{\delta}{1+r}\right)^t (V_B - p - c - (m-t)\mathrm{E}[r]p), & \text{if } \Phi = 1 \\ \sum_{t=0}^{m-1} \left(\frac{\delta}{1+r}\right)^t (D_I - V_B - c), & \text{if } \Phi = 0 \end{cases} \tag{5}$$

3.3 Termination Probability

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

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

$$u_A(t) = \begin{cases} \sum_{t=0}^{m-1} \left(\frac{\delta}{1+r}\right)^t (p - (t+1)\mathrm{E}[r]p - \mathrm{E}[r]D_I), & \text{if } \Phi = 1 \\ \sum_{t=0}^{m-1} \left(\frac{\delta}{1+r}\right)^t (\beta(n)V_A - \mathrm{E}[r]D_I - D_I), & \text{if } \Phi = 0 \end{cases} \tag{6}$$

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

4 Analysis

The core argument of Promise is that by locking multiple payments, service providers can reduce their initial collateral. Specifically, introducing Promise to a protocol π, does not increase the incentive for an intermediary A to not adhere to the specification. This means that π_{P} and π are equivalent in terms of security considering an economically rational intermediary A under Definition 1. More formally, we state that:

Theorem 1 (Security equivalence) *Given a protocol π that has a verifiable specification Φ and an economically rational service provider A that provides more initial collateral D_I than an incentive V_A to violate the specification, introducing Promise is secure if A does not gain additional utility by not fulfilling the specification considering A participates in at least two rounds in π.*

4.1 Action Choices

Alice's utility for choosing a specific course of action, i.e., fulfilling versus not fulfilling the specification, is given by Eq. (6). However, this makes an implicit assumption: Alice considers the entire period m as a basis for her decision. We depict her added utilities for an example of two rounds in Fig. 2.

Showing that Theorem 1 holds, requires considering that Alice might not participate for m rounds. Specifically, Alice might still consider the agreement as a single-shot game with a decision horizon of exactly one round. Following this, we can use $m = 1$ and Eq. (4) to conclude that Alice prefers to fulfill the specification if:

$$p - \mathrm{E}[r]p > \beta(n)V_A - D_I \tag{7}$$

Collateral Condition Comparing Alice's decision without Promise in Eq. (3) and her decision with Promise in Eq. (7) without considering β clearly shows that if Alice only considers a single round, introducing Promise *weakens* the security of π as Alice is not paid immediately and the initial collateral is reduced. Moreover, even if Alice considers multiple rounds, if Eq. (7) does not hold, Alice *has a higher utility*

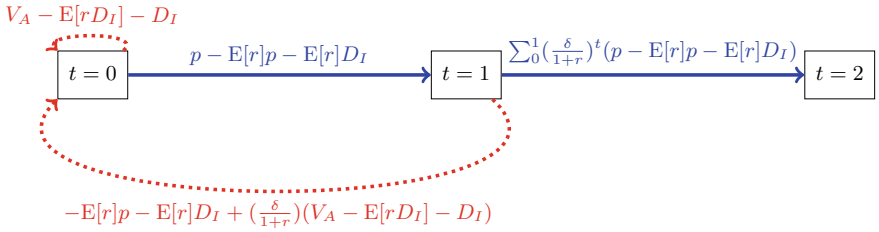


Fig. 2 Depicting the sum of utilities depending on different action choices made by Alice. At $t = 0$ Alice can choose between fulfilling the specification and receive the utility depicted in blue or choose the opposite and receive the utility depicted in red. If Alice at any point prefers to violate the specification, the game restarts and the action choices are essentially back to the $t = 0$ state. Furthermore, at $t = 1$, Alice will already have committed to adhering to the specification. In case Alice decides to misbehave at this point, she will not receive p that she was allocated when she transitioned to t_1. However, if she decides to continue to fulfill the specification, she will be rewarded with an additional payment allocation. This game continues until $t = m$

to not fulfill the specification if Bob is willing to continue to enter into agreements with her. Even worse, if Bob decides to continue using the protocol π and black-lists Alice for her violating Φ, Bob still might end up with a Sybil identity of Alice. Hence, for Promise to not weaken security the initial collateral needs to be set above $\beta(n)V_A - p + \mathrm{E}[r]p$.

In practice, this is achieved by over-collateralization or state-reversal. Over-collateralization is used in XCLAIM where a vault has to provide 200% collateral of the value it stands to obtain by violating the specification. In NOCUST and, generally, payment channel networks, participants are not able to steal funds, since an older state can be committed that reverses the stealing of funds.

4.2 Security Proof

Proof Under the assumption that A is economically rational, wants to participate in at least two rounds, e.g. $t > 0$, and Eq. (7) holds, we prove that π_{P} is secure. If Eq. (7) holds, Alice should fulfill the specification in the first round. We now show that if this holds, Alice should continue with the same course of actions in any subsequent round $t \in m$. 0375+ If if at any point $k \in m$, Alice decides to stop adhering to the specification she will receive the following pay-off:

$$u_A(t,k) = \sum_{t=0}^{k-1} \left(\frac{\delta}{1+r}\right)^t (-(t+1)\mathrm{E}[r]p - \mathrm{E}[r]D_I) + \left(\frac{\delta}{1+r}\right)^t (\beta(n)V_A - \mathrm{E}[r]D_I - D_I) \tag{8}$$

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

4.3 Cost Reduction for Service Providers

Service providers can reduce their initial collateral to the lower bound under the condition of Eq. (7). From this equation, we can determine the reduction if Alice only considers a single round of the game. We express D_I as $D - \rho$ where ρ is the reduction of the initial collateral and solve for ρ This yields:

$$\rho = p - \mathrm{E}[r]p + D - \beta(n)V_A \tag{9}$$

However, if we consider that Alice wants to participate in m rounds, we can express this based on Eq. (6) and solving for ρ. However, we argue that under the assumption of Eq. (7), Alice's decision is essentially between participating a single round and not fulfilling the specification, or participating multiple rounds over the pre-agreed period m while adhering to the specification. To calculate Alice's decision bound, we are assuming that from Eq. (9) the first reduction is set to the lowest possible value. This means that the term $\beta(n)V_A - p + \mathrm{E}[r]p - D_I = 0$, i.e., at the decision bound Alice is undecided if she should fulfill the specification since the utilities for both choices are equal. Thus, ρ can be expressed as:

$$\rho = \sum_{t=0}^{m-1} \left(\frac{\delta}{1+r}\right)^t (p - (t+1)\mathrm{E}[r]p - \mathrm{E}[r]D) \tag{10}$$

In Sect. 5.1 we give an example how collateral is lowered given a set of parameters. Note that to calculate the collateral reduction, both the service provider and the user only need to know the prior collateral requirement D as defined by π. For example, in XCLAIM this is 200%.

4.4 Cost Reduction for Users

Assume that Alice behaves honestly. If a user pays every round t for the service provided by Alice, then his pay-off per round is $V_B - p - c$ as described in Eq. (2). However, locking multiple payments incurs opportunity cost. This cost is lowered at every time step as the payments are assigned to the intermediary, as expressed in Eq. (5).

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

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

$$\sum_{t=1}^{m} \left(\frac{\delta}{1+r}\right)^t c = \sum_{t=0}^{m} \left(\frac{\delta}{1+r}\right)^t \mathrm{E}[r]p(m-t) \tag{11}$$

$$c = \mathrm{E}[r]p(m-t) \tag{12}$$

Provided the right hand of Eq. (12) is smaller, Bob should use Promise to lock multiple payments pm as it is his individually rational choice that maximizes his pay-off u_B.

