Cost-Effective Blockchain-based IoT Data Marketplaces with a Credit Invariant

James Meijers*, Guntur Dharma Putra†, Grammateia Kotsialou‡, Salil S. Kanhere† and Andreas Veneris*
*University of Toronto, †UNSW, Sydney, ‡London School of Economics and Political Science
j.meijers@mail.utoronto.ca, {gdputra, salil.kanhere}@unsw.edu.au, g.m.kotsialou@lse.ac.uk, veneris@eecg.toronto.edu

***Abstract*—Billions of Internet of Things (IoT) devices deployed today collect massive amounts of potentially valuable data. To efficiently utilize this data, markets must be developed where data can be traded in real time. Blockchain technology offers a potential platform for these types of markets. However, previous proposals using blockchain technology either require trusted third parties such as data brokers, or necessitate a large number of on-chain transactions to operate, incurring excessive overhead costs. This paper proposes a trustless data trading system that minimizes both the risk of fraud and the number of transactions performed on chain. In this system, data producers and consumers come to binding agreements while trading data off chain and they only settle on chain when a deposit or withdrawal of funds is required. A credit mechanism is also developed to further reduce the incurred fees. Additionally, the proposed marketplace is benchmarked on a private Ethereum network running on a lab-scale testbed and the proposed credit system is simulated so to analyze its risks and benefits.**

***Index Terms*—IoT, blockchain, data marketplace, mechanism design**

I. Introduction

Internet of Things (IoT) devices and the data they collect have a huge impact on society today. Billions of devices have been deployed, impacting large sectors of the economy, such as health care, automotives, manufacturing, and other infrastructure [1]. These devices produce vast amounts of potentially valuable data. However, as many of the devices operate in closed systems, this data is often unavailable to agents who may be able to use it [2]. This data hoarding can have a negative impact. Most obviously, it can stifle competition and innovation as it limits the number of people who can utilize the data. Additionally, the data owners forgo potential revenue from selling it. These issues can be avoided by giving data producers appropriate economic incentives to sell their data through efficient, fair, and secure marketplaces.

Selling IoT data has its own issues. IoT sensors generate data in real time and many applications must receive it as quickly as possible. Inevitably, as time passes its value may decrease and, therefore, there are incentives to deliver it in real time. MQTT [3] is a widely used protocol for the transfer of IoT data in real time. However, it does not include payment mechanisms and novel tools/methodologies are needed to enable efficient payment transmission channels over it.

Another issue with IoT data is that, depending on what data is being purchased, the producer or consumer of these transactions may not be fully "trusted". For example, a company that wishes to measure traffic flow in a city may wish to purchase location data from the users of phones moving around the city. The company may not wish to prepay for a large amount of data for fear that the phone users will accept the payment and then breach the agreement by withholding their data. On the other hand, users selling data may not wish to send large amounts of data at once without any prepayment for fear that the receiving party may eventually refuse to pay.

IoT data marketplaces do exist today [4], but require a trusted third party to mediate the data exchange. This introduces cost/privacy concerns, as the trusted third party generally has access to the data traded and may illicitly benefit from it. Further, similar concerns may prevent small players from being part of an IoT marketplace. Blockchain technology, and specifically *smart contracts* such as those that run on the Ethereum blockchain, can solve some of these problems. Smart contracts allow participants to enter into agreements secured by blockchain validators with no trusted third party required [5]. However, blockchains also introduce their own issues. First, it is impossible for smart contracts to directly track the flow of data over MQTT connections, making fair enforcement of the contract difficult. Second, blockchain transaction fees can be prohibitively expensive, *e.g.*, gas costs for Ethereum transactions occassionally amount to several dollars [6]. Such excessive fees can be a roadblock to "small" payments such as those often required by IoT data trades.

In this work, we propose a blockchain based data trading mechanism that allows the safe trading of real time data while still limiting the number of on-chain transactions required, thus reducing the associated fees. This is done through the use of binding off-chain agreements between data producers and consumers, secured through on-chain deposits. We show that the trading mechansim meets our objectives for fairness, privacy, and cost. We also propose a credit mechanism designed to further lower the fees incurred during data trades. Using this mechanism, producers estimate users' future behaviour based on the time the user has spent on the system in order to assign them credit that benefits both parties. We test our data trading mechanism by developing an implementation of the smart contract that is required and by analyzing the smart contract's performance. Our mechanism is shown to achieve significantly reduced transaction costs in comparison with the state-of-the-art. Further, simulation of the credit mechanism shows that producers and consumers can reduce fees by up to an additional 20% when compared to our base solution.

The remainder of this paper is organized as follows. Section II overviews existing technologies as well as problem definitions and objectives. Section III describes the data trading mechanism and analyzes it with respect to the underlying problem objectives. Section IV outlines the credit mechanism that reduces blockchain fees. Section V presents an implementation of the data trading methodology and an empirical evaluation

978-0-7381-1420-0/21/$31.00 ©2021 IEEE

of the credit mechanism. Finally, Section VI concludes this work.

II. Background

A. Related Work

Many blockchain based systems that facilitate trading of IoT data have been proposed in prior literature. In [7], Wörner et al. propose a data marketplace based on Bitcoin where sensors receive payments for data through a Bitcoin transaction and respond with the data itself in a subsequent transaction. While the authors admit the system is limited in that there is no guarantee that the consumer's payment will be honored by the producer, and that the method of communication imposes significant delays, it does open the possibility of blockchain based data marketplaces. In [8] the authors present a method for the sale of digital goods on blockchain that uses deposits from both the buyer and seller to incentivize honest behaviour. IDMoB [9] is a blockchain based marketplace specifically targeting machine learning applications, where obtaining large amounts of quality data is crucial. In that system, data is stored in a distributed database and data consumers can use smart contracts to pay producers for a key to access large data blocks. In [10], the authors propose a decentralized IoT data marketplace where two smart contracts, namely rating and product, play an important role in constructing a trusted, searchable registry for data owners and buyers to post and find available data products, which are traded using a Streaming Data Payment Protocol (SDPP) [11].

