Symposium on Foundations and Applications of Blockchain

Proceedings

Sumita Barahmand
Shahram Ghandeharizadeh
Bhaskar Krishnamachari

University of Southern California
Los Angeles, California
March 9, 2018

Message from the Chairs

The first Symposium on Foundations and Applications of Blockchain (FAB) brings together researchers and practitioners of blockchain to share and exchange research results. This one-day event is held at the beautiful campus of the University of Southern California, Los Angeles, CA, on March 9, 2018.

The program consists of six exciting refereed technical papers from around the world, a keynote by Professor Charalampos Papamanthou entitled "Applications of Verifiable Computation in Blockchains and Cryptocurrencies", a timely panel on future of blockchain, as well as poster and lightning talk sessions. The technical papers are given a 25 minutes presentation time with 5 minutes for questions. They were selected through a rigorous review process with 3 to 4 reviews per paper.

One of the technical papers titled "Unchain Your Blockchain" stood out and has been identified as the best paper award. We congratulate the authors for their timely technical contribution. The runner-up paper receiving the honorable mention is titled "Collaboration Among Adversaries: Distributed Workflow Execution on a Blockchain". Authors of both papers received certificates with the best paper receiving a trophy.

We thank our international program committee and the industrial sponsors. Our special thanks go to Heidi Pease for serving as the industrial chair, A.J. Nachikethas as the webmaster, and Brienne Jessica Moore for her logistical support of the event. Finally, we wish to thank the authors for their contributions and the panelists for an exciting discussion of the future of blockchain.

Sumita Barahmand, Proceedings Chair
Shahram Ghandeharizadeh, Program Chair
Bhaskar Krishnamachari, General Chair

Table of Contents

FAB 2018 Conference Organization

General Chair: Bhaskar Krishnamachari (*University of Southern California*)

Program Chair: Shahram Ghandeharizadeh (*University of Southern California)*

Industrial chair: Heidi Pease (*LA Blockchain Lab and Proof of Art*)

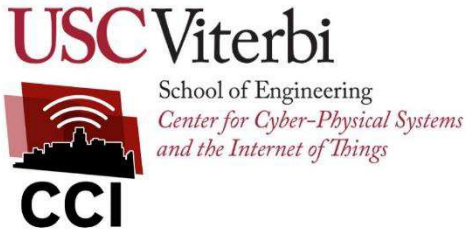
Proceedings chair: Sumita Barahmand (*Microsoft*)

Web chair: Nachikethas A. Jagadeesan (*University of Southern California*)

Program Committee Members:
Daniel Augot (*INRIA*)
Sumita Barahmand (*Microsoft*)
Luis Bathen (*IBM Research*)
Yu Chen (*State University of New York – Binghamton*)
Bhagwan Chowdhry (*UCLA*)
Eric Chung (*DApperNetwork*)
Ming-Deh Huang (*University of Southern California*)
Eric Diehl (*Sony Pictures Entertainment*)
Abdelkader Hameurlain (*Paul Sabatier University, Toulouse, France*)
Stephen Holmes (*Virtusa*)
Zhiyuan Jiang (*Tsinghua University*)
Lou Kerner (*Flight VC*)
Genevieve Leveille (*Otentic8*)
Chen Li (*UC Irvine*)
David MacFadyen (*UCLA*)
Beng Chin Ooi (*National University of Singapore*)
Avinash Sridharan (*Mesosphere*)
Vassilis J. Tsotras (*UC Riverside*)
Nick Vyas (*University of Southern California*)
Li Xiong (*Emory University*)
Kiran Yedavalli (*Cisco*)

FAB 2018 Conference Sponsors

University Sponsors



Industry Sponsors

Silver Sponsors:

Bronze Sponsors:

Applications of Verifiable Computation in Blockchains and Cryptocurrencies

Charalampos (Babis) Papamanthou

Assistant Professor

Department of Electrical and Computer Engineering and Institute for Advanced Computer Studies

University of Maryland, College Park

Abstract

Decentralized cryptocurrencies and smart contracts (e.g., Bitcoin and Ethereum) promise to revolutionize financial industries, forever changing the way money is transferred. They support monetary transactions based on programmable logic between users without requiring to trust a central authority (e.g., a government or a bank). At the heart of their design lies *the blockchain*, a distributed data structure that is stored and agreed upon by the participating parties and which is readily accessible to anyone in the world. However, due to its size and public nature, several scalability, security and privacy concerns have emerged. In this talk I will show how we can use protocols for verifiable computation (cryptographic techniques that enable an untrusted party to efficiently prove the validity of a statement to a verifier), to address these problems.

Bio

Charalampos (Babis) Papamanthou is an assistant professor of Electrical and Computer Engineering at the University of Maryland, College Park, where he joined in 2013 after a postdoc at UC Berkeley. At Maryland, he is also affiliated with the Institute for Advanced Computer Studies (UMIACS), where he is a member of the Maryland Cybersecurity Center (MC2). He works on applied cryptography and computer security---and especially on technologies, systems and theory for secure and private cloud computing. While at College Park, he received the NSF CAREER award, the Google Faculty Research Award, the Yahoo! Faculty Research Engagement Award, the NetApp Faculty Fellowship, the 2013 UMD Invention of the Year Award, the 2014 Jimmy Lin Award for Invention and the George Corcoran Award for Excellence in Teaching. His research is currently funded by federal agencies (NSF, NIST and NSA) and by the industry (Google, Yahoo!, NetApp and Amazon). His PhD is in Computer Science from Brown University (2011) and he also holds an MSc in Computer Science from the University of Crete (2005), where he was a member of ICS-FORTH. His work has received over 3,000 citations and he has published in venues and journals spanning theoretical and applied cryptography, systems and database security, graph algorithms and visualization and operations research.

Experiences from the Field: Unify Rewards - A Cryptocurrency Loyalty Program

Philip Shelper
Chief Executive Officer
LoyaltyX
Sydney, Australia
philip.shelper@loyaltyx.co

Andrew Lowe
Managing Director
PicoLabs
Sydney, Australia
andrew.l@picolabs.co

Salil S. Kanhere
School of Computer Science and Engineering
UNSW Sydney
Sydney, Australia
salil.kanhere@unsw.edu.au

ABSTRACT
The emergence of cryptocurrencies has created new opportunities for loyalty programs. In this paper, we present a proof-of-concept cryptocurrency loyalty program called Unify Rewards where participants earned Ether cryptocurrency by making purchases at participating retailers. We outline the experiences gained from conducting a field trial of the program with student and staff at UNSW Sydney. The results from the trial which included 177 participants suggests that cryptocurrency is a viable alternative to loyalty miles and points.

Categories and Subject Descriptors
K.4.4 [**Computers and Society**]: Electronic Commerce – *cybercash, digital cash, distributed commercial transactions, payment schemes.*

Keywords
Cryptocurrency, Ether, Blockchain, Blockchain Loyalty, Loyalty Program, Field Trial

1. INTRODUCTION
The invention of blockchain and cryptocurrencies has inadvertently created an opportunity for a paradigm shift in loyalty program design.

From the 1980's until present day, the dominant currencies within loyalty programs have been 'miles' or 'points'. This has been adopted by major coalition programs generating billions of dollars of revenue per annum, as well as individual retailers with niche programs, and everything in between. Creating a currency which can be controlled by an organization has become a very useful tool for customer engagement, and a viable alternative to product discounting.

Symposium on Foundations and Applications of Blockchain (FAB '18)
March 9, 2018, Los Angeles, California, USA.

Even so, miles and points have limitations which restrict their attractiveness to consumers; they expire, they can only be redeemed on a limited reward range, and the value can be manipulated by the issuer to increase profits.

With the rise of Bitcoin [1] and other cryptocurrencies, a number of specialized Blockchain loyalty companies have been created. These include Gatcoin, CampusCoin, Nexxus Rewards, LoyalCoin and EzToken. These companies tend to follow a similar business design; create a new cryptocurrency, raise funding via an Initial Coin Offering, build a loyalty platform, float the cryptocurrency on an exchange so it can be traded, then seek merchants and members to generate demand for the cryptocurrency to drive up the value. Many of the companies have positioned their approach as one which will disrupt the loyalty industry.

With millions of dollars being invested in these companies, numerous questions arise; Is cryptocurrency a viable alternative for miles or points in a modern loyalty program? Would offering cryptocurrency to members drive deeper engagement with the program than offering miles or points? Does a cryptocurrency-based loyalty program have the potential to disrupt the loyalty industry? Would consumers view cryptocurrencies as any different to cash?

To answer these questions, we designed a proof-of-concept loyalty program called Unify Rewards and tested it in the real-world on the UNSW Sydney campus. Students and staff of UNSW Sydney were invited to join the program, where by transacting with a choice of 12 campus retailers, they earned Ether cryptocurrency over a 5-week period.

The results from the trial which included over 170 participants indicate cryptocurrencies can indeed act as an effective substitute for loyalty points, with evidence indicating they have the potential to drive much deeper engagement with a program by solving a number of the limitations inherent in miles and points-based programs.

The rest of the paper is organized as follows. Section 2 provides a history of loyalty programs. Section 3 presents motivating arguments for using cryptocurrencies in loyalty programs. Section 5 presents an overview of the Unify

Rewards systems. Section 5 summarizes the evaluations from our field trial. Section 6 makes concluding remarks.

2. BACKGROUND

Egyptologists have uncovered evidence that ancient Egyptians practiced a type of reward program similar to modern frequent flyer programs, including status tiers and the ability to redeem on a wider variety of rewards. In [2], Professor Barry Kemp reminds us that for much of the Pharaoh's thousands of years of rule, they didn't have money. It simply wasn't invented yet. Instead they used a system much more aligned to a modern loyalty program. Citizens, conscripted workers and slaves alike were all awarded commodity tokens (similar to loyalty points or cryptocurrencies) for their work and temple time. The most common were beer and bread tokens. The tokens were made from wood, then plastered over and painted, and shaped like a jug of beer or a loaf of bread.

The tokens could also be exchanged for things other than bread and beer. Those high up enough to earn surplus tokens could redeem them on something else, just in the same way that frequent flyer members with lots of points can redeem them both on flights and on non-flight rewards such as iPads, KitchenAid mixers and Gucci handbags.

A more modern history of loyalty program currencies can be traced to the 1700's. In 1793, a U.S. merchant began rewarding customers with copper tokens, which could be used for future purchases, thereby generating repeat visits, a core focus of loyalty program design. The idea was quickly replicated by other merchants [3].

The Grand Union Tea Company was formed in 1872 in Pennsylvania. The owners chose to side-step merchants and sell their product directly to consumers, starting with door-to-door sales. They began rewarding customers with tickets which could be collected and redeemed for a wide selection of products from the company's Catalog of Premiums, which included such rewards as an Oak Roman Chair (100 tickets), lace curtains (120 tickets a pair), Ormolu clock (300 tickets), and dinner set Berlin 1903 (440 tickets). [4]

In the 1890's, marketers turned to the physical stamp to reward loyal customers. Customers earned stamps when making purchases and were encouraged to stick them into collecting books. The books could be exchanged for a wide range of rewards. The Sperry and Hutchinson Company came to dominate this type of loyalty currency approach with their S&H Green Stamps, which could be earned from a range of different merchants in an early form of coalition program. The program was so popular S&H even opened their own redemption center stores where merchandise could be purchased using books. At one point S&H claimed they were distributing 3 times as many Green Stamps as the US Postal Service was distributing postal stamps. [5]

The 1980's marked the beginning of the end for stamps when American Airlines launched the world's first currency-based frequent flyer program. They introduced a new currency, *miles*, which corresponded to how many miles a member had flown. Brought on by increasing competition with the deregulation of the US airline industry in 1978, the American Airlines AAdvantage program was soon followed by similar plays from United Airlines, TWA and Delta Airlines. Other airlines around the world quickly replicated. In 1987, Southwest Airlines launched a program which awarded '*points*' to members for trips flown, irrespective of the number of miles. Soon after the launch of the early programs, hotel and car rental companies partnered with the airlines and started offering miles and points as a way to grow their market share of the lucrative business travelers and high-value leisure travelers. The first roots of the modern, multi-billion dollar coalition loyalty programs took hold [6].

With the rapid expansion of the frequent flyer programs and their new currencies, other retailers soon replicated their approach, and miles & points became the dominant loyalty-program currencies.

3. MOTIVATION

From a loyalty perspective, the invention of cryptocurrencies is particularly interesting as it provides a viable alternative to miles or points.

Despite their dominance, miles and points (and indeed many of their predecessors) have limitations which restrict their attractiveness to consumers; (a) their lack of utility, (b) their ability to expire and (c) their systematic devaluation by loyalty program operators:

- Limited Utility: Most loyalty programs only allow miles/points to be used within their eco-system. This might be on flights, upgrades, an online store, retail vouchers or other company-specific discounts. One of the key frustrations for many frequent flyer program members is the lack of availability of flights when they try to use their miles/points i.e. they have miles/points but there are no flights they can spend them on. A cryptocurrency doesn't have any of these limitations. It can be bought, sold, transferred, gifted, sent overseas or converted into other cryptocurrencies or fiat currencies.
- Points Expiry: Members who aren't highly-engaged with a program can lose value when their miles or points expire. This might be because the miles/points aged and expired (e.g. points expire 2 years from issuance) or because there was no account activity for a specified period (e.g. points expire if there is no activity on the account for an 18-month period). Major coalition programs use actuaries to deliberately manage the program to maintain a set expiry rate in order to maximise their program profitability. Cryptocurrencies avoid these issues; they don't expire.

- Systematic Devaluation: A major Australian airline loyalty program launched an online store in 2008 which allowed members to redeem points for merchandise and gift cards. This included a $100 gift card for a popular department store for 13,500 points. Today, the same $100 gift card costs 16,800 points. The value of the points has been devalued by the airline to extract more profit from the program. Cryptocurrencies doesn't reduce in value as they become more popular. Market forces of supply & demand support a value increase as the cryptocurrency becomes more desired, ensuring the value accumulated by members also increases.

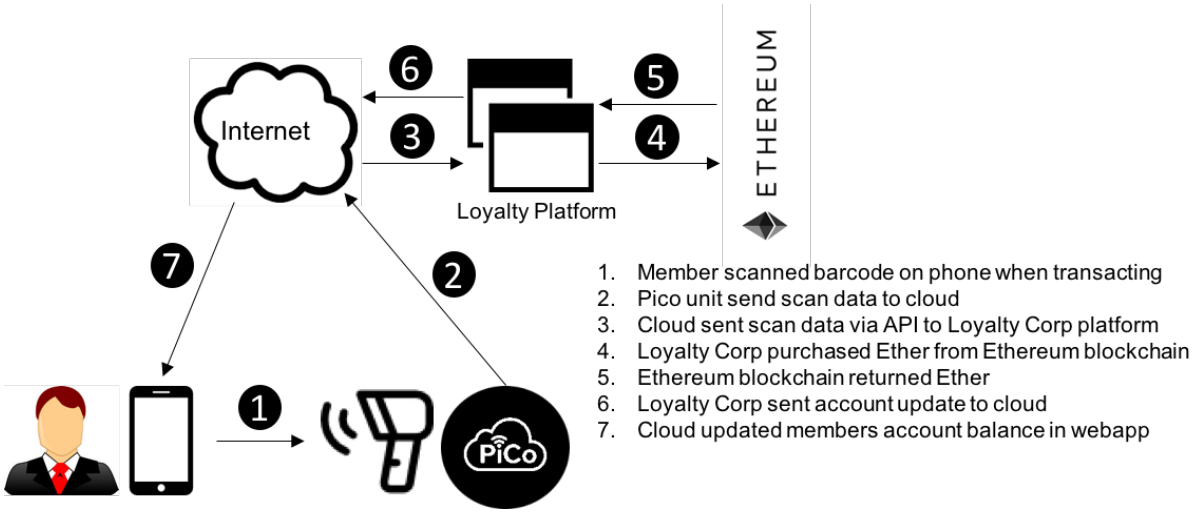


Figure 1: Unify Rewards System Architecture

Based on the benefits which cryptocurrencies provide compared to miles and points, we identified the potential for cryptocurrencies to deliver a more satisfying experience.

The other aspect of cryptocurrencies, the sometimes wild price fluctuations, were also identified as being a compelling characteristic of cryptocurrencies compared to miles/points. While the value of miles/points generally tend to remain static (ignoring any devaluation events), the value of cryptocurrencies such as bitcoin and Ether can fluctuate 25% in a single day. We were interested to understand whether this would be a significant element in affecting the member's overall engagement with the program. We also identified this as a key differentiator to earning cash.

4. UNIFY REWARDS: OVERVIEW OF THE SYSTEM AND THE FIELD TRIAL

It was agreed the best way to conduct the research was by creating a live-market loyalty program called Unify Rewards which mimicked other loyalty programs, with the main difference being the reward currency would be a popular cryptocurrency; Ether. The trial ran from 13th October 2017 to 18th November 2017.

4.1 Merchants

Twelve retailers at UNSW Sydney were enrolled as program merchants. With an actual loyalty program the merchant would be required to cover the cost of the reward currency provided to the participant, however for the purposes of the trial merchants were not required to contribute anything, with all currency costs covered by the project budget.

4.2 System

We recruited two loyalty companies to build the solution; Pico and Loyalty Corp. Pico provide a proprietary cloud-based point-of-sale data collection solution. Honeywell scanners connected to Pico units (comprised of a Raspberry Pi) were placed near the point-of-sale system at each merchant. When the participant scanned a unique barcode from their mobile device, the Pico unit sent the participant ID with a date & time stamp into the cloud, where it was captured and sent via API to Loyalty Corp's platform.

Loyalty Corp provided the front-end and back-end loyalty platform solution. A web app was developed which allowed students & staff to register for the program. Once registered they could access an account which showed their barcode, their account balance, plus it allowed them to process a redemption transaction. The back-end captured the transaction event from Pico and loaded the data into the participant's account real-time, allowing them to see that they had successfully earned for their scan.

When 10 stamps were collected, the Loyalty Corp platform purchased Ether from the Ethereum blockchain and added it to the member's account. Figure 1 depicts the system and outlines the various steps described above.

4.3 Participation Enrolment

The enrolment process was more extensive than most loyalty programs due to a range of additional requirements provided by the University's Ethics Committee.

As the participants were agreeing to a formal research project, they were not only required to provide standard loyalty program registration details (name, email address and password) but they also were required to agree to the university's extensive research participation criteria.
This may have dissuaded some students and staff from completing the registration process, however there is no evidence to support this.

4.4 Earning Cryptocurrency

To earn Ether cryptocurrency, participants conducted a transaction at any of the merchants. Irrespective of the size of the transaction, the participants were permitted to scan their unique barcode via the dedicated scanner. Scanning earned them one digital stamp, which appeared in their web app account. Participants were permitted to earn up to 5 stamps per day. When the participant earned 10 stamps, the stamps automatically converted into Ether.

To ensure participants had the experience of owning Ether for as long as possible, a $5 Ether join bonus was provided to them at the beginning of the trial.

At the start of the project participants could earn $5 of Ether for 10 stamps. From the second week, this was increased to $10 of Ether for a marketing exercise (Double Ether Week) but ended up being maintained for the remainder of the trial.

When a participant earned their Ether allocation, part of an Ether was provided to them, with the amount calculated on the dollar amount they had earned ($10) and the price of Ether at the time of the earn event.

Participants were not required to create a separate Ether cryptocurrency wallet, as their Ether balance was held for them in trust within their loyalty account.

4.5 Redemption

Participants had a range of options for redeeming their Ether balance. Throughout the trial they could:

- Cash their balance into an e-wallet. The Ether was sold at the actual market rate, and they funds were transferred into an e-wallet held within the web app. They could use the balance to access a discount on a range of popular gift cards.
- Cash their balance into a bank account. The Ether was sold at the actual market rate, and they funds were transferred into the participants nominated bank account.
- Transfer their balance to another participant, simply by using the recipient's registered email address.

At the end of the trial, participants were also provided with the opportunity to transfer their balance to their personal Ether Wallet. For those participants who didn't have a wallet, instructions were provided on how to create one.

4.6 Marketing

As with any consumer loyalty program, a range of marketing communications were sent to participants during the trial to educate them and stimulate engagement with the program.

The ambition of the marketing strategy was to persuade as many participants as possible to accumulate at least ten stamps, earning an Ether payout of $5 to $10, in order to provide them with a significant enough experience to be able to meaningfully complete a survey at the end of the trial.

Marketing campaigns during the trial included:

- Welcome email: Provided participants with relevant information to educate them about the essential elements of the trial.
- $5 Ether join bonus: Provided participants with an Ether balance early in the program so they could explore the concept of cryptocurrency ownership more deeply given the trial time constraints.
- Win One Ether competition: Participants received one entry for each stamp they earned to encourage early swiping and engagement.
- Cool earn tips from a member: An educational email detailing insight from a participant on how to maximise stamps earned.
- Double Ether week: Designed to drive ongoing engagement with the program by increasing the prize for earning ten stamps.
- Price of Ether: an educational email detailing the price fluctuations of Ether, designed to generate interest amongst participants in following the price changes.
- Last week of Unify Rewards: Designed to communicate the end date of the program and encourage participants to make the most of their last few days to scan and earn.
- Survey: An invitation to complete the research survey for the program.

5. EVALUATIONS

In the following we present results from the field trial.

5.1 Registration

177 participants registered for the program. Due to delays with the Ethics Committee approval process, the two-week registration window was reduced to 3 days, which included a weekend (thus one business day). Despite the severe reduction in time, the authors were very happy with the high number of registrations.

5.2 Participation

Scans were strongest in the first two weeks of the trial. They dropped off during exam period as many students were not on campus during that period (or frequented campus less regularly).

The spread of total stamps earned during the trial was as follows.

- 21% of participants earn 0 stamps (registered but didn't engage further)
- 18% earned 1-9 stamps
- 61% earned 10 stamps or more (achieving the project target for engagement as it allowed them to earn at least one allocation of Ether)

Even more encouragingly, 18% earned 20 stamps or more.

This is a very high engagement rate for a loyalty program compared to industry averages. By comparison, two major loyalty programs in Australia show member engagement rates of 57% (a major supermarket chain) and 37% (a major liquor chain).

5.3 Marketing Engagement

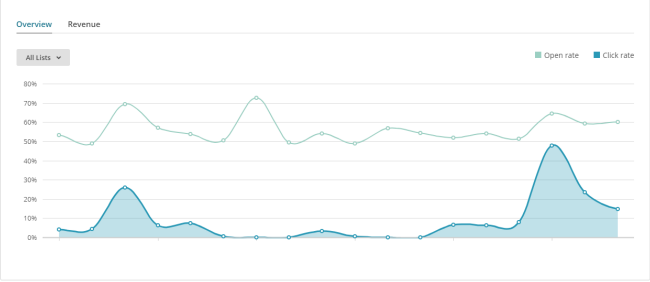


Figure 1: EDM Response

Engagement with the marketing communications was consistently high. The minimum open rate for Electronic Direct Mail (EDM) was 48.3% and the maximum was 72.7%, well above the industry average for loyalty programs which sits at around 20%. Figure 2 illustrates EDM open rates over the trial period. During the trial period, just two participants unsubscribed from communications. This indicates strong engagement with the program by a majority of participants.

5.4 Redemption Behavior

With respect to redemption behaviour:

- 67% of participants chose to transfer their Ether to their personal Ether Wallet
- 29% of participants chose to cash in their Ether for a deposit into their bank account
- 4% of participants chose to cash in their Ether to use for a gift card
- 0% of participants transferred their ETH allocation to another participant

The outcome indicates a strong propensity from a majority of participants to hold their Ether for speculative purposes, an advantage cryptocurrencies have over loyalty points, and one which the survey results identified as being particularly attractive to members. This provides a sharp contract to a program where they might earn cash, which has little speculative potential for the average consumer.

5.5 Survey Results

72 participants completed the post-project qualitative survey. Participants who didn't earn any stamps were not invited to fill in the survey, as it was felt they had not engaged with the program, therefore wouldn't have sufficient insight to provide a meaningful opinion.

Participants indicated they were generally well-exposed to points-based loyalty programs, with only one respondent indicating they didn't belong to any program. This meant participants had sufficient insight to compare a points program to a cryptocurrency program.

Overwhelmingly the results indicate participants found a cryptocurrency-based program to be more engaging than a points-based program.

Respondents reported the following:

- They found Unify Rewards to be more rewarding than their favourite loyalty program (7.58 vs 6.04/10)
- They felt Unify Rewards was more motivating in influencing them to spend their money with participating merchants than their favourite loyalty program (7.80 vs 5.98/10)
- They reported both Unify Rewards and their favourite loyalty program had motivated them to modify the way they spent money to maximise their loyalty currency earn (83% for Unify Rewards vs 80% for their favourite loyalty program). This is strong result for both approaches and provides evidence loyalty programs can be effective in influencing consumer spend behaviour.
- They provided a higher Net Promoter Score for Unify Rewards than their favourite loyalty program (8.53 vs 5.72/10)
- 59% spent more money on campus during the trial period. A further 41% reporting spending the same.
- 86% felt Unify Rewards was more appealing than their favourite loyalty program, and 11% felt it was just as appealing.

