A Smart City IoT Integrity-First Communication Protocol via an Ethereum Blockchain Light Client

Elizabeth Reilly *, Matthew Maloney *, Michael Siegel *, Gregory Falco *
*Massachusetts Institute of Technology
{reillye, maloneym, msiegel, gfalco}@mit.edu

***Abstract*—Smart city IoT is responsible for communicating system-critical data about urban infrastructure that keeps our modern cities functioning. Today, IoT devices lack communication protocols with data integrity as a priority. Without data integrity, smart city infrastructure is at risk of actuating urban environments on compromised data. Attackers can use this IoT communication flaw to wage cyber-physical attacks on Smart Cities. We designed and developed an integrity-first communication protocol for IoT that is distributed and scalable based on the Ethereum blockchain. Our light client ensures data communication integrity for systems that require it most.**

***Keywords*—** IoT Security, Blockchain, Smart Cities, Communication Integrity, Ethereum, IoT Communication Protocol

I. Introduction

Despite the prevalence and growing popularity of smart city IoT devices such as smart meters or CCTV security cameras, many of these devices lack security [9]. There are hundreds of videos dedicated to demonstrating how to hack into IoT devices in minutes [4]. The lack of communication standards and various IoT device manufacturers further complicates the security of these memory and processing power-constrained devices [8]. Our research focuses on a specific security gap for urban critical infrastructure IoT - improving communication integrity. Prioritizing integrity, what we call integrity-first communication, is important for smart city IoT because integrity failures can result in cyber-physical damages.

Because many IoT devices have different operating systems and configurations, it is hard to establish one single communication protocol that can be universally applied [6]. IoT devices with limited memory and computational resources pose an even greater concern as they often do not have the resources to host a communication protocol at all [12].

Several IoT communication protocols have been proposed that attempt to prioridize communication integrity, yet they are not scalable [13].

Given the distributed nature of IoT devices and their scale, a suitable integral communication protocol is needed. We propose using the blockchain for this purpose. Blockchain provides a scalable, distributed ledger that requires consensus across all participating nodes [14]. Once a transaction has been approved by the majority of nodes, it cannot be tampered with or erased unless an attacker has control over 51% of the nodes. Therefore, blockchain is a good candidate for IoT communication integrity, assuming the blockchain is sufficiently large.

However, hosting a blockchain node on IoT has been problematic because each node in the network must store the chain of transactions. Many IoT devices have memory and computational limits which keep them from being able to store entire chain data. Current solutions such as the tangle, IoT chain, IoTex and NeuroMesh have tried to overcome this issue by using lighter clients or hardware solutions [15][5][1][8]. However, each has limitations. We have designed a light version of the Ethereum blockchain that overcomes these outstanding issues. Our light client provides integrity-first communication for IoT devices.

II. Related Works

Today, there is no industry standard for how IoT devices should communicate securely. Many current communication protocols for IoT devices either poorly address security, or are not scalable. Researchers have only recently begun investigating blockchain as a potential IoT communication protocol. However, most IoT blockchain implementations are either too large, too centralized, too expensive or use hardware solutions.

A. Legacy Communication Protocols

Modbus was one of the original proposed communication mechanisms for IoT devices [10]. Modbus was originally designed for isolated systems so the integrity of messages was not taken into consideration [10]. DNP3 is another early communication protocol for IoT devices built upon TCP/IP. It has a master-slave structure and likewise does not provide integrity of messages and no authentication mechanism between master and slave [10]. Both of these systems do not ensure integrity-first communication and were designed for early industrial IoT.

B. Modern IoT Communication Protocols

IoT devices with limited resources are often referred to as constrained nodes. Currently, DTLS is the default security protocol used for application messages between constrained nodes [2]. DTLS is built upon the UDP protocol [12]. However, DTLS lacks scalability. Similarly, the IETF has attempted to create a standard for communication protocols for constrained nodes. One such protocol is CoAP, which is also built upon UDP and includes a subset of HTTP functions that are optimized for resource scarce environments [16]. However, similar to DTLS, this protocol does not perform well under high load or congestion [16]. Also, UDP is known for being unreliable and messages are often lost [1]. Therefore, none of the security protocols are able to achieve a distributed, scalable form of integral communication.

MQTT is another communication protocol popular with constrained nodes. It is built on TCP and IP [17]. MQTT has 3 different reliability standards called Quality of Service Levels [17]. The first level is 0, in which a message is delivered at most once and no acknowledgement of reception is required [17]. The second is level 1, in which every messaged is delivered at least once and acknowledgement of reception is required [17]. The third level is 2, in which a four-way handshake mechanism is used to deliver a message exactly once. Despite these different levels of reliability, MQTT also lacks scalability as more nodes connect to a centralized broker that must perform handshake procedures [13].