In [12], Bajoudah et al. describe a trustless, blockchain based IoT data marketplace targeting data streams. Their system requires the data producer to set a batch size. Once the consumer has received a full batch of data, a checkpoint is reached and the consumer must send a data receipt to a smart contract to confirm that they received the data and pay the producer. This may cause tension between the producer and consumer because the producer may prefer the smallest possible batch size to reduce risk of data loss, while the consumer may prefer the largest possible size to reduce the number of on-chain transactions required for the trade. A similar mechanism is described in [11]. The authors of [13] describe a model where a producer, consumer, and mediator all agree to a payment model and then continually update a shared smart-contract to track the data to be traded. When the producer and consumer disagree about how much data has been sent, the mediator settles the dispute. This adds complexity to the model, thus increasing its delay and/or cost.

Another important aspect of data marketplaces is that of matching consumers (who require a service) to producers (who supply it). In [14], the authors present a method for device owners to advertise their services, and potential consumers to choose between services that meet their requirements.

Ensuring the integrity of data produced by IoT devices is a major concern, and many works have proposed blockchain based solutions. In [15], the authors describe an efficient method for securing surveillance camera footage using blockchain. The work of B-FICA [16] presents a blockchain based system that records sensor data for use in auto-insurance claims. As cost is a major concern when it comes to storage on blockchain, [17] uses multiple low cost blockchains to temporarily secure data before final commitment on Ethereum.

Scaling blockchain capacity is crucial to reduce transaction costs and allow more services to run on chain. The Lightning Network [18] uses payment channels to increase the capacity of the Bitcoin network. These channels involve two users depositing coins to a shared wallet using a bitcoin transaction. Following this, those two users can trade those funds back and forth off chain in a secure, trustless manner, and only settle on chain when a user becomes uncooperative or the deposited funds are required for other uses.

Trust, reputation, and credit is another area of research that impacts IoT marketplaces. In [19], the authors propose a review system for IoT devices selling data on blockchains, allowing consumers to review the quality of the data they have received. Additionally, [20] presents a trust score system that rates the trustworthiness of consumers so that producers can decide whether or not to release their data to them. In [21], Luecking et al. propose a trust mechanism based on a web of trust model, which allows users to leverage their existing trust relationships to obtain the trust of new users.

Our mechanism design builds on prior work by utilizing deposits from the data consumer to set up a payment channel and trade real time IoT data. In this design the consumer will never pay for data they did not receive and the producer can reduce their potential data loss to be as low as that of a single sensor reading. This high level of safety is realized while still reducing the number of required on-chain transactions and their associated cost.

B. Problem Definition

We develop a system for a data producer to sell IoT data streams to an anonymous consumer using a blockchain to facilitate the payment(s). We model the consumer as wishing to purchase D worth of data from the producer over the course of some time interval. To do this, the consumer is willing to deposit up to d onto a smart-contract, where $d \leq D$. In this notation, we assume that D and d are measured in fiat money or some native blockchain tokens. It is also assumed that the producer and consumer are able to communicate in order to agree on the data to be sent and its monetary price [22]. We assume a linear pricing model, where each datum is of the same value, however the same system could be used for more complicated pricing models. Periodic sensor readings from IoT devices where data are delivered individually and are all of the same priority are well suited to linear pricing models. Once they come to an agreement, the producer can stream data to the consumer. We assume the consumer can verify the correctness and completeness of the data, that both parties can track the amount of data sent over this stream, and that the producer can stop the data stream at any time.

The system relies on an underlying blockchain platform for settlement. This blockchain can be permissioned or permissionless, but it must support smart contracts (*e.g.,* Ethereum, Hyperledger, etc.). Further, fees charged by the blockchain for a call to the proposed smart contract are bounded from above and below by $f_{\max}$ and $f_{\min}$, respectively. We assume that the producer monitors the blockchain for calls to their smart contract. When such a contract call occurs, the producer is able to post a transaction to the smart contract in response within at most t_{tx} blocks of the earlier contract call. This assumption is needed to ensure the producer is able to retrieve funds they

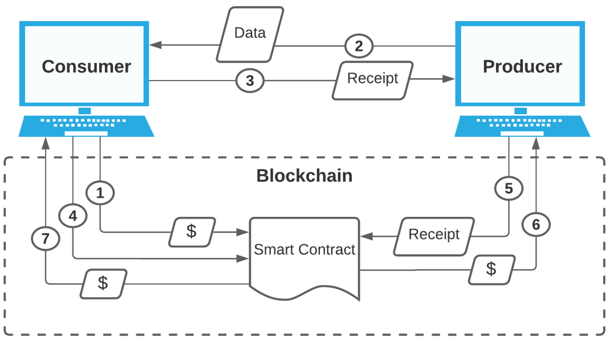

Fig. 1: Diagram of the data trading system.

are owed from the smart contract in response to the consumer signalling they wish to retrieve their deposit.

This work sets the following underlying objectives (criteria) for the design of an efficient IoT data trading system:

- **Objective 1** (*Consumer Fairness*): Consumers should not pay for data they did not receive.
- **Objective 2** (*Producer Fairness*): The risk of data loss to producers should be minimized.
- **Objective 3** (*Privacy*): Publicly visible data should be limited to maximize user privacy.
- **Objective 4** (*Cost*): The system should minimize overhead and operational costs.