Some of the positive reasons cited included:

- The concept is interesting since the value can fluctuate.

- There's a bit of mystery about Ether - it's a bit of a wild card so there's an element of speculation and potential that makes it exciting. But it's not a guaranteed thing.
- Cryptocurrency is cool, exposed me to it
- Cryptocurrency is a very exciting currency as it fluctuates and you never know what to expect the next day. It might go up, or go down, and it is a great experience to learn about how it works and what influences it.
- The possibility of growing value and ability to cash out when you like is very attract.
- More appealing because of the tangible dollar value of the ether as opposed to less tangible points
- Ether feels like you're getting money rather than "points". When Ether was low, I was incentivised to spend and reach the next 10 before Ether spiked.

Some of the negative reasons cited included:

- There is too much fluctuation with cryptocurrency.
- It was an interesting reward, but also felt to be of little difference to cash.

Further analysis of the survey data identified evidence to suggest surveyed participants who were less satisfied with the level of reward from existing loyalty schemes were more likely to find earning Ether more appealing.

While the research was focused on members and not merchants, the verbal feedback from merchants was positive due to the increase in spend by members seeking to earn more Ether. This would only increase with scale.

6. CONCLUSIONS AND FUTURE WORK

The evolution of currencies in loyalty programs shows a long and varied history. Tokens, tickets, stamps, miles and points have all been invested as a device to stimulate loyalty from worshippers and customers, often with great success. They also carry limitations, including limited utility, expiry and devaluation characteristics. With miles & points dominating as the main loyalty currency for over 35 years, it would not be unusual in the history of loyalty for them to be replaced by a new currency design.

Our world-first field trial has shown cryptocurrencies have the potential to be that new currency. The research demonstrated offering Ether as an alternative to miles/points generated very strong engagement with the Unify Rewards program, with 86% of survey respondents reporting they found it to be more appealing that the points they earn from their favorite loyalty program.

While some members drew comparisons with cash, the overwhelming opinion from members indicated they felt cryptocurrencies were more exciting and desirable due to value fluctuations (‘you never know what to expect the next day’) and the potential for a significant future value increase (‘there's an element of speculation and potential that makes it exciting’). It is also telling that 67% of participants chose to hold (or HODL) their Ether rather than cash it in. In that sense, we argue cryptocurrencies injects a unique and highly-engaging gamified element into the program which is absent from points & miles programs, and cash programs.

Some merchants may not appreciate that the cryptocurrencies earned within the program can be transferred externally, rather than reinvested with them. This issue can be offset via quality customer experience design in two ways; firstly, by making it really easy and worthwhile to spend with the merchant, and secondly by allowing the member to transfer other cryptocurrencies into the eco-system to be easily spent with the merchant.

Further research is required to explore the potential of cryptocurrencies in future loyalty program design. The Unify Rewards earn approach, where 10 stamps were required to earn $10 Ether, was simplistic and didn’t take into account the amount of spend made in each transaction. A new research project which ties the amount of cryptocurrency earned to the amount spent would provide an additional insight; whether cryptocurrency loyalty programs are more effective in driving higher transactional spend than miles & points-based programs.

Another aspect which was not possible to measure with Unify Rewards is the effectiveness of a new cryptocurrency in driving engagement behavior. While some companies may choose to utilize existing, popular cryptocurrencies such as Bitcoin and Ether, the bigger opportunity is for a company to create an original cryptocurrency with full control over the amount created and how it is distributed. This would likely involve a greater investment to build awareness of, and desire for, the currency, and would require a longer timeframe to determine any results.

Our research indicated cryptocurrencies do have a key role to play in the future design of loyalty programs, and companies around the world already running a miles/points-based program, or considering implementing one, should seriously consider cryptocurrencies as a viable alternative.

7. REFERENCES

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf. [Accessed 11 12 2017].

[2] B. J. Kemp, Ancient Egypt: Anatomy of a Civilization, 2nd Edition ed., Routledge, 2006.

[3] C. T. &. B. Innovator, "The Loyalty Evolution," New York, 2016.

[4] G. U. T. Company, Grand Union Tea Company - Catalogue of Premiums 1903, 1903.

[5] S. &. Hutchinson, S & H Green Stamps Ideabook of Distinguished Merchandise: 70th Anniversary Edition, 1966.

[6] R. Petersen, "History of Frequent Flyers Program," 2001. [Online]. Available: http://www.webflyer.com/company/press_room/facts_and_stats/history.php. [Accessed 11 12 2017].

Collaboration among Adversaries: Distributed Workflow Execution on a Blockchain

Mads Frederik Madsen
IT University of Copenhagen
Rued Langgaards Vej 7
2300 Copenhagen S,
Denmark
mfrm@itu.dk

Mikkel Gaub
IT University of Copenhagen
Rued Langgaards Vej 7
2300 Copenhagen S,
Denmark
mikg@itu.dk

Tróndur Høgnason
IT University of Copenhagen
Rued Langgaards Vej 7
2300 Copenhagen S,
Denmark
thgn@itu.dk

Malthe Ettrup Kirkbro
IT University of Copenhagen
Rued Langgaards Vej 7
2300 Copenhagen S,
Denmark
maek@itu.dk

Tijs Slaats
University of Copenhagen
Emil Holms Kanal 6
2300 Copenhagen S,
Denmark
slaats@di.ku.dk

Søren Debois
IT University of Copenhagen
Rued Langgaards Vej 7
2300 Copenhagen S,
Denmark
debois@itu.dk

ABSTRACT

We study distributed declarative workflow execution in an adversarial setting. In this setting, parties to an agreed-upon workflow do not trust each other to follow that workflow, or suspect the other party might misrepresent proceedings at a later time. We demonstrate how distributed declarative workflow execution can be implemented as smart contracts, guaranteeing (I) enforcement of workflow semantics, and (II) an incontrovertible record of workflow execution history. Crucially, we achieve both properties *without* relying on a trusted third party.

The implementation is based on the Ethereum blockchain, inheriting the security properties (I) and (II) from the guarantees given by that chain. A recurring challenge for both the implementation and the analysis is the cost of operations on Ethereum: This cost must be minimised for honest parties, and an adversary must be prevented from inflicting extra cost on others.

1. INTRODUCTION

Mutually distrusting organisations must often collaborate, as illustrated by the following example. On the Danish labour market employer-employee disputes are resolved not by the parties themselves, but by the umbrella organisations for respectively Danish employers (abbreviated here "DE") and Danish unions (abbreviated here "DU")[1]. A dispute may be resolved through negotiations between the two parties, or if negotiations break down, in court.

[1]The actual Danish names are "Dansk Arbejdsgiverforening" (DE) and "Landsorganisationen" (DU).

Symposium on Foundations and Applications of Blockchain (FAB '18) March 9, 2018, Los Angeles, California, USA.

Given their conflicting interests, DU and DE are mutually distrusting collaborators. They follow an agreed-upon process when negotiating a dispute, a process which defines simple things like who proposes meeting dates, who submits which document to whom and how, etc.

However, depending on the strength of their respective cases, they may not have equal incentives to follow this process. If an employee has a strong claim to unpaid salary, DE may be less forthcoming in responding to meeting date proposals. Conversely, if an employer is planning legal but unpleasant mass firings, DU may similarly stall the process. Should a case go to court, either party's intransigence may have legal repercussions.

This reluctant collaboration is an example of a cross-organisational workflow between adversaries. System support for such a workflow must provide two key guarantees:

(I) **Workflow correctness.** The system must enforce the agreed-upon workflow, so that no party can obtain an advantage by acting out of turn or failing to fulfil an obligation to act.

(II) **Consensus on history.** The system must provide an incontrovertible record of execution, e.g., to decide in court which party did in fact violate the agreed upon workflow.

The usual way to achieve (I) and (II) is having participants agree on a trusted third party. This third party verifies that the actions taken are within the bounds of the agreement, and meticulously records the proceeding of the case. However, such a third party is not always practical: It may be difficult for the parties to agree on one, and it may be expensive to retain one, especially at large case volumes.

In this paper, we show how (I) and (II) may be achieved without a trusted third party by implementing an executable workflow specification as an Ethereum smart contract.

Our solution is based on recent advances in executable workflow specifications on the one hand and blockchain technologies on the other. A blockchain can be used as a mechanism to produce a trusted, immutable record of workflow execution. E.g. if DU and DE were to store the history of

their common processes on a blockchain, they could both trust this history to be correct with very high probability.

Executable workflow specifications. Agreeing on a record of workflow history is only a part of the puzzle. We must also enforce adherence to the agreed-upon workflow, i.e., the rules governing the exact order in which work can be done. Instead of encoding a workflow directly as a part of the source code of the system, it is typically modelled separately in a *workflow notation* such as BPMN [32], Workflow Nets [1], DECLARE [39], DCR graphs [7, 10], GSM [24], or CMMN [31]. In the best case, such a model is executed by an execution engine embedded in the overall system, enabling straightforward adaptation of work practices by changing the model rather than redeveloping the system itself.

Traditionally, workflow notations have been flow-based, describing processes in a style similar to transition systems, representing precisely the steps that one may go through to satisfy the goals of the process. Such notations work well for strict production processes with little variation, but when applying them to knowledge intensive processes [11], which usually allow a large degree of flexibility and many different paths towards the goals of the process, the models tend to become overly complex and unreadable [39].

Declarative process notations [39, 19, 31, 38] address this deficiency by capturing not explicit flow but rather the constraints and goals of a process, letting the system deduce the allowed paths to the goal. As shown in [20], the declarative approach is highly relevant in the case of DU and DE, whose processes are strongly knowledge intensive.

A declarative process model may be implemented as a *smart contract* [36, 40]: a blockchain where blocks represent not only a common history, but also contracts in the form of executable code. For example, DU and DE have agreed that DU will always propose meetings first; encoding this rule in a smart contract, we can ensure that any attempts to add new events in violation of this rule are rejected.

Contributions. We show that a declarative workflow engine can be employed in an adversarial setting by embedding it on a blockchain as smart contracts. We demonstrate how this approach can be implemented in practice on the Ethereum [40] blockchain, using the smart contract language Solidity and the process execution semantics of DCR graphs [7, 10]. This implementation guarantees (I) correctness with regard to the agreed-upon workflow and (II) the recording of an incontrovertible history. Of course, these guarantees extend no further than the security of the underlying Ethereum blockchain technology, i.e., we assume no adversary can construct Ethereum blocks faster than the honest nodes.

Cost is an issue: Both the cost of participation in the workflow, and the possibility of attacks that inflict cost on honest nodes. In particular, to minimise cost caused by the Ethereum smart contract model (where each computational operation incurs a micro-fee), our cost-effective implementation required both counter-intuitive contract design as well as other non-trivial performance enhancements.

1.1 Related Work

In [41, 16, 30] the authors propose encoding workflows as a smart contract on a blockchain. An implementation of these ideas was given in Caterpillar [28]. In these works, workflows are modelled by BPMN diagrams [32]. This choice of notation clearly separates it from the present work: rather than structured, flow-based processes, we apply the approach to declarative process notations, thereby providing support for knowledge-intensive processes.

In [15], the authors introduce a high-level language, inspired by institutional grammars, that can be compiled into Solidity code. The notation has a declarative feel to it, but describes business contracts rather than workflows. Moreover, the authors do not provide an implementation.

In [23], the authors argue for the suitability of the business artefact paradigm towards modelling business processes on a distributed ledger. The paper lays out their vision, but does not go into detail on neither exact syntax or semantics, nor the exact guarantees offered by smart contracts.

In [20], the DU and DE case was studied in the context of declarative workflow specifications, but relying on a trusted third party for their collaboration. We use this collaboration as a running example; there is otherwise no special relation between the present and this older work.

Both of the properties (I) and (II) are closely related to classical security properties. It was demonstrated in [5] that a workflow notation may encompass security policy specifications. Enforcing distributed adherence to a workflow definition is related to enforcing distributed adherence to a security policy, e.g., [4, 33]. Achieving consensus on the history of a distributed workflow execution is reminiscent of distributed monitoring, e.g., [43, 25].

2. ETHEREUM

Ethereum [42, 40] is a blockchain extended with user-created code and arbitrary data encapsulated in smart contracts. When a transaction is included in a block, part of the verification of that block comprises running the code specified by the transaction, mutating the state of the contract accordingly.

This code is executed on the Ethereum Virtual Machine (EVM), in which each operation has an associated cost denoted in *Gas*. Once the sum of Gas has been calculated for an execution, it is paid for in the Ethereum cryptocurrency *Ether* by the user calling the code, at an Ether/Gas rate specified by that user. This rate allows miners to prioritise those calls paying the most.

The EVM is in principle Turing-complete [42]. However, all computations are in practice finite, limited by the amount of Gas that a caller is willing to spend.

Ethereum allows one to verify the existence of specific source code on the blockchain, whether it has been run, and whether a run was completed successfully or not. Moreover, Ethereum certifies that code was executed as specified, and that only authorised parties execute contract calls [42, 40]. This means that when implementing workflows as smart contracts, any participant can be certain that the source code is unchanged and that every execution is validated with respect to both the contract logic and execution rights.

Like the Bitcoin blockchain, the Ethereum blockchain relies on mining being hard to ensure that the probability of an attacker overtaking the main chain, rewriting history, is low. However, whereas the Bitcoin blockchain and variants has seen work on analysing under what circumstances and with what probabilities that might happen [3, 27, 35, 26, 6, 2, 37, 18] we are unaware of similar analyses for Ethereum.

3. DCR GRAPHS

In this Section, we recall DCR Graphs, a vehicle for specifying admissible sequences of event executions. A DCR Graph specifies an "agreed-upon" workflow, where the events are the activities of the workflow. A DCR Graph comprises *events* (nodes) and *relations* between events (edges); events have state which is recorded in a *marking*. Relations indicate how executability of one event may depend on the states of others, and how execution changes such states.

DEFINITION 1 (DCR GRAPH [19]). *A* DCR Graph *is a tuple* $(\mathsf{E}, \mathsf{R}, \mathsf{M})$ *where*

- E *is a finite set of* events*, the nodes of the graph.*
- R *is the edges of the graph. Edges are partitioned into five kinds:* conditions $(\rightarrow\bullet)$*,* responses $(\bullet\rightarrow)$*,* milestones $(\rightarrow\diamond)$*,* inclusions $(\rightarrow+)$*, and* exclusions $(\rightarrow\%)$*.*
- M *is the* marking *of the graph, a triple* $(\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$ *of sets of events, respectively the previously executed* (Ex)*, the currently pending* (Re)*, and the currently included* (In) *events.*

When G is a DCR Graph, we write, e.g., $\mathsf{E}(G)$ for the set of events of G, as well as, e.g., $\mathsf{Ex}(G)$ for the executed events in the marking of G.

We give in Figure 1 an excerpt of the workflow of DU and DE reported in [20]. The events are nodes in the graph; the marking of each event is shown graphically: Hold Meeting is pending, viz. the blue exclamation mark; both Accept events are excluded viz. the dashed border.

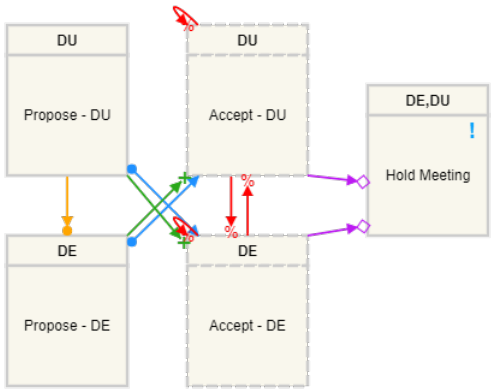


Figure 1: The DU/DE example—a DCR model of a cross-organisational workflow

By default, every activity may execute any number of times. We regulate the sequencing of such activity executions by adding relations between activities. There are five such relations: Three which mutate the state of some events when another executes, and two which constrain the ability of one event to execute depending on the state of others.

3.1 Execution of Events

To specify what happens when an event executes, we have the *response*, *inclusion*, and *exclusion* relations.

First, the *response*. When either DE or DU proposes a date, the other is required to eventually accept one. The blue *responses* $(\bullet\rightarrow)$ from Propose - DU to Accept - DE and Propose - DE to Accept - DU model this requirement: Executing the first event makes the second event *pending*.

The red *exclusions* $(\rightarrow\%)$ temporarily remove events from the process. This can be both an event removing itself after being executed, as is the case for each instance of Accept, or an event removing another event, exemplified by Accept - DU removing Accept - DE and vice versa. We say that such a removed event is *excluded*, indicated by a dashed border, as seen in the two Accept events.

Exclusions are dynamic and may be reverted: When DU or DE proposes new dates, the other is expected to accept these dates again. This is modelled through the green *inclusions* $(\rightarrow+)$ from Propose - DU to Accept - DE and Propose - DE to Accept - DU. Because Accept - DU and Accept - DE are excluded (dashed border), either requires its including event to happen before it can itself happen.

We formalise the notion of executing an event.

Notation. For a binary relation $\rightarrow \subseteq X \times Y$ and set Z, we write "$\rightarrow Z$" for the set $\{x \in X \mid \exists z \in Z.\ x \rightarrow z\}$, and similarly for "$X \rightarrow$". For singletons we usually omit the curly braces, writing $\rightarrow e$ rather than $\rightarrow\{e\}$.

DEFINITION 2 (EXECUTION [19]). *Let $G = (\mathsf{E}, \mathsf{R}, \mathsf{M})$ be a DCR Graph with marking $\mathsf{M} = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$. If we* execute *$e$ in G, we obtain the* resulting *DCR graph $(\mathsf{E}, \mathsf{R}, \mathsf{M}')$ with $\mathsf{M}' = (\mathsf{Ex}', \mathsf{Re}', \mathsf{In}')$ defined as follows.*

1. $\mathsf{Ex}' = \mathsf{Ex} \cup e$
2. $\mathsf{Re}' = (\mathsf{Re} \setminus e) \cup (e\bullet\rightarrow)$
3. $\mathsf{In}' = (\mathsf{In} \setminus (e\rightarrow\%)) \cup (e\rightarrow+)$

That is, to execute an event e one must: (1) add e to the set Ex of executed events; (2) update the currently required responses Re by first removing e, then adding any responses required by e; and (3) update the currently included events by first removing all those excluded by e, then adding all those included by e.

3.2 Enabled Events

Not all events in a graph are necessarily allowed to execute. To specify which events are in fact executable we have *conditions* and *milestones*. A condition indicates that when the source is included but not executed, the target cannot execute. For example, by convention, DU is always the first to propose dates. This is modelled by the condition relation$(\rightarrow\bullet)$ between Propose - DU and Propose - DE.

When dates have been proposed but not yet accepted, the meeting cannot be held. The milestone relations $(\rightarrow\diamond)$ from Accept - DU and Accept - DE to Hold Meeting ensure this: a milestone indicates that whenever the source is included and pending, the target cannot execute. In the diagram, the Accept events are not yet pending. This is intentional: DU and DE may skip proposing dates and hold ad hoc meetings.

Unlike the condition relation, an event constrained by a milestone can become blocked again. In our example, if a date was accepted but later new dates are proposed, accepting dates becomes pending again, blocking Hold Meeting.

We give formal meaning to these relations.

DEFINITION 3 (ENABLED EVENTS [19]). *Suppose $G = (\mathsf{E}, \mathsf{R}, \mathsf{M})$ is a DCR Graph with marking $\mathsf{M} = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$. We say that an event $e \in \mathsf{E}$ is* enabled *and write $e \in \mathsf{enabled}(G)$ iff (a) $e \in \mathsf{In}$, (b) $\mathsf{In} \cap (\rightarrow\bullet e) \subseteq \mathsf{Ex}$, and (c) $\mathsf{In} \cap (\rightarrow\diamond e) \subseteq \mathsf{E}\setminus\mathsf{Re}$.*

That is, enabled events (a) are included, (b) have their included conditions already executed, and (c) have no included pending milestones. The enabled events for the DCR Graph in Figure 1 are Propose - DU and Hold Meeting.

General DCR Graphs have labelled events, allowing distinct events to exhibit the same observable activity, a detail we have elided in the current paper. In the general case, DCR Graphs express the union of regular and ω-regular languages [10].

3.3 Distributed DCR Graphs

Distributed implementations of DCR Graphs were studied in [22] and [9]. In both cases, the core idea is that workflows are partitioned in subsets of events, with each participant owning a particular subset. The owner of an event is responsible for maintaining the marking (M, Definition 1) of that event. Moreover, only the owner of an event can execute it (Definition 2).

Since executing one event may modify others via exclusion, inclusion and response arrows (Definition 2), whenever a party executes an event, it may have to notify owners of affected events. E.g., in the DU/DE example (Figure 1), events are naturally owned by either DU or DE as indicated at the top of each event. The event Propose - DU is owned by DU and so can only be executed by DU; however, executing this event includes the event Accept - DE, and so DU must notify DE whenever it executes Propose - DU, in order that DE may toggle the state of Accept - DE to included.

Similarly, before executing an event, the owner must verify that the event is enabled (see Definition 3). Whether an event is enabled is a function of the marking of other events via condition or milestone relations, hence the owner may have to query owners of such other events. E.g., in the DU/DE example, DE cannot execute Propose - DE before querying DU about the state of Propose - DU because of the condition relation from the latter to the former.

As queries for enabledness may interleave with effects of an execution, distributed implementations of DCR Graphs generally need some form of concurrency control [9].

4. DISTRIBUTED DCR GRAPHS AS ETHEREUM SMART CONTRACTS

In this section, we consider in the abstract an implementation of distributed DCR Graphs as Ethereum smart contracts. We shall see how such an implementation achieves the goals (I) and (II) of Section 1 *provided* an adversary has no feasible attack on the Ethereum blockchain.

The naive implementation of DCR Graphs as Ethereum smart contracts is to simply implement a contract comprising a DCR Graph (Definition 1) represented as an Ethereum data structure, and calls for computing execution and enabled events (Definitions 2 and 3). Only the owner of an event has access rights to execute that event. This appealingly simple idea turns out to mask considerable pitfalls, in particular regarding who bears the cost of executing that call. In this section, we analyse this situation.

4.1 Cost of Relations

Previous treatments of distributed DCR graphs [9, 22] do not emphasise ownership of *relations*. Adding a relation to a DCR Graph induces additional computation in either enabledness (condition, milestone) or effect of execution (inclusion, exclusion, response). On Ethereum, additional computation translates directly to additional cost, so an adversary can inflict cost on honest parties if he can add new relations. For example, adding 100 distinct conditions $A_i \rightarrow\bullet\ X$ to some event X would increase the cost of computing enabledness of X by 100 additional checks whether each A_i is executed or excluded.

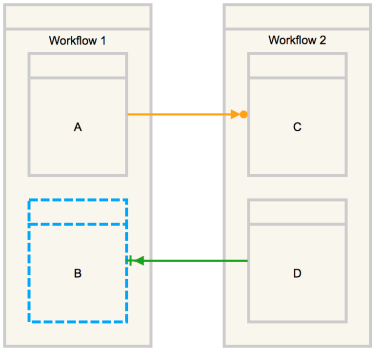


Figure 2: Inter-workflow relations

For *conditions and milestones*, each such relation induces computational cost at the owner of the *target* event. For example, in Figure 2, Workflow 2 must consult Workflow 1 to learn the state of A before it can execute event C. In general, adding an incoming relation such as $A \rightarrow\bullet\ C$ increases the cost of computing enabledness of its target C. To avoid cost-inflicting attacks, only the owner of the target C should be allowed to add incoming relations to it.

However, because executing D requires an update of the state of B, if that update is to be performed by the owner of B, there is again an opportunity for an adversary to inflict cost. In that case, we must again require that only the owner of B and D jointly may add relations.

We summarise where adding relations incurs cost Table 1.

Added relation	cost on A	cost on B
$A \rightarrow\bullet\ B$		✓
$A \rightarrow\diamond\ B$		✓
$A \rightarrow+\ B$	✓	
$A \rightarrow\%\ B$	✓	
$A \bullet\rightarrow B$	✓	

Table 1: Incurred cost of added relations

4.2 Correctness

In general, adding relations to a workflow can make that workflow both more and less restrictive [8, 10]. For example, in Figure 2, the condition relation (top) means that Workflow 2 must wait for Workflow 1 to execute A before it can execute activity C. If we imagine we have just added that condition, the new combined workflow has less behaviour than the old one, but no new behaviour. Thus, an adversary who can add relations can mount a potential denial-of-service attack by adding enough relations that the resulting combined workflow has *no* behaviour left.