5 Applications

In this section we apply Promise to the XCLAIM protocol. We show analytically how Promise is able to reduce the initial amount of locked deposits. Further, we give a sketch how Promise can be implemented in NOCUST.

5.1 XCLAIM

XCLAIM is a protocol that allows users to transfer assets between heterogeneous decentralized ledgers using a collateralized service provider called a vault [21]. Instead of relying on a trusted third party like a centralized exchange, the vault must provide collateral to ensure that it does not steal the coins it holds in custody. It has to verify correctness of her actions by submitting transaction inclusion proofs to the smart contracts that augments the protocol. Promise can be applied such that the vault, Alice, locks some initial collateral D_I and issues backed-tokens using this collateral. Bob, using the service, is able to lock the future payments of Alice to allow him to transfer more assets between the ledgers.

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

- Initial Collateral D: A service provider needs to provide 200% collateral D in comparison to the value held in custody V_A.
- Payments p: Although payments are not specified in the original XCLAIM paper, similar services such as tBTC require users to pay 0.9375% of fees as payment.[1]
- Rate of return r: A service provider needs to lock collateral in the ETH currency to participate. Possible alternatives offer a maximum of 3.75% APR rates.[2]
- Discount factor δ: Service providers can discount future payments. As the price of cryptocurrencies is relatively volatile we adopt a strongly discounted future income at 0.75 from [14].

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

[1] Based on https://docs.keep.network/tbtc/index.pdf from 3 May 2020.

[2] Based on https://www.coingecko.com/en/earn/ethereum from 3 May 2020.

1. The user and a vault agree on a subscription period m. For example, the user and the vault can agree that the vault will be responsible for the next ten Bitcoin-backed tokens issued or redeemed by this users that are each 1 BTC in size.
2. The user and the vault set-up a Promise contract in which the user pre-pays $m = 10$ fees at 0.9375% of 1 BTC for the next ten requests at a set price of $p = 0.009375$ to issue or redeem Bitcoin-backed tokens.
3. The vault then deposits the initial deposit D_I into the contract.
4. Each time the user issues or redeems tokens with the vault, the vault is allocated a part of the payment p.
5. Finally, after ten requests have been made, the vault can withdraw pm and D_I.

Cost Reduction For simplicity, we are going to denote all monetary amounts in BTC. In XCLAIM, a vault would have to provide the equivalent of 2 BTC in collateral to hold custody over 1 BTC in value. First, we calculate the possible D_I collateral given Eq. (9). This gives us the minimum collateral required to also protect against a *vault that plays a single-shot game*. For simplicity, we are assuming that the vault has an incentive of 1 to steal the BTC (the current value of the Bitcoin) as well as a hidden motivation to steal BTC such that $V_A = D = 2$. Moreover, the vault is not interested in any future collaboration with the user, hence $\beta(n) = 1$. Last, we divide the 3.75% APR through 365 days to get the average return.

$$\begin{aligned} \rho &= p - \mathrm{E}[r]p + D - \beta(n)V_A \\ \rho &= 0.009375 - \frac{0.0375}{365}0.0093750 + 2 - 2 \\ \rho &= 0.00937404 \end{aligned} \tag{13}$$

Second, we are using Eq. (10) to explore the reduction factor ρ if the *vault plays a sequential game*. Note that, at a minimum, a vault has at least a private value of V_A to not follow the specification of XCLAIM: if the vault can take the 1 BTC and is punished with less collateral D_I being taken away, it is incentive compatible for the vault to take the 1 BTC. Using the example values above, we calculate ρ as:

$$\begin{aligned} \rho &= \sum_{t=0}^{9} \left(\frac{\delta}{1+r}\right)^t (p - (t+1)\mathrm{E}[r]p - \mathrm{E}[r]D) \\ \rho &= \sum_{t=0}^{9} \left(\frac{\delta}{1+r}\right)^t \left(0.009375 - (t+1)\frac{0.0375}{365}0.009375 - \frac{0.0375}{365} * 2\right) \end{aligned} \tag{14}$$

Discussion We plot the results from Eqs. (13) and (14) in Fig. 3. The collateral reduction ρ can be subtracted from D. In practice, the user and the service provider can agree on the desired reduction. We note three findings: (i) If the user and the service provider want to maintain security w.r.t. no additional incentive for the service provider to violate the specification, the maximum reduction is given by the single-shot game reduction from Eq. (13) (the orange line in the Figure). (ii) Note that the

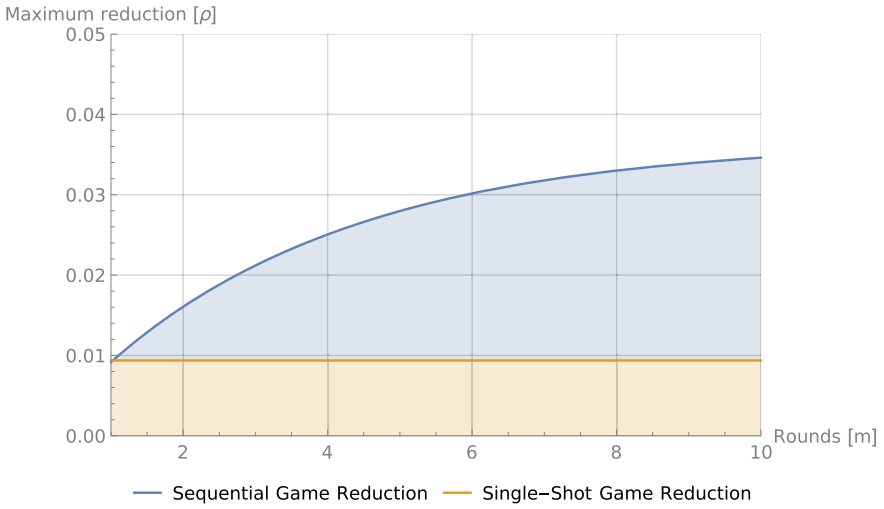


Fig. 3 The possible collateral reduction ρ under the assumption that the vault considers a single round of execution (i.e., a single-shot game) as depicted in the orange line or that the vault considers a sequential game with multiple rounds as depicted in the blue line. The colored areas show in which collateral reduction ranges the vault does not receive an additional incentive to violate the specification as agreed with the user. The more rounds m the game last, the higher the collateral reduction ρ can be under the sequential game scheme. Collateral reductions are constant in case the vault only plays a single-shot game

main reason that the single-shot reduction is comparably low since the user has a "buffer" of 1 unit of BTC that was added to V_A. If the user is willing to accept a lower buffer, say 0.5, the collateral can be consequently lowered. This would still cover the value of the 1 BTC in our example plus a 0.5 potential malicious intend on the vaults side. (iii) The user and the service provider can agree to lower the initial collateral D_I in a sequential game setting if they agree on a longer period m. The collateral reduction, however, is finite: as $m \to \infty$, ρ stabilizes to a constant value.

5.2 NOCUST

NOCUST is a second-layer payment protocol whereby an untrusted intermediary operates a commit-chain to facilitate payments between its users [15]. The application of Promise to NOCUST follows a similar approach as the XCLAIM example. Hence, we are only giving a sketch of Promise's applicability here.

