Incentivizing global crowdsensing with mediated Bitcoin micropayment channels

Abstract—Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Index Terms—Computer Society, IEEE, IEEEtran, journal, LaTeX, paper, template.

1 Introduction

MOBILE phones have an unprecedented scale in the diffusion of technology. It is the first computing technology that reaches even the rural and remote areas in the developing world. As of 2014, there are 3.6bn unique subscribers accounting for 50% of the world population. The total number of cellular connections without machine-to-machine connections was 7.3bn. Of those, 2.6bn connections originate from smartphones and the number is growing rapidly. Smartphones are powerful connected mobile general purpose computing devices that can be considered as the largest mobile sensing platform in existence. Essentially the same base technology is now being deployed in all kinds of other devices from cars to lightbulbs, extending the scope of sensing services even further. However thus far, the emerging opportunity of a planetary nervous system [1] has not yet come to scale. Mobile crowdsensing, as will be defined in Section 2.1, has only been used either in small scale scenarios or in application-specific scenarios. A global platform for crowdsensing is missing. In this paper, we argue for the need of a decentralized platform with limited power and the need for control and incentives for the end user. A fundamental building block of a platform, either centralized or decentralized, is a global, permissionless payment system that reaches as far as the diffusion of smartphones, and is capable of providing micropayments to remunerate ad-hoc sensing services. We argue that Bitcoin is such a global payment network. Although Bitcoin is based on a heavily replicated, shared ledger on top of a gossip peer-to-peer network, and an energy and capital intensive distributed consensus protocol using proof-of-work, we present technologies that leverage the built-in scripting language to enable peer-to-peer and mediated micropayments on the order of $\$10^{-6}$. In particular, we present an implementation of mediated unidirectional payment channels based on hash timelocked contracts. We discuss the merits and challenges of this approach to incentivize crowdsensing and provide an overview of the technological advancements in this space.

2 Background

2.1 Crowdsensing and Sensing-as-Service

Crowdsensing is where individuals with sensing and computing devices collectively share data and extract information to measure and map phenomena of common interest [2]. Predominately, crowdsensing has been engaged in the form of mobile crowdsensing, utilizing the ubiquity of smartphones, in application-specific scenarios. Examples are transit tracking [3], road and traffic monitoring [4], site characterization [5], and on-street parking [6], [7]. Besides these small-scale academic field studies, notable examples for successful large-scale, application specific mobile crowdsensing application are Google Maps and Waze which was acquired by Google in 2013.

[8] provide an extensive recent review of the various mobile crowdsensing paradigms, their applications, as well challenges and opportunities. The main challenges that have been identified repeatedly (see also [9]) are privacy and incentives. [10] provides an analysis of the privacy implications and threats as well as a survey of available privacy-preserving techniques.

In the application-specific scenario participants are typically incentivized by receiving an application-specific service. In the case of Google Maps a participant gets routing and traffic information in exchange for providing location and speed information. Especially in the case of participatory sensing where explicit user input is required gamification [11] has been used successfully (e.g. Waze) to incentivize participation. However incentivization by service provision leads to contribution only from users who are in need of that particular service and only during specific times. In contrast monetary incentives are a general purpose incentive mechanism. Thus far monetary incentives have mostly studied in small-scale, localized field studies, as well as in form of game-theoretic incentive mechanism design [12]. Related to crowdsensing there is also the paradigm of sensing as a service [13], [14] which emphasizes the ad-hoc access to sensing infrastructure or real-time data without investing in infrastructure itself. The sensing as a service model can be seen as an extension of the crowdsensing model to incorporate commercial sensor networks.

In recent years a multitude of architectures enabling general purpose crowdsensing and sensing as a service applications have been presented [1], [15], [16], [17], [18]. However none of them has reached reasonable scale.

What has been mainly neglected is how monetary incentives can be provided efficiently under the conditions of a global mobile crowdsensing platform. Individual rewards are rather small and data requesters as well as data providers might be distributed globally. Furthermore traditional online payment mechanisms provide additional means for de-anonymization of participants.