Conversely, adding inclusions and exclusions can make a workflow less restrictive [8]. Without the inclusion relation

(bottom) in Figure 2, Workflow 1 can never execute activity B. If we imagine we have just added that inclusion, the new combined workflow has *more* behaviour than the original one, since the new one admits the sequence DB which the old one did not. This means that an adversary who can add relations can violate correctness of the original workflow. E.g. if the activity B were "pay out lump sum", the adversary has succesfully orchestrated a payout in violation of the original workflow policy.

Assume a correct implementation of (1) the computation of enabled events and (2) the effect of executing event in Solidity. Assume moreover that this implementation is used for implementing the distributed workflow in such a way that each party to the workflow can execute only the events they own, and only when these events are enabled. In this case, running this implementation on Ethereum, we get an implementation of the workflow which *automatically* achieves the goals of workflow correctness (I) and consensus on history (II) *provided* the adversary cannot produce valid blocks fast enough to outpace the Ethereum miner network.

Note that in workflows with more than two participants, we do not preclude colluding actors *within* the bounds of concurrent workflow semantics. In such a workflow, two or more participants could mount a denial-of-service attack against other participants by coordinating executions of activities on the same block, thereby skipping states in which specific activities were enabled. This is an inherent consequence of allowing concurrent executions of activities in DCR-graphs, and *not* a violation workflow correctness (I).

5. COST REDUCTIONS

Our practical experiments have revealed two major insights about executing DCR Graphs on Ethereum:

1. It is indeed feasible to implement distributed workflows in an adversarial setting on the Ethereum blockchain.
2. However, to keep costs manageable, our implementation must take some counter-intuitive design decisions, including implementing only one contract and implementing set operations as bitvectors.

DCR Graphs as presented in Section 3 are simple enough that the core data structures (relations and markings, Definition 1) as well as operations on them (execution and enabledness, Definitions 2 and 3) are straightforward to implement in contemporary programming languages.

A naive implementation implements DCR Graphs straightforwardly as an Ethereum contract for each workflow instance, representing marking and relations straightforwardly using standard data structures. This naive implementation has two shortcomings:

1. The Gas costs of Ethereum are dominated by the price of creating a smart contract, which is an order of magnitude more expensive than other operations [42].
2. The cost of computing enabledness respectively execution grows linearly with the number of incoming respectively outgoing relations.

5.1 Relations

To reduce the impact of additional relations on the cost of computing enabledness and execution, we exploit that the core EVM datatype is a 256-bit value, noting that the core operations of DCR Graphs (Definitions 3 and 2) are all simple set-manipulations and can be implemented efficiently as *bit vectors*.

Our prototype for this reason assumes at most 256 events in a DCR Graph, an assumption that is both practically reasonable [29, 34], and straightforward to remove if necessary.

For such fixed-size bit vectors, we get an upper bound of the cost of executing an activity: execution is implemented as a static check of the legality of the execution, followed by 3 bitwise-operations between bit vectors representing relations. We give the implementation of the enabledness computation in Listing 1; we encourage the reader to compare that listing, in particular lines 16, 20–21, and 25–27 to the clauses (a)-(c) in Definition 3.

```
1  function canExecute(uint256 wfId, uint256 activity)
2    public constant returns (bool)
3  {
4    var workflow = workflows[wfId];
5    uint32 i;
6
7    // sender address must have execute rights
8    for (i = 0; i < workflow.authAccounts.length; i++)
9      if (workflow.authAccounts[i] == msg.sender)
10       break; // sender authorised
11
12   if (i == workflow.authAccounts.length)
13     return false; // sender not authorised
14
15   // activity must be included --- Def. 3(a)
16   if ((workflow.included & (1<<activity)) == 0)
17     return false;
18
19   // all included conditions executed --- Def. 3(b)
20   if(workflow.conditionsFrom[activity] &
21      (~workflow.executed & workflow.included) != 0)
22      return false;
23
24   // no included milestones pending --- Def. 3(c)
25   if(workflow.milestonesFrom[activity]
26      & (workflow.pending & workflow.included) != 0)
27      return false;
28
29   return true;
30 }
```

Listing 1: Enabled computation

Besides the optimisations we have mentioned so far, our prototype implementation uses additional tricks to minimise Gas costs, notably packaging call data to conserve storage space. We refer the interested reader to [17], which contains additional implementation detail.

As mentioned in Section 2, execution of Ethereum smart contracts is paid for by setting an exchange rate between Gas, the cost of execution instructions, and the cryptocurrency Ether. We compare the cost of the naive and optimised implementations in Table 2.

Note that the cost of executing some activities actually *increases* from naive the to the optimised implementation: the bit vector implementation give lower Gas cost on execution only when events have many relations. The DU/DE example is too small to exhibit this effect; however, practical workflows tend to have many more relations [21].

Event	Naive		Optimised	
	GAS	USD*	GAS	USD*
1. Initialisation**	2,185,061	14.582	717,709	4.790
2. Propose - DU	61,126	0.408	66,293	0.442
3. Propose - DE	62,592	0.418	52,615	0.351
4. Accept - DU	46,126	0.308	51,293	0.342
5. Accept - DE	46,226	0.308	52,615	0.351
6. Hold Meeting	37,353	0.249	49,665	0.331
Sum	2,392,258	16.273	990.190	6.608

* Prices in USD are computed from average Gas- and Ether prices at the time of writing [13, 14].

** Prices for the naive implementation includes contract creation and workflow creation; prices for the optimised implementation only workflow creation.

Table 2: Cost comparison, naive and optimised implementation.

5.2 Contract Creation & Access Control

The cost of creating an instance of the DU/DE example workflow is given in Table 2, column "Naive". Notice that creating the contract, "initialisation" is two orders of magnitude more expensive than subsequent event executions.

To reduce this cost, we propose a *mono-contract implementation*, that is, a single contract which hosts *all* workflows, and new workflows can be added at any point after contract creation. In this mono-contract implementation methods each take an index of the workflow to work on. In such a setting, the cost overhead for creating a contract is incurred only *once*[2]: As subsequent workflows are hosted by this single contract, the cost of creating a contract does not reoccur. The cost of constructing a new workflow is reduced substantially, see the column "optimised" in Table 2.

The mono-contract provides access control by accepting, on workflow creation, a list of public keys/addresses that are authorised to subsequently execute events; the implementation manually checks that the caller is authorised before executing an event. Because state in Ethereum can only be changed through contract calls, this mechanism provides complete mediation: there is no way to alter the state of running workflows without going through the contract operations. Returning to the code for the enabledness computation in Listing 1, we see access control computed in line 7–10, using the Ethereum provided `msg.sender` constant.

As mentioned, workflow creation is an order of magnitude cheaper in the optimised implementation. Moreover, in the naive implementation, workflow creation is two orders of magnitudes more expensive than event executions; in the optimised implementation only one.

6. IMPLEMENTATION

We have implemented a software tool which converts a DCR Graph to a Solidity smart contract. To show that our DCR engine can be used in practice, we have implemented a graphical user interface (GUI), where users can create workflows and execute activities on a deployed Ethereum contract. We host the source code of the contracts at https://github.com/DCReum/dcreum.github.io for perusal, and the GUI for anyone to use at https://dcreum.github.io. An Ethereum node and client are required to view and use the GUI. We recommend Parity alongside the Google Chrome extension Parity Ethereum Integration.

Multiple high-level languages compiling to EVM bytecode exist; our implementation was done in the statically typed object-oriented language Solidity. However, compared to main-stream programming languages, in EVM/Solidity we have to contend additionally with quirks of the Solidity interpreter. For example, Solidity limits the number of variables allowed in scope at one time, as these are always kept on the stack, and the EVM only allows access to the 16 topmost items [42]. Other limitations include externally available functions not being allowed structs or nested arrays as arguments or return type.

Ethereum execution causes a delay in event execution, as the network has to process and accept such an execution. We can have the acceptance of a transaction (event execution) prioritised by offering above-market Gas prices. Unless we send the request at almost exactly the time of new block propagation, in experiments run late spring 2017, our requests have been included in the next mined block when paying market Gas prices. However, even though a block is accepted, it may still find itself on a less difficult chain, and thus eventually discarded. In general, like in Bitcoin and other blockchain-based transactional systems, one must wait some number of blocks before one can reasonably assume that the transaction is permanently included.

The frequency of event executions is bounded by the (dynamic) Gas limit [42]. This limit is currently at 6,718,941, which for the DU/DE example (Figure 1) in theory would allow between 101 and 135 executions, depending on the exact activity executed. If we consider instead single-participant, non-concurrent executions, the limit is the mining time, which should average 12 seconds, although at the time of writing, the average for the last 5000 blocks is ca. 30 seconds. In general, it has been our experience that mining time varies between a few seconds and several minutes. We estimate we have seen an average of 1-2 executions per minute at market Gas prices.

7. CONCLUSION

We have demonstrated how to implement distributed declarative workflow execution in an adversarial setting, without the assistance of a trusted third party, by implementing DCR Graph declarative process models as Solidity contracts running on the Ethereum blockchain. Within the the security guarantees given by this blockchain, this implementation guarantees that the execution does follow the agreed-upon workflow—the DCR Graph—(I) and that the sequence of executions recorded on the blockchain is incontrovertibly the actual recorded history (II).

Cost is an issue, both because an adversary must be prevented from inflicting cost on an honest party, and because cost of contract execution is high enough that we must optimise. Particularly helpful optimisations are the mono-contract and bitvector representation of sets and relations.

We have demonstrated the economic feasibility of the implementation: see actual costs in Table 2. Moreover, we have discussed bounds on delay and frequency of event executions in Section 6, estimating that the Ethereum blockchain can likely sustain 1-2 execution per minute at market prices.

[2]This contract was created in the transaction [12] at a cost of 2,976,162 Gas/USD 8.73.

8. REFERENCES

[1] W. M. P. v. d. Aalst. Verification of Workflow Nets. In *Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, ICATPN '97, pages 407–426, London, UK, UK, 1997. Springer-Verlag.

[2] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis. Chainspace: A Sharded Smart Contracts Platform. *arXiv:1708.03778*, 2017.

[3] S. Barber, X. Boyen, E. Shi, and E. Uzun. Bitter to better—how to make bitcoin a better currency. In *Proc. of FC '12*, pages 399–414. Springer, 2012.

[4] D. Basin, M. Harvan, F. Klaedtke, and E. Zalinescu. Monitoring usage-control policies in distributed systems. In *Proc. of TIME '11*, pages 88–95. IEEE, 2011.

[5] D. A. Basin, S. Debois, and T. T. Hildebrandt. In the Nick of Time: Proactive Prevention of Obligation Violations. In *Proc. of CSF '16*, pages 120–134. IEEE, 2016.

[6] E. Buchman. *Tendermint: Byzantine Fault Tolerance in the Age of Blockchains*. PhD thesis, 2016.

[7] S. Debois and T. Hildebrandt. The DCR Workbench: Declarative Choreographies for Collaborative Processes. In *Behavioural Types: from Theory to Tools*, River Publishers Series in Automation, Control and Robotics, pages 99–124. River Publishers, June 2017.

[8] S. Debois, T. Hildebrandt, and T. Slaats. Safety, liveness and run-time refinement for modular process-aware information systems with dynamic sub processes. pages 143–160, 2015.

[9] S. Debois, T. T. Hildebrandt, and T. Slaats. Concurrency and Asynchrony in Declarative Workflows. In *Proc. of BPM '15*, volume 9253 of *LNCS*, pages 72–89. Springer, 2015.

[10] S. Debois, T. T. Hildebrandt, and T. Slaats. Replication, refinement & reachability: complexity in dynamic condition-response graphs. *Acta Informatica*, Sept. 2017.

[11] C. Di Ciccio, A. Marrella, and A. Russo. Knowledge-intensive processes: characteristics, requirements and analysis of contemporary approaches. *J. on Data Semantics*, 4(1):29–57, 2015.

[12] Mono-contract creation transaction. https://etherscan.io/tx/0x003fb07eb74b4a2557dc48fa-8e7799e481f98e0c4ed0857ce646dbe3b9b90cda.

[13] Ethereum average gasprice chart. https://etherscan.io/chart/gasprice.

[14] Ethereum/us dollar (eth/usd) price chart. https://www.coingecko.com/en/price_charts/ethereum/usd.

[15] C. K. Frantz and M. Nowostawski. From institutions to code: Towards automated generation of smart contracts. In *Proc. of FAS*W '16*, pages 210–215, Sept 2016.

[16] L. García-Bañuelos, A. Ponomarev, M. Dumas, and I. Weber. Optimized execution of business processes on blockchain. In *Proc. of BPM '17*, pages 130–146. Springer, Cham, 2017.

[17] M. Gaub, M. E. Kirkbro, F. Madsen, and T. Högnason. Consensus in declarative process models using distributed smart-contracts. 2017.

[18] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun. On the Security and Performance of Proof of Work Blockchains. In *Proc. of SIGSAC '16*, CCS '16, pages 3–16, New York, NY, USA, 2016. ACM.

[19] T. Hildebrandt and R. R. Mukkamala. Declarative Event-Based Workflow as Distributed Dynamic Condition Response Graphs. In *Post-proc. of PLACES '10*, volume 69 of *EPTCS*, pages 59–73, 2010.

[20] T. Hildebrandt, R. R. Mukkamala, and T. Slaats. Designing a cross-organizational case management system using dynamic condition response graphs. In *Proc. of EDOC '11*, pages 161–170. IEEE, 2011.

[21] T. T. Hildebrandt, R. R. Mukkamala, and T. Slaats. Nested Dynamic Condition Response Graphs. In F. Arbab and M. Sirjani, editors, *Proc. of FSEN '11*, volume 7141 of *Lecture Notes in Computer Science*, pages 343–350. Springer, Apr. 2011.

[22] T. T. Hildebrandt, R. R. Mukkamala, and T. Slaats. Safe distribution of declarative processes. 7041:237–252, 2011.

[23] R. Hull, V. S. Batra, Y.-M. Chen, A. Deutsch, F. T. Heath, and V. Vianu. Towards a shared ledger business collaboration language based on data-aware processes. In *Proc. of ICSOC*, 2016.

[24] R. Hull, E. D. R. D. Masellis, F. Fournier, M. Gupta, F. Heath, S. Hobson, M. Linehan, S. Maradugu, A. Nigam, P. N. Sukaviriya, and R. Vaculín. *A Formal Introduction to Business Artifacts with Guard-Stage-Milestone Lifecycles*. 2011.

[25] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a distributed firewall. In *Proc. of CCS '00*, pages 190–199. ACM, 2000.

[26] A. Kiayias and G. Panagiotakos. On Trees, Chains and Fast Transactions in the Blockchain. *IACR Cryptology ePrint Archive*, 2016:545, 2016.

[27] Y. Lewenberg, Y. Sompolinsky, and A. Zohar. Inclusive block chain protocols. In *Proc. of FC '15*, pages 528–547. Springer, 2015.

[28] O. López-Pintado, L. García-Bañuelos, M. Dumas, and I. Weber. Caterpillar: A Blockchain-Based Business Process Management System. In *Demo Track, BPM '17*, 2017.

[29] M. Marquard, M. Shahzad, and T. Slaats. Web-based Modelling and Collaborative Simulation of Declarative Processes. In *Proc. of BPM '15*, pages 209–225, 2015.

[30] J. Mendling, I. Weber, et al. Blockchains for business process management-challenges and opportunities. *arXiv:1704.03610*, 2017.

[31] Object Management Group. Case Management Model and Notation. Technical Report formal/2014-05-05, Object Management Group, May 2014. Version 1.0.

[32] Object Management Group BPMN Technical Committee. *Business Process Model and Notation, Version 2.0*. 2013.

[33] A. Pretschner, M. Hilty, and D. Basin. Distributed usage control. *Comm. of the ACM*, 49(9):39–44, 2006.

[34] T. Slaats, R. R. Mukkamala, T. T. Hildebrandt, and M. Marquard. Exformatics Declarative Case Management Workflows as DCR Graphs. In *Proc. of BPM '13*, pages 339–354, 2013.

[35] Y. Sompolinsky, Y. Lewenberg, and A. Zohar. SPECTRE: A Fast and Scalable Cryptocurrency Protocol. *IACR Cryptology ePrint Archive*, 2016:1159, 2016.

[36] N. Szabo. Formalizing and Securing Relationships on Public Networks. *First Monday*, 2(9), Sept. 1997.

[37] J. Teutsch and C. Reitwießner. A scalable verification solution for blockchains. 2017.

[38] R. Vaculín, R. Hull, T. Heath, C. Cochran, A. Nigam, and P. Sukaviriya. Declarative business artifact centric modeling of decision and knowledge intensive business processes. In *Proc. of EDOC '11*, pages 151–160, 2011.

[39] W. M. van Der Aalst, M. Pesic, and H. Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science-Research and Development*, 23(2):99–113, 2009.

[40] B. Vitalik. A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper, 2014.

[41] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling. Untrusted business process monitoring and execution using blockchain. In *Proc. of BPM '16*, pages 329–347. Springer International Publishing, 2016.

[42] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[43] X. Zhang, J.-P. Seifert, and R. Sandhu. Security enforcement model for distributed usage control. In *Proc. of SUTC '08*, pages 10–18. IEEE, 2008.

Unchain Your Blockchain

Tamraparni Dasu
AT&T Labs-Research
tamr@research.att.com

Yaron Kanza
AT&T Labs-Research
kanza@research.att.com

Divesh Srivastava
AT&T Labs-Research
divesh@research.att.com

ABSTRACT

Blockchain is emerging as a preeminent decentralized ledger and receiving increasing attention from researchers, practitioners, organizations and the public. Initially, blockchain was developed to address the "double spending" problem in cryptocurrencies, but recently, many new applications of blockchain have been proposed or are being developed. Blockchain allows sharing data in a decentralized, transparent and immutable way, using a peer-to-peer network, without the need to trust any particular entity. To achieve that in public blockchain, where the peers are *a priori* unknown, efficiency and scalability are often sacrificed.

In this paper we present a novel partition of the blockchain into smaller chains, to allow association of sub-chains, wallets and transactions with real-world concepts, such as geographical areas, and by this, improve scalability and security. Our contribution is threefold. First, we discuss the utilization of a real-world hierarchical structure, such as a geospatial subdivision, to partition the ledger into a tree of connected blockchains, in order to increase scalability and provide a tradeoff between privacy and transaction latency. Second, we illustrate the use of a geospatial partitioning to support geofencing, in order to add security to cryptocurrencies and other blockchain applications. Third, we present *proof-of-location* as an alternative to proof-of-work, to cope with the large waste of energy caused by proof-of-work, which may be inflated by the partitioning.

CCS Concepts

•**Security and privacy** → **Distributed systems security;** •**Information systems** → *Spatial-temporal systems;* •**Computing methodologies** → Distributed algorithms;

Keywords

Blockchain, Proof-of-Location, PoL, decentralized ledger, hierarchical partitioning, immutable storage, cryptocurrency, Bitcoin, geofencing, localized ledger

Symposium on Foundations and Applications of Blockchain (FAB '18) March 9, 2018, Los Angeles, California, USA.

1. INTRODUCTION

Blockchain and cryptocurrencies have a growing economic influence. They are beginning to revolutionize industries [36, 37], and are considered by many to be a *game changer* in areas like finance, insurance, notary, copyright protection, distribution of digital arts, and so on. Hence, they are receiving a growing attention from researchers and practitioners.

Blockchain is a decentralized public ledger that was initially introduced as a solution to the "double spending" problem in cryptocurrencies like Bitcoin [25]. It provides an immutable storage of transactions in a chain of blocks. The chain is created in a decentralized fashion by peers, using a peer-to-peer network, without any central node to govern it or enforce rules. The chain structure provides a serialization of the stored transactions, to prevent double spending.

Recently cryptocurrencies have flourished, and in particular, the importance of Bitcoin has increased, as it becomes an acceptable method of payment to a growing number of organizations and companies. Cryptocurrencies facilitate micropayments, provide anonymity to both the payer and the payee, and lay the basis for an economy without regulation. This challenges the traditional economic order [40].

Blockchain is receiving growing attention not just as the underlying technology of cryptocurrencies, but also as a public ledger in various domains, as elaborated below.

- **Financial transactions:** Financial institutions are examining the use of blockchain as a ledger for financial transactions, to cut out the middleman to reduce costs and expedite the processing of transactions [39].

- **Digital assets:** Blockchain can be used to maintain digital assets such as stocks, bonds, land titles, etc. Stored transactions record the transfer of assets between users [10].

- **Evidence of data and documents:** The blockchain stores data and documents, either in full or merely a digest of the data (e.g., using a cryptographic hash like SHA-256). The aim is to provide an evidence of the existence of data or documents, such as contracts, patents, scientific publications, deeds, insurance policies, etc. [36].

- **Identity management:** Using blockchain for identity management is examined [2, 3]. Hashed features of a person (digest of verifiable attributes of the person) are stored with a public key or some other means to electronically sign documents or access remote ser-

vices. The aim is to protect people from identity theft and fraudulent impersonation.

- **Sharing data:** Blockchain has the potential to provide a secure infrastructure for smart cities [8, 35], and could facilitate the creation of a marketplace of social data [21] where people share their private data for public benefit.
- **Commercial use:** Blockchain-based applications are developed for tracking diamonds from the mines to the market, managing data provenance in IoT systems [5, 24], to provide transparency in product manufacturing and supply chain management [42], and support vehicle provenance [39].

While the importance of blockchains is growing rapidly, it still has drawbacks and limitations that raise concerns regarding its scalability and suitability to large-scale applications. A notable concern is that the creation and maintenance of a public blockchain cause a significant waste of energy due to excessive work by the involved peers. Leading blockchains, such as Bitcoin, are based on Proof-of-Work [4, 19], where the peers, called *miners*, need to execute a demanding computation to create a block. It was estimated that the energy consumption of maintaining Bitcoin exceeds the energy consumption of Ireland [26]. The energy consumption continues to grow as more miners join the network.

Another concern is the low rate of transactions. In Bitcoin, a block is created approximately every 10 minutes, and the size of a block is fixed (1 MB in Bitcoin, 2MB in SegWit2x, and 8MB in Bitcoin Cash), and the rate of adding transactions to the blockchain is around 7 transactions per second.[1] Such a limitation exists in other blockchains as well, e.g., it is estimated that in Ethereum the transaction rate is about 10–30 transactions per second.[2] This is several orders of magnitude smaller than the transaction rate that modern financial institutions are able to process (e.g., more than 30,000 transactions per second in VISA). Changing the block-creation rate or the size of a block is difficult because a blockchain is decentralized, without any entity that can force a change or enforce new rules. In addition, rapid block-creation may result in frequent forks, which would make the blockchain less stable and more vulnerable to attacks.

Anonymity in cryptocurrencies like Bitcion provides some advantages but also creates risks. A money transfer from an owner of coins to a payee requires merely a signature using the private key of the payer. If the private key of a coin owner is revealed or stolen, the coin can be stolen. A lost private key is like lost money. Thus, cryptocurrencies are susceptible to theft and money loss.

In this paper, we envision a partitioning of blockchain into a hierarchy of sub-chains, reflecting a real-world subdivision, to increase scalability and security. We illustrate a geospatial partitioning and explain how localization and location certificates [20, 31] can be used to reliably establish association with sub-chains. The levels of the hierarchy provide a tradeoff between privacy and confirmation time of transactions. To prevent inflated energy consumption when replacing a single blockchain by many sub-chains, we introduce a novel *proof-of location* (PoL) approach that mitigates the energy consumption problem.

[1] `https://blockchain.info/charts/transactions-per-second`

[2] `https://etherchain.org/charts/tps`

2. BACKGROUND

We start by providing some background. We mainly refer to cryptocurrencies, to simplify the discussion, but the methods we suggest can be applied to other domains as well.

2.1 Blockchain

Blockchain is a decentralized ledger that stores transactions in a chain of blocks. In cryptocurrencies, a transaction can be a reward to the creator of a block, or a transfer of coins from the owner to a payee. Each transaction includes the public key of the payee. Transactions form a chain of coin transfers. To transfer money, the owner of the coins signs the transfer using the private key that matches the public key in the transaction that granted her/him the coins. Given coins and the transaction t that granted them, only someone who possesses the private key that matches the public key in the transaction t can spend the coins, i.e., transfer them on. In many blockchains, user identities are not revealed, to provide anonymity, hence, money transfer is between *wallets*, where a user may have many wallets.

We denote by $t = (x \to y, m)$ a transaction that transfers m coins from wallet x to wallet y. We denote by $t = (\to y, m)$ a transaction that grants m coins to y as a reward.

To prevent double spending, the transactions are added to the blockchain and are publicly visible. The chain defines a serialization of the transactions, so that if two transactions transfer the same coins (double spending), after the insertion of one of the transactions into the blockchain, the other transaction is considered invalid, and should not be added to the blockchain. The blockchain, thus, represents a consensus of the peers on what are valid transactions.