We consider a scenario where Alice is the intermediary commit-chain operator, and Bob is a payment recipient. In this setting we propose to employ Promise as follows: Any fee to be paid by Bob to Alice in exchange for the delivery of an incoming payment would be locked as collateral that Bob could claim if the NOCUST

Table 2 Overview of Promise functions and their cost

Function	Description	Gas cost	Cost
Create	Setup function	112196	USD 0.02895
Deposit	Called by intermediary to provide deposit	43291	USD 0.01116
Payment	Called by user to provide pre-payment	43770	USD 0.0113
Deliver	Called as part of task provision	50703	USD 0.01309
Withdraw	Called by intermediary after the service period is up to receive payment and deposit	31788	USD 0.0082

protocol fails. Over time, the fees locked in Promise would grant Bob instant finality over larger payments, increasing the utility of the service.

For example, in a sales scenario, Bob could release some goods immediately after Alice promises to deliver the payment for them, instead of waiting two rounds for guaranteed finality. If Alice fails to deliver the payment, her collateral would paid to Bob to cover the cost of the goods.

5.3 Implementation

We implement Promise in Solidity in around 100 lines of code. We use the implementation to experimentally assess the cost of executing the contract functions. Our cost calculations are summarized in Table 2 based on an Ether exchange rate of USD 172.61 and 1.5 Gwei gas price. The implementation is available as an open source project.[3]

6 Related Work

There are two strands of related literature. The first one comes from the financial world covering (advance) payments for financial contracts. The second strand comes from the more recent work in decentralized ledgers. In the economics literature, a wide range of work focuses on secured debt, such as [17, 18]. However, these concepts rely on trust on third parties to maintain security in the debt and payment positions. Promise replaces this third-party trust by holding advance payments in a smart contract escrow.

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

On the second strand, Balance is a protocol that allows intermediaries to lower their collateral over time [14]. It operates at the other end of Promise: instead of lowering the initial collateral, the more an agent behaves honestly, the higher the reduction of collateral. Balance requires the highest collateral to be provided at the start of the interaction between agents and makes the assumption that payments are close to 0 (i.e., there is perfect competition). Promise and Balance can be combined together to first reduce initial collateral when bootstrapping a new protocol and then lower collateral requirements for established agents over time. Teutsch et al. discuss bootstrapping a token for verifiable computations [19]. This work discusses how to enable users, like Bob, to obtain the required funds to participate in TrueBit. Their proposal includes a governance game that allows to exchange special governance tokens into collateral tokens (for intermediaries) and utility tokens (for users). Lastly, the idea of bundling payments together is also introduced in [7] to create subscriptions for services of agents. Promise extends this idea to allow collateral reduction for intermediaries.

7 Conclusion

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

Acknowledgements The authors would like to thank Arthur Gervais and William Knottenbelt for their helpful feedback on this paper. Further, the authors thank the anonymous reviewers for their excellent feedback and suggestions for improvement.

References

1. MakerDAO Whitepaper. https://makerdao.com/whitepaper. Accessed: 2018-11-28
2. Aiyer, A. S., Alvisi, L., Clement, A., Dahlin, M., Martin, J. P., & Porth, C. (2005). Bar fault tolerance for cooperative services. In *ACM SIGOPS operating systems review* (Vol. 39, pp. 45–58). ACM.
3. Al-Bassam, M., Sonnino, A., & Buterin, V. (2018). Fraud proofs: Maximising light client security and scaling blockchains with dishonest majorities. arXiv:1809.09044.

4. Avarikioti, G., Kogias, E. K., & Wattenhofer, R. (2019). Brick: Asynchronous state channels. arXiv preprint arXiv:1905.11360.
5. Avarikioti, G., Laufenberg, F., Sliwinski, J., Wang, Y., & Wattenhofer, R. (2018). Towards secure and efficient payment channels. arXiv preprint arXiv:1811.12740.
6. Bartoletti, M., & Zunino, R. (2018). BitML: A calculus for bitcoin smart contracts. In *CCS '18 Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (pp. 83–100). https://doi.org/10.1145/3243734.3243795.
7. Berg, P. R. (2018). ERC-1620: Money streaming. https://github.com/ethereum/EIPs/issues/1620.
8. Böhme, R. (2019). A primer on economics for cryptocurrencies.
9. Douceur, J. R. (2002). The sybil attack. In *International Workshop on Peer-to-Peer Systems* (pp. 251–260). Springer.
10. Dziembowski, S., Eckey, L., & Faust, S. (2018). FairSwap: How to fairly exchange digital goods. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security - CCS '18* (pp. 967–984). New York, USA: ACM Press. https://doi.org/10.1145/3243734.3243857, http://dl.acm.org/citation.cfm?doid=3243734.3243857.
11. Dziembowski, S., Faust, S., & Hostáková, K. (2018). General state channel networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (pp. 949–966). ACM.
12. Fanti, G., Kogan, L., Oh, S., Ruan, K., Viswanath, P., & Wang, G. (2019). Compounding of wealth in proof-of-stake cryptocurrencies. In *Financial cryptography and data security 2019*.
13. Garay, J. A., Kiayias, A., & Leonardos, N. (2016). The bitcoin backbone protocol with chains of variable difficulty. http://eprint.iacr.org/2016/1048.pdf. Accessed: 2017-02-06
14. Harz, D., Gudgeon, L., Gervais, A., & Knottenbelt, W. J. (2019). Balance: Dynamic adjustment of cryptocurrency deposits. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19)*. New York, NY, USA: ACM. https://eprint.iacr.org/2019/675.pdf.
15. Khalil, R., Gervais, A., & Felley, G. (2019). NOCUST - A securely scalable commit-chain. https://eprint.iacr.org/2018/642.
16. McCorry, P., Bakshi, S., Bentov, I., Miller, A., & Meiklejohn, S. (2018). Pisa: Arbitration outsourcing for state channels. *IACR Cryptology ePrint Archive*, *2018*, 582.
17. Scott, J. H, Jr. (1977). Bankruptcy, secured debt, and optimal capital structure. *The Journal of Finance*, *32*(1), 1–19.
18. Stulz, R., & Johnson, H. (1985). An analysis of secured debt. *Journal of Financial Economics*, *14*(4), 501–521.
19. Teutsch, J., Mäkelä, S., & Bakshi, S. (2019). Bootstrapping a stable computation token. arXiv:1908.02946.
20. Teutsch, J., & Reitwießner, C. (2017, March). A scalable verification solution for blockchains. https://truebit.io/. Accessed: 2017-10-06.
21. Zamyatin, A., Harz, D., Lind, J., Panayiotou, P., Gervais, A., & Knottenbelt, W. J. (2019). XCLAIM: Trustless, interoperable, cryptocurrency-backed assets. In *Proceedings of the IEEE Symposium on Security & Privacy*, May 2019 (pp. 1254–1271). https://eprint.iacr.org/2018/643.pdf.

Step on the Gas? A Better Approach for Recommending the Ethereum Gas Price

Sam M. Werner, Paul J. Pritz, and Daniel Perez

Abstract In the Ethereum network, miners are incentivized to include transactions in a block depending on the gas price specified by the sender. The sender of a transaction therefore faces a trade-off between timely inclusion and cost of his transaction. Existing recommendation mechanisms aggregate recent gas price data on a per-block basis to suggest a gas price. We perform an empirical analysis of historic block data to motivate the use of a predictive model for gas price recommendation. Subsequently, we propose a novel mechanism that combines a deep-learning based price forecasting model as well as an algorithm parameterized by a user-specific urgency value to recommend gas prices. In a comprehensive evaluation on real-world data, we show that our approach results on average in costs savings of more than 50% while only incurring an inclusion delay of 1.3 blocks, when compared to the gas price recommendation mechanism of the most widely used Ethereum client.