Our transaction system builds on the periodic payment system of [12]. In that design, the producer sends batches of data to the consumer before getting paid. As such, the producer risks losing up to a full batch's worth of data due to a nonpayment, and thus they are incentivized to set the batch size to be small. However, this increases the transaction costs for the consumer. In contrast, our design disentangles the batch size from the consumer's transaction costs, allowing the producer to limit their maximum loss to an arbitrarily low value without affecting transaction costs. This design provides a high level of safety without requiring the complexity and frequent transactions required by some other systems. Further, it puts a strong emphasis on privacy. The smart contract only controls the storing of deposits and the transfer of payments from the consumer to the producer. No information about the agreement between the producer and consumer, such as the type and price of data to be delivered, is stored on the smart contract. This is done to protect the privacy of both parties.

III. Data Trading System

The setup of our transaction system is outlined in Fig. 1. Initially, the producer and consumer come to an agreement on a pricing model and data transfer mechanism off chain, and the consumer deposits money on the smart contract to begin trading. Data is sent to the consumer, who then signs and returns a receipt acknowledging they received the data. When desired, the consumer can retrieve their deposit from the smart contract and the producer can send the latest receipt to the smart contract to retrieve the funds they are owed and complete the payment. This process is detailed below.

A. Trade Setup

The trading relationship can be set up as follows:

1) The producer publishes a smart contract S on the blockchain. This contract serves as a trusted mediator between the producer and consumer.
2) The consumer selects the producer as their data provider and contacts them in order to negotiate a pricing model for the data and set a value ϵ, which is the maximum amount of data (measured in the unit of exchange) the producer sends to the consumer without receiving a signed receipt in return.
3) The consumer sends an *initial deposit* transaction (see ① in Fig. 1) which deposits d funds on the smart contract. A variable on the smart contract, $S.balance$, tracks the total amount that the consumer has stored on the smart contract and is initialized to d. Another smart contract variable, $S.paid$, tracks the total value paid to the producer and is initialized to 0.
4) When the producer observes the *initial deposit* transaction on S, they can begin to send data to the consumer.

The consumer can retrieve their deposited funds at any time by sending a *deposit retrieval* transaction (see ④ in Fig. 1) to the smart contract. This initial message informs the producer that the consumer wishes to retrieve their funds, but does not retrieve the funds from the smart contract S. After waiting a number of blocks specified by S, they can send another *deposit retrieval* transaction (see ⑦ in Fig. 1) to transfer $S.balance$ funds from the smart contract to their wallet and set $S.balance$ to 0. The required delay between the *deposit retrieval* transactions, $t_{\mathcal{D}}$, must be larger than the time it takes for the producer to respond to a transaction, t_{tx}, so that the producer has the chance to retrieve the funds they are owed before the deposit is removed from the smart contract. This security measure is referred to as a *speed bump* [23].

B. Trade Process

Once the trade setup has completed, the producer and consumer can follow the below process to safely trade data using off-chain interactions.

1) The producer sends ϵ worth of data to the consumer (see ② in Fig. 1) along with a receipt r with two fields, $r.sent$, set to be the total value of data that has been delivered so far, and a cryptographic signature field, $r.signature$, to be filled by the consumer.
2) If the consumer agrees that they have received $r.sent$ data, they sign and return the receipt (see ③ in Fig. 1).
3) The producer verifies the consumer's signature and the $r.sent$ amount on the receipt. Until these values are verified the producer will not send any further data to the consumer.
4) Steps 1) through 3) above are repeated, with $r.sent$ being increased with each receipt. The process continues until the outstanding payment due to the producer, determined as the difference between $r.sent$ and $S.paid$, is equal to the funds that the consumer has deposited, $S.balance$.

At any time, the producer can send the latest receipt r to the smart contract through a *receipt* transaction (see ⑤ and ⑥ in Fig. 1) in order to transfer funds from the deposit to their wallet. If $r.signature$ is a valid signature of the consumer, the following occur on the smart contract:

- The outstanding payment, $r.sent - S.paid$, is sent to the producer from the deposit.
- $S.balance$ is reduced by the payment value.
- $S.paid$ is increased by the payment value.

Through this mechanism, the producer can retrieve the funds they are owed without requiring any further cooperation from the consumer. Both $r.sent$ and $S.paid$ are cumulative values and do not reset between *receipt* transactions. This prevents the producer from performing a replay attack by sending the same receipt twice. The producer can retrieve funds as rarely or as often as they like, however, the more often funds are retrieved, the more *receipt* transactions they must send, and the more they must pay in transaction fees.

C. Trade Completion

The trade process continues until one of the following events happens:

a) The outstanding payment is equal to the remaining deposit, $S.balance$.
b) The consumer refuses to sign a receipt.
c) The producer or consumer wishes to end the trading relationship.

If the trading process is halted because of case a) above, but the consumer wishes to acquire more data, they can make another deposit on the contract through a *deposit* transaction, increasing $S.balance$ by the deposit amount in the transaction. When this occurs, trading can continue as before. Obviously, to avoid any interruption to the data flow, the consumer can use the *deposit* transaction to top up their balance at any time. If they do not wish to acquire more data, case a) is no different from case c). If cases b) or c) occur, then the trading relationship is terminated. In this case, the producer and consumer can proceed as follows:

1) The consumer sends a *deposit retrieval* transaction to signal the end of the relationship.
2) The producer sends a final *receipt* transaction to retrieve the funds they are owed, as recorded by the latest signed receipt.
3) After waiting $t_{\mathcal{D}}$ after step 1), the consumer can send a second *deposit retrieval* transaction to remove their remaining balance from the smart contract.