The transactions are organized into blocks, which are created and added to the blockchain by members of a peer-to-peer network. In Bitcoin, these peers are called *miners*. The first block in the chain is the *genesis block*. Every other block contains a hash of the previous block in the chain, e.g., using SHA-256. This means that a change in one of the blocks would either result in an incorrect chain or would require changing the hash values in all the following blocks.

A blockchain is maintained in a decentralized manner. It is immutable, where changes of past blocks are practically impossible. To achieve that and to prevent forks, where a separation of the chain cannot be resolved, blockchains like Bitcoin rely on *proof-of-work* (PoW)—a computation that is hard and time consuming, e.g., a cryptographic riddle. In Bitcoin, each block includes a *nonce* such that the hash of the block (with the nonce) has at least k leading zeros. Computing the nonce is hard, hence it is a PoW. The value k is determined such that the overall computation by all the peers (miners) would require approximately 10 minutes for computing a block. In a case of a conflict, or a fork, miners are expected to add blocks to the longest branch. This causes short branches to be abandoned and prevents forks. A block that contains invalid transactions, e.g., double spending, will be ignored by the majority of the peers, and eventually will not be part of the chain.

An attacker that tries to change a block in the blockchain needs to create an alternative branch and compete with all the other miners, in an attempt to make the alternative branch the longest one. The chances of succeeding are slim, due to the hardness of block creation. This provides immutability, stability and reliability. A comprehensive survey of blockchain technologies is provided in [1].

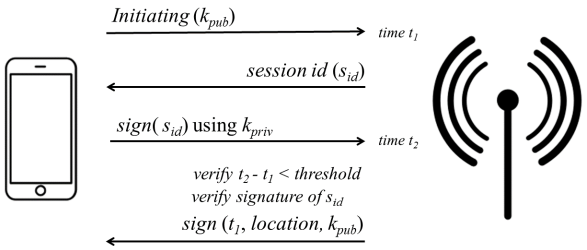


Figure 1: Issuing a location certificate for a requester (left) by a corroborator (right).

2.2 Location Certificate

Geospatial partition is natural in many blockchain applications. It is based on reliably mapping transactions to their location and time, and providing a certificate of that. The *location certificate* is a digital proof that a device was at a particular place at a specific time. GPS cannot be used for that because GPS can be spoofed [38].

One way to produce location certificates is based on the existence of a trusted *localized corroborators* that could provide the certificate [20]. A localized corroborator is a server that has a known location, and that can only be accessed from a short range. It can be a server that is directly, physically, connected to stationary devices. For mobile devices it can be a cellular tower, a wireless access point (Wi-Fi, Bluetooth, ZigBee), an optical access point (based on infrared light), etc. A device can only be connected to the localized corroborator if it is near the corroborator—a few meters in a case of Bluetooth, ZigBee or infrared sensor; dozens of meters for Wi-Fi; and a few kilometers for a cellular tower. Higher accuracy can be achieved by taking the signal strength into account [30, 44, 45]. The trustworthiness of certificates can be strengthen by adding cryptographically signed geotags to IP packets [11]. We assume that the corroborator has a unique pair of a private key and a public key, as part of a public-key cryptosystem.

For our purposes, we consider issuing a location certificate for a device that holds a specific private key—the private key remains concealed and only the public key is revealed to the corroborator or to a verifier. For a given pair (k_{priv}, k_{pub}) of private and public keys, the certificate attests that a device containing the private key k_{priv} was near the corroborator at the time of the issuing.

The protocol involves the following steps.

1. The requester sends an initiation message to the server, including the public key k_{pub}.

2. The corroborator sends a random session id s_{id} to the requester.

3. The requester sends back the session id s_{id} signed using the private key k_{priv}.

4. The corroborator checks the time that elapses between sending s_{id} and getting it back (signed) and verifies the authenticity of the signature using k_{pub}. When the time difference is a few milliseconds (less than a threshold of say 5 milliseconds), the corroborator issues a certificate consisting of the time, location and requester public key k_{pub}, signed by the private key of the corroborator.

The requester cannot create a certificate without the corroborator because a valid certificate requires the signature of the corroborator. The session id can only be signed after the beginning of the session, because it is unknown before the session starts. Therefore, after the session initiation, a device that can sign the session id with k_{priv} must be near the corroborator, to provide a response in a latency that is smaller than the threshold. The certificate can include a precise location or a general one, e.g., a city, a county, a state, to increase privacy.

A *certified transaction* is a pair (t, C) of a transaction $t = (x \to y, m)$ and a location certificate C, where the public key of y is used to create the certificate. As explained, the certificate is created by a device that at the certified time is near the corroborator and contains the private key of y.

3. BLOCKCHAIN PARTITIONING

We present now our partitioning approach. In public blockchains like Bitcoin and Ethereum, the transaction rates are low. One of the reasons for the low transaction rate is the serialization of all the transactions, even those that are not conflicting. Had there been a partition of the transactions into groups so that transactions from different groups could never conflict, non-conflicting transactions could have been processed in parallel, and blocks of non-conflicting transactions could have been generated in parallel. This can be achieved by creating a partition of the blockchain into a hierarchy of blockchains (sub-chains) and associating transactions with different nodes of the hierarchy. Each sub-chain is managed independently, so blocks of different sub-chains can be created and added to the appropriate chain in parallel.

The study of parallel creation of blocks led to the development of the BlockDAG data structure, where a new block can extend several previous blocks, not just one, and the "heaviest" tree is selected in a greedy fashion, e.g., using the GHOST protocol [34]. The SPECTRE protocol [33] utilizes BlockDAG for a virtual vote on the order of the blocks, to achieve high throughput and fast confirmation time. Two other notable attempts to cope with the low transaction rates in public blockchains are Bitcoin-NG [14] and Algorand [17]. Bitcoin-NG speeds up block creation by electing a leader for a specified epoch, and allowing the leader to create a large number of blocks till the next leader is elected. Algorand employs a sophisticated method of randomly selecting a small group of users (who are replaced when their identity is revealed) and executing a Byzantine Agreement protocol by the chosen users, to prevent forks altogether. Our approach is orthogonal to these systems. First, in a hierarchy of linked sub-chains, any blockchain implementation can be used, including Bitcoin, Bitcoin-NG, Algorand, and others. The hierarchical structure may even link different types of blockchain. Second, scalability is achieved by adding new sub-blockchains to the hierarchy without changing the technology or performing a hard fork.

Different hierarchies can be used. Geospatial hierarchy is a natural one, e.g., a partition into neighborhoods, cities, counties, states and countries. Such a partition is suitable, for example, when using blockchains to record real-estate transactions. Another partition example is a partition into business units of a large global company, e.g., teams, depart-

ments, divisions, sub-organizations, etc. Such a partition can be applied when a company ledger is used for recording processes, data sharing, code transfer, etc.

We elaborate on geospatial partition. Our underlying assumption is that most transactions are local, e.g., cash exchange is often between people who are geographically near, and this may also be true in a cryptocurrency that aims to replace cash. Other usages of geospatial partition are real estate transactions, supply chains, management of data in smart cities, and so on. The hierarchy provides a tradeoff between privacy and efficiency, where local transactions are more efficient and non-local ones are more private.

A localized blockchain is defined with respect to a given area A, e.g., the area of the USA. Localization is with respect to a *hierarchical partition* of A, and each wallet is associated with a sub-area in A.

EXAMPLE 1. *In a hierarchical partition of the USA, the country is partitioned into states, states are partitioned into counties, and counties are partition into cities and towns. A transaction within a city is registered merely in the city. A transfer of coins from a city in one county to a city in another, within the same state, is registered in the relevant cities, counties and the state. A transfer across states is recorded in all the levels of the hierarchy.*

The partitioning of the blockchain makes local transactions faster and cheaper than non-local ones, because a local transaction is notarized for a local area and "competes" with less transactions. When moving higher in the hierarchy, each transaction may need to compete with transactions from a wider area—this will increase privacy, but also expected to increase the transaction delay (i.e., lengthen the wait time till the transaction is recorded in the blockchain).

The hierarchy is the result of a recursive partitioning of A. Formally, let $\mathcal{A}$ be a set of subareas of A. The hierarchical partition $H = (T, \alpha)$ of A comprises a tree $T = (V, E, v_{root})$ and a function $\alpha : V \to \mathcal{A}$, where V, E, and v_{root} are the vertexes, edges and root of T, respectively. The function α maps each vertex v to a subarea in $\mathcal{A}$, such that for each node v that is not a leaf it must hold that: *(1)* $\alpha(v) = \cup_{u \in children(v)} \alpha(u)$, and *(2)* $\alpha(u_1) \cap \alpha(u_2) = \emptyset \quad \forall u_1 \neq u_2 \in children(v)$. That is, the areas associated with the children of a vertex v are a partition of the area associated with v.

A wallet is localized by associating it to a node of H. Let W be the set of all wallets, then $\lambda : W \to V$ is a function that maps wallets to nodes of H. A wallet $w \in W$ is associated with the area $\alpha(\lambda(w))$. A transaction $t = (x \to y, m)$ is *local* if $\lambda(x) = \lambda(y)$ is a leaf of H. Otherwise, the *LCA* of t is the least-common ancestor $lca(x, y)$ in T. The area of t is $\alpha(lca(x, y))$.

Certification. A *certification requirement* allows only processing of certified transactions (t, C). When including a certified transaction in a blockchain, it is required to verify that the certificate C is valid and includes the public key of the receiving wallet y. The location in C should be inside the area of the receiving wallet, i.e., in $\alpha(\lambda(y))$.

We consider three types of transfers. A *lateral transfer* between wallets in the same node. An *ascending transfer* from a wallet in a node v to a wallet in the parent of v. A *descending transfer* from a wallet in a node v to a wallet in a child of v. The blocks of each node of H are managed separately from the blocks of the other nodes, with a distinct chain for each node. To increase the efficiency, blocks

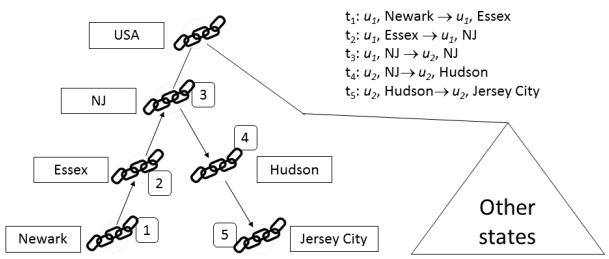


Figure 2: Hierarchical partitioning and a transfer.

associated with different nodes can be created in parallel.

To prevent double spending, each transaction $t = (x \to y, m)$ must be added to the blockchain of the node associated with x, to get accepted. The transaction $t' = (y \to z, n)$ that follows t is added to the blockchain of the node associated with y. A local transaction that is related to node v is added to $blockchain(v)$, as a lateral transfer. A non-local transaction from x to y is translated to a sequence of transfers along the shortest path from x to y in T.

For example, a transfer of coins within Chicago is local and requires a single lateral transfer. A transfer from a wallet of user u_1 in Newark, NJ to a wallet of u_2 in Jersey City, NJ requires the five transfers depicted in Fig. 2.

The geographic partition can be done in different ways depending on how people use money. Several transfers are needed for non-local transactions, but blocks of different chains are created in parallel. For anonymity, users can choose the level at which they execute transactions—a higher level provides a more obfuscated exposure of the user location. There is, however, a tradeoff between privacy and the time that elapses till a transaction is added to the blockchain.

Non-geographic partitions could be applied as well. In a large corporation, for instance, a partition based on the divisions and subdivisions of the company could be used to manage company transactions, as in the geospatial partition.

Geofencing. Partitioning of blockchains can be used to strengthen security. We describe geofencing as an example.

A private key of a wallet can be stolen, which may lead to the loss of the coins. By *geofencing wallets*, coins can be more secure. In geofencing, a wallet is associated with an area, as explained in Section 3. For executing a transaction, the payee needs to provide a location certificate for a place within the area of the payer's wallet, at the time of the transaction. If, for example, Alice associates her wallet with her neighborhood, a malicious attacker from a different country, say Mallory, would be limited in her ability to spend the money. Even if Mallory would steal the private key of Alice, to create a certificate and transfer the coins she would need to have a device in Alice's neighborhood with the private key of the receiving wallet. If Mallory would use as a proxy a device in Alice's neighborhood, to create a certificate on her behalf, she would need to surrender her private key to the proxy. Hence, the taken money could be spent by the proxy. This would make cryptocurrencies more secure. The stronger security would also make it safer to create backups for a lost key. Note that Alice could transfer money from her local wallet to a wallet associated with her state, if she wants to use the money when traveling within the state.

Geofencing can be done by requiring a certificate from the payee, the payer or from both, to restrict, at the time of the transaction, the location of the payer, the payee or of both. Note that geofencing strengthens the security provided by the private keys, it does not replace private keys. There is a tradeoff between security and privacy here—smaller area provides more security but less privacy, and vice versa.

Geofencing can be applied to various applications of blockchain, e.g., in a blockchain that supports a supply chain, transactions of item transfer could be limited to the warehouses, i.e., they could only be recorded at the warehouses, to provide strict control over transfers and their registration.

4. PROOF-OF-LOCATION

Blockchains that are based on Proof of Work (PoW) are wasteful, that is, consume an excessive amount of energy. A partition of the blockchain could increase the amount of energy that is required to sustain the system. In this section we show how location certificates can be used to establish Proof-of-Location as a non-wasteful alternative to PoW, to achieve consensus in a public blockchain.

4.1 Proof of Work

Over the years, PoW has been proven to be a successful and reliable consensus mechanism for a public (permissionless) blockchain like Bitcoin, and capable of preventing a Sybil Attack [13]. Its main limitation, however, is the immense energy consumption that is required to maintain the system. Miners who create a block are rewarded for that by receiving transaction fees or a block-creation incentive. They compete to create blocks, and thus, many miners spend significant computation power on finding a suitable nonce, for each block. Furthermore, if miners would collude, they could issue a 51% attack or in some cases, even a 25% attack [15]. This is a real threat because Bitcoin miners are already organized into large groups and share their computational resources to create blocks [16].

4.2 Alternatives to PoW

Several methods were proposed as an alternative to PoW. One of them is *proof-of-stake* (PoS) [6, 7, 22], where the voting power is given to "stake holders" of the system, i.e., to those who have coins. The creator of a block needs to provide a cryptographic proof of existence of a certain amount of coins in its possession, and these coins are locked till some conditions are met. This approach was criticized as non-resilient to forks, since, unlike in PoW, the expected gain from working on more than one branch is often higher than the cost of doing so. Furthermore, in this method peers with many coins could delay the creation of new blocks (when they are selected to create the next block) and could use that for extortion, or in an attempt to attack the system for an external gain [23].

In *proof-of-disk-space* the creators of blocks need to waste disk space to create a block [27, 29]. Like PoW, it is a wasteful approach. A consensus protocol to cope with the case where an unknown number of peers could be offline was suggested in [28].

Several solutions were designed for private (permissioned) blockchains, see an analysis in [12]. Practical Byzantine Fault Tolerance (PBFT) [9, 41] was proposed as a method to reach consensus by voting, but it requires knowing the number of peers, so it is unsuitable for a public blockchain in which joining the peer-to-peer network is open to the public. *Proof of authority*[3] was developed for private blockchains, with trusted entities as authorities. It relies on establishing trust in the peer-to-peer network, e.g., see [43].

4.3 Implementing PoL

We introduce now *proof-of-location* (PoL)—a novel alternative to PoW. It aims to avoid waste when creating a block, and yet keep the process decentralized and independent of knowledge about the reputation of peers, or their number. It is based on the ability to create a location certificate to provide a location proof [20, 31, 32], for a particular place, to create the next block.

Block creation. The blockchain is created such that a location ℓ is selected in each step, in an unpredictable way, and the next block is the one that was created by the peer with the PoL closest to the selected location. If two location certificates have the same distance from the selected location, the one with the smallest time stamp is selected.

The selection of a location ℓ can be done in different ways. One way is as follows. Consider the geographical area in which the block creators (peers) are active, e.g., USA. Let G be a grid that covers this area. Let $c_1, \ldots, c_m$ be the cells of G. Let B be the last block in the blockchain, so far, and $h(B)$ the hash of B. The selected location is the center of the cell number $h(B) \bmod m$, i.e., $c_{h(B) \bmod m}$ of G. This yields a cell whose coordinates cannot be computed without knowing B. Note that for a hash function h whose digest has a size of 256 bits, even if the remainder of the division $2^{256}/m$ is non-zero, the difference between $\left\lfloor \frac{2^{256}}{m} \right\rfloor$ and $\left\lfloor \frac{2^{256}}{m} \right\rfloor + 1$ is negligible, so if h is uniform then the selection of cells can, practically, be regarded as uniform.

To control the hardness of block creation, so that an attacker could not create an alternative branch fast, we suggest that the distance of the certificate from ℓ would be limited by an adaptable inflating bound. One option to do so, is as follows. Let t_{prev} be the creation time of the last block. The *inflating distance limit* is $d(t) = \delta \cdot minutes(t - t_{prev})^k$, for given k and δ. A location certificate with location and time (l_p, t_p) satisfies the distance limit if $distance(l_p, \ell) < d(t_p)$. For $k = 3$ and $\delta = 100$ meters, in the first minute (time difference < 1), the certificate should be for a location that is less than 100 meters from ℓ. In the second minute (time difference < 2), the certificate should be for a location that is less than 800 meters from ℓ. The distance limit (in meters) as a function of the time difference (in minutes) evolves as follows: $(2, 800), \ldots, (4, 6400), \ldots, (8, 512, 000), \ldots, (10, 100, 000), \ldots$. With these parameters, the distance limit is 100 kilometers after 10 minutes, and covers the area of the USA after about half an hour. (These parameters can be changed to control the block creation rate, and guarantee that blocks will be created within a reasonable time.)

An attacker that would try to change a block and then create the longest branch, by competing with the other miners, would need to produce location certificates faster than the other miners. However, without a machine and a corroborator near any arbitrary location ℓ, the attacker would need to wait, e.g., if its nearest machine to ℓ is 100 kilometers, it would need to wait 10 minutes, and at that time the other

[3] https://github.com/paritytech/parity/wiki/Proof-of-Authority-Chains

miners would add blocks to the main chain. Note that with machines that cover an area of 10 km^2, about 1,000,000 machines would be needed to cover the area of the USA.

An advantage of the proposed method is that, unlike in Bitcoin, if the locations of the peers (miners) are arbitrary, a group of miners that collude do not have an advantage over a group that do not collude. This would make the system less vulnerable to colluding peers. Furthermore, for an attacker it will be hard to create blocks fast, even with a large computation power, because the computation power would not help arriving at ℓ or getting close to ℓ faster.

Fork Prevention. When two or more branches are constructed in parallel without being abandoned, forks occur. Forks cause the blockchain to be less reliable, and reduce consistency. To cope with that, the rule of thumb is that the miners would continue the longest branch so far. But there is also a need to discourage the miners from extending other branches. In PoW, the computation of a nonce is demanding, so miners have an incentive to invest their computation power only on the branch with the highest chance of success (the longest one). This can be achieved in PoL if there would be a cost to each certificate, e.g., where miners would pay to the corroborators for each creation of a location certificate. (Note that in PoW miners pay for block creation in their electricity bills.) A payment would encourage miners to only "invest" in a branch with a high chance of success. The payment can be adaptive, e.g., including ℓ in the certificate and making the fee proportional to the distance between the corroborator and ℓ, to discourage miners that are geographically far from ℓ from creating a block.

Effect on Miners. In PoL, the miners create location certificates and reveal their location. This, however, does not affect users, i.e., there is no disclosure of the locations of the payers or the payees whose transactions are added to a block. It is an open question, however, whether revealing the location of miners is much different from revealing their IP addresses, as being done anyway in the peer-to-peer network. (Miners can hide their IP address, e.g., by using onion routing [18], but this would slow them down in the "race" to create a block. Such a tradeoff between privacy and effectiveness can be made also in PoL, where a miner may decide only to create location certificates by a mobile device when she/he is far from her/his home or office.)

Decentralized System. In PoL, the system remains decentralized, because location certificates are not produced by a single entity. The certificate may be produced by different companies and organizations using network access points, e.g., modifying all the cell towers to serve as corroborators. A company that would not provide reliable certificates, the blocks with its certificates would not be accepted by the majority of the miners, and hence, users will stop acquiring certificates from it. Hence, the incentive of certificate providers to be honest is similar to that of miners in a public blockchain like Bitcoin.

Sybil Attack. To create a certificate there is a need to be near the corroborator. Therefore, forging many identities that are located in a single place does not increase the ability to create a block if PoL is used. Also, having more machines or stronger machines in proximity to a single corroborator does not give an advantage. A miner could try to deploy many machines in many remote places. This, however, would require investment in equipment and would incur maintenance costs, and unlike Bitcoin mining farms could not be in a single location.

An attacker may try to apply *cryptojacking*, i.e., use machines of other users to create location certificates, somewhat like unauthorized use of machines for Bitcoin mining. But in such a case, to create the certificate, the attacker would need to expose the private key of the wallet that would receive the incentive fee (this key is necessary to create the certificate). Any hijacked machine would then have the private key that would allow it to spend the new coins.

To increase security, there should be many corroborators distributed over a large area. More importantly, each corroborator should have a different private key—if the security of a corroborator will be breached, using its key for creating fake certificates would be limited to a single location.

5. CONCLUSION AND DISCUSSION

Blockchain has the potential to revolutionize data sharing among organizations and individuals, by providing a decentralized, transparent and tamper-proof storage of transactions. It is the underlying technology of many cryptocurrencies, and is adapted for other uses. However, currently blockchains are not scalable (they have a low transaction rate), and public blockchains are wasteful (require a high usage of electricity to support PoW), and insecure (provide no protection from theft of a private key). In this paper, we present a novel approach of partitioning the blockchain into a tree of sub-chains based on a real-world hierarchy, like a geographical or an organizational partition, where transactions of different sub-chains do no conflict with one another. Such a partition provides a tradeoff between efficiency and privacy—high levels provide more privacy than low levels but a longer expected wait till the transaction is added to a block, and vice versa. Scalability can be achieved by partitioning leaf nodes in which the transaction rate is too high. Creating an optimal hierarchy and adapting the hierarchy to changes are challenging research directions.

An important advantage of the hierarchical partitioning is that there is no need to develop a new technology or perform hard forks to cope with scalability issues. The recent debate about how to increase the block size of Bitcoin illustrates how difficult it is to make changes in public blockchains.

We explain how a geographic partitioning combined with location certificates can be used to increase security by applying geofencing. With the growing popularity of cryptocurrencies and their usage in applications that do not require privacy, strengthening security by restricting usage of coins to specified locations could proliferate utilization of cryptocurrencies. How to further increase security of cryptocurrencies at the expense of privacy, but without completely revealing user identities, is an open question.

The partition of the blockchain may inflate the excessive energy consumption cause by PoW. Thus, we suggest a novel non-wasteful proof-of-location (PoL) method, to achieve consensus for block creation. In PoL, unlike PoW or PoS, having a strong computation power or many coins does not increase the chances of creating the next block. This has the potential of providing higher stability than that of PoW or PoS, however, further research is required to prove that.

Note that our vision of using partitions to create subchains can be generalized from hierarchies to a network of blockchains, e.g., by connecting existing blockchains. We defer a detailed discussion in the interest of space.

6. REFERENCES

[1] D. T. T. Anh, M. Zhang, B. C. Ooi, and G. Chen. Untangling blockchain: A data processing view of blockchain systems. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[2] D. Augot, H. Chabanne, T. Chenevier, W. George, and L. Lambert. A user-centric system for verified identities on the bitcoin blockchain. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 390–407. Springer, 2017.

[3] D. Augot, H. Chabanne, O. Clémot, and W. George. Transforming face-to-face identity proofing into anonymous digital identity using the bitcoin blockchain. *arXiv preprint arXiv:1710.02951*, 2017.

[4] A. Back. Hashcash–a denial of service counter-measure, 2002.

[5] N. Baracaldo, L. A. D. Bathen, R. O. Ozugha, R. Engel, S. Tata, and H. Ludwig. Securing data provenance in internet of things (IoT) systems. In *International Conf. on Service-Oriented Computing*, pages 92–98, 2016.

[6] I. Bentov, A. Gabizon, and A. Mizrahi. Cryptocurrencies without proof of work. In *International Conference on Financial Cryptography and Data Security*, pages 142–157. Springer, 2016.

[7] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake. *ACM SIGMETRICS Performance Evaluation Review*, 42(3):34–37, 2014.

[8] K. Biswas and V. Muthukkumarasamy. Securing smart cities using blockchain technology. In *High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on*, pages 1392–1393. IEEE, 2016.

[9] M. Castro and B. Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002.

[10] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman. Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2:6–10, 2016.

[11] T. Dasu, Y. Kanza, and D. Srivastava. Geotagging IP packets for location-aware software-defined networking in the presence of virtual network functions. In *Proc. of the 25th ACM SIGSPATIAL International Conf. on Advances in Geographic Information Systems*. ACM, 2017.