2.2 Bitcoin

Bitcoin is the first digital currency that does not rely on a central authority to prevent double spending. Instead the ledger of which ownership of coins can be derived is replicated across all network participants. Identities are pseudonymous and are based on public key cryptography. Hence new identities can be created locally without any permission. This means in particular that everyone and everything can accept bitcoins anywhere in the world. Transactions which update the state of the shared ledger, in order to transfer ownership of coins, can be submitted to the network by anybody but have to provide digital signatures to authenticate ownership of the respective coins. Each node validates the transaction locally, keeps it in memory, and relays it to connected nodes. Consensus on the new state of the ledger is achieved by a novel consensus process now called Nakamoto consensus. Consensus in a distributed system where adversaries can create new identities at no cost is an unsolved problem. In Nakamoto consensus nodes accept a state change based on proof of work, a computational puzzle based on calculating hashes until a particular criteria is met. Hence, updating the ledger involves the expenditure of physical resources in order to prevent Sybil attacks [19]. Today, only a small ratio of the nodes, called miners, are grouping the collected transactions into an authenticated data structure [20] called a block[1] and attempt to solve the computation puzzle to propose a state change. A miner is allowed to add a special transaction to the block, the coinbase transaction. The coinbase transaction allows to capture transaction fees that may be attached to individual transactions, and allows to create a limited number of new coins. Hence, the mining process provides monetary incentives to provide state changes that will be accepted by the other network participants and solves the problem of how to distribute coins at the same time.

When a new node enters the network it does not just request the current state of the shared ledger but requests the whole sequence of blocks beginning with the genesis block. Each block references its predecessor by means of a hash pointer leading to an authenticated data structure. If any transaction in any block is tampered with the chain of hashes breaks and thus tampering can be verified. The *blockchain* that involves the largest amount of accumulated proof of work is thus identified as the state of the system.

At each point in time every miner is expending resources trying to solve the computational puzzle which references the current tip of the blockchain. Neglecting alternative mining strategies, a miner who solved the puzzle will broadcast the block across the network and start the process anew. Each node in the network that gets the block will verify the proof of work and update its state accordingly. If the node is a miner she will dismiss her current block and start again based on the new state. Since the speed of information propagation in a global distributed network is finite it happens that temporarily different subsets of the network have a different view on the state. This leads to the requirement that the expected time between two solutions of the computational puzzle should be larger than the time it takes a block to propagate across the network. This requirement was met by Nakamoto by making particular parameter choices. First, the difficulty of the computational puzzle is adapted every 2016 blocks such that the expected block generation rate is 10 minutes based on the average rate provided by those last 2016 blocks. Second, the maximal size of a block is limited to 1 MB. Furthermore, Nakamoto suggests that a user should only accept a transaction to be final if it is in a block with a depth of at least six. Consequently, the Bitcoin system currently allows for a maximum transaction rate of 7 transactions per second and payment *finality*[2] takes at least 60 minutes. Moreover, as space in blocks gets scare transactions have to provide higher fees to be included by the miners. One of the parameters, the block size limit, will probably be increased rather soon but there is always a trade off and changes in consensus-critical code in a distributed system such as Bitcoin are not without risk [22].

Decentralized systems are inferior to centralized systems in terms of performance and efficiency. However, decentralized systems provide resilience against attacks from inside and outside. All other global digital currency systems that have existed so far have not survived. A decentralized system that is based on peer-to-peer technology in contrast will live as long as the participants perceive it as valuable. Moreover, although the last paragraph might have created the impression that Bitcoin is ill-suited for the task at hand, i.e. to incentivize low-value sensing task, we will see that Bitcoin is able to provide a base layer on which instant micropayment systems can be built on top.

3 Instant Bitcoin Micropayments

We have seen that Bitcoin and other cryptocurrencies based on current permissionless blockchain technology face inherent scalability issues due to heavy replication as well as the proof-of-work based consensus and security mechanism. Another consequence are non-negligible transaction fees for low-value transactions. In particular since transaction fees are tied to the size of transaction in bytes rather than to the transfered amount of bitcoins. In the following, we present approaches that enable instant Bitcoin micropayments despite those underlying conditions.

3.1 Off-chain payments

An obvious strategy to enable instant micropayments is to use a centralized service to provide managed accounts for

1. More precisely, the authenticated data strucutre is a Merkle hash tree [21] of transaction hashes.

2. There remains always a risk of blockchain reorganization. However the probability quickly diminishes with the increasing depth of a block.

participants. After users deposit funds in their accounts a payment between users of the system involves just a change in an internal database. Thus, payments can be confirmed practically instantly and transaction costs are (can be) vanishingly small. However, in this model users have to trust a centralized service. Not only in the honesty of the service but even more so in their security against attacks. Examples of such a model are Coinbase and Changetip but also Mt. Gox which lost 850000 bitcoins (worth $450 million at that time) most probably because of a long lasting security breach.