Keywords Smart contracts · Ethereum · Gas price oracle · Gas mechanism · Blockchain

Sam M. Werner—The author would like to thank the *Brevan Howard Centre for Financial Analysis* for its financial support.
Daniel Perez—The author would like to thank the *Tezos Foundation* for its financial support.

S. M. Werner (✉) · P. J. Pritz · D. Perez
Imperial College London, London, UK
e-mail: sam.werner16@imperial.ac.uk

P. J. Pritz
e-mail: paul.pritz18@imperial.ac.uk

D. Perez
e-mail: daniel.perez@imperial.ac.uk

P. Pardalos et al. (eds.), *Mathematical Research for Blockchain Economy*, Springer Proceedings in Business and Economics,
https://doi.org/10.1007/978-3-030-53356-4_10

1 Introduction

Since the introduction of Ethereum [4] and it's virtual machine, participants have been able to create so-called *smart contracts*, i.e. programs that encapsulate the logic for governing funds. As these contracts have to be executed by all participating nodes in the Ethereum network, the sender of a transaction has to pay for the computational cost of execution in units of *gas*. The amount of gas to be paid by the sender of a transaction depends on the complexity of executing a smart contract's logic. Additionally, the sender is required to specify the *gas price*, which he will have to pay per unit of consumed gas. The product of the gas cost and price determines the transaction fee, which is received by the miner who includes the transaction in a block. Hence, setting an appropriate gas price is critical for having a transaction included in a timely manner. While Ethereum employs a hard coded and transparent gas cost model, there does not exist any embedded mechanism for computing how much a sender of a transaction should pay per unit of gas. The gas price is instead determined by the supply and demand for computational resources. Therefore, choosing an optimal gas price can be challenging, as underpaying likely results in a transaction not being included by miners, whereas overpaying leads to avoidable costs.

The most widely used gas price prediction mechanism is implemented by the popular Ethereum client Geth [13]. This and comparable mechanisms only use recent gas prices and merely aggregate past data to heuristically recommend the gas price for a transaction.

In this paper, we present a novel approach for gas price prediction, motivated by the empirical analysis of a period of 522,213 blocks. We find significant seasonality in the gas price data, suggesting that this can be predicted using a machine learning model. We propose the use of Gated Recurrent Units [7] as these have been shown to be suitable for capturing such patterns. Consequently, we design an algorithm for choosing the gas price for a transaction, which leverages the predictions of our model while allowing to specify the transaction's urgency. Our evaluation on real-world data shows that the proposed approach significantly outperforms the most widely-used Ethereum client Geth [10].

Contributions. Our contributions are as follows:

1. We present a comprehensive empirical analysis of the Ethereum gas price over a period of three months and identify seasonal patterns in the data,
2. We propose a deep-learning based model to predict the gas price and combine this with a novel algorithm for recommending the gas price for a transaction,
3. We evaluate our model on real-world data and show that it outperforms the most widely used gas price recommendation approach, resulting on average in costs savings of more than 50% while only incurring an inclusion delay of 1.3 blocks compared to Geth.

Structure. The remainder of this paper is organized as follows. Section 2 introduces the background of Ethereum and its embedded gas mechanism. An empirical analysis of Ethereum gas prices is presented in Sect. 3. We propose a methodology for better

gas price recommendation in Sect. 4, before evaluating our model's results in Sect. 5. Related work is discussed in Sect. 6. Lastly, we conclude in Sect. 7.

2 Background

In this section, we first provide a brief overview of the workings of the Ethereum network. Subsequently, we examine in greater detail the gas cost and pricing mechanisms used in Ethereum.

2.1 Ethereum

Ethereum employs a Proof-of-Work consensus mechanism as first introduced by Bitcoin [20]. In such a protocol, transactions are grouped into blocks and Ethereum's block arrival time is approximately 13 s [11]. Ethereum allows for the creation of so-called *smart contracts*. These are programs which define a set of rules using a Turing-complete programming language, typically Solidity [8], that can be invoked by network participants. An Ethereum account balance is expressed in the underlying currency Ether (ETH) and directly altered via state transitions caused by transactions. The consensus rules governing transaction validity are implemented by the *Ethereum Virtual Machine* (EVM), a low-level stack machine which executes the compiled EVM bytecode of the smart contract. Operations performed by the EVM consume *gas*, a virtual unit of account used to measure the computational cost of executing a transaction. By design, each EVM instruction has a hard-coded[1] gas cost [26]. The total execution cost has to be paid for by the sender of a transaction.

2.2 Gas Mechanism

The total execution cost for a contract consists of two components, namely the gas cost in units and gas price per unit. The gas cost is split into a fixed base cost of 21 000 gas and an execution cost dependent on the instructions executed while running the contract.

Gas Limit. Due to the Turing-completeness of the EVM, the exact computational cost of a transaction cannot be predetermined. Hence, the sender is required to specify a *gas limit*, or the maximum amount of gas that may be consumed. As the computational steps of a transaction are executed, the required gas is subtracted from the paid gas. Once a transaction is completed, any unused gas will be refunded to the sender.

[1] Note that via a hard-fork, the Ethereum Improvement Proposal 150 [3] re-aligned gas costs for instructions involving I/O-heavy operations.

Should a transaction try to consume gas in excess of the gas limit, an Out-of-Gas exception is thrown by the EVM. Even though such a transaction would fail, it would be recorded on-chain and any used gas will not be refunded to the sender. Note that in addition to the per transaction gas limit there is also a *block gas limit*,[2] which specifies the total amount of gas that may be consumed by all transactions in a block.

Gas Price. Apart from setting a gas limit, a sender will also have to specify the *gas price*, which refers to the amount of Ether the sender is willing to pay per unit of gas, generally expressed in *wei* (1 wei = 10^{-18}ETH) or *Gwei* (1 Gwei = 10^{-9}ETH). Miners set a cut-off gas price to choose which transactions to include in their memory pool. When constructing a new block, they then choose the transactions with the most lucrative gas prices from their memory pool. A higher gas price will increase the fee which miners receive from a transaction, thereby motivating a miner to include a transaction in a block. The total amount of wei to be paid by a sender is referred to as the *transaction fee* and amounts to the product of the gas price and gas cost.

Gas Price Oracles. The sender of an Ethereum transaction is exposed to the non-trivial task of having to decide on a gas price. Since a higher gas price will increase the likelihood of having a transaction included quickly, there is a clear trade-off between waiting and paying. We define the optimal gas price as the minimum gas price such that the transaction is included in a block within the period of time that the sender of the transaction is prepared to wait for.

In order to avoid risks of overpaying, *gas price oracles* exist [1, 12–14]. These oracles aim to recommend the gas price a transaction requires in order to be included in a block within a specified time period. Commonly, the recommendation mechanism uses some rule-based approach analyzing the gas prices of previous blocks. We provide a more detailed summary on existing approaches in Sect. 6.

3 Empirical Analysis

In this section, we empirically analyze Ethereum block data to develop a better understanding of the gas price behavior. We use data from the period of 1 October, 2019 to 31 December, 2019, which amounts to a total of 522,213 blocks. When comparing mean, minimum and maximum gas prices averaged over 3 h intervals during this period, we can see in Fig. 1 that substantial spreads exists in the gas price. More specifically, the maximum gas price exceeds the minimum gas price by an order of magnitude for the entire period. The gas price volatility throughout the examined 522,213 blocks is further indicated by the standard deviation of the average gas price, which is 46.4645 Gwei at an average gas price of 13.9598 Gwei, as shown in Table 1. The same can be said about the average gas utilization per block. Figure 2 shows the cross-correlations between the average gas price, maximum gas price, minimum gas price, number of transactions and gas utilization per block. Surprisingly, the average gas price and utilization are not correlated. In fact, the average gas price

[2] At the time of writing the average block gas limit was around 10,000,000 units of gas.