Note that in the case where the consumer is unresponsive, the producer can send their final *receipt* transaction before the first *deposit retrieval* transaction.

D. System Analysis

We analyze the system described above in terms of each of the objectives from section II-B, and also argue that the proposed data trading system adheres to them all.

Objective 1 (*Consumer Fairness*): The system above meets the consumer fairness objective because the consumer will never pay for data they did not receive. The only way for the producer to retrieve funds from the deposit on the smart contract is to produce a valid receipt for those funds signed by the consumer. Therefore, for the consumer to pay for data they did not receive, the producer would have to possess a receipt signed by the consumer acknowledging receipt of data that they did not receive. Of course the consumer would never sign such a receipt.

Objective 2 (*Producer Fairness*): The system above meets the producer fairness objective because the producer will never lose more than ϵ worth of data, and ϵ can be set to be arbitrarily low. For the producer to not be paid some amount, either the latest receipt returned to the producer has an $r.sent$ value less than the total value of data sent, or $S.balance$ is insufficient to cover the outstanding payment owed to the producer.

In the first case, the producer must have sent the consumer data without receiving a receipt in return. However, the producer will only ever send up to ϵ worth data without receiving a receipt, and so their loss is limited to ϵ.

In the second case, either the producer sent more data than the consumer had deposited on the smart contract, which they will not do if they follow the system guidelines, or the consumer managed to retrieve their deposit before the producer was able to send a *receipt* transaction. However this is not possible as the consumer must wait at least $t_{\mathcal{D}}$ blocks before retrieving funds from the smart contract, and the producer is able to submit a transaction to the chain within t_{tx} blocks of the consumer's transaction, where $t_{tx} < t_{\mathcal{D}}$. As such, the consumer would not be able to retrieve their deposit without the producer first having an opportunity to retrieve the funds they are owed through a *receipt* transaction.

It is possible that after receiving ϵ worth of data, the consumer will simply retrieve their funds without ever signing a receipt and create a new identity in order to repeat the attack. We argue that this is not a likely scenario, because in order to do this, the consumer would have to send both an *initial deposit* transaction and two *deposit retrieval* transactions. The minimum cost of the fees for these transactions would be $3 \cdot f_{\min}$, where $f_{\min}$ is the minimum possible transaction fee for a contract call. Therefore, as long as $\epsilon \leq 3 \cdot f_{\min}$, this attack would not be in the interest of the consumer, presuming they are interested in more than ϵ worth of data. In cases where they only require ϵ worth of data, this attack would be in their interest, however in this case the loss to the producer is still limited to ϵ.

Objective 3 (*Privacy*): We argue the system meets the privacy objective because the only data visible on chain are deposits and value transfers. No data specifying the price of data, the amount of data sent, or the type of data are visible on chain, except what can be deduced from the total value paid for the data. Therefore, this effectively limits the information visible to the public.

Objective 4 (*Cost*): We argue the system meets the cost objective because it limits the ongoing transaction fees paid by the consumer to those incurred from sending *deposit* transactions. Therefore, given the average of the transaction fees f which are charged to the consumer over a time interval, the consumer's spending per interval D, and their preferred deposit size d, we can calculate the average fees incurred per time interval. We call this the consumer fee and define it as:

$$F = f \cdot \frac{D}{d}.$$

Clearly, the consumer can reduce these fees by increasing d, and so the fees can be set to be as low as the cost of a single *initial deposit* transaction. Likewise, the producer's only ongoing costs are those incurred from running *receipt* transactions. As they decide how often they wish to receive funds from the smart contract, their fees can also be reduced to be as low as the cost of a single *receipt* transaction.

Compared with [12], our system provides significant improvements. In that work, the producer and consumer must come to an agreement on the batch size, *i.e.* the amount of data to be sent between payments. Until the producer gains trust in the consumer, they run significant risk of losing this data, incentivizing them to make the batch size as small as possible. However, when transaction fees are high, this can have a significant impact on the consumer's costs. By adding binding, off-chain transactions, our system allows the producer to limit their risk to an arbitrarily low amount, while passing minimal costs and no additional risk to the consumer. This also improves the speed of transfers, because payment to the producer is confirmed as soon as they receive the receipt signed by the consumer. Finally, our system has privacy advantages as less data is publicly available on chain.

IV. A Credit Scheme

We showed that the consumer's fees could be set arbitrarily low by increasing their deposit size. However there may be some circumstances where the consumer is unable or unwilling to increase their deposit d. For this situation, we propose a *credit mechanism* which allows the consumer to lower their fees as they spend more time purchasing data from the producer. This has the potential to benefit both the producer and the consumer. However, by giving credit the producer trades-off on risk, thus increasing their potential loss.

A. Effect of Credit

To reduce the fees paid by the consumer, the producer may consider giving them a credit, c. With this credit, the producer can continue to send data until the outstanding payment is equal to $S.balance + c$. Therefore, with this credit, and while still only having d as a deposit on the smart contract, the consumer fee over a given time interval can be reduced to:

$$F_c = f \cdot \frac{D}{d+c},$$

where f is the average of the fees charged to the consumer over the interval. fig. 2 shows the impact of a credit on the consumer fee F_c. More specifically, this graph shows that the reduction of fees achieved by increasing c is most significant when c is small. Additionally, observe that for a fixed c, the largest effect is realized when f is high and when the ratio $\frac{D}{d}$ is large, as expected.