3.2 One-to-one unidirectional payment channels

Another approach that conserves the peer-to-peer nature of Bitcoin are One-to-one payment channels which were first introduced by [?]. To understand payment channels we have to take a deeper look at the structure of a Bitcoin transaction. Bitcoin is not build on an account model but on an *UTXO* model. UTXO stands for unspent transaction output and the set of UTXOs represents the current state of Bitcoin. A transaction accordingly attempts to spend one or more UTXOs and creates one or more UTXOs. Thus, a transaction entails inputs which reference outputs of earlier transactions and a field called *sigScript*, as well as outputs which entail the value assigned and a *pubScript* field. In a simple Bitcoin transaction the pubScript demands a signature belonging to a particular Bitcoin address or public key and a sigScript of the input attempting to spend this output has to provide this signature. However, more generally, the entries of referencing inputs and outputs get concatenated and evaluated by a stack-based interpreter which implements a number of cryptographic primitives. A transaction is valid if the sum of the value assigned to the outputs does not exceed the sum of the value in the inputs, and if the scripts evaluate to *true*. Thus the conditions to spend an output can be complex any many useful applications have been found. A simple and very useful challenge script demands more than one signature.

The idea of a payment channel is simple. Assume two parties, a payer and a payee. The payer sends a certain amount of bitcoin C to an output demanding a signature by the payer and payee. In Bitcoin this is called a 2-of-2 multisignature output. Essentially C has been locked in a shared account and can only be spent if payer and payee cooperate. This transaction is the *funding transaction* that establishes the channel. This transaction is broadcasted to the Bitcoin network. Assume the payee is a sensor that provides a datastream with a datagram every 5s and each datagram costs ϵ satoshi. The payer prepares a transaction that attempts to spend the multisignature output to split C such that the payee gets ϵ and herself gets $C - \epsilon$. She will provide her signature and send the transaction to the payee over a private or public communication channel. Due to the security of the funding transaction in the blockchain the payee can accept the payment immediately since she could sign the transaction any time and broadcast it to the Bitcoin network. Thus, she sends the datagram to the payer. If the payer wants to buy another datagram she creates a new transaction that splits C such that the payee gets 2ϵ, and herself gets $C - 2\epsilon$. Thus, once a channel is established

[redacted].

this setup enables repetitive off-chain micropayments from a payer to a payee. However, while the payee can close the channel at any time by signing and broadcasting one of the payment transactions, the payer is at the payee's mercy. In order to prevent the hostage scenario we can add a channel lifetime T (see A for details) after which the payer is able to retrieve the total amount C.

Essentially we have seen how the Bitcoin blockchain provides a trustless escrow service that enables two parties to transact repeatedly with each other using off-chain transactions. Payment channels are a simple demonstration of how the built-in scripting language allows to build systems on top of the native Bitcoin payment system. However, if we think of a typical crowdsensing task, like getting barometric pressure in a specific geographic region, a buyer would have to open hundreds to thousands of payment channels. Each consuming transaction fees and locking funds.

3.3 N-to-M mediated payment channels

In order to remedy the problem of having payment channels between each payer-payee pair we introduce a central hub for payment routing. Thereby each payer and each payee have to have only one payment channel with the hub in order to trade with everyone else. In the simplest case payer-hub and hub-payee channels are independent of each other. Hence, the hub would be able to take the money from a payer without paying the payee accordingly. Although we are concerned with micropayments where the financial gain would probably not outweigh the loss of reputation and trust we will use hashed time-locked contracts (HTLCs). These HTLCs allow trustless routing of payments through intermediaries.

We assume a starting position where payer and hub, and hub and payee have established payment channels by having broadcasted funding transactions to the Bitcoin network as described in Sec. 3.2. The basic idea behind HTLCs is to connect both channels temporarily by conditioning a payment in both channels on the disclosure of a shared secret. If one payment is publicly claimed by broadcasting a transaction to the Bitcoin network the corresponding payment in the other channel can be claimed as well since the secret has been published in the blockchain. In order to initialize payment flow using a HTLC the payee generates a secret s and communicates the hash $h(s)$ to the hub and the payer. The payer then creates a *setup transaction* that spends the shared output and allocates the payment amount in a HTLC output (c.f. Listing 1), and a *settlement transaction*. The settlement transaction attempts to spend the HTLC output by satisfying the first branch of the if [redacted].

statement. Both transactions are signed and sent to the hub. In addition the hub repeats the procedure with the payee. The payment get finalized by communicating the secret and creating the *teardown transactions* which allocate the values of the HTLC outputs to the respective receivers. A diagram illustrating the process is shown in Figure 1. Under normal conditions all transactions are kept off-chain. Only if a party becomes unavailable or attempts defraud a counterparty the counterparty is able to enforce the contract by sending the appropriate transactions to the Bitcoin network. For this reason the HTLC output has three branches to be spent under different conditions. The first branch can be used by the hub if the payee reveals the secret but the payer refuses to create the *teardown transaction* which allocates the value of the HTLC output to the respective receiver. The second branch can be used by the payer to get her locked funds back if the hub does not cooperate, and the third branch allows to close the channel in agreement immediately if something went wrong.