[12] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan. BLOCKBENCH: a framework for analyzing private blockchains. In *Proc. of the ACM International Conf. on Management of Data*, pages 1085–1100, 2017.

[13] J. R. Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[14] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse. Bitcoin-NG: A scalable blockchain protocol. In *NSDI*, pages 45–59, 2016.

[15] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International Conf. on Financial Cryptography and Data Security*, pages 436–454, 2014.

[16] A. Gervais, G. Karame, S. Capkun, and V. Capkun. Is bitcoin a decentralized currency? *IEEE security & privacy*, 12(3):54–60, 2014.

[17] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.

[18] D. Goldschlag, M. Reed, and P. Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.

[19] M. Jakobsson and A. Juels. Proofs of work and bread pudding protocols. In *Secure Information Networks*, pages 258–272. Springer, 1999.

[20] Y. Kanza. Location corroborations by mobile devices without traces. In *Proc. of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2016.

[21] Y. Kanza and H. Samet. An online marketplace for geosocial data. In *Proc. of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2015.

[22] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

[23] J. A. Kroll, I. C. Davey, and E. W. Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *Proceedings of WEIS*, 2013.

[24] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu. Blockchain based data integrity service framework for IoT data. In *Web Services (ICWS), 2017 IEEE International Conf. on*, pages 468–475, 2017.

[25] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[26] K. J. O'Dwyer and D. Malone. Bitcoin mining and its energy footprint. In *25th IET Irish Signals & Systems Conference*, Limerick, Ireland, 2014. IET.

[27] S. Park, K. Pietrzak, J. Alwen, G. Fuchsbauer, and P. Gazi. Spacecoin: A cryptocurrency based on proofs of space. Technical report, IACR Cryptology ePrint Archive, 2015.

[28] R. Pass and E. Shi. The sleepy model of consensus. In *International Conf. on the Theory and Application of Cryptology and Information Security*, pages 380–409, 2017.

[29] L. Ren and S. Devadas. Proof of space from stacked expanders. In *Theory of Cryptography Conference*, pages 262–285. Springer, 2016.

[30] S. Saha, K. Chaudhuri, D. Sanghi, and P. Bhagwat. Location determination of a mobile device using IEEE 802.11 b access point signals. In *Wireless Communications and Networking*, volume 3, pages 1987–1992. IEEE, 2003.

[31] S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *Proc. of the 10th Workshop on Mobile Computing Systems and Applications*. ACM, 2009.

[32] N. Sastry, U. Shankar, and D. Wagner. Secure

verification of location claims. In *Proceedings of the 2nd ACM workshop on Wireless security*, pages 1–10. ACM, 2003.

[33] Y. Sompolinsky, Y. Lewenberg, and A. Zohar. Spectre: A fast and scalable cryptocurrency protocol. *IACR Cryptology ePrint Archive*, 2016:1159, 2016.

[34] Y. Sompolinsky and A. Zohar. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.

[35] J. Sun, J. Yan, and K. Z. Zhang. Blockchain-based sharing services: What blockchain technology can contribute to smart cities. *Financial Innovation*, 2(1):26, 2016.

[36] M. Swan. *Blockchain: Blueprint for a new economy.* " O'Reilly Media, Inc.", Sebastopol, CA, USA, 2015.

[37] D. Tapscott and A. Tapscott. *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World*. Penguin Random House, New York, NY, USA, 2016.

[38] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun. On the requirements for successful gps spoofing attacks. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 75–86. ACM, 2011.

[39] S. Underwood. Blockchain beyond bitcoin. *Commun. ACM*, 59(11):15–17, Oct. 2016.

[40] P. Vigna and M. J. Casey. *The age of cryptocurrency: How bitcoin and digital money are challenging the global economic order*. St. Martin's Press, New York, NY, 2015.

[41] M. Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.

[42] N. Vyas. Disruptive technologies enabling supply chain evolution. *Supply Chain Management Review*, 2016.

[43] L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE transactions on Knowledge and Data Engineering*, 16(7):843–857, 2004.

[44] K. Yedavalli and B. Krishnamachari. Sequence-based localization in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 7(1), 2008.

[45] K. K. Yedavalli. *Location Determination using IEEE 802.11 b*. PhD thesis, University of Colorado, 2002.

Blockchain Protocols: The Adversary is in the Details

Rachid Guerraoui
EPFL
rachid.guerraoui@epfl.ch

Matej Pavlovic
EPFL
matej.pavlovic@epfl.ch

Dragos-Adrian Seredinschi*
EPFL
dragos-adrian.seredinschi@epfl.ch

ABSTRACT

Blockchain-like protocols are flourishing. Maybe not surprisingly, the differences among these protocols are often subtle and difficult to understand. More importantly, it is often unclear what the weaknesses of each of these protocols are and how easily they can be attacked. The goal of this paper is to shed light on the important differences between blockchain protocols and the impact these differences can have in terms of their vulnerabilities. We cover well-studied protocols ranging from those inspired from the distributed systems literature (e.g., PBFT), to recent research prototypes (e.g., ByzCoin or Algorand), including the popular Bitcoin protocol.

Towards reaching our goal, we first precisely define the problem that these protocols seek to solve. Then we propose a unifying scheme that captures, at a high level, the behavior of any blockchain protocol. Interestingly, this scheme is also sufficiently low level to highlight the important differences between these protocols. We show that blockchain-like protocols can be differentiated according to their degree of indulgence—i.e., tolerance towards node misbehavior or towards network asynchrony—which translates into the different vulnerabilities of each of these protocols.

1. INTRODUCTION

Since the advent of Bitcoin, tens—if not hundreds—of variations on this protocol have been proposed. These variations, which we call simply Bitcoin-like protocols, usually have a twofold purpose: (1) to improve reliability with respect to the original Bitcoin protocol by withstanding severe attacks, or (2) to improve efficiency by either decreasing latency or increasing throughput. Usually, these objectives are antagonist. For example, decreasing the time it takes to commit a transaction in a blockchain protocol can make that protocol more vulnerable to double spending [22].

The differences between various blockchain protocols are often subtle, and each improvement to a certain protocol may open vulnerability breaches which are not clear a priori. For instance, it is perhaps alarming to see the recent uncovering of critical errors in algorithms that can be used to implement blockchains, namely in Tangaroa [10], Zyzzyva and FaB [3]. This makes the current blockchain ecosystem very chaotic, which, to say the least, is rather disappointing, given that these protocols are mainly aimed at implementing distributed trust.

The motivation of this paper is to help clarify this state of affairs. Our aim is not a rigorous formalization of a specific blockchain protocol and its properties, as done in a significant body of related work [8, 18, 29]. Instead, we propose a high-level, *adversary-oriented* approach to deconstructing blockchain protocols. Ultimately, our goal is to offer a better understanding of blockchain variations (e.g., efficiency or reliability enhancements), discussing which variation opens which vulnerability breach. Our principled approach is inspired from the theory of distributed computing. As we will recall, blockchain protocols are solving a classical distributed computing problem. We proceed in several steps.

First, we define precisely the blockchain problem, namely the problem that seeks to be solved by the original Bitcoin protocol and its many variants. Roughly speaking, the problem consists of building a highly available (replicated) set of single owner bank accounts while avoiding double spending. Indeed, blockchain is both the name of (a) the chain of transaction blocks that need to be replicated and maintained consistently to enable Bitcoin transactions, as well as (b) the protocol that maintains this consistency. In a sense, blockchain is the name of the *solution* (b), as well as the name of the *problem* (a) being solved. Whilst solutions have been discussed at great length, we believe the problem specification has been largely ignored.

We define the problem precisely in terms of safety and liveness properties [5]. We then use classical results in distributed computing to highlight how the blockchain problem is harder, for example, than building a replicated file system [7], but is equivalent to the celebrated consensus [31] and State Machine Replication (SMR) problems [32]. It is known that there is no deterministic solution to consensus if we assume that the network can be asynchronous and at least one node can crash (even if no node can act adversarially) [17]. No matter what solution is designed, adverse network conditions can defeat it.

We then present a general scheme that unifies solutions to the blockchain problem. We show how all solutions to this problem restrict the power of the adversary in one way or another, e.g., either by assuming a bound on the number of misbehaving nodes which an adversary controls, or by assuming a bound on network asynchrony. We study here well-known protocols like PBFT [11] or Bitcoin [27], as well as recent research efforts

*This work has been supported in part by the European ERC Grant 339539 - AOC

Symposium on Foundations and Applications of Blockchain (FAB '18) March 9, 2018, Los Angeles, California, USA.

such as ByzCoin [24], Bitcoin-NG [15], and Algorand [19].

In short, our general scheme builds upon two fundamental components: a *leader election* subprotocol and a *commitment* subprotocol. The goal of the first subprotocol is to elect a node (or a set of nodes) to lead the task of ordering transactions. The goal of the second subprotocol is to make sure the ordering is global and the decision is unique, in case a new (or concurrent) leader is elected and considers a different ordering. This intuitive decomposition helps describe the avenues for attack which an adversary can take to subvert a blockchain protocol. It also enables us to point out critical differences between blockchain protocols as well as draw parallels between protocols.

We point out the existence of two classes of protocols. A protocol which represents the first class is Castro and Liskov's PBFT [11]. This class of protocols preserves its safety property, namely, consistency, despite the harshest conditions of the network (i.e., asynchrony). We say that this class is indulgent towards asynchrony and call it *asynchrony-indulgent* (or A-indulgent). A representative of the second class is Bitcoin [27]. This protocol continues executing (i.e., preserves liveness) despite an adversary mounting a Sybil attack, polluting the system with many misbehaving nodes. We say that this protocol is *behavior-indulgent* (or B-indulgent).

We organize the rest of this paper as follows. We discuss the problem addressed by blockchain protocols through the lens of distributed computing, and introduce the A-indulgent and B-indulgent classes of blockchain protocols (§2). We then introduce a general scheme which captures the essential behavior of any blockchain protocol, and use this scheme to discuss two notable blockchain protocols—PBFT and Bitcoin—showing how each is a typical example of respectively the A-indulgent and B-indulgent class (§3). We also relate a few other protocols to our general scheme and discuss their indulgence (§4), and then we conclude this paper (§5).

2. THE PROBLEM

2.1 State Machine Replication and Consensus

On-line services often employ replication to ensure their availability despite failures in the underlying systems. A common method to achieve this is via *state machine replication (SMR)* [32]. In SMR, a service, such as a financial ledger or an online shopping cart, is modeled as a deterministic state machine. The service consists of (1) a service state, and (2) operations that can be applied on this state. Typically, each replica (or node) of the system maintains its own local copy of the state, and updates this state as a result of applying client operations.

In SMR, the operations have to be deterministic, i.e. the operation result and the new state it produces are a function of only the previous state and the operation itself. Any service state can thus be uniquely defined by the initial state and a sequence of operations applied on this initial state.

In order to keep the service state consistent, replicas need to apply the *same operations in the same order*. In other words, SMR requires that the sequence of operations applied at all replicas is the same; the main challenge in implementing SMR is ensuring this requirement. The challenge can be reduced to the fundamental problem of *consensus* (agreement) in a distributed system, where all replicas need to agree on what the n-th operation of the sequence will be, for an ever-increasing n. In Figure 1 we sketch the typical architecture of an SMR system as we have presented it so far. The system comprises 6 replicas, labeled from 0 to 5. At the heart of the system

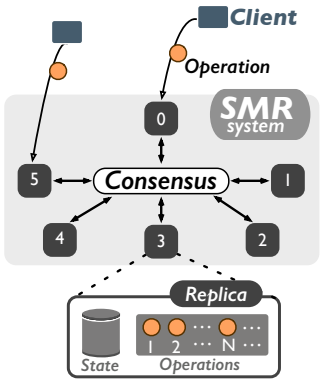


Figure 1: State Machine Replication in action.

lies a distributed consensus algorithm which the replicas use to agree on the sequence of incoming client operations and maintain the consistency of the replicated state.

A consensus algorithm must satisfy three essential properties:

- **Validity:** The agreed-upon operation must be the input of one of the replicas (e.g., a client operation).
- **Agreement:** The agreed-upon operation is the same for all correct replicas.
- **Termination:** The replicas will eventually agree on some operation.

Validity and agreement are *safety properties*: they define events that must never happen in a correct execution. Termination, on the other hand, is a *liveness property*, defining that a correct execution must make some progress [5].

2.2 Replicated Ledgers

We focus on a specific type of service: a *ledger*. Without loss of generality, we assume that a ledger describes the movement of money across different bank accounts. Concretely, a ledger is an ever-growing sequence of transactions, each of which transfers money between the users (i.e., clients) of the system.

Replicating a ledger is non-trivial, and even the relation between a ledger and consensus is not immediate. To understand this relation, we start from the simple observation that a ledger is no different than a *fetch-and-add* object [20]. As shown by Herlihy [23], such an object has consensus number 2. This means that, in a shared memory model, up to two processes (but not more) can solve consensus among themselves if they have access to fetch-and-add, i.e., a ledger object. Any object weaker than a ledger, such as simple *read-write register*, is insufficient to solve consensus in shared memory. In more practical terms, the problem of implementing a replicated ledger is strictly more difficult than that of implementing a file system or a key-value store, both of which have a *read-write* interface [7].

Note, however, that we are not interested in the shared memory model, but in the message passing model (detailed in §2.4). Interestingly, in a message passing system, Delporte-Gallet et al. [13] showed that replicating any object that has a consensus number greater than one—such as *fetch-and-add*—is equivalent to solving consensus. In other words, replicating a ledger is equivalent to solving consensus. We also know that solving consensus allows us to replicate *any* object via the SMR approach. Thus, we conclude that in the message passing model there is no object that is harder to replicate than a ledger.

In the following, we explain how the abstract notions of consensus and SMR relate to distributed ledgers.

2.3 Ledgers as Replicated State Machines

Modeling a ledger as a replicated state machine is straightforward. The SMR state is an ordered sequence of all transactions performed in the past, while each new SMR operation represents the appending of a new transaction to the ledger.

The system replicas must agree (i.e., solve consensus) on which transaction should take the n-th position in the ledger. Validity is easy to satisfy in this context using standard cryptography, so we do not focus on it henceforth. The challenge is to achieve both agreement (safety) and termination (liveness).[1] We require a replicated ledger to have the following properties:

1. **Safety**: If a replica accepts a transaction T ordered after some transaction T', then no replica accepts T without having ordered T' before T.
2. **Liveness**: If a client issues a transaction, the transaction is eventually accepted at all correct replicas.

We note that *accepting* a transaction has a nuanced meaning. In certain protocols, like PBFT, there is a specific point where a transaction becomes irrevocable (this is often referred to as *consensus finality* [33]). We say that such a transaction is accepted. Protocols which lack consensus finality, like Bitcoin, always permit the revocation of transactions—albeit with diminishing probability. When a revocation occurs, it represents a safety violation; in this sense, Bitcoin is prone to violating safety (§3.2.2).

Informally, safety prescribes that different replicas never have a different view of what the n-th transaction is. To understand the importance of safety, imagine, for example, two replicas R_0 and R_1 participating in a protocol that replicates a ledger. Both R_0 and R_1 have the same view of the first $n-1$ transactions; only one of these transactions states that a client called Eve receives 100$. However, Eve manages to make replica R_0 believe that the n-th transaction is "Eve transfers 100$ to Alice", while convincing R_1 that the n-th transaction is "Eve transfers 100$ to Bob". Such a situation clearly violates agreement. The balance in Eve's account was 100$, so only *one* of these transactions should be accepted by the system. If Alice and Bob consult R_0 and R_1 respectively to obtain the state of the ledger, they may both believe to have received money from Eve and provide her with some goods or services in exchange.

Such situations, where Eve effectively spends the same money twice, are known as *double spending*. Indeed, in the context of ledgers, safety violations can always be related to double spending. Different protocols have different approaches to prevent this problem. We elaborate later how double spending can occur in notable blockchain protocols (§3.2 and §4).

We recall that replicating a ledger is equivalent to replicating any state machine, as we argued earlier (§2.2). Traditionally, protocols like PBFT are employed for general-purpose SMR. Recently, however, protocols in the vein of Bitcoin are used as well towards implementing SMR (e.g., Ethereum, which generalizes transactions to so-called smart contracts [2]). For simplicity, we restrict ourselves to distributed ledgers which contain only monetary transactions; these are sufficient to explain all principles discussed in this paper, and a generalization to arbitrary state machines is straightforward.

[1]Note that, in general, achieving only one of liveness and safety is trivial. A protocol doing nothing never violates safety, but violates liveness. On the other hand, it is easy to make progress (satisfying liveness) if the output need not be correct (violating safety).

2.4 The Adversary

We consider a message passing model where nodes communicate by exchanging messages. The adversary can control various parts of the system, and there are two kinds of assumptions on the capabilities of this adversary.

Behavior assumptions define how much control the adversary can exert over the behavior of nodes (i.e., over the correctness of their computation). These are typically known as fault-threshold assumptions in distributed computing [26], and a common example is that at least two thirds of replicas are correct and follow the protocol faithfully [11], i.e., these replicas are not corrupted by the adversary.

Synchrony assumptions define how much control the adversary has over the speed of (otherwise corect) computation at nodes, as well as over the message transmission delays and delivery guarantees of the network. For example, the reliable message delivery or the absence of network partitions both fall under synchrony assumptions.

The precise definition of what the adversary can and cannot do plays a great role. A truly Byzantine adversary has no restrictions [25], but such a model is very restrictive in terms of the solutions it allows. Perhaps the most common assumption on a Byzantine adversary is that it cannot subvert cryptographic primitives. For instance, standard cryptographic assumptions prevent the adversary from inverting a secure hash function or producing a valid cryptographic signature without knowledge of the corresponding private key [11].

Another common assumption on the Byzantine adversary limits its interference with the nodes which it does not directly control. For example, it is often assumed that the adversary cannot prevent correct nodes from making progress (e.g., communicating with each other) indefinitely through a denial of service attack, a permanent network partition, or unremitting dropped messages. Typically, the assumption is that messages sent by correct nodes eventually reach their destination.

As we will see in the following sections, it is very often exactly these fine details in what the adversary can and cannot do—and for how long—that make the difference between the guarantees various protocols offer. We will discuss different protocols also from this point of view, i.e., we show how these assumptions influence the protocols' safety and liveness guarantees.

We ask ourselves the following question: *What is necessary to compromise the liveness and / or safety of a blockchain protocol?* Obviously, correctness of a protocol (i.e., upholding both safety and liveness) always depends on every assumption a protocol makes, otherwise the assumption would not be needed in the first place. However, not necessarily both of safety and liveness break when certain assumptions are violated. Focusing primarily on safety, we discuss two classes of protocols:

- **A-indulgent** protocols: protocols indulgent to *asynchrony*. These are protocols which focus on maintaining safety while putting minimal restrictions on the adversary in terms of synchrony.
- **B-indulgent** protocols: protocols which are indulgent towards bad or malicious node *behavior*. These protocols focus on maintaining safety while tolerating a relatively large number of malicious nodes.

The famous FLP impossibility result [17] shows that solving consensus without restricting the adversary is impossible. Protocols thus rely on various assumptions to circumvent this impossibility and guarantee both safety and liveness. PBFT, for example, can guarantee safety without any synchrony

assumptions on the adversary whatsoever. Synchrony assumptions are, however, required for liveness. Bitcoin, on the other hand, remains correct even when allowing the adversary more control over node behavior (an overwhelming fraction of nodes can behave maliciously as long as they do not possess enough computing power). Bitcoin, however, requires additional synchrony assumptions to remain safe. In this sense, PBFT is A-indulgent and Bitcoin is B-indulgent.

In the rest of this paper, we examine in more detail why PBFT and Bitcoin are each a representative of one of these classes. We then discuss other protocols, which, interestingly, can lie in between the two classes (specifically, Algorand), or are a combination of both classes (ByzCoin).

3. GENERAL SCHEME

In this section we introduce a general scheme that captures, at a high level, the behavior of any protocol implementing a blockchain. We also discuss how two notable blockchain protocols can be expressed using our scheme, and show how these protocols lie in sharp contrast to each other, representing the A-indulgent and the B-indulgent class, respectively.

Implementing a blockchain that prohibits double-spending is a challenging multi-level problem. First, most protocols rely on a leader election mechanism. Leader election must ensure that a unique leader presides over the protocol steps; this is important for maintaining consistency, since two leaders can engender disagreements. Generally, the existence of a leader simplifies implementations and reasoning about distributed protocols [28]. Blockchains operate in a Byzantine environment, however, where some replicas—including the leader—may fail arbitrarily. Even if a unique leader is correctly chosen, it can act maliciously, e.g., by equivocating. A second problem, then, is ensuring that transactions do not conflict and all correct replicas maintain the same view on the blockchain data structure. Yet a third problem can appear in protocols which are optimistic and permit temporary conflicts to exist across replicas; in such cases, additional measures are necessary to resolve conflicts. We capture all these difficulties in a general scheme that most algorithms follow in one way or another.

3.1 General Scheme

In broad strokes, we argue that the behavior of any protocol implementing a blockchain comprises four basic steps. These steps are as follows:

① a client *issues* a transaction;

② a *leader election* protocol determines a leader to marshal the transaction;

③ the replicas *commit* on an ordering proposed by the leader, i.e., they externalize the output, for instance, by replying to the client or by executing the transaction on their local state;

④ if the protocol allows conflicts to arise (which are often called *forks*), then a *recovery scheme* triggers to reconcile such conflicts.

The same replica may play different or multiple roles—be it client, leader, or ordinary replica—in this four-piece scheme. Indeed, in most protocols, each replica may become a leader at some point. The most notable differences between blockchain protocols arise at the level of steps ② and ③, namely in the protocol's method of dealing with the problems of leader election and commitment on a proposal. These steps are particularly difficult because it is at either of these two points where disagreements among replicas may arise, which can lead to safety violations (i.e., double-spending). In some protocols (such as PBFT), leader election takes place a priori, and the same leader tends to be reused across multiple transactions, as long as that leader behaves correctly and is able to communicate with a certain fraction of the system replicas [11]. In other protocols (such as Bitcoin), leader election happens on the critical path of handling transactions. Often, step ④ is absent from blockchain protocols. Unless we explicitly state what this step entails, we consider it to be absent because the protocol avoids disagreement by design, and hence no recovery is necessary.

3.2 Two Extremes of Blockchain Algorithms

We use our general scheme to present a breakdown, at a very high level, of PBFT [11] and Bitcoin [27] protocols. Interestingly, these protocols represent two extremes with respect to leader election and commitment, which translates into each of them belonging to one of the two indulgence classes, as we show next.

3.2.1 PBFT – Practical Byzantine Fault Tolerance

In PBFT, clients send their transactions to some replica i which they believe to be the current leader; if replica i is not the leader, then i simply forwards the request to the actual leader. This is step number ① of our scheme.

In PBFT, the leader role switches from one replica to another in a round-robin manner. Leader election—i.e., step ②—takes place only if there is a suspicion that the current leader has failed, prompting the system to switch to the next leader by executing a *view-change* sub-protocol.[2] If the leader acts correctly and the network is synchronous so as to permit progress, then no leader election occurs.

Step ③ in PBFT takes the form of a three-phase protocol. This protocol is essentially a quorum-gathering technique with Byzantine fault-tolerance, and ensures that if a correct replica commits on the leader's proposal, then no correct replica commits on a different proposal. Any correct replica in the PBFT protocol commits on some proposed ordering for a transaction after that replica is certain that a majority of replicas also commit on the same ordering.

A major drawback of PBFT-like protocols is that all replicas must have complete and consistent knowledge of all other replicas (i.e., the *membership set*) in the system at a given time. This information can be statically setup at system deployment and never changed, which is impractical for real-world replicated ledgers, as participants are expected to change over time. Dynamic membership can be achieved through a reconfiguration module [4, 9]. For instance, the very same mechanism that is used to agree on the contents of the ledger can also be used to agree on the membership. In this case, membership is redefined using a special transaction which changes the membership set. When nodes commit on such a special transaction, they also agree to update their view of the current membership set.

PBFT as an A-indulgent algorithm.

Informally, in PBFT any decision happens after the leader

[2]In PBFT, the leader is called the *primary* replica, and each replica is a primary in a given *view*. The view-change sub-protocol achieves the switching from a view to the next (hence, it also switches the primary to a different one), which we do not explain here. We refer the interested reader to the original PBFT description for more details [11].

asks permission from a quorum; in this case, a quorum comprises more than 2/3 of the replicas [11, 26]. Hence, a supermajority of the system replicas must coordinate via a three-phase protocol to agree on committing any decision. Irrespective of how badly the network behaves—such as being asynchronous or affected by a severe partition—PBFT always remains *safe* as long as the 1/3 threshold of faulty replicas is maintained. For this reason, we call PBFT an *A-indulgent* algorithm. In the classic sense defined by Guerraoui [21], PBFT is indulgent towards asynchrony in the network, permitting arbitrary periods of such asynchrony, because each correct replica refrains from taking any decisive step before consulting with a majority of replicas to agree on that step.