For $f, d > 0$, we show that both the consumer and producer are incentivized to use the credit scheme. We define the *fee savings* s to be the difference between the consumer fee and the credit consumer fee. Then we have:

$$\mathsf{s} = F - F_c = f \cdot D \cdot \left(\frac{1}{d} - \frac{1}{d+c}\right) = f \cdot \frac{D}{d} \cdot \frac{c}{d+c}.$$

Observe that s is increasing in c, which implies that we get higher fee savings if we use the credit scheme with high values of credit. To incentivize both consumers and producers to use this scheme, we suggest that s is distributed between the producer and consumer. In our analysis, without loss of generality, we assume that s is evenly split between the two.

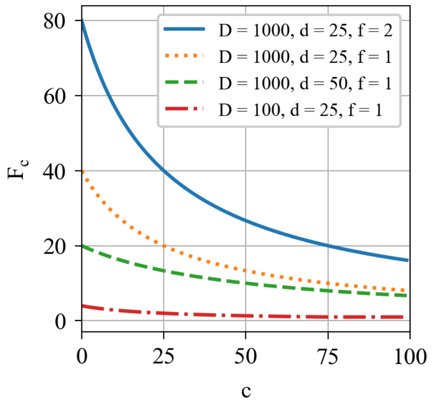

Fig. 2: Effect of credit on fees.

The producer's share of s can be periodically paid by the consumer through the same receipt method used to pay for the data itself.

B. Credit Risk

By giving credit, the producer risks losing up to c worth of data. Therefore, we can also refer to c as the *cash out value* V_{co}, where $V_{co} = c$. This is the value gained by the consumer and lost by the producer in the case where the consumer decides to terminate their trading relationship. The credit has another possible value, that is the potential savings for each one of the producer and the consumer which are realized when both use the credit scheme over an estimated number of time intervals $M \geq 0$. We call this the *savings value* V_s and it is defined by $V_{\mathsf{s}} = \frac{\mathsf{s}}{2} \cdot M$. Note M is an estimated (by the producer) number of time intervals that the consumer remains on the platform and purchases D data per interval. We determine how the producer estimates M in Section IV-C.

Consider an estimation for M and a universal lower bound on the fees $f_{\min}$ charged at each interval $m \in [1, \ldots, M]$. Then we give an upper bound to the credit value that a producer can allow to minimise their risk. This upper bound is derived from our proposal that the minimum savings value V_{s} of the producer (which equals the savings value of the consumer in our model design) must be higher than the cash out value of the credit. Then we get:

$$V_{co} < V_{\mathsf{s}} \Leftrightarrow c < f_{\min} \cdot \frac{D}{2 \cdot d} \cdot M - d. \tag{1}$$

There is another factor limiting c. As the consumer can profit V_{co} by "cashing out" the credit, we set V_{co} to be less than the minimum cost of obtaining the credit. Observe that in any other case, the consumer is either unincentivized to build credit or incentivized to build credit in order to "cash it out" at first opportunity, start with a new identity, and iterate this process. Therefore, the minimum cost of obtaining credit must be greater than its "cash out" value.

We define as $\mathcal{C}_c$ the cost of obtaining credit (for the first time), which is given by the difference between the fees paid by a user since joining the system, before receiving a credit value, and the fees that would have been paid by the user had they been given this credit during that time. Let M_p be the number of time intervals since a consumer entered the system purchasing D worth of data with a $d > 0$ deposit and let s^m

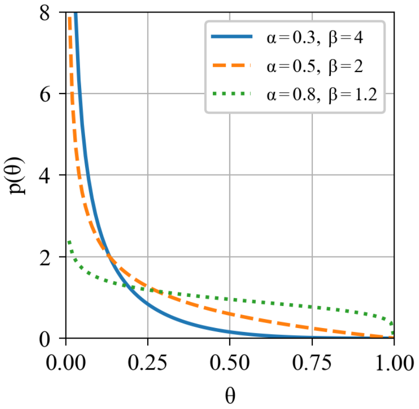

Fig. 3: Examples of various beta distributions.

be the (total) fee savings that would have been realized by the consumer at interval $m \in [1, \ldots, M_p]$ if they were given credit c in that interval. Then, the cost to the consumer of obtaining credit c is equal to:

$$\begin{aligned}\mathcal{C}_c = \sum_{m=1}^{M_p} \frac{\mathbf{s}^m}{2} &= \sum_{m=1}^{M_p} \frac{1}{2} \cdot f^m \cdot \frac{D}{d} \cdot \frac{c}{d+c} \\ &\geq \frac{1}{2} \cdot f_{\min} \cdot \frac{D}{d} \cdot \frac{c}{d+c} \cdot M_p, \end{aligned} \tag{2}$$

which expresses their minimum possible cost of obtaining a credit of value c. By the desirable property of our model design $\min_f \mathcal{C}_c > V_{co}$ and inequality (2), we get that:

$$\begin{aligned} &\frac{1}{2} \cdot f_{\min} \cdot \frac{D}{d} \cdot \frac{c}{d+c} \cdot M_p > c \\ \Leftrightarrow\ & f_{\min} \cdot \frac{D}{2 \cdot d} \cdot M_p - d > c. \end{aligned} \tag{3}$$

From inequalities (1) and (3), we show that a producer can minimize their potential loss of giving out credit when the credit offer to the consumer does not exceed the value:

$$f_{\min} \cdot \frac{D}{2 \cdot d} \cdot \min\{M, M_p\} - d - \epsilon,$$

where ϵ a small positive number. Note that the upper bound (3) depends on the time length that a user has been a data consumer on the system, while (1) depends on the estimated time that a current consumer would remain as a consumer on the system. Observe that bound (1) becomes effective when the estimated additional time period is less or equal to the (known) time period since the consumer joined.