Listing 1. Pseudocode for HTLC output (pubScript)

```
If
    Signatures A and B, and S  (time lock $\Delta$)
else if
    Signature $A'$ (time lock $\Delta$+1)
else
    Signatures A and B
```

In principle HTLCs can be used to route payments over multiple intermediaries. Together with clever tricks that enable bi-directional payments routable off-chain payment networks can be built. Currently there are two main designs: (1) The lightning network [23] and (2) Duplex Micropayment channels [24]. However the complexity is high and fully functional implementations are not yet available. McCorry et al. [25] provide a comparison between the two designs.

4 System Design

In this section we first state the main principles that lead to the system's design. Thereafter, we describe the architecture and explain the components in more detail.

4.1 Principles

4.1.1 Low barrier for participation

The usefulness of mobile crowdsensing depends primarily on the participation of a large number of data providers. Therefore, it is essential to keep barriers to participation as low as possible.

4.1.2 Incentivize participation with micropayments

For the same reason

4.1.3 Limited trust in hub

We aim for minimizing the trust in the hub provider

4.1.4 Anonymity

Neither data producers nor data buyers should need to disclose their identity.

4.2 Architecture

The system consists of three main components. A data requester or data buyer application, a data provider application, and a hub application. In order to fulfill the aforementioned principles the payment of the measurement data is done by means of N-to-M mediated payment channels as presented in Section 3.3 where the hub acts as the mediator over which payments are routed. At the same time the hub provides a query-able data provider registry. Noteworthy, the consolidation of payment routing and registry slightly violates our third design principle. However, both components are logically separated and we will discuss approaches to further decentralization later on. An illustration of the system's architecture is shown in Figure 2.

5 Prototype Implementation

The implementation of N-to-M mediated payment channels is based on extending the payment channel implementation of the bitcoinj library [26]. Bitcoinj is a Java library for working with the Bitcoin protocol and was the first Bitcoin library that had implemented support for payment channels. The Java implementation was chosen because it allows to reuse the implementation across all of the system's components. JavaScript provides the same benefits but libraries were not as mature at that point in time. Furthermore, bitcoinj implements simplified payment verification (SPV). It allows to verify on-chain transactions without needing to store and verify the entite Bitcoin blockchain. SPV nodes store only the block headers (80 byte instead of a few hundred kilo byte for a typical block). In order to validate a particular transaction, a SPV node demands the Merkle branch that entails the transaction and is thus able to verify that the transaction is in a particular block by comparing the calculated Merkle root with the one in the stored block header. Only due to the SPV approach it is viable to use Bitcoin on resource constrained devices without trusting third party services.

The data requester and the hub are implemented as native Java applications. The data provider is implemented as an Android application. Before presenting the implementation of the individual components we start with the common building block on which all peer-to-peer payments are based.

5.1 Mediated payment channels using HTLCs

We follow the client-server architecture of the payment channel implementation of bitcoinj [27]. A client is always on the sending side of a payment while a server is on the receiving side (c.f. Figure 2). Since the hub acts as mediator

Payer		Hub		Payee
				Generate s Calculate h(s)
	$\xleftarrow{h(s)}$		$\xleftarrow{h(s)}$	
Create T_{setup}, T_{settle}				
	$\xrightarrow{T_{setup},T_{settle}}$			
		Create $T^{'}_{setup}$, $T^{'}_{settle}$		
			$\xrightarrow{T^{'}_{setup},T^{'}_{settle}}$	
	$\xleftarrow{s}$		$\xleftarrow{s}$	
Create $T_{teardown}$				
	$\xrightarrow{T_{teardown}}$			
		Create $T^{'}_{teardown}$		
			$\xrightarrow{T^{'}_{teardown}}$	

Fig. 1. The flow of a mediated payment using HTLCs.