The only attack vector on PBFT's safety is controlling a fraction of at least a third of all replicas. In a dynamic setting, as we described earlier, gaining control over a third of replicas can be easy for an adversary. In a Sybil attack [14], the adversary simply spawns many replicas and makes them all join the system. As creating replicas is comparatively cheap in terms of computational and communication resources, enacting such an attack is realistic in practice. When controlling more than a third of replicas, such an adversary can convince two correct nodes to accept different transactions, *t1* and *t2*, at the same position in their ledger. These transactions can be crafted so as to permit double-spending, i.e., both *t1* and *t2* can be spending the same money, as in our example with Eve from §2.3.

To prevent the adversary from controlling a big fraction of replicas, PBFT requires an additional protection mechanism. This mechanism can take various forms, such as an access control scheme based on a certificate authority, a mechanism able to identify Sybil identities [6], or requiring participants to dispose of important amounts of a scarse resource (similar in spirit to Bitcoin's proof-of-work, which we will discuss next). To wrap-up, PBFT is mainly susceptible to an adversary that can manipulate the behavior of a large number of replicas in the system. As noted, however, PBFT is A-indulgent in the sense that its safety is resilient towards asynchrony.

The next algorithm we visit is Bitcoin. This protocol lies in sharp contrast with PBFT in its indulgence, i.e., in the way it deals with asynchrony or with adversarial behavior.

3.2.2 Bitcoin

Nodes in Bitcoin-like protocols are called miners. At step ①, clients issue their transactions to multiple miners, typically through a gossip-based broadcast scheme. Each miner independently assembles a block of transactions and then starts executing a proof-of-work (PoW) algorithm.

The PoW algorithm serves, primarily, as a leader election scheme, that is step ②. In contrast to PBFT (where leaders succeed each other whenever the views change), in Bitcoin all miners are striving to become a leader for every new block of transactions. Briefly, any miner can become the leader if it successfully solves a cryptographic puzzle—essentially, inverting a hash function [27]—faster than other miners.

Upon finding a puzzle solution, the miner becomes the de facto leader, and it broadcasts its solution, proposing an ordering and commencing step ③. The solution is uniquely bound to the block which the miner assembled earlier, and chained to the whole history of older blocks via a hashing algorithm, hence giving the name of blockchain to the resulting data structure.

Step ③ is more nuanced in Bitcoin than in other systems. Briefly, when another node observes a block, committing on it simply means appending the corresponding block at the end of its local blockchain. Note, however, that leader election does not guarantee a unique leader, because multiple miners can solve the puzzle for the same position in the blockchain. Since all puzzle solutions are equally valid, this gives rise to a fork. To select a specific branch of the fork and recover from the conflict, nodes in Bitcoin simply wait until one of the branches is extended with subsequent solutions. This is known as "the longest chain wins" rule, and represents step ④ in our general scheme. Typically, before committing some block b, nodes wait until additional blocks—called *confirmations*—are found, thereby extending b and raising the confidence that b will not be abandoned. The number of confirmations can vary; a node might even choose to wait for no confirmations and instantly accept a block without any waiting time [1].

Bitcoin as a B-indulgent algorithm.

At a high level, PBFT and Bitcoin differ in one critical aspect. Nodes in PBFT choose leaders and commit on values *after* consulting with a majority of the system. Nodes in Bitcoin commit on a value after some time passed and that value accumulated a certain number of confirmations; the Bitcoin protocol, briefly, relies on timing assumptions in the commitment step. For this reason, Bitcoin does not qualify as an A-indulgent algorithm: if timing assumptions do not hold (e.g., an adversary partitions the system), then Bitcoin is vulnerable to attacks on its safety. Instead, we label Bitcoin as a B-indulgent algorithm, since this protocol tolerates adversarial behavior in the nodes, even if the adversary controls the vast majority of the identities (as long as their combined computation power stays small enough). This difference between PBFT and Bitcoin reflects mainly at steps ② and ③ of these protocols.

Another important difference between Bitcoin and PBFT is the notion of quorum. Unlike PBFT that relies on a quorum in terms of the number of participating replicas, Bitcoin requires the cooperation of replicas that together possess a majority of the computing power in order to make decisions. While this has severe impact on energy efficiency and performance of the system, it serves as a protection against Sybil attacks, towards which Bitcoin is not vulnerable.

Various attacks have been crafted against the Bitcoin protocol. Several attacks have to do with increasing the rewards a miner can obtain in an unfair way, e.g., through selfish mining [16], block withholding, or fork after withholding attacks. Perhaps more important than fairness in rewards, we are interested in correctness attacks, that is, attacks that potentially lead to double spending. An eclipse attack is interesting for our discussion because it illustrates a concrete exploitation of the optimistic nature of the Bitcoin protocol [22]. An attacker with a sufficient number of IP addresses (in the order of thousands) can pollute—i.e., eclipse—an honest node's membership view, so that the honest node only has connections with the attacker and no other honest node in the Bitcoin network. Essentially, the attacker partitions the honest node from the rest of the network. Thereafter, the attacker creates a fork in the blockchain: in one branch the attacker spends money on certain goods, while in another branch the attacker is buying something from the honest node. The former branch is part of the actual Bitcoin network (and hidden from the eclipsed node), while the latter branch is eventually orphaned. Note that the attacker spent the same money on both branches. To summarize, an eclipse attack is a way to exploit Bitcoin's timing assumptions towards carrying out a double spending attack, i.e., violating Bitcont's safety.

4. BLOCKCHAIN VARIATIONS

In this section, we position a few notable blockchain protocols (Algorand, ByzCoin and Bitcoin-NG) in relation to the general scheme we introduced earlier (§3.1). For brevity, we adopt a high-level view on each protocol and the focus will be on their properties and assumptions (not on algorithmic details) with respect to A-indulgence and B-indulgence.

4.1 Algorand

Algorand [19] is a recent blockchain protocol designed for a permissionless environment, and exhibits very good performance improvements over Bitcoin. In a nutshell, step ② in this algorithm uses verifiable random functions to sample nodes across the whole system and elect a small committee. This committee is meant to act, as a whole, in the role of the leader. Algorand does not assign voting power based on identities (as PBFT does), nor based on computational power (as done in Bitcoin), but instead associates a weight with each node in the system based on the amount of money that node possesses. This approach is known as proof-of-stake.

Step ③ in Algorand takes the form of a novel Byzantine agreement algorithm, called BA⋆. At a high level, this is a two-phase agreement protocol. Each phase comprises a series of steps (typically between 2 and 11) leading either to agreement among correct nodes or to a recovery subprotocol. Recovery is triggered in case agreement is not reached (e.g., because of a network partition) and a fork occurred. This subprotocol represents step ④, that is, reconciling conflicting views, and highlights the importance of this step in our general scheme.

To ensure safety, Algorand assumes that the network experiences bounded periods of asynchrony, and each such period is followed by a bounded period of synchrony [19]. The exact length of these periods is irrelevant; what matters to our discussion is that safety in Algorand relies on the existence of synchronous periods. A malicious adversary can therefore exploit the assumption on the length of the synchronous period towards subverting this system's safety.

Algorand is an example of a protocol lying in between PBFT and Bitcoin in terms of its indulgence. It is more A-indulgent than Bitcoin: a violation of synchrony assumptions is not sufficient to violate safety if all participants are honest (unlike in the case of Bitcoin). Algorand is also more B-indulgent than PBFT, in the sense that controlling the majority of nodes is not enough to subvert the system (a substantial fraction of the money is also necessary). To subvert Algorand's safety, two conditions must be met: (1) malicious nodes must control some part of the money (not necessarily a big fraction), and (2) the network must experience some asynchrony. Thus, Algorand is both less A-indulgent than PBFT and less B-indulgent than Bitcoin.

4.2 Bitcoin-NG

Bitcoin-NG [15] is an important protocol because it introduces the idea of decoupling leader election (step ②) from agreement on blocks (step ③). In the original Bitcoin protocol, as described in §3.2.2, these two steps are entangled: a single leader is elected via PoW and nodes may immediately commit on the leader's proposal (if they choose to do so), or may wait for some confirmations, but no additional mechanism is necessary towards commitment.

Using this decoupling strategy, Bitcoin-NG can reach superior throughput compared to Bitcoin. Clearly, however, the elected node has too much responsibility: on its own, this leader can throw the system into a state of inconsistency by introducing forks and allowing double-spending. Essentially, this system has very similar assumptions and vulnerabilities as Bitcoin, and thus we consider it a typical B-indulgent protocol.

4.3 ByzCoin

ByzCoin [24] is an instance of a hybrid blockchain protocol [30], as it combines PBFT-style with Bitcoin-style agreement. Other such protocols exist [12, 30]; we focus here on ByzCoin, but we believe that our conclusions equally apply to other hybrid algorithms.

The common motivation underlying hybrid protocols is that neither Bitcoin's agreement (based on PoW), nor PBFT-style agreement are perfect, but they are in a sense complementary to each other. The former is very slow but has the advantage of being resilient to Sybil attacks. The latter is faster, but performs optimally if running on a small set of nodes (e.g., 4 to 7 replicas), and has no in-built protection against Sybil attacks. Hybrid protocols combine these two styles of agreement in the hope of avoiding their individual pitfalls and merging their specific strengths.

Similarly to Bitcoin-NG, ByzCoin decouples leader election (step ②) from agreement on transactions (step ③). However, leader election in ByzCoin means electing a whole committee of nodes (not just a single one). This committee runs PBFT, which is used to quickly serialize new transactions.

To be part of a committee, a node must pass a simple requirement: run PoW and be one of the last w nodes to find a solution. The parameter w, called a "share window" [24], defines the number of successful miners that will form the committee. This parameter encapsulates an important trade-off, from the point of view of this protocol's guarantees. On one hand, smaller w means smaller committee and a bigger threat to safety, as it is more likely that an adversary with sufficient computational resources gains control of the committee—i.e., obtain a third of total shares in the PBFT committee—and hence subvert the system's safety, as we explained earlier (§3.2.1). On the other hand, bigger w means larger committees: less potential for an adversary to control the committee, but increased risk of losing liveness if a third of the nodes in the committee are unresponsive, e.g., by being inactive at certain times or just leaving the network.

From the point of view of indulgence, ByzCoin is a combination of an A- and B-indulgent algorithm. The PBFT-like consensus algorithm ensures safety in periods of asynchrony, while the Bitcoin-like leader election protects against malicious node behavior such as Sybil attacks.

5. CONCLUSIONS

In this paper, we have presented a brief overview of several notable blockchain protocols, relating them to well-established concepts from distributed computing. We proposed a general scheme which unifies classical (PBFT-like) state machine replication protocols with the increasingly popular blockchain protocols. Our main goal was to shed light on the differences between these protocols. In doing so, we also pointed out the existence of two classes of protocols, defined in terms of how an adversary can go about subverting their safety. Asynchrony-indulgent protocols maintain their safety despite the harshest conditions of the network. A second class is that of malicious behavior-indulgent protocols, which maintain safety while tolerating big numbers of malicious nodes. We have shown how some protocols in the blockchain ecosystem are representatives of one class or another, or how they combine these two classes.

6. REFERENCES

[1] Confirmation – Bitcoin wiki. https://en.bitcoin.it/wiki/Confirmation.

[2] Ethereum project. http://www.ethereum.org.

[3] I. Abraham, G. Gueta, D. Malkhi, L. Alvisi, R. Kotla, and J.-P. Martin. Revisiting fast practical byzantine fault tolerance. 2017.

[4] I. Abraham and D. Malkhi. Bvp: Byzantine vertical paxos. 2016.

[5] B. Alpern and F. B. Schneider. Defining liveness. *Information processing letters*, 21(4):181–185, 1985.

[6] L. Alvisi, A. Clement, A. Epasto, S. Lattanzi, and A. Panconesi. Communities, random walks, and social sybil defense. *Internet Mathematics*, 10(3-4), 2014.

[7] H. Attiya, A. Bar-Noy, and D. Dolev. Sharing memory robustly in message-passing systems. *Journal of the ACM (JACM)*, 42(1), 1995.

[8] C. Badertscher, U. Maurer, D. Tschudi, and V. Zikas. Bitcoin as a transaction ledger: A composable treatment. In *Eurocrypt*, 2017.

[9] A. Bessani, J. Sousa, and E. E. Alchieri. State machine replication for the masses with BFT-SMaRt. In *IEEE/IFIP DSN*, 2014.

[10] C. Cachin and M. Vukolić. Blockchains consensus protocols in the wild (keynote talk). In *DISC*, 2017. arXiv preprint arXiv:1707.01873.

[11] M. Castro and B. Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4), 2002.

[12] C. Decker, J. Seidel, and R. Wattenhofer. Bitcoin meets strong consistency. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, page 13, 2016.

[13] C. Delporte-Gallet, H. Fauconnier, and R. Guerraoui. Tight failure detection bounds on atomic object implementations. *JACM*, 57(4), 2010.

[14] J. R. Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[15] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse. Bitcoin-NG: A Scalable Blockchain Protocol. In *NSDI*, 2016.

[16] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.

[17] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *JACM*, 32(2), 1985.

[18] J. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Eurocrypt*, 2017.

[19] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *SOSP*, 2017.

[20] A. Gottlieb and C. P. Kruskal. Coordinating Parallel Processors: A Partial Unification. *SIGARCH Comput. Archit. News*, 9(6), 1981.

[21] R. Guerraoui. Indulgent algorithms (preliminary version). In *PODC*, 2000.

[22] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *USENIX Security Symposium*, 2015.

[23] M. Herlihy. Wait-free synchronization. *ACM TOPLAS*, 13(1), 1991.

[24] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *USENIX Security*, 2016.

[25] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM TOPLAS*, 4(3), 1982.

[26] D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed computing*, 11(4), 1998.

[27] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[28] D. Ongaro. *Consensus: Bridging theory and practice*. PhD thesis, Stanford University, 2014.

[29] R. Pass, L. Seeman, and A. Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Eurocrypt*, 2017.

[30] R. Pass and E. Shi. Hybrid consensus: Efficient consensus in the permissionless model. *Cryptology ePrint Archive*, 2017. http://eprint.iacr.org/2016/917.pdf.

[31] M. C. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

[32] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, 1990.

[33] M. Vukolić. The Quest for Scalable Blockchain Fabric: Proof-of-work vs. BFT Replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.

A Case Study for Grain Quality Assurance Tracking based on a Blockchain Business Network

Percival Lucena
IBM Research

Alécio P. D. Binotto
IBM Research

Fernanda da Silva Momo
UFRGS

Henry Kim
York University

ABSTRACT

One of the key processes in Agriculture is quality measurement throughout the transportation of grains along its complex supply chain. This procedure is suitable for failures, such as delays to final destinations, poor monitoring, and frauds. To address the grain quality measurement challenge through the transportation chain, novel technologies, such as Distributed Ledger and Blockchain, can bring more efficiency and resilience to the process. Particularly, Blockchain is a new type of distributed database in which transactions are securely appended using cryptography and hashed pointers. Those transactions can be generated and ruled by special network-embedded software – known as smart contracts – that may be public to all nodes of the network or may be private to a specific set of peer nodes. This paper analyses the implementation of Blockchain technology targeting grain quality assurance tracking in a real scenario. Preliminary results support a potential demand for a Blockchain-based certification that would lead to an added valuation of around 15% for GM-free soy in the scope of a Grain Exporter Business Network in Brazil.

1. INTRODUCTION

According to a WorldBank Report [1], in 2015 Brazil exported 195 billion dollars, making it the 21st largest exporter in the world. Exports are led by soybeans which represent 11% of the total exports of Brazil. Other grains such as coffee 3% and corn 2.6% also represent a significant percentage of Brazilian exports.

The United States Department of Agriculture annual production, supply and distribution report states that Brazil is the second largest soybean producer in the world, behind only the United States[2]. The sum of exports from January to August 2017 is over 57 million tonnes [3]. In comparison with 2016, in this same period, there was an increase of more than 8.7 million tons exported.

The Brazilian cereal grains production for the 2016/17 harvest increased by 28% reaching 238.8 million tons as reported by the 12th Grain Survey conducted by Conab (Brazilian National Supply Company) [3]. A FIESP survey (Federation of Industries of the State of São Paulo) shows that the planted area dedicated to grains in Brazil is estimated at 60.9 million hectares which represents the largest area registered in the historical series [4].

The increasing productivity of grains in Brazil is related to a greater use of technology in the field, present not only in machines and implements but also in seeds, cultivation techniques and also the use of irrigation. Regarding the production of maize, soybean, and wheat, there is a variation of productivity in relation to the 15/16 harvest, respectively, 32.9%, 17.2% and -14.8% [3].

Forasmuch as technology has helped Brazilian cereals grain production, agriculturalists still face logistics and warehouse challenges to move the production from the farm to port terminals and processing industries. More than sixty percent of the Brazilian soybean production is transported by truck from the production areas to the port terminals [5].

The existing transportation and warehouse storage processes often affect grain quality causing grain damage, moisture, and contamination [6]. Grain quality control information is often kept in spreadsheets and spread among diverse ledgers which often provides inaccurate information causing financial losses on negotiation between producers and traders.

This case study aims to describe and highlight the gains obtained with the implementation of a Blockchain Business Network for Brazilian Agriculture exports; the lessons learned from the implementation of this project; the challenges in the development of Blockchain platform and future opportunities of using Blockchain in other contexts.

This paper presents the GEBN Blockchain Business Network, an enterprise consortium that aggregates data from certified quality assurance processes and provides information for diverse business partners of The Brazilian Grain Exporters Business Network. This platform helps producers track grains stored in warehouses optimizing trading with global exporters. The remainder of this paper will explain the solution implemented and it is structured as follows: Section 2 introduces Blockchain Business Networks. Section 3 describes the Brazilian Grain Exporters Business Network. Section 4 presents the case study context. Section 5 presents a Proof of Concept developed, and Section 6 presents results and conclusions.

Symposium on Foundations and Applications of Blockchain (FAB '18) March 9, 2018, Los Angeles, California, USA.

2. BLOCKCHAIN TECHNOLOGY APPLICATIONS ON AGRICULTURE

Blockchain can be seen as a "disruptive innovation with a wide range of applications, potentially capable of redesigning our interactions in business, politics, and society in general" [7] . Focusing on the business field, Cohen, Amorós, and Lundy (2017) [8] point out that the use of Blockchain in solving different business problems, from different segments, allows for modifications in existing business models and even the creation of new business models. Therefore, this technology allows new opportunities for creating customer value in a business model suitable for its exploration [8].

Blockchain has originated as a shared database for recording the history of Bitcoin transactions [9]. These transactions are grouped in blocks, including hashed pointers to previous blocks, that provide the accepted history of transactions since the inception of the Blockchain. This architecture has been implemented and extended by several general purpose ledgers such as Hyperledger Fabric [10], R3 Corda [11] and Ethereum [12].

The Blockchain can be regarded as a complex, network-based software connector, which provides communication, coordination (through transactions, smart contracts, and validation oracles) and facilitation services. Every node in the blockchain network has two layers, namely, application layer and blockchain layer. Part of the application is implemented inside the blockchain connector in terms of smart contracts [13].

Szabo [14] has coined the "smart contract" term as software representation for many kinds of contractual clauses in a way to make a breach of contract expensive for the breacher. Smart contracts could be used to represent liens, bonding, delineation of property rights, and other paper-based contracts. Smart contracts may be operated by a consortium comprising of parties in a multilateral contract [15].

2.1 Blockchain Business Networks

Advanced globalization has created several complex supply chain scenarios where different companies cooperate to form a "quasi-organization" [16]. A Business Network describes the structures and processes that exist in the exchange of assets among participants in economic networks.

In the business network, every company "connects different people, various activities and miscellaneous resources with varying degrees of mutual fit" and "operates within a texture of interdependencies that affects its development" [17]. Technology, knowledge, social relations, administrative routines, and systems are some resources that are encountered in business relationships.

Therefore, participants, assets, registries, and transactions are shown as fundamental elements of a business network. Participants are the actors in the business network and might be an individual or an organization. Assets are created by participants and subsequently exchanged between them through transactions. Assets can have a rich lifecycle, as defined by the transaction in which they are involved. As assets move through their lifecycle and through different registries they can be in more than one registry at the same time.

Complex business networks can be represented in different ways. Previous to the advent of Blockchchain, most multi-party business collaborations were implemented in the form of multiple binary relationships. Alternatively, the business partners would rely on a Trusted Third Party (TTP) to facilitate interactions, since a TTP acts for building and repair the trust [18]. Another possibility is the use of technology as a way to reduce TTP's participation in the business network maintaining trusting. Blockchain can represent complex business networks by nodes for each one of the different companies that need to cooperate and exchange information.

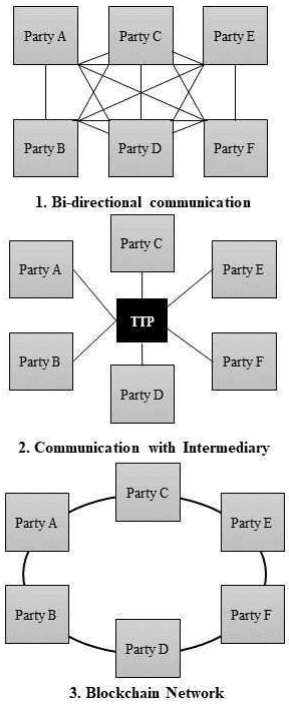


Figure 1: External business collaborations implemented on Blockchain.

Contracts are usually created as a set of promises to formalize relationships in a Business Network. Whenever a trusted intermediary is removed, the organizations involved in interdependent processes must find alternative means to provide a division of labor as effectively as the displaced contractor. Smart contracts can provide such division of labor and near decomposability [15].

Blockchain provides a new approach to Business Networks. According to several authors [19, 20, 21, 22], this technology has a great potential of impact and revolution in the world economy from the generation of changes in organizations and in the way business is done. Most of the blockchain networks assume organizations running the peers have no trust relationship established between them. Encryption, consensus, and other algorithms of blockchain guarantee trusted outcomes in this context [23].

In this regard, Figure 1 presents the relationship configurations mentioned before between the parties involved in a transaction of a complex business network. We highlight from these configurations that the use of Blockchain allows a better flow between the members of the business network and remove the role of some intermediaries in some business processes [9].

3. THE BRAZILIAN GRAIN EXPORTERS BUSINESS NETWORK (GEBN)

The Grain Exporters Business Network (GEBN) in Brazil is composed of a diverse set of players including grain producers, rural credit cooperatives, warehouse companies, tradings exporters, agrochemical companies, freight forwarders, and ports authorities. There are different types of transactions and contracts among business partners for financing, sales, transportation, warehousing, among others.

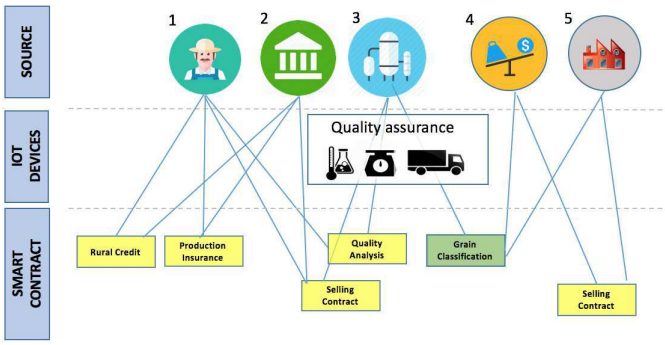


Figure 2: Grain Exporters Business Network

Figure 2 illustrates part of the Grain Exporters Business Network consortium. Actor 1 represents a grain producer who is usually concerned that the grains ingest and classification are properly conduced, so he can be fairly paid. The ingest receipt is also important for the Producer in order to obtain credit on Rural Credit Banks and to contract production insurance as well. Actor 2 represents a Rural Credit Bank agent who depends on accurate data from producers to reduce credit giving risks and offer lower interest credit rates for credit operations. Actor 3 represents a private warehouse agent who depends on accurate grain classification data in order to wholesale grains. Actor 4 represents a trading company agent who is concerned in buying large amount of grains with the right quality from several warehouses in order to fulfill an export request. Actor 5 represents a food processing company who is concerned in buying special selected grains that have specific characteristics such as high protein, high carbohydrates and low moisture levels.

The different players from the grain supply chain were not able to trust a centralized system due to the different business goals of the business actors. The Brazilian Government National Supply Company (CONAB) GeoSafras Report [24] offers yearly total estimates of grains production. Unfortunately, real time information about grain products is segregated to local actors in the supply chain.

Blockchain smart contracts provides a fair trading system of exchange that honours producers, communities, wholesalers, traders and the environment. Blockchain immutable ledger provides a compliance mechanism in order to avoid frauds in the grain classification process. As described in Figure 2, GEBN platform was created so that laboratory quality assurance test devices can connect directly to the Blockchain so the process information cannot be changed. This provides extra trust for the GEBN business partners who depend on accurate grain information. Blockchain can increase its likelihood of export to international markets since compliance with international standards becomes a transparent and undisputed matter [25].