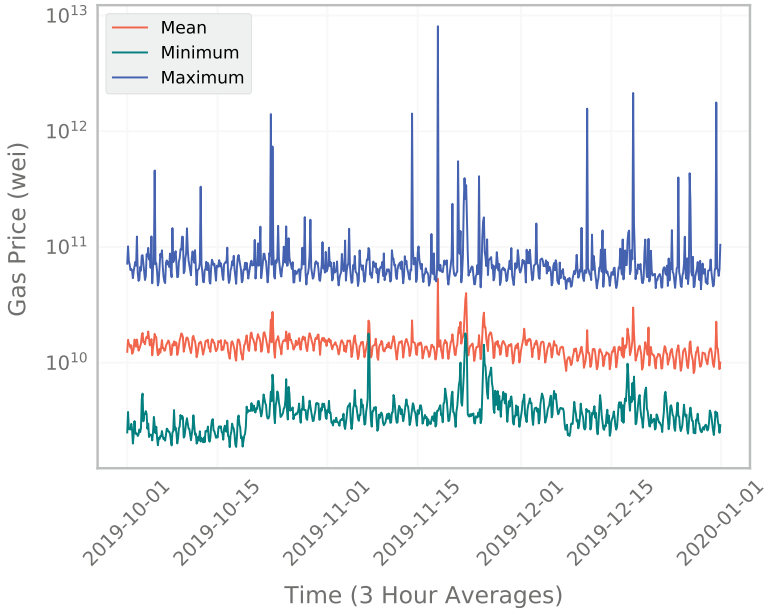


Fig. 1 The mean, maximum and minimum gas price averaged over 3 h intervals from block 8,653,173 (1 October, 2019) to 9,193,265 (31 December, 2019)

Table 1 Mean, median and standard deviation of average gas price per block, as well as mean and standard deviation of gas utilization per block from block 8,653,173 (1 October, 2019) to 9,193,265 (31 December, 2019)

Number of blocks:	522,213
Mean gas price:	13.9598 Gwei
Median of average gas price:	10.3260 Gwei
Standard deviation of average gas price:	46.4645 Gwei
Mean gas utilization:	79.36%
Standard deviation gas utilization:	32.00%

is only significantly correlated with the maximum gas price. The gas utilization is only correlated with the transaction count. However, apart from these two correlated pairs, the remainder of the variables are not significantly correlated.

To investigate the presence of seasonality in the data, we examine the autocorrelation of each variable on a per block and hourly basis. Most interestingly, we find that even though the gas price does not exhibit any significant seasonality on a per block basis, there does exist seasonality when looking at the gas price averaged over one hour intervals, as indicated by the autocorrelation in the left plot of Fig. 3.

It can be seen that especially for a lag of 24 h significant seasonality can be found in the data, which could be linked to different time zones of the countries where most

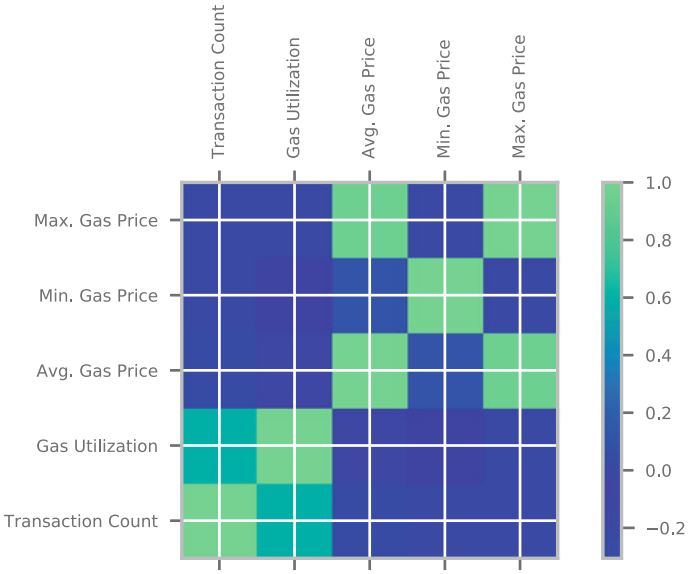


Fig. 2 Correlation matrix for the average gas price, maximum gas price, minimum gas price, number of transactions and gas utilization per block

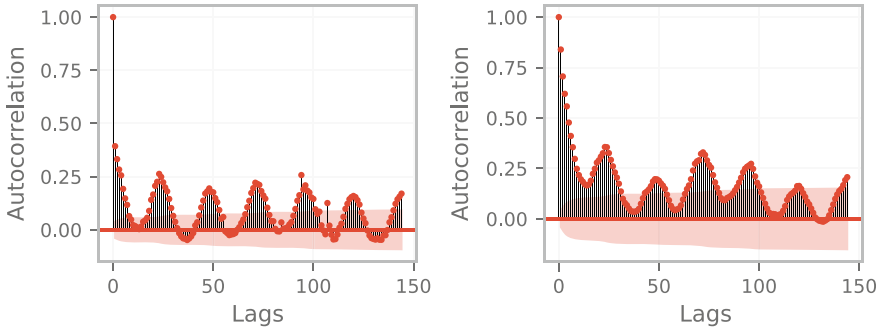


Fig. 3 Autocorrelation function (ACF) plot of mean (left hand side) and minimum (right hand side) gas prices averaged over one hour periods for 144 lags

transactions are conducted. This seasonality can be found to an even greater extent in the autocorrelation of the minimum gas price averaged over one hour intervals. The presence of seasonal patterns in the data alludes to the viability of machine learning models for predicting future gas prices.

4 Methodology

The gas price recommendation methodology we propose consists of two key components. First, we present a deep-learning based model to predict the gas price for a pre-defined period of time. Second, we introduce an algorithm that uses these predictions to recommend a gas price for a transaction, parameterized by the sender's willingness to delay the transaction. Both components, as well as the employed data pre-processing steps are presented in this section.

4.1 Gas Price Prediction

The methodology we propose requires a forecasting model to predict the gas price trajectory over a pre-defined number of time steps s. In particular, we are interested in predicting the minimum gas price under rational miner behavior, since this can be seen as a lower bound for setting the gas price for a given transaction. From the preliminary data exploration in Sect. 3, it is apparent that the per-block data is extremely noisy, which can be attenuated by averaging over a longer period of time. We therefore average the minimum gas price of all blocks in consecutive 5 min intervals and forecast on this level of granularity, instead of using per-block data directly. A time step is then defined as a 5 min interval. We denote the complete time series of average minimum gas prices by y. Furthermore, we define the aggregated time series of all features used as model input as $\mathcal{D}$, where $d_t \in \mathcal{D}$ denotes the feature vector for a single time step t. For both model training and inference, we use a sliding window model that uses a fixed-size window of historical data with l time steps for prediction. The problem of forecasting a window of s time steps using a window of size l can then be defined as

$$\hat{y}_{t+1}, \ldots, \hat{y}_{t+s} = \underset{y_{t+1},\ldots,y_{t+s}}{\operatorname{argmax}} \; p(y_{t+1}, \ldots, y_{t+s} | d_{t-l}, \ldots, d_t) \,. \tag{1}$$

In the remainder of this section we present our pre-processing methodology and proposed forecasting model.