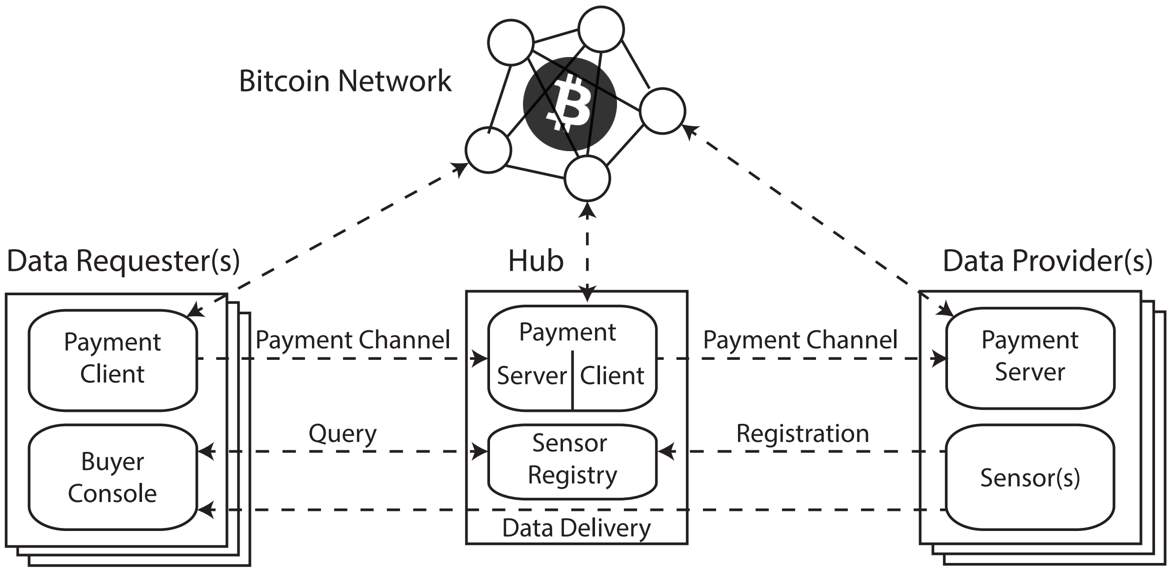

Fig. 2. Overview of system's architecture.

or router for payments between requesters and providers it has to receive payments from requesters and forward those payments to providers. Hence, the hub incorporates both, server and client.

In order to implement hashed HTLCs we extended the four layers of bitcoinj's payment channel implementation with a fifth layer that is concerned with keeping track of the HTLC flow. The five layers are described briefly in Table 1. Details about the state machines and sequence diagrams can be found in BLINDED FOR REVIEW. Messages and transactions are serialized using Google protocol buffers[3] and are exchanged between the components over TCP connections. The payment channel setup and the HTLC payments in the implementation are slightly more complex and involve even more communication than it is shown in Table 1 because we use the interactive approach for channel refunds.

TABLE 1
The five layers of the HTLC payment channel implementation.

Layer	Description
Driver Application	Provides the API to access lower layers
Channel Connection	Provides the interface to the network
Payment Channel	Handles messages from/to the network and passes instructions/gets results to/from the lower layer
Channel State Machine	Handles the state of the payment channel
HTLC State Machines	Handles the state of the HTLC

3. https://developers.google.com/protocol-buffers/

TABLE 2
Examples of virtual and physical sensors available in smartphones.

Sensor	Type	Application
Barometric pressure	physical	Weather prediction
Location	both	People flow
Network strength	physical	Coverage maps
Installed apps	virtual	Inter-app correlations
Transportation mode	virtual	City monitoring
Steps	virtual	Health monitoring

TABLE 3
Available commands of the data requester console.

Command	Description
STATS NODES	Returns all connected sensor nodes
STATS SENSORS	Returns available sensor types
SELECT SENSOR=<type>	Returns sensors of type <type> with pricing information
BUY <type> FROM <node>	Buys the current measurement value of sensor <type> from node <node>

5.2 Data provider

The data provider is implemented as an Android application. The application allows users to offer measurement data from various sensors at an adjustable price denominated in satoshis[4]. Table 2 provides an overview of possible sensors. With the initial boot of the application, a local wallet is generated. The wallet is responsible for generating and storing key pairs and signing transactions. Furthermore a background service is started that manages the payment channel server and data delivery. If a user decides to offer measurement data from a specific sensor, the application registers the sensor at the global sensor registry. The initial registration then triggers the setup of a payment channel between the hub and the Android application. As soon as the payment channel is established the application is ready to serve requests from buyers.

5.3 Data requester

The data requester is implemented as a Java application. The requester application allows to query the sensor registry. In the prototype this is achieved using simple console commands. Table 3 shows the available commands.