The designed solution plans to be extended to cover most common agriculture processes allowing Agribusiness partners to collaborate seamlessly, reducing communication time, helping to track product provenance, reducing costs, and proving trust among GEBN partners.

4. CASE STUDY CONTEXT

The grain mass stored in the silos deteriorates in relation to the interaction between physical variables (temperature, humidity, warehouse structure, meteorological variables), chemical variables (oxygen availability in intergranular air) and biological variables of internal sources (longevity, respiration, post-harvest maturity and germination) and biological variables from external sources (fungi, yeasts, bacteria, insects, mites, rodents and birds). The degree of deterioration depends on the rate of increase of these variables which, in turn, are mainly affected by the interaction of temperature and humidity and secondarily by their interrelation with the grain, between them, and with the structure of the silo [26].

The search for the quality of grains and by-products is a priority for producers, processors, and for distributors of these products. Quality assurance processes are used on GEBN to provide grain classification according to international standards. Due to deterioration factors, it is important to have accurate information of the grains when they arrive and leave the warehouse silos. The intrinsic and extrinsic analysis processes are executed on both incoming and outgoing trucks, so grains quality assurance compliance process provides decision information for both grains buyers and sellers.

In a standard quality assurance process, an automatic crane located in the warehouse entrance, removes samples from a grain loaded truck. Those samples are sent by pipes to a packer in the laboratory. A laboratory technician collects the samples and submits them to intrinsic and extrinsic analysis tests.

4.1 Intrinsic Analysis

This process analyses soybeans and corn samples in order to detect if the grains are genetically modified (GMO) and if mycotoxins levels are at acceptable levels. Sample grains are ground so that 60-70% of the sample must pass through a 20-mesh sieve. Grains are then mixed with water in a (1:5) proportion. A 12 ml sample is removed from the solution using a pipette and dispensed into a reaction cup where a reaction strip is placed for 5 minutes. Inside the reaction tube, the sample travels by capillary action of an end to the other extreme of the tape. When passing through the membrane, the sample comes in contact with the antibodies that react with the target analyte [27].

After the testing period, the strip is then immediately placed into Envirologix QuickScanner device which is connected to a computer that reads the information and stores it on GEBN Blockchain. Each batch of test strips for GMOs or mycotoxins is tested and compared to known quantification patterns generating specific batch curves. The standard curve data is encoded in a 2-D strip itself. When the strip is read, Envirologix QuickScanner software measures the information of the standard curve present in the barcode and calculates the amount of analyte specific to each test tape.

4.2 Extrinsic Analysis

Brooker et al. [28] considers several properties for extrinsic analysis of grains, such as: moisture content, specific mass, the percentage of broken grains, impurities, damages caused by drying temperature, susceptibility to breakage, grinding, the presence of insects and fungi, type of grain and year of production. GEBN Blockchain extrinsic analysis process determines some of those grains characteristics such as moisture levels, broken and damage levels.

Gehaka Moisture Analyzers installed in the Quality assurance laboratory provide information directly to the GEBN Blockchain. The quality process that requires visual analysis is stored including information of the testing operator and exam date in order to avoid disputes.

5. EXPERIMENT

On our GEBN study, each business partner owns a node in the network with a full copy of transaction data shared among all the participants. Three nodes were created on GEBN: one for the cooperative producers, another for the warehouse originator company and a third for a rural credit bank.

GEBN was deployed on a Hyperledger Fabric Blockchain Cloud instance. Figure 3 illustrates quality assurance transactions stored on GEBN Blockchain. Hyperledger 1.0 query mechanisms are used to trace the origin of an outgoing lot helping audit processes through the supply chain. Hyperleger Fabric 1.0 blockchain also provides private communication channels, allowing computation to occur only among business partner nodes involved in a transaction. Permissioned blockchains such as Hyperledger Fabric have a set of trusted parties to carry out verification, and additional verifiers can be added to the agreement of the current members of the consortium. [29]. Permissioned blockchains offer clear advantages in security and privacy while potentially reducing costs of compliance with regulations [30].

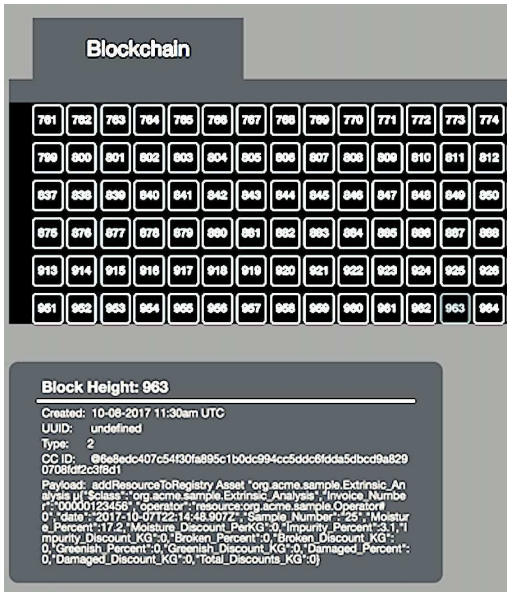


Figure 3: Quality Assurance Transaction on GEBN Blockchain.

Smart contracts were created using the Hyperledger Composer Framework [31] as described on Figure 4. Hyperledger Composer Framework includes a standalone Node.js process that exposes a business network as a LoopBack REST API [32]. The communication between the Grain Controller Desktop Application (GDPA) and the GEBN Blockchain server is protected by PassportJS open source authentication middleware [33] configured with passport-local strategy that implements an access control list.

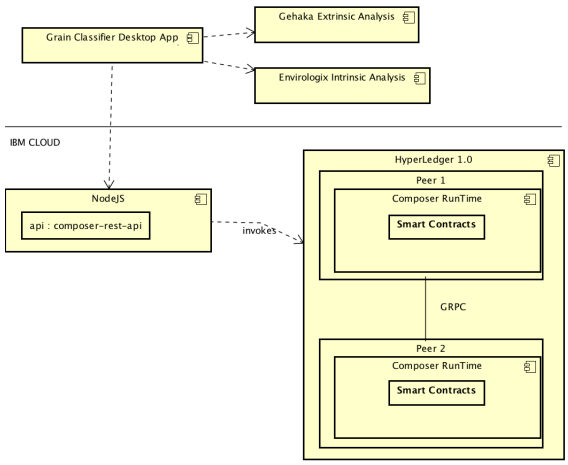


Figure 4: High Level Architecture

The Grain Controller Desktop Application was installed at a single Warehouse Quality Control lab in Ribeirão do Sul, Brazil. All the grains received at this facility pass through the quality control procedure before they are stored in silos. The quality control process allows the segregation of grains by levels of quality so prices are set according to the characteristics of the stored and marketed cargo. Listing 1 shows a sample smart contract developed for calculating price discounts for soybeans.

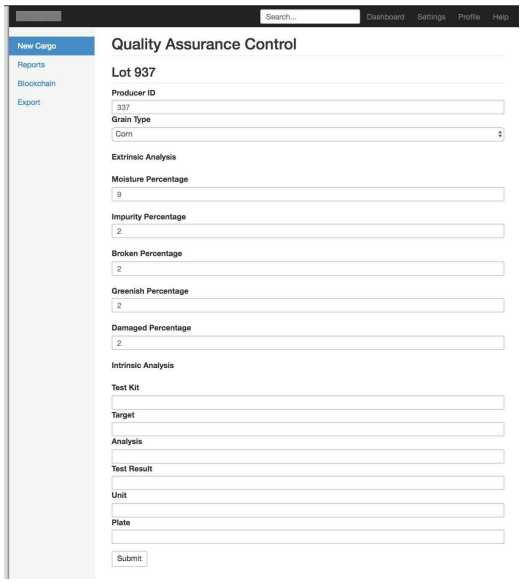


Figure 5: Grain Controller Desktop Application

The quality control procedure starts when the trucks arrive at the warehouse facility. The cargo electronic weighting information is stored on GEBN Blockchain. Afterwards, samples are taken from all incoming and outgoing trucks. Those samples are submitted to extrinsic and intrinsic analysis processes.

A certified quality assurance user authenticates in GDPA and inputs information about the producer's cargo that arrives at the facility as described on Figure 4. GDPA reads quality data directly from Intrincic and Extrinsic analysis devices. Producer's identity is also stored on GEBN Blockchain. The analysis results determine the pre-cleaning and drying process for the cargo and the appropriate storage silo. When a grain cargo is sold, a similar process is executed on the outgoing truck.

Listing 1: DiscountsTransaction

```
asset Extrinsic_Analysis identified by Invoice_Number {
    o String Invoice_Number
    --> Operator operator
    o DateTime date
    o String Sample_Number
    o Double Moisture_Percent
    o Double Impurity_Percent
    o Double Broken_Percent
    o Double Greenish_Percent
    o Double Damaged_Percent
    o Double Total_Discounts_KG
}

transaction DiscountsTransaction {
 --> Extrinsic_Analysis asset
}

event DiscountsEvent {
 --> Extrinsic_Analysis asset
}

function discountsTransaction(tx) {
    var d=0;
    if(tx.asset.Moisture_Percent>12)
    d+=(tx.asset.Moisture_Percent-12)*4;
    if(tx.asset.Impurity_Percent>3)
    d+=(tx.asset.Impurity_Percent-3)*2.5;
    if(tx.asset.Broken_Percent>5)
    d+=(tx.asset.Moisture_Percent-5)*1;
    if(tx.asset.Damaged_Percent>3)
    d+=(tx.asset.Impurity_Percent-3)*3.5;
    tx.asset.Total_Discounts_KG = d;

return
getAssetRegistry('com.agritech.Extrinsic_Analysis')
        .then(function (assetRegistry) {
            return assetRegistry.update(tx.asset);
        })
        .then(function () {
         var event =getFactory()
         .newEvent('com.agritech', 'DiscountsEvent');
         event.asset = tx.asset;
         tx.asset.Total_Discounts_KG;
         emit(event);
   });
}
```

6. CONCLUSIONS

This study aims to describe and highlight the gains obtained with the implementation of the blockchain platform in the agricultural context; the lessons learnt from the implementation of blockchain in the agricultural context; the challenges in the development of blockchain platform and future opportunities of using blockchain in other contexts.

One of the main advantages of using Blockchain, in spite of other software development platforms, is that all the members of the GEBN can now share the same business rules and transaction data in their nodes reducing disputes among business partners, information asymmetries and consequently improving governance.

The transactions transparency provided by Blockchain requires that the companies involved in the supply chain to collaborate effectively defining common rules that can be expressed in smart contracts. The formation of consortia like GEBN is an interesting form of organization that allows members of a supply chain to vote on the rules and consensus principles for transactions settlements.

We found a controversial use of Blockchain regarding its legal value. In order to prevent disputes, our approach focused on signing all transactions with an identity of a recognized member of the consortia. Brazilian legislation implemented by Medida Provisoria 2.200-2/2001 recognizes digital signatures on documents to have legal value. Nonetheless, complex scenarios involving international trade and arbitration laws are yet to be proven.

Although our blockchain application focused on grains quality control we believe there is a huge opportunity for Blockchain applications on Global Trade. Trading partners today rely on a complex and paper-heavy process to secure their transactions. Buyers, sellers, banks, transporters, inspectors, regulators have their own forms and records to fill out in separate systems of records. Capital is tied up as paper documents are sent back and forth, checked and rechecked, reviewed and reconciled. Delays caused by errors and manual processing can make it difficult for companies to access financing, which could cause business inefficiencies. Blockchain can provide a shared version of the truth, so trade partners can interact with greater trust. Third-party verification processes could be simplified by the use of smart contracts reducing the potential for errors or tampering. This can increase the efficiency with which companies access funding as well as save time and costs throughout the trade process.

Like all prior disruptive technologies, there will be beneficial and detrimental aspects of Blockchain technologies that will be tested as the first Blockchain Business networks start to operate. As the technology matures, Blockchain Business networks should provide several new business models that can revolutionize several industries worldwide.

7. REFERENCES

[1] World Bank. Brazil vegetable exports by country 2015 - online: http://wits.worldbank.org/countryprofile/en/country/bra, 2016.

[2] United States Department of Agriculture (USDA). Top 10 countries for oilseed, soybean. -http://bit.ly/2bjnoyq, September 2017.

[3] Companhia Nacional de Abastecimento (CONAB). Acompanhamento da safra brasileira de grãos, September 2017.

[4] Federação das Indústrias do Estado de São Paulo (FIESP). Informativo deagro, September 2017.

[5] Mary-Grace C Danao, Rodrigo S Zandonadi, and Richard S Gates. Development of a grain monitoring probe to measure temperature, relative humidity, carbon dioxide levels and logistical information during handling and transportation of soybeans. *Computers and Electronics in Agriculture*, 119:74–82, 2015.

[6] José Vicente Caixeta Filho. The determinants of transport costs in brazil's agribusiness. Technical report, Inter-American Development Bank, 2008.

[7] Marcella Atzori. Blockchain technology and decentralized governance: Is the state still necessary? 2015.

[8] Boyd Cohen, José Ernesto Amorós, and Lawrence

Lundy. The generative potential of emerging technology to support startups and new ecosystems, 2017.

[9] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[10] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. *arXiv preprint arXiv:1801.10228*, 2018.

[11] R Brown. Introducing r3 corda^{TM}: A distributed ledger designed for financial services. *R3 Cev*, 2016.

[12] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[13] Xiwei Xu, Cesare Pautasso, Liming Zhu, Vincent Gramoli, Alexander Ponomarev, An Binh Tran, and Shiping Chen. The blockchain as a software connector. In *Proceedings of the 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, 2016.

[14] Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.

[15] Henry M Kim and Marek Laskowski. A perspective on blockchain smart contracts: Reducing uncertainty and complexity in value exchange. 2017.

[16] Håkan Håkansson and David Ford. How should companies interact in business networks? *Journal of Business Research*, 55(2):133 – 139, 2002. Marketing Theory in the Next Millennium.

[17] Ivan Snehota and Hakan Hakansson. *Developing relationships in business networks*. Routledge London, 1995.

[18] Susan E Brodt and Lukas Neville. Repairing trust to preserve balance: A balance-theoretic approach to trust breach and repair in groups. *Negotiation and Conflict Management Research*, 6(1):49–65, 2013.

[19] Ray Valdes and David Furlonger. Prepare for a multiple blockchain world, 2016.

[20] Don Tapscott, Alex Tapscott, and Rik Kirkland. How blockchains could change the world, 2016.

[21] Joe McKendrick. Why blockchain may be your next supply chain, 2017.

[22] Don Tapscott and Alex Tapscott. *How Blockchain Will Change Organizations*. Winter, 2017.

[23] Richard Hull, Vishal S. Batra, Yi-Min Chen, Alin Deutsch, Fenno F. Terry Heath III, and Victor Vianu. *Towards a Shared Ledger Business Collaboration Language Based on Data-Aware Processes*, pages 18–36. Springer International Publishing, Cham, 2016.

[24] Divino Cristino Figueiredo. Projeto geosafras sistema de previsão de safras da conab. *Revista de Política Agrícola*, 14(2):110–120, 2005.

[25] Yu-Pin Lin, Joy R Petway, Johnathen Anthony, Hussnain Mukhtar, Shih-Wei Liao, Cheng-Fu Chou, and Yi-Fong Ho. Blockchain: The evolutionary next step for ict e-agriculture. *Environments*, 4(3):50, 2017.

[26] Ranendra N Sinha and William E Muir. *Grain storage: part of a system*. Avi Pub. Co., 1973.

[27] Envirologix. Quickscan user manual. *online: http://www.envirologix.com/wp-content/uploads/2015/05/M120.pdf*, 2017.

[28] Donald B Brooker, Fred W Bakker-Arkema, and Carl W Hall. *Drying and storage of grains and oilseeds*. Springer Science & Business Media, 1992.

[29] Gareth W Peters and Efstathios Panayi. Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money. In *Banking Beyond Banks and Money*, pages 239–278. Springer, 2016.

[30] David Yermack. Corporate governance and blockchains. *JReview of Finance*, 21(1):7 – 31, 2017.

[31] Ashray Kakadiya. Block-chain oriented software testing approach. 2017.

[32] Azat Mardan. Sails. js, derbyjs, loopback, and other frameworks. In *Pro Express. js*, pages 205–214. Springer, 2014.

[33] Caio Ribeiro Pereira. Authenticating users. In *Building APIs with Node. js*, pages 49–59. Springer, 2016.

Towards Trusted Social Networks with Blockchain Technology

Yize Chen
University of Washington
Seattle, WA, USA
yizechen@uw.edu

Quanlai Li
University of California, Berkeley
Berkeley, CA, USA
quanlai_li@berkeley.edu

Hao Wang
University of Washington
Seattle, WA, USA
hwang16@uw.edu

ABSTRACT

Large-scale rumor spreading could pose severe social and economic damages. The emergence of online social networks along with the new media can even make rumor spreading more severe. Effective control of rumor spreading is of theoretical and practical significance. This paper takes the first step to understand how the blockchain technology can help limit the spread of rumors. Specifically, we develop a new paradigm for social networks embedded with the blockchain technology, which employs decentralized contracts to motivate trust networks as well as secure information exchange contract. We design a blockchain-based sequential algorithm which utilizes virtual information credits for each peer-to-peer information exchange. We validate the effectiveness of the blockchain-enabled social network on limiting the rumor spreading. Simulation results validate our algorithm design in avoiding rapid and intense rumor spreading, and motivate better mechanism design for trusted social networks.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; J.4 [**Social and Behavioral Sciences**]: Computer Applications—*Social Networks*

Keywords

Blockchain, Rumor Spreading, Social Networks, SIR, SBIR

1. INTRODUCTION

Rumor has been existing for thousands of years in human history. A rumor often refers to a piece of unverified information (e.g., explanation of events, media coverage, and information exchange) circulating from person to person or pertaining to an object, event, or issue of public concern [1]. In the age of the Internet, denser connections among individuals along with faster information transmission rate also trigger rapid rumor propagation, and could cause more intense social panics and negative effects [2].

Past studies have put emphasis on both the modeling techniques and the avoidance mechanisms of rumor spreading. Yet considering the complexities of rumor transmission dynamics, the diversity of social networks, along with the emergence of information transmission media, most of these studies cannot find the root of rumor blast nor a general yet effective approach to eliminate rumor dissemination [3].

The blockchain technology then becomes a good fit, which has seen its success in financial area for trusted and secure contracts. This has motivated us to re-design the information exchange process as a "contract"-based process in modern social networks. In addition, the pair-wise spreading style of rumor also lets blockchain-based contract to become a good fit for future information propagation and exchange platform.

In this work, we introduce a mechanism for smart contract design that makes full use of the expressive power fulfilled by blockchain technologies. By allocating a virtual accumulated credit for each member in the social network, we design an innovative approach for information exchange. Such credits are a reflection of the credibility of both social network members and corresponding information. The proposed algorithm is designed to avoid the spreading of "untrusted information", which is information without sufficient endorsement.

To illustrate that such a mechanism design would help to avoid the large-scale propagation of fake news through the network, we design and set up a graphical model along with the nonlinear systems for the social networks that are of interests. We show that for peer-to-peer information exchange and propagation, individuals under blockchain are more cautious about the authenticity of the information. Our simulation examined the propagation of information with and without the proposed mechanism, and showed that our proposed approach can effectively reduce the social and economic damage by rumor. To our knowledge, this paper is the first work aiming to utilize the characteristics of blockchain to address and solve the rumor spreading problems in social networks. The general architecture for our proposed approach is shown in Fig. 1.

1.1 Related Work

1.1.1 Rumor Spreading Model

A rumor is a piece of *unverified circulating information*. Past research on rumor involved multidisciplinary efforts from physics, sociology, and psychology. Several approaches to the modeling of rumor spreading and control of its damage were discussed [4, 5]. Previous rumor models regarded the heterogeneous social network as a graph where rumors propagate. Studies in [6] used stochastic processes to simulate rumor spreading to get a better understanding. In [7], the authors observed that more influential spreaders exist on social networks. They assigned higher probabilities for them to spread the information. In the context of new types of social media and networks (e.g., micro-blogging), studies in [8] proposed

Symposium on Foundations and Applications of Blockchain (FAB '18) March 9, 2018, Los Angeles, California, USA.

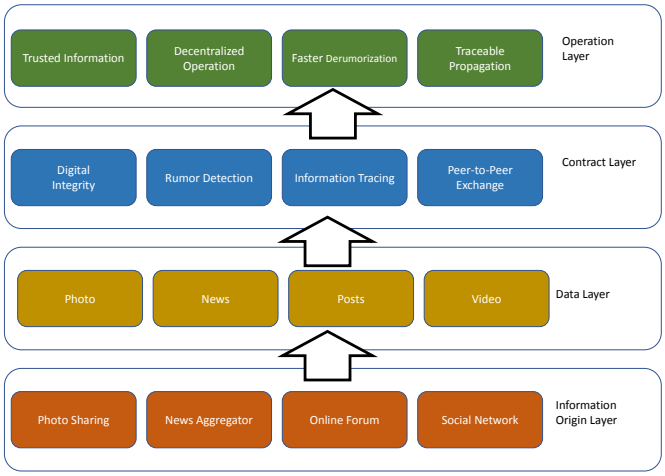


Figure 1: The architecture of blockchain-enabled information exchange system.

a SIR (Susceptible, Infected, Recovered) rumor spreading model. In this model, the spreading process is classified as susceptible, infected, and recovered. The work is based on the assumption that ignorants are easily influenced by the spreader, and that accordance with reality will change the probabilities of converting a spreader into a stifler.

Previous studies have shown that the cessation or blast of rumors is mainly related to the stifling and forgetting mechanisms for a given network [9, 10]. New forms of social networks, such as bidirectional information exchanges, also emerge. In this case, the receiver could also have an influence on the spreader. We will leverage the blockchain technology in this type of social network, and examine how this will affect the spread of rumors (e.g., change of immunity).

1.1.2 Blockchain Technology

A blockchain is a linked chain of growing list of blocks [11]. Every block contains its corresponding record and the timestamp. The blockchain is designed with a peer-to-peer network, where each node propagates its records to other nodes. This design prevents unvalidated modification of data.

Researchers have implemented blockchain-based protocols to build a decentralized network [13]. In the network, the third party is replaced by an automated access-control manager, enabled by the distributed blockchain system. Other researchers proposed to adopt blockchain in supply chain management for a better quality [14]. Blockchain can solve the traceability and trustability problems in this scenario. People also find blockchain useful in power grid industry [15]. Both utilities and consumers benefit from this technology by recording and validating the information on a distributed network affordably and reliably. Meanwhile, a combination of blockchain and the internet of things (IoT) increases utilization of cloud storage [16]. The blockchain is also suitable for other applications, such as online transaction, identity management, notarization [13].

The main contributions of this paper are as follows:

- We propose an innovative decentralized mechanism for social network information exchange based on blockchain;
- We build a blockchain-enabled SIR model and show from numerical simulations that from the “regulator" perspective proposed algorithm could effectively control rumor spreading on social networks.

The remainder of the paper is organized as follows. In Section 2, we present the rumor spreading model for social networks. In Section 3, we develop the blockchain-enabled architecture for social networks. In Section 4, we conduct numerical simulations to validate the effectiveness of our blockchain-enabled algorithm, and Section 5 concludes the paper.

2. RUMOR SPREADING MODEL

In this section, we present the SIR epidemiological model, which can be used to characterize the rumor spreading dynamics for a social network with a group of fixed participants. We analyze the temporal characteristics of such a stochastic model (e.g., the peak value, the convergence rate and the final state), and introduce the potential roles that the blockchain technology could play to reduce the spread of rumors. We also discuss several practical issues in real-world social networks.

2.1 Model Setup

Consider an undirected graph $G = (V,E)$, where V is a set of vertices representing individuals in the social network, and E is a set of edges representing the social interactions. We assume that the social network has a fixed number of homogeneously mixed population, and the degree distribution for nodes in G conforms to the Poisson distribution:

$$P(k) = e^{-\bar{k}} \frac{\bar{k}^k}{k!}, \tag{1}$$

where $\bar{k}$ is the average degree for G and $P(k)$ denotes the probability of observing k degrees for $v \in V$.

To better investigate how rumors are propagated through the network, we adopt a rule-based classification method that divides the vertices into three convertible sets [8, 10, 17]: the spreader set S, the ignorant set I, and the stifler set R. The dynamics of these three classes are as follows:

- *Ignorant with Density* $I(t)$. The ignorants are similar to susceptible individuals in classic SIR models. At time $t > 0$, an ignorant has a probability λ to become a spreader when it has contact with a spreader who is quite certain of the truth of the rumor. Afterwards, it’s willing to spread the rumor in the following time steps. Meanwhile, the ignorant has probability η to become a stifler, who has no interests in the rumor anymore.