C. Determining Credit

To determine the credit to give to the consumer, the producer must obtain an estimate for how many intervals the consumer will continue to purchase D data. We argue for this estimate to be made using a Bayesian model.

In this approach we model the consumer as performing a Bernoulli trial at the start of each interval in order to decide whether to continue purchasing data or leave the system. The trial returns true with probability θ, and if it does, they will participate for another interval. If the trial returns failure, they will drop off of the system and cash out their credit.

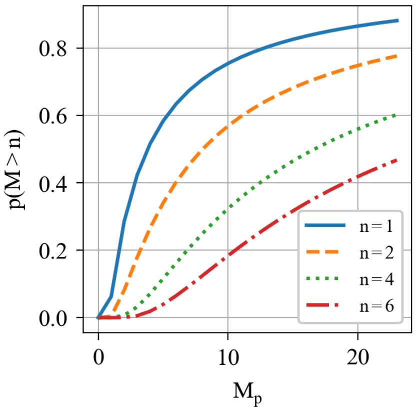

Fig. 4: Graph of how the probability of a consumer staying for n periods of time grows the longer they remain on the platform. Calculated with a beta prior, where $\alpha = 0.2$, $\beta = 4$.

Our goal is to estimate θ based on the number of intervals the consumer has been continuously purchasing data, M_p. To do this, we first define a prior distribution, $p(\theta)$ which defines the prior likelihood of each possible θ value. There are certain features that we want to have in this distribution. We must ensure that new users are not given any credit. Therefore $p(\theta)$ must assign its highest probability at $\theta = 0$. Likewise, since no user will stay on the system indefinitely, the prior should assign $p(1) = 0$.

One distribution that is able to model these features is the beta distribution, defined such that:

$$p(\theta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}.$$

In the above, parameters $\alpha > 0$ and $\beta > 0$ are set by the producer. The desired constraints can be achieved by setting $\alpha < 1$, and $\beta > 1$. Beyond this, α and β can be further tuned by the producer to better model observed consumer behaviour. Fig. 3 shows some examples of beta distributions that meet our requirements. As seen in that figure, lower α values assign higher probabilities to low values of θ, and similarly, lower β values assign higher probabilities to higher values of θ.

Given the prior, $p(\theta)$ and M_p for the consumer, the producer can produce an estimate for θ by choosing it to maximize the likelihood of the observation. To maximize the likelihood of the observation, they only need solve the below:

$$\theta^* = \max_\theta p(\theta) \cdot p(M_p|\theta) = \max_\theta p(\theta) \cdot \theta^{M_p},$$

where M_p is the number of intervals the consumer has been purchasing data. With this estimate of θ, the producer can estimate the probability of the consumer staying at least n intervals to be θ^n. Based on these estimates and their personal tolerance for risk, the producer can determine a final M value to use in their credit calculation.

Fig. 4 shows how the estimate for the likelihood of a consumer staying n intervals grows the longer they remain part of the system, for various values of n. As expected, the estimates are lower for higher values of n, and they grow the longer the user is on the system.

V. Performance Evaluation

In this section, we present a quantitative evaluation of our proposed solution with regard to the performance of the data trading system on a blockchain testbed and the performance of the credit system on a data marketplace simulation.

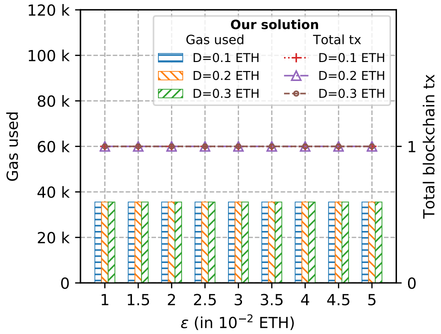

Fig. 5: Total of gas consumption and incurred blockchain transactions by the consumer for trading scheme using our proposed solution.

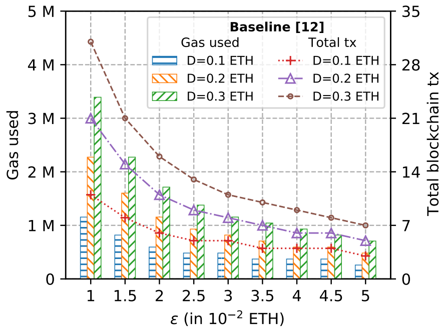

Fig. 6: Total of gas consumption and incurred blockchain transactions by the consumer for trading scheme on baseline [12].

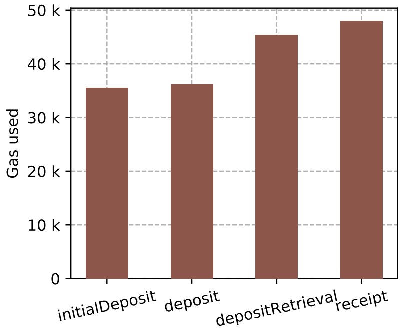

Fig. 7: Comparison of gas consumption for each core transaction of our smart contract implementation.