5.4 Hub

The hub acts as the central point of coordination. Data providers register their sensors with the sensor registry that the hub provides, and data requesters are able to query the registry. Furthermore, the hub is responsible for mediating the payments between buyers and sellers. The hub instantiates a payment channel client and a payment channel server

6 Evaluation

6.1 Conceptual Evaluation

6.1.1 *Low barrier for participation*

A potential participant on the data provider side has to download and install the data provider smartphone application. The data provider does not need to own any bitcoins. Furthermore no manual sign-up or registration is required. This means also the system has global reach and could provide a low-income stream to smartphone users in developing countries.

A data requester does not have to register but the wallet of the data requester application needs to have some amount of bitcoin in order to initiate a payment channel with the hub.

6.1.2 *Incentivize participation with micropayments*

Individual payments can be as low as 1 satoshi. In April 2015, the exchange rate is around 450\$/BTC. In regard to this exchange rate individual payments can be as low as 4.5 μ\$. Even smaller payments could be implemented with probabilistic payment schemes [28], [29]. Transaction fees for setting up and closing payment channels will be discussed in Sec. 6.4.

6.1.3 *Limited trust in hub provider*

The usage of HTLCs to interconnect payment channels of data providers and data requesters allows atomic payments between those parties. This means either the payment happens in both channels or it happens not at all.

6.1.4 *Anonymity*

Obviously there is a risk of de-anonymization for data providers depending on the data they are selling. However we restrict the evaluation of anonymity to the payment aspect. Bitcoin payments themselves are pseudonymous. Transactions only entail cryptographic public keys and hashes thereof. Thus, if a user is careful, Bitcoin provides a high level of anonymity. In addition, due to the usage of mediated payment channels, there is no public link between buyers and sellers. Only transactions between buyers and the hub, and sellers and the hub end up in the blockchain. Most of the transactions entailing the linking secret are kept private. Only if a payment channel has to be closed before its capacity is depleted a revealing transaction ends up in the Bitcoin blockchain.

In addition, there is a risk of de-anonymization based on the IP addresses. However since all communications are based on TCP, it would be possibly to run the system over Tor[5].

6.2 Empirical Evaluation

6.3 Performance

For setting up and closing of payment channels the Bitcoin network itself is dictating the performance. If fees are sufficient a transaction can be assumed to be included in the blockchain in 10 min on average. Depending on the value of the transaction the receiver of the funds might wait until the transaction has six confirmations. This means that there are five additional blocks on top of the block entailing the transaction. In the following we assume that payment channels are already in place and evaluate the time is takes to complete a payment. The experiment was done by instantiating 100 data requesters and one data provider. Each requester buys one measurement value from the data

4. A satoshi is the smallest fraction of a bitcoin and corresponds to 10^{-8} bitcoins

5. https://www.torproject.org/

TABLE 4
Payemt duration breakdown

Payment	Mean	St. Dev.
Requester-Hub setup	479.3 ms	78.8 ms
Hub-Provider setup	1015.1 ms	151.4 ms
Hub-Provider teardown	199.6 ms	68.3 ms
Buyer-Provider teardown	146.6 ms	10.1 ms

TABLE 5
Approximate sizes and costs for Bitcoin transactions

Transaction Type	Size	Cost
Standard P2PKH		
Funding		
HTLC setup		
Settlement		
Forfeiture		

provider. Table 4 shows a breakdown of the mean times and the standard deviation according to the payment steps. The results request some explanation. First, the HTLC setup process between hub and provider seems to take almost twice as long as the the HTLC setup between requester and hub. The reason for this is that the hub-provider setup is initiated with generating the secret but then has to wait until requester-hub setup is finished. If the hub would initiate the HTLC setup with the provider immediately the hub would risk paying the provider without being pull the respective funds from the requester. Second, the teardown (i.e. updating the setup transacting by assigning the value of HTLC output to the payee.) process takes less than half the time of the setup process. The reason for that is mainly that in the actual implementation the interactive refund approach was used. This means that there is an additional transaction which has to be signed by both parties and has to be transferred from payer to payee and back. Thus, we expect a comparable performance between setup and teardown if the non-interactive approach is used. Furthermore, we see that the steps that involve the provider take more time. This is because signing and signature verification are computation intensive tasks and the data provider is a comparably weak smartphone.