4.1.1 Pre-processing

We introduce a number of pre-processing steps to the data, which specifically aim to reduce the impact of noise while still capturing seasonal components and trends. Table 2 lists the features used as input for the predictive model. Due to the daily seasonality in the data, some variables are also included with a lag of 24 h. To reduce the impact of noise in the data, we first remove outliers using a heuristic criterion, where we delete all data points that are more than 1.5 standard deviations higher or lower than the mean. Subsequently, all data is normalized to values between 0

Table 2 Features used as input data for the predictive model to forecast the minimum gas price. Lagged variables are included both with and without lag

Feature name	Lagged by 24 h
Average gas price per block	Yes
Transaction count per block	No
Max. gas price per block	No
Min. gas price per block	No
ETH price at block timestamp	No

and 1. Since the main goal of the predictive model is to capture the seasonality and predict the gas price on a fairly coarse level, we employ a further pre-processing step presented in [23]. This additional step applies a discrete Fourier transform to each window in the input data and truncates the frequency domain representation of the time series using an adaptive energy-based criterion. We then convert it back to the time-domain using an inverse Fourier transform. This methodology allows us to adaptively reduce the impact of short-term fluctuations in each window of input data, while still capturing the seasonal components and overall trend.

4.1.2 Model

As a forecasting model, we propose the use of a Gated Recurrent Unit (GRU) [7]. GRUs are a specialisation of recurrent neural networks, where a computationally efficient gating mechanism is used. Gating has been shown to improve the network's ability to learn longer term dependencies [17], making this kind of model well-suited to the problem at hand. The GRU architecture is given by

$$z_t = \sigma(W_z d_t + V_z h_{t-1} + b_z)\,, \tag{2}$$

$$r_t = \sigma(W_r d_t + V_r h_{t-1} + b_r)\,, \tag{3}$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \phi(W_h d_t + V_h(r_t \circ h_{t-1}) + b_h)\,, \tag{4}$$

$$\hat{y} = \hat{y}_{t+1}, \ldots, \hat{y}_{t+s} = f(h_t)\,, \tag{5}$$

where $\circ$ denotes the Hadamard product, W, V and b are parameter matrices and biases, $\sigma(\cdot)$ and $\phi(\cdot)$ denote the sigmoid and hyperbolic tangent functions, respectively, z_t, r_t and h_t are the update and reset gates and the hidden state and $f(\cdot)$ denotes the final linear layer of the network. The network is trained using gradient descent and backpropagation with an Adam optimiser [18].

4.2 Recommendation Algorithm

We now describe our recommendation algorithm which leverages the gas prices predicted by our model. We use the 20th percentile of the predicted prices as the initial gas price, which we note $\hat{g}$. One of the main objectives of our algorithm is to scale $\hat{g}$ such that the faster the predicted gas prices are decreasing, the lower the gas price recommended by the algorithm. On the other hand, if the prices are increasing, the predicted prices should not be significantly lower than the current gas price. We incorporate this objective by finding a coefficient $0 < c \leq 1$ that is multiplied with the predicted gas price $\hat{g}$. Furthermore, we want c to increase or decrease exponentially with respect to the trend to achieve aggressive gas pricing if the predicted prices decrease quickly.

First, we compute the trend of the predictions $\hat{y}$ returned by our forecasting model. We fit a linear function such that $\hat{y} = aX + b$, with $X = 1, 2, \ldots, s$, and store the slope a, which captures the trend in the predicted gas prices. We then normalize a to $\tilde{a}$ to lie in the range between 0 to 1. This is achieved by computing the maximum A_{max} and minimum A_{min} values of the slopes we obtain for our training data and computing $\tilde{a}$ according to Eq. (6).

$$\tilde{a} = \frac{a - A_{min}}{A_{max} - A_{min}} \tag{6}$$

Then, to obtain the described exponential behavior, we exploit the fact that the exponential function in the interval $[-2, 0]$ has the desired properties and hence, compute c using Eq. (7).

$$c = e^{2\tilde{a}-2} \tag{7}$$

Finally, to allow the user to configure the urgency of a transaction, we define an urgency parameter $\mathcal{U}$, which we use to scale the obtained coefficient c to arrive at a recommended gas price $\mathcal{G}$ given by

$$\mathcal{G} = \hat{g} \cdot c \cdot \mathcal{U}\,. \tag{8}$$

Algorithm 1 Evaluation procedure of the gas recommendation efficiency

```
function EvaluateRecommender(StartBlock, EndBlock, Recommend)
    Pending ← ∅
    Results ← ∅
    Block ← StartBlock
    while Block ≤ EndBlock ∨ (Pending ≠ ∅ ∧ Block ≤ LastBlock) do
        Price ← GetMinimumPrice(Block)
        while Pending ≠ ∅ ∧ min_{t∈Pending} (t_1) ≥ Price do        ▷ t_1 is the transaction price
            Transaction ← argmin_{t∈Pending} (t_1)
            Pending ← Pending \ {Transaction}
            Results ← Results ∪ {(Transaction, Block, Price)}
        end while
        if Block ≤ EndBlock then
            Recommended ← Recommend(Block)
            Pending ← Pending ∪ {(Block, Recommended)}
        end if
        Block ← Block + 1
    end while
    return Results
end function
```

4.3 Measuring Gas Recommendation Efficiency

Up to here, we have described how we recommend a price at a given block number. However, to understand how optimal a gas price is, we need to measure the difference between the recommended and the optimal gas price.

To evaluate the efficiency of our approach, we iterate over a range of blocks, where we do the following. For each block, a new transaction using the recommended gas price is added to a set of pending transactions. Each transaction in the pending set is processed upon encountering a block with a minimum gas price lower than that specified in the transaction. We keep track of the recommended price, the inclusion price, i.e. the minimum gas price of the block where the transaction is included, and the number of blocks elapsed until inclusion. We show the detailed steps in Algorithm 1. The `EvaluateRecommender` function takes a start block, an end block and a recommendation function to evaluate. `LastBlock` is the number of the last block which we evaluate and `GetMinimumPrice` returns the minimum gas price for a given block.

To be able to evaluate the efficiency of our algorithm, we use the Geth gas price recommendation algorithm as the main baseline, as it is by far the most widely used Ethereum client [10].

5 Results

In this section, we present the results we obtain when using the methodology presented in the previous section and compare them with our baselines.

5.1 Model Training

All models are implemented in Python, using the PyToch library [24]. We train all models on a personal computer with 32GB of RAM, an 8th generation Intel Core i7-8700 with 3.20 GHz and 6 cores and a 256GB SATA hard drive. Model training and hyper parameter tuning is performed on the data between 10 November, 2019 to 20 November, 2019, where we use the first 70% of the data for training and the remaining 30% for validation. We show exemplary predictions of our model in Fig. 4.