C. IoT Blockchains

Blockchain has been proven to be highly scalable but has not been largely applied to IoT devices nor explored as a source of integrity-first communication. The most well known variation of blockchain specifically designed for IoT is the tangle, which is accompanied by the IOTA coin [15]. The tangle stores the chain in a DAG structure to reduce space and also has a light variation specifically for small devices [15]. In order for nodes to get their transactions approved, they must approve 2 other transactions first [15]. This makes the transactions fee-less but also means that there are no miners in the system. Without miners, the system has less incentive to grow. Furthermore, nodes can decide for themselves which transactions to approve. This can prevent certain transactions from being approved. Therefore, a coordinator that decides the order in which transactions will be approved is currently used. This is problematic because it creates a level of centralization and single point of failure within the network [2].Also, the tangle is a relatively small blockchain variation making it vulnerable to a 51% takeover attack.

D. IoT Chain

IoT chain uses a DAG structure similar to the tangle and also uses Simplified Payment Verification (SPV) to facilitate operations on smaller devices [5]. SPV allows devices to conduct payment verification without maintaining complete blockchain information as long as block headers are preserved [5]. IoT chain also uses practical byzantine fault tolerance (PBFT) to ensure fast consensus [5]. PBFT is a state machine replication algorithm that is based on the consistency of message passing [5]. The combination of the DAG structure and PBFT allow transaction times for IoT chain to be milliseconds [5]. Despite these benefits, the primary limitation of IoT chain is that it requires a specific operating system and linking module [5]. This makes it a hardware solution which is more difficult to integrate with older devices. Also, the relatively small size of IoT chain makes it vulnerable to a 51% attack.

E. IoTex

IoTex similarly uses PBFT and SPV to ensure fast transaction times and limited storage space [1]. The key concept for IoTex is the idea of blockchains within blockchains [1]. Essentially, different blockchains are used for different kinds of IoT devices [1]. This is done because different devices have different features and by separating these features into different blockchains, no one device has to store large chains [1]. For example, one chain might record transactions and another one might have turing-complete contracts on it [1]. There is a root blockchain that manages all the blockchains [1]. However, this root blockchain is problematic because it introduces a layer of centralization. Like the tangle and IoT chain, the small size of IoTex makes it vulnerable to a 51% attack.

F. NeuroMesh

One blockchain that successfully provides secure communication for IoT devices, which is most similar to our design, is NeuroMesh [8]. NeuroMesh functions as a "friendly" botnet to fight against other botnets and communicates security commands to IoT devices using the Bitcoin blockchain as the communication protocol [8]. While this technology does provide integral communication because of the size of the Bitcoin network (minimizing the 51% attack risk), it has several operational constraints. First, NeuroMesh is only able to send 80 bytes of characters at a time [8]. This might be fine for sending security commands to IoT devices, but is insufficient to handle substantial data transfers. Second, the Bitcoin network is slow and expensive compared with Ethereum [18]. Finally, while NeuroMesh is only 1MB, it uses SPV which must store block headers [8][14]. This limits how small it can become.

III. Design Parameters

For our light client, we focused on allowing constrained nodes to participate in global blockchain networks without needing to host an entire node. There were many parameters we had to consider in this light design. First, we needed to select an implementation of blockchain to use. There are many different implementations of blockchain, including both Bitcoin and Ethereum [18][14]. We initially considered creating our own blockchain, however, small blockchain networks are often vulnerable to 51% attacks. Also, we wanted to choose a blockchain implementation that was fast, inexpensive and reliable. Therefore, we decided on Ethereum.

Next, we had to find a way to scale down the size of Ethereum, both in terms of code size and chain data. We chose not to store the chain data at all and to reduce the code size using compiler tricks as well as manual stripping of the code.

Finally, we had to figure out how to avoid storing the chain data while also maintaining the correct parameters of the network, such as nonce and gas price. We achieved this by first storing these parameters locally, and occasionally querying the network to ensure that locally stored data matched global data. With all of these considerations in mind, we have implemented a light client version of Ethereum that is able to send and receive data from other devices.

IV. Light Client Implementation

A. Using Ethereum as a Base

As previously mentioned, the integrity of blockchain communication comes from the size of the network. For example,

when the number of participants in the blockchain is small, it is easy for a malicious actor to execute a 51% attack. While Bitcoin is the largest blockchain network, we sought a large blockchain that does not limit data transfer to 80 bytes nor has a high cost of transaction. For these reasons we chose the Ethereum blockchain [18]. Like Bitcoin, Ethereum uses proof of work as a decentralized consensus algorithm to hash blocks to the blockchain [18]. The coins exchanged in the Ethereum network are ethers rather than bitcoins [18].