6.4 Transaction costs

Transaction fees in Bitcoin are in principle voluntary. The party that creates a transaction specifies the transaction fees as the difference of the sum of the value of all inputs and the sum of the value of all outputs. However, each miner can decide individually if she will accept the transaction for a candidate block. Since there is a limit on the maximum size of a block, and larger blocks need more time to propagate the network, rational miners will select transactions with higher transaction fees. This is also the reason why transaction fees are based on the size of the transaction instead of its value. The transaction size is mainly defined by the number and size of input and output scripts and signatures. Figure 3 shows the daily averages of transaction fees in USD/kb for the period of one year.

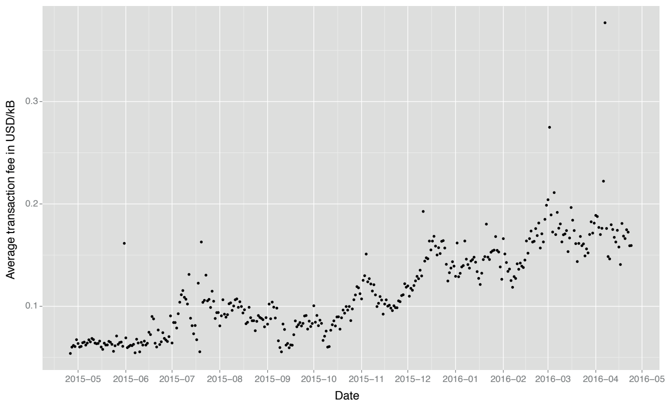

Fig. 3. Bitcoin transaction fees in USD/kb. Each point represents a daily average.

7 Discussion

7.1 Intermittent connectivity

In order for the payment mechanism to be trustless payment receivers

7.2 Locking of funds

The usage of payment channels involves locking of funds. This is a problem in particular for the payment hub. One of our principles was that the onboarding of new data producers should be frictionless. Hence, we assume that data producers do not have any bitcoins to begin with. When a data producer registers a sensor with the registry the payment hub initiates a payment channel with the data producer. As described in Sec. 3.3 this involves broadcasting a funding transaction to the Bitcoin network. Thereby the hub has to lock capital in the channel and has to provide a fee in order for the transaction to be mined. The operator has to balance between locking a large amount and the transaction costs involved in closing and reopening a channel because of depletion. Since there is no cost for a data producer to register, a malicious actor might attack the hub by registering a large amount of fake sensors. A strategy to disincentive such a behavior would be to request proof of work from the sensor as part of the registration process. In addition the locking of funds is a problem for benevolent data requesters since the earned satoshis are only usable for payments after closing of the channel.

7.3 Improving anonymity

7.4 Discovery

7.5 From smartphones to wireless sensor networks

7.5.1 Additional challenges

8 Conclusion

The conclusion goes here.

Appendix A
Refund Transactions and Time locks

There are two ways to allow the payment channel creator to retrieve her funds after a specific *locktime* if the payee is not cooperative. The first, an interactive method, has been available in Bitcoin for a long time. The second, a non-interactive method, has only been available since December 2015. The non-interactive methods using the *CHECKLOCKTIMEVERIFY* operator provides various advantages. Thus we present the payment channels in terms of this operator although our implementation is based on the interactive approach since at time of the implementation *CHECKLOCKTIMEVERIFY* was not available. For that reason we briefly present the interactive method.

Bitcoin transactions provide a data field that allows to specify a locktime in terms of a block number or an UTC timestamp. The locktime prevents a transaction to enter the blockchain before that event. This feature can be used to create a refund transaction that can spent the 2-of-2 multisignature output of the funding transaction after some time. It is important to note that the refund transaction has to be signed by the payee before the funding transaction is broadcasted to the Bitcoin network. Otherwise the payee would be able to refuse signing the refund transaction and hence the funds could be locked forever. After the payee has signed the refund transaction, the payer has to store the transaction in order to be able to retrieve the funds after the event specified with the locktime.

APPENDIX B

Appendix two text goes here.

ACKNOWLEDGMENTS

The authors would like to thank...

REFERENCES

[1] F. Giannotti, D. Pedreschi, A. Pentland, P. Lukowicz, D. Kossmann, J. Crowley, and D. Helbing, "A planetary nervous system for social mining and collective awareness," *The European Physical Journal Special Topics*, vol. 214, no. 1, pp. 49–75.