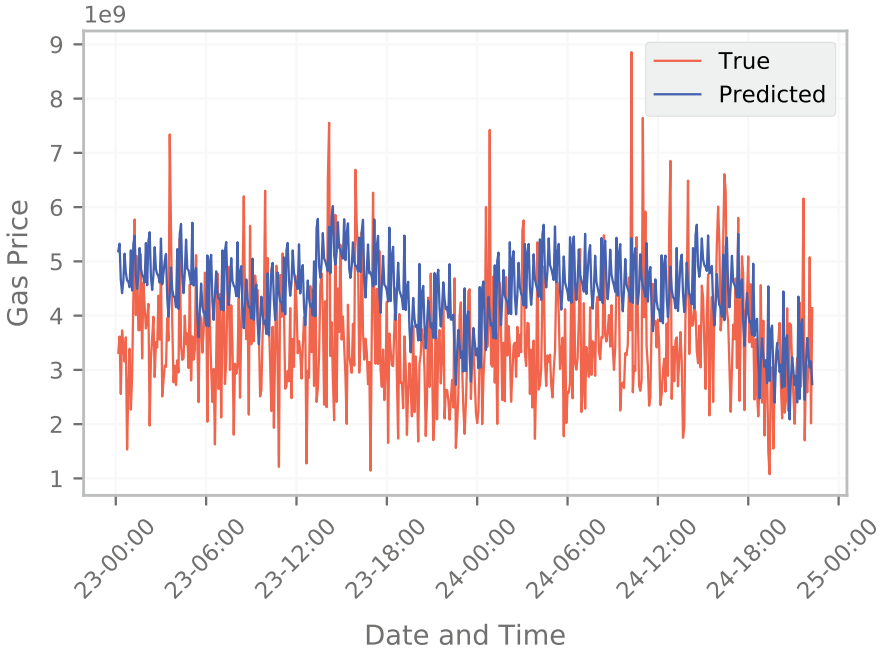


Fig. 4 Exemplary gas price predictions obtained with our forecasting model for the period between the 23 November, 2019 and 25 November, 2019

Table 3 Parameters used in the different strategies

Model	Parameter	Description
Geth	Scaling ($\mathcal{S}$)	Ratio by which to scale the price (0.8 means use 80% of the recommended price)
Proposed approach	Urgency ($\mathcal{U}$)	Urgency tuning parameter to trade-off price for time
Look-ahead	Blocks ($\mathcal{B}$)	Number of blocks to look ahead

5.2 Evaluation

We use a sample of around five days of data—from 20 November, 2019 (block 8,965,759) to November 24, 2019 (block 8,995,344)—and evaluate the different price recommendation strategies using the procedure presented in Algorithm 1. We first describe the parameters of each strategy in Table 3. For Geth we use a scaling ratio parameter $\mathcal{S}$ with which the recommended gas price is multiplied. The main purpose of this parameter is to ensure that giving a lower gas price does have a direct impact on the number of blocks waited. Our proposed recommendation strategy accepts a single parameter $\mathcal{U}$ representing the urgency. The urgency parameter is used to trade off gas price for waiting time: the lower the urgency, the lower the gas price and hence, the longer the waiting time. Empirically, reasonable values for these parameters are roughly between 0.7 and 1.3, where 0.7 will result in cheap but long to be accepted transactions and 1.3 will result in more expensive but faster transactions. Finally, our look-ahead model, which we use to estimate the lowest possible price takes a parameter $\mathcal{B}$ representing the maximum look-ahead as a number of blocks. We note that the look-ahead strategy is for validation purposes only as it uses information about future blocks, which would obviously not be available in practice.

We present a summary of the results for the different recommendation strategies in Table 4. We use several values for the parameters of each strategy and order its results so that the gas price decreases and the number of blocks to wait increases. We can see that using the price recommended by Geth, the waiting time is very short—on average less than 2 blocks—with an average gas price of around 4.4 Gwei. However, by just using 90% of the recommended price, the waiting time increases to an average of 15.5 blocks. Comparing these results to the minimum possible gas price, obtained from the look-ahead model, we can see that by only waiting for an average of 4.8 blocks a saving of 75% could be obtained. Although these numbers are hypothetical, they suggest the potential for significant improvement.

We now show how our model performs in comparison to the price recommended by Geth and the hypothetical minimum price. With the urgency parameter set to 1.0, our model recommends a gas price on average twice as low as the Geth price, while waiting for an average of approximately 3.3 blocks. When decreasing the urgency parameter, we can see that the number of blocks elapsed increases fairly slowly at

Table 4 Results of the different recommendation strategies presented. Gas price and wait time are averaged over the number of blocks processed. Parameters are described in Table 3

Strategy	Parameter	Gas price	Blocks waited
Geth	$\mathcal{S} = 1.0$	4,414,902,746	1.97
Geth	$\mathcal{S} = 0.9$	4,080,968,868	15.49
Geth	$\mathcal{S} = 0.8$	3,531,922,197	25.52
Look-ahead	$\mathcal{B} = 15$	1,166,965,099	4.80
Look-ahead	$\mathcal{B} = 30$	969,559,938	8.52
Look-ahead	$\mathcal{B} = 60$	782,105,012	18.84
Proposed approach	$\mathcal{U} = 1.0$	2,120,108,703	3.28
Proposed approach	$\mathcal{U} = 0.9$	1,908,097,833	3.79
Proposed approach	$\mathcal{U} = 0.8$	1,696,086,963	5.13
Proposed approach	$\mathcal{U} = 0.7$	1,484,076,092	10.06

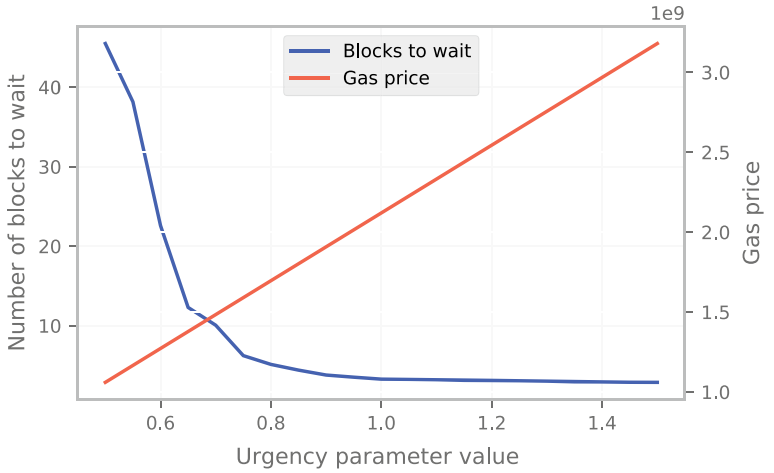


Fig. 5 Effect of the urgency parameter on the average gas price paid and number of blocks waited

first but doubles between 0.8 and 0.7, showing that at this point the gas price becomes too low for the transaction to be included in a timely manner. In Fig. 5, we show the effect of our urgency parameter on the average gas price paid and the average number of blocks elapsed until the transaction is included.

6 Related Work

For Ethereum in particular, there has been extensive research on smart contract correctness, upper-bound gas consumption and imperfections in the current EVM gas cost model. Nonetheless, very little work has been done with the goal of determining

optimal gas prices. In this section, we first present existing work on the gas mechanism, before examining the most widely used gas price recommendation methods.

6.1 Gas Mechanism

The overconsumption of gas can be harmful for the contract user for two main reasons: higher monetary costs and potential vulnerabilities. Gas overconsumption is examined by Chen et al. [5], who focus on gas usage optimization by introducing Gasper, a tool leveraging symbolic execution for detecting costly patterns in the bytecode of smart contracts which are not optimized by the Solidity compiler. Potential issues in the form of gas-related vulnerabilities are carefully examined by Grech et al. [15], who propose a static analysis tool, called MadMax, predominantly suitable for detecting out-of-gas exceptions which may cause contract funds being locked. Elvira et al. [2] present Gastap, a static analysis tool for inferring gas upper bounds for smart contracts and are thereby able to detect whether any out-of-gas vulnerabilities could exist. A further approach for computing gas consumption upper bounds was introduced by Marescotti et al. [19], however, the authors are yet to implement and test their algorithms in an EVM setting. For a more general summary of existing smart contract verification tools we point the reader to [16].