- *Spreader with Density* $S(t)$. A spreader is contributing to the propagation of rumor within G. Any spreader involved in a pair-wise meeting attempts to “infect" other individuals with the known rumor. At time $t > 0$, when a spreader contacts with a stifler, the spreader has a probability γ to convert to a stifler. In addition, we take the forgetting mechanism [10] into consideration and assume that at a certain time, a spreader itself has forgotten the rumor and then turns into a stifler at rate δ.

- *Stifler with Density* $R(t)$. A stifler is contributing to the final elimination of the rumor. In general, it is an absorbing state in our stochastic model, and are accumulating its density by turning both ignorants and spreaders into stiflers.

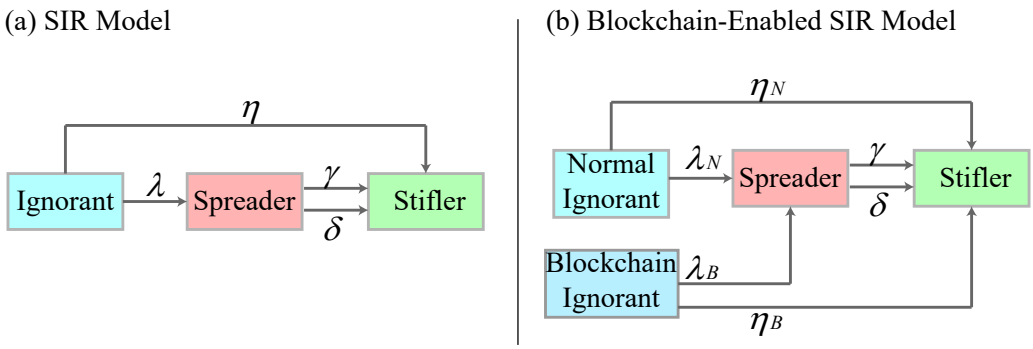


Figure 2: The comparison of classic SIR model for social networks (Fig. 2a) and SIR model under blockchain technology (Fig. 2b). A new group, the blockchain-enabled ignorants I_B, are emerged and taking part in the information exchange.

To summarize all the dynamics considered above, we derive a nonlinear system consisting of the following differential equations for $I(t), S(t)$ and $R(t)$, respectively.

$$\frac{dI(t)}{dt} = -(\lambda+\eta)\bar{k}I(t)S(t), \tag{2a}$$

$$\frac{dS(t)}{dt} = \lambda\bar{k}I(t)S(t) - \gamma\bar{k}S(t)(S(t)+R(t)) - \delta S(t), \tag{2b}$$

$$\frac{dR(t)}{dt} = \eta\bar{k}I(t)S(t) + \gamma\bar{k}S(t)(S(t)+R(t)) + \delta S(t), \tag{2c}$$

with the corresponding model structure plotted in Fig. 2a.

2.2 System Dynamics and Practical Issues

We initialize a social network G with $|V| = N$ with a spreaders who know the rumor and are willing to spread:

$$I(0) = \frac{N-a}{N}, \quad S(0) = \frac{a}{N}, \quad R(0) = 0. \tag{3}$$

We are interested in the dynamics of the rumor spreading model, e.g., the peak density of spreaders and the velocity of rumor spreading. It is also shown in [8] that when the system approaches to the final states, there are only ignorants and stiflers left in the network, while spreaders for untrusted information will die out. It is then important to observe the final state of $R(t)$, since a smaller $R(t)$ indicates that when the rumor appears again, the group of $I(t)$ will have to face the rumor spreading issue throughout the network.

The applicability of the model described in Section. 2a is also justified in several previous studies [10, 18]. In [18] it showed that such model is well fitted for real Twitter data on a set of real-world news (e.g., Boston Marathon Bombings and Pope Resignation).

3. SYSTEM ARCHITECTURE

In this section, we first describe the proposed blockchain-enabled protocol for information exchange, which can be integrated into the social network model described in Section. 2. We then illustrate how such a blockchain-enabled algorithm can propel a trusted social network as a whole.

3.1 Blockchain Protocols for Rumor Spreading

To ensure both security and privacy of the information change process and avoid large-scale spreading of untrusted piece of messages, we adopt the blockchain technology and design a protocol consisting of *private contract* and *public contract*. We allocate an accumulated virtual information credit for each participant in the network, and use such credits to motivate the propagation of trusted information.

3.1.1 Private contract

The private contract is negotiated and signed between the spreader and the receiver offline. Consider a group of spreaders $s_i \in S$ and a receiver r. At timestep t, validation between s_i and r is executed before a private contract is negotiated. For example, once the receiver r's desires have been accomplished, it "pays" virtual credit $cred_{rs}(t)$ to the spreader s_i (denoted as $cred_{s_i}$), while the spreader is in charge of sending the information $info_{sr}(t)$ to the receiver r (denoted as $info_r$). This "investment" of credit can pay off once this piece of information is validated to be trustworthy. The accumulated credits increase for receiver r. On the contrary, once the information is validated as a rumor, the accumulated credits of receiver r would decrease.

In Algorithm 1, we illustrate the working principle of such a peer-to-peer information-credit exchange program.

Algorithm 1 Private Smart Contract

Initialize: Initial spreader set S
Initialize: $info \leftarrow \emptyset$, $cred \leftarrow \emptyset$
Input: Contract Receiver r, r's accumulated credit $c(t)$
  for s_i in connection of r **do**
    if $s_i \in S$ **then**
      # *s_i and r form a secure channel to negotiate contract*
      if Contract made **then**
        $c(t+1) = c(t) - cred_{rs}(t)$
        $info_r \leftarrow info_{s_i r}(t)$, $cred_{s_i} = cred_{s_i} + cred_{rs_i}(t)$
        break
      end if
    end if
  end for

3.1.2 Public contract

The public contract is updated at every time step to record the links of information propagation as well as the credit flows throughout the social network. It serves as the public ledger for all information transactions. A transaction on information is used as evidence of contractor consent. This contract also makes the highest transaction credit C_{max} public to all existing participants of the information exchange, which is available for decision-making in private contract negotiation stage. The logs recording each transaction time,

credits along with the hash form the block. The pseudocode for achieving such a chain of contract is sketched in Algorithm 2.

Algorithm 2 Public Smart Contract

Initialize: highest credit $C_{max} = 0$; Credit of each member $cred_i$
Initialize: $info_{list} \leftarrow \emptyset$, $cred_{list} \leftarrow \emptyset$
  for $t \in T$ **do**
    if Contract made **then**
      Update C_{max} through the network
      Update $cred_{list}$, $info_{list}$
    end if
  end for

By employing the two-layer contract design for information exchange, we are able to construct a distributed, synchronized contract network which is secure and resilient. Moreover, as the network evolves, C_{max} increases with respect to the network consensus, which indicates either higher risk or higher credibility for ignorant to trust given information. Such public transaction information would guide each member under blockchain contract make their private decisions. Moreover, note that our blockchain-based contract architecture is not closed only for contractors. For normal individuals in the social networks, they possess their original information exchange process. In Section. 4 we compare the simulation results with different ratio of blockchain-based individuals involved in the social networks.

3.2 Rumor Spreading Model Under Blockchain

We begin by firstly introducing a trusted model along with the modeling assumptions. Then we justify how the blockchain technology would help inhibit the propagation of rumors on social networks.

The only difference between the model described here and the model described in Section. 2 is that we have a group of initial social network participants who have signed a blockchain-enabled trust contract. For the simplicity, we denote the density of initial members of such contract as I_B and initial members without signing the contract as I_N. Note that I_N conforms to the similar ignorants' dynamics as described in 2a, and the corresponding probability of converting to a spreader or a stifler is λ_N and η_N, respectively. Whenever there is information exchange between two individuals with at least one individual belonging to I_B, they run the secure and reliable consensus protocol to agree upon the pre-defined virtual credits for members under the trust contract. Since individual coming from I_B has access to the public information provided by all existing blockchain contracts, she/he has a different estimate of virtual credits of information exchange. Therefore, I_B have different dynamics compared to I_N, and we denote the corresponding probability as λ_B and η_B, respectively. We can derive the dynamics of $I_N(t)$, $I_B(t)$, $S(t)$ and $R(t)$ on the blockchain-enabled social networks:

$$\frac{dI_B(t)}{dt} = -(\lambda_B + \eta_B)\bar{k}I_B(t)S(t), \tag{4a}$$

$$\frac{dI_N(t)}{dt} = -(\lambda_N + \eta_N)\bar{k}I_N(t)S(t), \tag{4b}$$

$$\frac{dS(t)}{dt} = \lambda_B\bar{k}I_B(t)S(t) + \lambda_N\bar{k}I_N(t)S(t) - \gamma\bar{k}S(t)(S(t)+R(t)) - \delta S(t), \tag{4c}$$

$$\frac{dR(t)}{dt} = \eta_B\bar{k}I_B(t)S(t) + \eta_N\bar{k}I_N(t)S(t) + \gamma\bar{k}S(t)(S(t)+R(t)) + \delta S(t). \tag{4d}$$

3.3 Mechanisms Analysis and Discussion

Rumor Spreading Rate: Once an individual has signed the trust contract under blockchain protocols, it has an extra record coming from the blockchain "transactions" list of information within the whole network. This gives her/him an additional estimate of the "value" for possible information exchange measured by accumulated credits, and thus can make a better judgment of the authenticity of information based on the risk of losing credits in certain transactions.

In general, once the private contract between two individuals has reached a higher value of the virtual credit, members from I_B are more cautious about the ongoing information exchange. Then members from I_B are less vulnerable when exposing to a rumor. Meanwhile, they are more likely to lose interest in a rumor and thus convert to stiflers directly.

Hence, with blockchain-enabled contract signed, ignorants from I_B have limited contributions to the spreader S but more contributions to the stifler R:

$$\lambda_B < \lambda_N, \quad \eta_B > \eta_N. \tag{5}$$

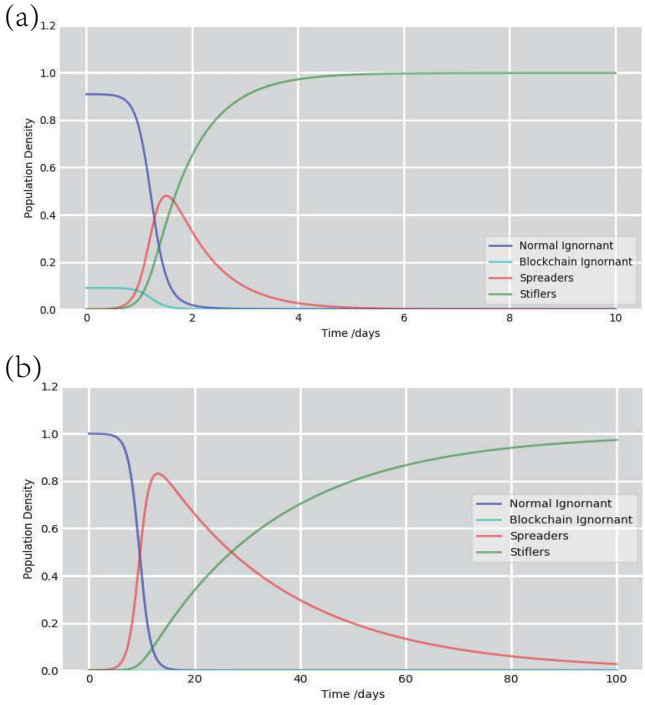


Figure 3: The temporal dynamics of the proposed blockchain-enabled system (Fig. 3a) compared to a blockchain-free system (Fig. 3b). Blockchain ignorants are under the blockchain contracts throughout the time.

Forgetting Mechanism: with blockchain-enabled SIR model for social networks, the forgetting mechanism not only takes account of the "forget" process of spreaders, it also takes account of those spreaders under blockchain contract, who are less likely to keep a "fake" news in their public records. So with blockchain technology enabled in a given G, δ tends to approach a higher value than the model in 2a. This indicates that the social network could get a higher absorbing rate from spreaders to stiflers.

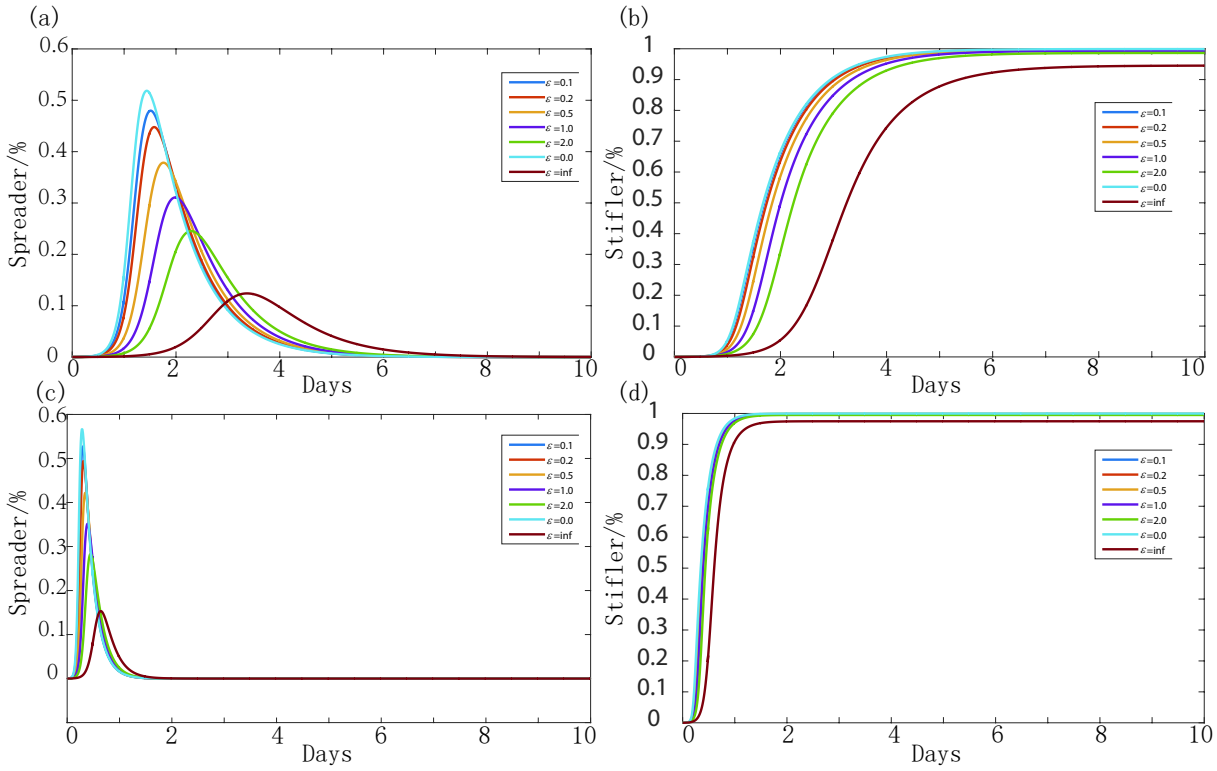


Figure 4: The temporal dynamics of the spreader density $\frac{S(t)}{N}$ (Fig. 4a and Fig. 4c) and the stifler density $\frac{R(t)}{N}$ (Fig. 4b and Fig. 4d) under different ε.

4. NUMERICAL SIMULATIONS

Following the model introduced in Section.2, in this section, we conduct numerical simulations to validate the performance of blockchain-enabled rumor spreading on social network models, and compare the results with blockchain-free rumor spreading performance.

We are particularly interested in the rumor spreading process from the initial stage $t = 0$ till the terminal stage $t = T$. That is to say, in all our simulations over $t \in \{0, 1, ..., T\}$, we start with $S(0) = 1, R(0) = 0$ out of a fixed overall population of $S(t)+R(t)+I(t) = 10000$, where $I(t) = I_B(t)+I_N(t)$. Note that we constrain our simulation to the case that stifler is the final absorbing state. Based on the discussion in Section. 3.3, we set $\lambda_B = 0.3$, $\lambda_N = 0.8$, $\eta_B = 0.7$, $\eta_N = 0.2$ to investigate the influence of blockchain contract. To better evaluate the performance of the blockchain contract in our model, we introduce $\varepsilon = \frac{I_B(0)}{I_N(0)}$ to control the initial population ratio. We also consider two group of settings for $\bar{k}$, where $\bar{k} = 10$ corresponds to a sparse, traditional information exchange platform, and $\bar{k} = 50$ corresponds to a dense, newly-emerged information exchange platform.

In the first experiment, we investigate two situations for the dynamics of rumor spreading process, which are shown in Fig. 3. In Fig. 3a we show a group of setting with $\varepsilon = 0.1, \delta = 0.3, \gamma = 0.1, \bar{k} = 10$. In Fig. 3b, we simulate the extreme circumstance in which no member signs the blockchain contract with $\varepsilon = 0$, while members from I_N easily trust the rumors with $\lambda_N = 0.99, \eta_N = 0.01, \bar{k} = 10$. We observe from Fig. 3b that the rumor exists much longer on the social networks (over 100 days compared with less than 6 days in Fig. 3a). Moreover, the peak density for spreaders is over 81%, which indicates that most of the members easily trust rumors. In contrast, with a portion of the population enrolled in the blockchain contract, the peak value of rumor density can be cut down to 48% as is shown in Fig. 3a.

We then conduct simulations to evaluate the impact of the number of individuals who sign the blockchain contract, as depicted in Fig. 4. Fig. 4a and Fig. 4b show the simulation results under $\bar{k} = 10$, implying everyone on the social network possesses relatively sparse connections (e.g., information exchange through newspapers and phone calls). Fig. 4c and Fig. 4d are simulated under $\bar{k} = 50$, in which everyone on the social network is densely connected (e.g., information exchange through online social networks). Note that $\varepsilon = \inf$ indicates the scenario where all the individuals except the initial spreader have signed the blockchain contract.

From Fig. 4a and Fig. 4c, we observe that as the number of initial blockchain contractors increases (larger ε value), the peak value of spreader density drops significantly. In addition, the peak is deferred compared to the case with a smaller ε value, indicating that the rumor spreading process has been delayed. This delayed and weaken rumor spreading process also provides an opportunity for external intervention (e.g., a credible clarification) to control rumor spreading. One interesting finding is that if we consider a dense social network which is more prevalent in today's new media as well as online social networks, the rumor spreading process is much quicker (Fig. 4c) compared to traditional rumor spreading media. Results depicted in Fig. 4b and Fig. 4d also verify that with a higher penetration of blockchain-enabled members, initial ignorants are more skeptical to the rumors and thus take a longer time to finally convert to stiflers. In addition, the results also show that $\frac{R(T)}{N} < 1$ decreases as ε increases in the terminal state.

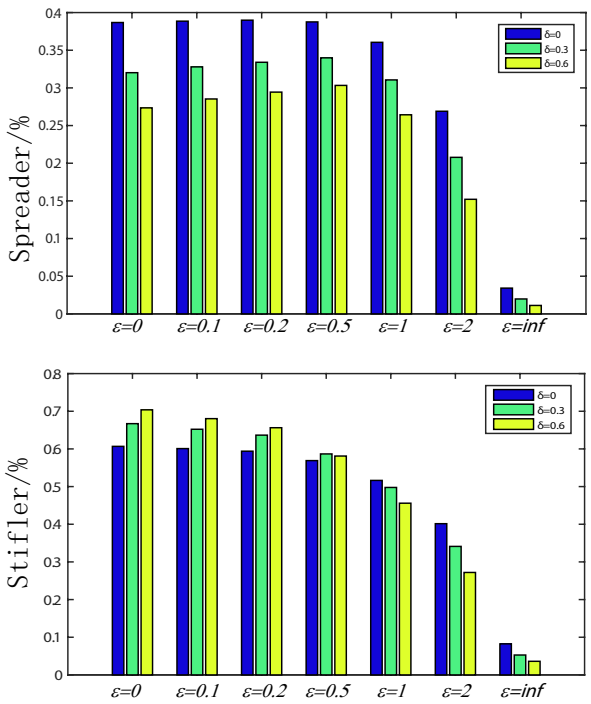


Figure 5: The spreader density $\frac{S(t)}{N}$ (Fig. 5a) and the stifler density $\frac{R(t)}{N}$ (Fig. 5b) at the end of day 2 under different ε and δ.

Our final simulation evaluates the mixed impact of ε and δ. Since the blockchain contract would also trigger some converted spreaders to be skeptical of rumors, δ can increase, and spreaders "forget" rumors and finally convert to stiflers. In Fig. 5a and Fig. 5b, we show the density of spreaders and stiflers at the end of day 2, respectively. We observe that a larger δ would drag down the spreader density, because a larger δ represents the spreaders are more willing to reconsider the rumor as fake information and convert to stiflers. Meanwhile, a larger ε can eventually drag down the spreader density, but as we observed in Fig. 4a and Fig. 4c, a larger ε would also change the rumor spreading speed. Therefore, there is no significant change in spreader density when ε is relatively small.

Results shown in Fig. 5 also motivate the mechanism design for a blockchain-based information propagation contract. The design of an appropriate virtual credit would not only control the system dynamics (e.g., rumor spreading speed and the peak value), but also regulate each participant's behavior (e.g., distributing different initial information credits).

5. CONCLUSION AND DISCUSSION

In this work, we investigated the dynamics of rumor dissemination in social networks with and without blockchain-enabled technology. We firstly introduced the graphical model setup for social networks. We then illustrated how to incorporate the blockchain contract into peer-to-peer information exchange process by employing virtual credits. The re-designed blockchain-enabled rumor spreading model along with numerical simulation demonstrated that blockchain technology would help in avoiding large-scale rumor spreading. Such model setup and simulation results would motivate us to design trust-based information exchange system with blockchain technology enabled.

In the future work, we would also like to inspect the extreme case that is not included in this work, e.g., at initial point, most of members are spreaders, or during the information propagation, members are with low immunity. Contracts designed for extreme conditions and large-scale social networks may be designed and considered.

6. REFERENCES

[1] W. A. Peterson and N. P. Gist, "Rumor and public opinion," *American Journal of Sociology*, vol. 57, no. 2, pp. 159–167, 1951.

[2] P. Bordia, "Studying verbal interaction on the internet: The case of rumor transmission research," *Behavior research methods*, vol. 28, no. 2, pp. 149–151, 1996.

[3] B. Doerr, M. Fouz, and T. Friedrich, "Why rumors spread so quickly in social networks," *Communications of the ACM*, vol. 55, no. 6, pp. 70–75, 2012.

[4] G. W. Allport and L. Postman, "The psychology of rumor." 1947.

[5] R. L. Rosnow and G. A. Fine, *Rumor and gossip: The social psychology of hearsay.* Elsevier, 1976.

[6] Y. Moreno, M. Nekovee, and A. F. Pacheco, "Dynamics of rumor spreading in complex networks," *Physical Review E*, vol. 69, no. 6, p. 066130, 2004.

[7] D. A. Vega-Oliveros, L. da F Costa, and F. A. Rodrigues, "Rumor propagation with heterogeneous transmission in social networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2017, no. 2, p. 023401, 2017.

[8] L. Zhao, H. Cui, X. Qiu, X. Wang, and J. Wang, "Sir rumor spreading model in the new media age," *Physica A: Statistical Mechanics and its Applications*, vol. 392, no. 4, pp. 995–1003, 2013.

[9] D. J. Daley and D. G. Kendall, "Epidemics and rumours," *Nature*, vol. 204, no. 4963, pp. 1118–1118, 1964.

[10] M. Nekovee, Y. Moreno, G. Bianconi, and M. Marsili, "Theory of rumour spreading in complex social networks," *Physica A: Statistical Mechanics and its Applications*, vol. 374, no. 1, pp. 457–470, 2007.

[11] J. Brito and A. Castillo, *Bitcoin: A primer for policymakers*. Mercatus Center at George Mason University, 2013.

[12] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[13] G. Zyskind, O. Nathan *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *Security and Privacy Workshops (SPW), 2015 IEEE*. IEEE, 2015, pp. 180–184.

[14] S. Chen, R. Shi, Z. Ren, J. Yan, Y. Shi, and J. Zhang, "A blockchain-based supply chain quality management framework," in *2017 IEEE 14th International Conference on e-Business Engineering (ICEBE)*. IEEE, 2017, pp. 172–176.

[15] J. Basden and M. Cottrell, "How utilities are using blockchain to modernize the grid," *Harvard Business Review*, 2017.

[16] H. Shafagh, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of iot data," *arXiv preprint arXiv:1705.08230*, 2017.

[17] E. Beretta and Y. Takeuchi, "Global stability of an sir epidemic model with time delays," *Journal of mathematical biology*, vol. 33, no. 3, pp. 250–260, 1995.

[18] F. Jin, E. Dougherty, P. Saraf, Y. Cao, and N. Ramakrishnan, "Epidemiological modeling of news and rumors on twitter," in *Proceedings of the 7th Workshop on Social Network Mining and Analysis*. ACM, 2013, p. 8.

Author index