[2] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges." *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[3] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson, "Cooperative transit tracking using smart-phones," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '10. New York, NY, USA: ACM, 2010, pp. 85–98. [Online]. Available: http://doi.acm.org/10.1145/1869983.1869993

[4] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 323–336. [Online]. Available: http://doi.acm.org/10.1145/1460412.1460444

[5] Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao, "Automatically characterizing places with opportunistic crowdsensing using smartphones," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp '12. New York, NY, USA: ACM, 2012, pp. 481–490. [Online]. Available: http://doi.acm.org/10.1145/2370216.2370288

[6] X. Chen, E. Santos-Neto, and M. Ripeanu, "Crowdsourcing for on-street smart parking," in *Proceedings of the Second ACM International Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, ser. DIVANet '12. New York, NY, USA: ACM, 2012, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/2386958.2386960

[7] V. Coric and M. Gruteser, "Crowdsensing maps of on-street parking spaces," in *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, May 2013, pp. 115–122.

[8] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, R. Huang, and X. Zhou, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, p. 7, 2015.

[9] D. He, S. Chan, and M. Guizani, "Privacy and incentive mechanisms in people-centric sensing networks," *Communications Magazine, IEEE*, vol. 53, no. 10, pp. 200–206, 2015.

[10] D. Christin, "Privacy in mobile participatory sensing: Current trends and future challenges," *Journal of Systems and Software*, pp. –, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121215000692

[11] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: Defining "gamification"," in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, ser. MindTrek '11. New York, NY, USA: ACM, 2011, pp. 9–15. [Online]. Available: http://doi.acm.org/10.1145/2181037.2181040

[12] D. Yang, G. Xue, G. Fang, and J. Tang, "Incentive mechanisms for crowdsensing: Crowdsourcing with smartphones," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, pp. 1–13, 2015.

[13] X. Sheng, J. Tang, X. Xiao, and G. Xue, "Sensing as a Service: Challenges, Solutions and Future Directions," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3733–3741, Oct. 2013.

[14] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by internet of things," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp. 81–93, 2014. [Online]. Available: http://dx.doi.org/10.1002/ett.2704

[15] X. Hu, T. H. S. Chu, H. C. B. Chan, and V. C. M. Leung, "Vita: A crowdsensing-oriented mobile cyber-physical system," *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 1, pp. 148–165, June 2013.

[16] G. Cardone, L. Foschini, P. Bellavista, A. Corradi, C. Borcea, M. Talasila, and R. Curtmola, "Fostering participaction in smart cities: a geo-social crowdsensing platform," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 112–119, June 2013.

[17] N. Haderer, F. Paraiso, C. Ribeiro, P. Merle, R. Rouvoy, and L. Seinturier, *Crowdsourcing: Cloud-Based Software Development*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, ch. A Cloud-Based Infrastructure for Crowdsourcing Data from Mobile Devices, pp. 243–265. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-47011-4_13

[18] G. Merlino, S. Arkoulis, S. Distefano, C. Papagianni, A. Puliafito, and S. Papavassiliou, "Mobile crowdsensing as a service: a platform for applications on top of sensing clouds," *Future Generation Computer Systems*, vol. 56, pp. 623–639, 2016.

[19] J. R. Douceur, "The sybil attack," in *Peer-to-peer Systems*. Springer, 2002, pp. 251–260.

[20] R. Tamassia, *Algorithms - ESA 2003: 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, ch. Authenticated Data Structures, pp. 2–5. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-39658-1_2

[21] R. C. Merkle, "A certified digital signature," in *Advances in CryptologyCRYPTO89 Proceedings*. Springer, 1989, pp. 218–238.

[22] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, and E. Gün, "On scaling decentralized blockchains," in *Proc. 3rd Workshop on Bitcoin and Blockchain Research*, 2016.

[23] J. Poon and T. Dryja, "The bitcoin lightning network."

[24] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Stabilization, Safety, and Security of Distributed Systems*, ser. Lecture Notes in Computer Science, A. Pelc and A. A. Schwarzmann, Eds. Springer International Publishing, 2015, vol. 9212, pp. 3–18. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-21741-3_1

[25] P. McCorry, M. Möser, S. F. Shahandashti, and F. Hao, "Towards bitcoin payment networks."

[26] "Bitcoinj," (accessed January 11, 2016), https://bitcoinj.github.io.

[27] "Working with micropayment channels," (accessed January 11, 2016), https://bitcoinj.github.io/working-with-micropayments.

[28] R. L. Rivest, *Financial Cryptography: First International Conference, FC '97 Anguilla, British West Indies February 24–28, 1997 Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, ch. Electronic lottery tickets as micropayments, pp. 307–314. [Online]. Available: http://dx.doi.org/10.1007/3-540-63594-7_87

[29] R. Pass and a. shelat, "Micropayments for decentralized currencies," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: ACM, 2015, pp. 207–218. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813713

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.