There have been several pieces of work focusing on imperfections in the current gas cost mechanism. Both Yang et al. [27] and Perez and Livshits [21] identify inconsistencies in the pricing of EVM instructions in the current gas cost model. The latter propose a new type of attack aimed at exploiting EVM design flaws by generating resource exhaustive contracts, which are on average significantly slower in terms of throughput than typical contracts. As a means of preventing Denial-of-Service attacks stemming from under-priced EVM instructions a modification of the current gas cost mechanism has been proposed by [6].

While several pieces of existing work examine the current gas cost mechanism, limited work exists on gas price recommendation. Pierro et al. [22] investigate potential factors that influence transaction fees in Ethereum from a technical and economic perspective, yet leave a gas price prediction model for future research.

6.2 Gas Price Oracles

In the following, we examine existing approaches for gas price recommendation that are used in practice.

Geth. The Ethereum client implementation in go, namely Geth [13], accounts for over 79% of all Ethereum clients [10]. To recommend a gas price, Geth uses the minimum gas price of the previous blocks. It looks back at the 100 blocks preceding the current one and then uses the value of the 60th percentile of the minimum gas prices as the price recommendation.

EthGasStation. A further gas price oracle has been introduced by EthGasStation [1], a third-party tool, which estimates the expected number of blocks required to confirm a transaction at a given gas price using a Poisson regression model based on data of the previous 10,000 blocks. This approach has also been implemented by the popular Ethereum block explorer Etherchain [9]. Unfortunately, no historical data was available for comparison.

GasStation—Express. EthGasStation also released a more simple gas price oracle called "GasStation – Express" [12]. This approach predicts the likelihood of a transaction being included in the next block at a given gas price by examining the percentage of the last 200 blocks that included a transaction with the same or lower gas price [25]. The percentage thresholds of recent block inclusions are fixed for the categories *Fast* (90%), *Standard* (60%) and *SafeLow* (35%). Additionally a *Fastest* option is given, whereby the suggested gas price was included by all of the previous 200 mined blocks, which likely results in the sender overpaying considerably. Just like the threshold percentages, the associated expected confirmation times are also hard-coded, which limits the speed at which the system can react to changes.

7 Conclusion

Motivated by an empirical analysis of 3 months of data, we have proposed a novel approach for recommending the Ethereum gas price that outperforms the method of the most widely used Ethereum client. Our approach uses a deep-learning based price forecasting model as well as an algorithm parameterized by an urgency value that can be set by the user. In a comprehensive evaluation, we show that our approach is able to reduce the average gas price paid by the sender of a transaction by more than 50% while only introducing an average additional waiting time of 1.3 blocks compared to Geth.

Our evaluation of the proposed approach aimed to focus on common-sized transactions. For more computationally intensive transactions, the gas price would likely need to be increased to ensure timely inclusion in a block. However, this could be easily accomplished by adjusting the urgency parameter.

Future work can examine the usefulness of additional data, such as memory pool data, as model inputs. Additionally, the evaluation and comparison of our approach and previous approaches in a larger simulation may be a fruitful avenue for further research.

References

1. Ethgasstation. (2020). https://ethgasstation.info/. [Online; Accessed 14-January-2020].
2. Albert, E., Gordillo, P., Rubio, A., & Sergey, I. (2018, November). GASTAP: A gas analyzer for smart contracts. *CoRR*, abs/1811.1.
3. Buterin, V. (2019). EIP 150: Gas cost changes for IO-heavy operations. https://eips.ethereum.org/EIPS/eip-150. [Online; Accessed 05-June-2019].
4. Buterin, V. (2014). A next-generation smart contract and decentralized application platform. *Ethereum*, (January), 1–36.
5. Chen, T., Li, X., Luo, X., Zhang, X. (2017). Under-optimized smart contracts devour your money. In *SANER 2017 - 24th IEEE International Conference on Software Analysis, Evolution, and Reengineering* (pp. 442–446).
6. Chen, T., Li, X., Wang, Y., Chen, J., Li, Z., Luo, X., et al. (2017). An adaptive gas cost mechanism for ethereum to defend against under-priced dos attacks. In J. K. Liu & P. Samarati (Eds.), *Information security practice and experience* (pp. 3–24). Cham: Springer International Publishing.
7. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, arXiv:1406.1078.
8. Dannen, C. (2017). *Introducing ethereum and solidity: Foundations of cryptocurrency and blockchain programming for beginners* (1st ed.). Berkely, CA, USA: Apress.
9. etherchain.org. (2020). Gas price oracle. https://www.etherchain.org/tools/gasPriceOracle. [Online; Accessed 15-January-2020].
10. ethernodes.org. (2020). Ethereum mainnet statistics. https://www.ethernodes.org/. [Online; Accessed 15-January-2020].
11. Etherscan. (2020). Ethereum average block time chart. https://etherscan.io/chart/blocktime. Accessed 28-January-2020.
12. Github. (2020). Gasstation-express. https://github.com/ethgasstation/gasstation-express-oracle. [Online; Accessed 15-January-2020].
13. Github. (2020). Official go implementation of the ethereum protocol. https://github.com/ethereum/go-ethereum/. [Online; Accessed 14-January-2020].
14. Github. (2020). Parity ethereum: The fastest and most advanced ethereum client. https://github.com/paritytech/parity-ethereum. [Online; Accessed 16-January-2020].
15. Grech, N., Kong, M., Jurisevic, A., Brent, L., Scholz, B., & Smaragdakis, Y. (2018). MadMax: Surviving out-of-gas conditions in ethereum smart contracts. *SPLASH 2018 Oopsla*, *2*(October).
16. Harz, D., & Knottenbelt, W. (2018). Towards safer smart contracts: A survey of languages and verification methods. arXiv preprint arXiv:1809.09805.
17. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.
18. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization.
19. Marescotti, M., Blicha, M., Hyvärinen, A. E. J., Asadi, S., & Sharygina, N. (2018). Computing exact worst-case gas consumption for smart contracts. In T. Margaria & B. Steffen (Eds.), *Leveraging applications of formal methods, verification and validation. Industrial practice* (pp. 450–465). Cham: Springer International Publishing.
20. Nakamoto, S. (2008, December). Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf. Accessed: 2015-07-01.
21. Perez, D., & Livshits, B. (2019). Broken metre: Attacking resource metering in EVM. arXiv preprint arXiv:1909.07220.
22. Pierro, G. A., & Rocha, H. (2019). The influence factors on ethereum transaction fees. In *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)* (pp. 24–31). IEEE.
23. Pritz, P. J., Perez, D., & Leung, K. K. (2020). Fast-fourier-forecasting resource utilisation in distributed systems.

24. PyTorch Contributors. (2019). PyTorch online documentation. https://pytorch.org/docs. Accessed 13-August-2019.
25. Station, M. E. G. (2017). Gasstation express: A simple gas price oracle for anyone running a full ethereum node. https://medium.com/@ethgasstation/gasstation-express-a-simple-gas-price-oracle-for-anyone-running-a-full-ethereum-node-f1bde46260f5. [Online; Accessed 15-January-2020].
26. Wood, G. (2014). Ethereum yellow paper.
27. Yang, R., Murray, T., Rimba, P., & Parampalli, U. (2019). Empirically analyzing ethereum's gas mechanism. *CoRR*, abs/1905.0.