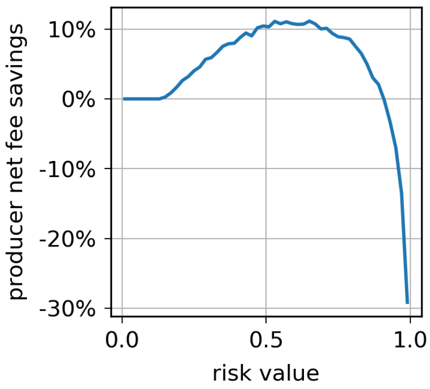

Fig. 8: Effect of risk on producer fee savings

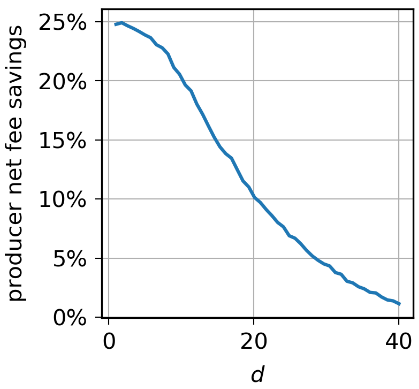

Fig. 9: Effect of d on producer fee savings

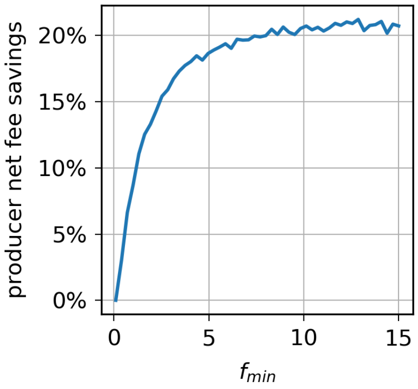

Fig. 10: Effect of $f_{\min}$ on producer fee savings, where $f_{\max} = 2f_{\min}$

A. Evaluation of the Blockchain Performance

We evaluate our trading model on a private Ethereum blockchain on a lab-scale testbed consisting of 15 Raspberry Pis, which mimic IoT nodes and a MacBook Pro that serves as a miner. All Raspberry Pis and the MacBook Pro computer run `geth` v1.9 as an Ethereum client to form the blockchain network. We simulate interactions between producers and consumers using programming scripts written in bash and Python v3.7.7. with `web3py` v5.13 and `py-solc` v3.2.0 library for Ethereum connectivity and smart contracts compilation, respectively.

We execute a trade scenario wherein a consumer is willing to purchase D worth of data from a producer over a period of time. Both the consumer and producer agree to set a certain value for ϵ, which defines the amount of data that may be sent before a signed receipt is returned. We presume that the consumer is honest and adheres to the trading mechanism by always returning a valid signed receipt upon receiving the data. As a baseline, we compare our solution against the trading mechanism in [12], in which the consumers are obliged to track each batch of data sent by the producer by logging transactions onto the blockchain. We measure the resulting overall gas consumption and the total amount of stored transaction on the blockchain by the consumers. We vary the value of D and ϵ to observe the resulting effects in gas consumption and we plot the results in Figs. 5 and 6.

As seen in Fig. 5, our proposed solution incurs a constant gas consumption and required number of blockchain transactions by the consumer for different values of D and ϵ. Recall from section III that our solution only requires the consumer to deposit the funds at the beginning of the trade setup and does not require both of the producer and consumer to submit additional transactions to log the data transfer process, as the data transfer logging occurs off chain. Note that this mechanism significantly reduces the resulting blockchain fees and stored transactions. On the other hand, different values of D and ϵ affect the overall gas consumption and incurred blockchain transactions for the baseline trade mechanism [12]. As shown in Fig. 6, the overhead costs and required transactions are proportional to D, while inversely proportional to ϵ, as a smaller ϵ value would require more frequent payments by the consumer. However, even with a larger ϵ (in which case the producer takes a larger risk), our solution requires fewer transactions and has a lower total gas cost for the consumer.

In Fig. 7, we also plot the required gas to execute the core transactions of our trade process. We can see that both transactions for the consumer to deposit some funds to the smart contract, i.e., `initialDeposit` and `deposit`, incur relatively similar gas usage of approximately $35,000$ units of gas. On the other hand, the amount of gas used for `depositRetrieval` and `receipt` transactions are relatively higher than funds deposit transactions, with `receipt` being slightly higher as a signature checking is required. Note that the gas usage in Ethereum depends on the number and types of EVM opcodes executed when running a particular transaction [24].

B. Evaluation of the Credit System

To measure the performance of the credit system described in section IV, we use a simulation of a data marketplace. In the simulation, 200 consumers are purchasing data from a producer over the course of 20 time intervals. The system's transaction fees are bounded from above and below by $f_{\max}$ and $f_{\min}$. Each consumer is assigned a value θ, sampled from the uniform distribution over the interval $(0, 1)$. Before each time interval, each consumer runs a Bernoulli trial. The trial returns true with probability θ. If the trial returns true, the consumer participates for another interval, if it returns false, the consumer leaves the system and cashes out any credit they have been given. If the consumer participates in a time interval, they will purchase D worth of data. Additionally, each consumer is willing to deposit d worth of funds at a time to purchase the data.

The producer determines how much credit to give to each consumer using the system described in section IV-C. The producer must decide on a risk value, which measures the percent chance they are willing to take that a consumer will remain long enough to make giving them credit pay off. They also need to decide on parameters α and β for their beta prior distribution. In our simulation, the producer uses $\alpha = 0.5$ and $\beta = 2.0$.

Some results from the simulation (averaged over multiple runs) are shown in Figs. 8, 9, and 10. In Fig. 8 it can be seen that the producer can maximize savings when a moderate level of risk is taken. When the producer takes on too much risk, they lose money as consumers are not staying long enough to pay back their credit. However, if too little risk is taken, the producer will not give any credit and not see any savings. In Fig. 9 it can be seen that the savings realized by giving credit decrease as the consumer's deposit size increases. This is because as the consumer increases their deposit size, their fees decrease, and so more credit is needed to have an impact. In Fig. 10 it is seen that as fees increase, so does the percentage of those fees that can be saved by giving credit.

Under the right circumstances, the credit system can provide significant fee savings, on top of those already achieved by the data trading system. It is especially notable that the percent savings increase as fees do, up to over 20%. This means that if fees rise the credit mechanism not only becomes more needed, but also more effective.

VI. Conclusion

This paper proposes a blockchain based IoT data trading mechanism that aims to minimize the number of required on-chain transactions and overhead blockchain costs, while still being able to provide secure payment mechanisms for real time data transfers. A mechanism is presented for the safe trading of funds and data off chain, secured by cryptographic signatures and on-chain deposits to a smart contract. Later, a credit mechanism is introduced to further lower the fees incurred during the data trading process, bringing benefits to both the producer and consumer. Simulation results indicate that the proposed marketplace mechanism incurs a constant overhead cost and requires limited transactions. The credit mechanism is also able to reduce the incurred fees by up to an additional 20%. Some interesting directions for future work include further validation of the proposed solution in a real world data-trading ecosystem, including more in-depth evaluation of the scalability and latency analysis; the development of methods to validate the origin and validity of IoT data; the application of the system to different data and pricing models; game theoretic analysis of the credit scheme; and the development of an additional trust mechanism to deliver a more robust and rigorous credit scheme.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] B. Ahlgren, M. Hidell, and E. C. . Ngai, "Internet of things for smart cities: Interoperability and open data," *IEEE Internet Computing*, vol. 20, no. 6, pp. 52–56, 2016.

[3] M. B. Yassein, M. Q. Shatnawi, S. Aljwarneh, and R. Al-Hatmi, "Internet of things: Survey and open issues of mqtt protocol," in *2017 International Conference on Engineering MIS (ICEMIS)*, 2017, pp. 1–6.

[4] F. Stahl, F. Schomm, and G. Vossen, "Marketplaces for data: An initial survey," *ERCIS Working Paper*, vol. Working Papers, European Research Center for Information Systems, 07 2012.

[5] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, 2014.

[6] Z. Voell and W. Foxley, "Decentralized finance frenzy drives ethereum transaction fees to all-time highs," Aug 2020. [Online]. Available: https://www.coindesk.com/decentralized-finance-frenzy-drives-ethereum-transaction-fees-to-all-time-highs

[7] D. Wörner and T. von Bomhard, "When your sensor earns money: Exchanging data for cash with bitcoin," in *2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 09 2014, pp. 295–298.

[8] A. Asgaonkar and B. Krishnamachari, "Solving the buyer and seller's dilemma: A dual-deposit escrow smart contract for provably cheat-proof delivery and payment for a digital good without a trusted mediator," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019, pp. 262–267.

[9] K. R. Özyilmaz, M. Doğan, and A. Yurdakul, "Idmob: Iot data marketplace on blockchain," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, 2018, pp. 11–19.

[10] G. S. Ramachandran, R. Radhakrishnan, and B. Krishnamachari, "Towards a decentralized data marketplace for smart cities," in *2018 IEEE International Smart Cities Conference (ISC2)*, 2018, pp. 1–8.

[11] R. Radhakrishnan, G. S. Ramachandran, and B. Krishnamachari, "Sdpp: Streaming data payment protocol for data economy," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019, pp. 17–18.

[12] S. Bajoudah, C. Dong, and P. Missier, "Toward a Decentralized, Trust-Less Marketplace for Brokered IoT Data Trading Using Blockchain," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 339–346.

[13] W. Badreddine, K. Zhang, and C. Talhi, "Monetization using blockchains for iot data marketplace," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020, pp. 1–9.

[14] S. Gheitanchi, "Gamified service exchange platform on blockchain for iot business agility," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020, pp. 1–3.

[15] R. A. Michelin, N. Ahmed, S. S. Kanhere, A. Seneviratne, and S. Jha, "Leveraging lightweight blockchain to establish data integrity for surveillance cameras," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020, pp. 1–3.

[16] C. Oham, R. Jurdak, S. S. Kanhere, A. Dorri, and S. Jha, "B-fica: Blockchain based framework for auto-insurance claim and adjudication," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1171–1180.

[17] S. Mercan, M. Cebe, E. Tekiner, K. Akkaya, M. Chang, and S. Uluagac, "A cost-efficient iot forensics framework with blockchain," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020, pp. 1–5.

[18] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," Jan 2016. [Online]. Available: https://lightning.network/lightning-network-paper.pdf

[19] J.-S. Park, T.-Y. Youn, H.-B. Kim, K.-H. Rhee, and S.-U. Shin, "Smart contract-based review system for an iot data marketplace," *Sensors*, vol. 18, p. 3577, 10 2018.

[20] G. D. Putra, V. Dedeoglu, S. S. Kanhere, and R. Jurdak, "Trust management in decentralized iot access control system," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020, pp. 1–9.

[21] M. Luecking, C. Fries, R. Lamberti, and W. Stork, "Decentralized identity and trust management framework for internet of things," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020, pp. 1–9.

[22] P. Gupta, V. Dedeoglu, K. Najeebullah, S. S. Kanhere, and R. Jurdak, "Energy-aware demand selection and allocation for real-time iot data trading," in *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2020, pp. 138–147.

[23] M. Wohrer and U. Zdun, "Smart contracts: security patterns in the ethereum ecosystem and solidity," in *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, 2018, pp. 2–8.

[24] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.