B. Avoiding Storing Chain Data

A considerable challenge was avoiding the need to store the Ethereum chain data (including headers) on constrained nodes so that they still could participate in the Ethereum network. Nodes rely on chain data to determine their nonce [18]. A node's nonce is a count of its outgoing transactions [18]. Every time a node wants to make a new transaction it must include its current nonce, otherwise the transaction will be rejected. Keeping the nonce consistent is crucial to the function of the light client.

In the light client implementation, nodes keep track of their nonce locally and occasionally query other nodes in the network to gain a consensus on their correct nonce.

In addition to the nonce, nodes need to have correct gas limit and gas price values in order to ensure that their transactions are mined and stored in the chain. Gas pays for the computation of the transaction regardless of whether it is accepted or not. This transaction fee is equal to gas limit * gas price. These values have been hard coded at a gas price of 30 Gwei (equal to 0.00000003 ether) and a gas limit of 210000 units of gas because these values are highly likely to get transactions added to the chain. Therefore, light nodes do not need to query for this data. With the gas and nonce set correctly, the light node is able to exchange transactions without storing chain data.

C. Reducing Code size

Eliminating the need to store chain data greatly reduces the size of the code that needs to be stored on the node. However, the current open source code for Ethereum is still 30 MB. Therefore, we have stripped as much code as possible from the node in order to reduce its size. Since the light nodes are not acting as miners in the system, there are many aspects of the code base that can be removed for the light client, such as any code related to mining. Furthermore, there are a few compiler flags that we also added in order to cut down on size. To date, we have stripped the code base to 5 MB while preserving the ability to send transactions and query other nodes. Although 5 MB is sufficient for many smart city devices (many CCTVs), we aim to reduce this size even further.

D. Architecture and Communication

As seen in Figure 1, the light client architecture interacts with full nodes and other light nodes in a distributed manner. The light nodes do not store any chain data but can both send and receive data as indicated by the two-way transaction broadcast arrows. The light nodes send data by adding that data to a transaction and then sending the transaction to another node. When the light node wishes to send a transaction, it pings all Ethereum nodes, just like any full node would. Only a full node can process the light node's data, which is indicated by the red "x" shown between light node communications. When a full node successfully mines the light node's block (indicated by a checkmark), other full nodes will no longer be able to mine the same block (indicated by an "x"). After the light node's block is successfully mined it is posted to the Ethereum network as an immutable value on the ledger. The light client must pay a small gas fee should it wish to send data, usually on the order of a few cents [11].

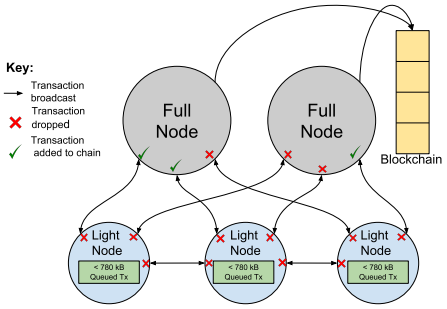

Fig. 1. Distributed Node Architecture

Light clients are able to receive data from the full nodes by pulling data from the blockchain. Although these light nodes will not store the blockchain themselves, they can query other nodes for information about the chain. Therefore, these nodes can query to see whether any recent transactions have been sent to them and then download these specific, small transactions. These transactions are signed and can be verified using cryptography [18]. Therefore, the light nodes and full nodes are able to send and receive messages effectively, and the small node does not need to store the entire chain data.

If the light client has a large amount of information it wants to send all at once, it can queue transactions and aggregate the transfer. The gas limit of an Ethereum transaction is about 3,000,000 gas which allows up to 780 kB of data per transaction [3]. Therefore, at any given time we queue up to this amount. The extent of transaction queuing varies by the amount of available storage on the device.

V. Discussion

Our Ethereum light client communication protocol for smart city IoT devices provides assurance that data was not compromised in transit. The light client authenticates the origin of a data source based on the unique Ethereum address that initiated the transaction. This can be trusted because of the proof of work consensus algorithm and the size of the Ethereum chain. Many urban critical infrastructure sectors rely on integral communications. For example, electric grid infrastructure requires device state information to be transferred over networks at regular intervals. If an attacker compromised the integrity of state data, there could be cyber-physical damages.

Despite the benefits of using the light client for communications, there are limitations of this approach which restrict its practical implementation to certain IoT use cases.First, the light client will require an operating system to run. This considerably limits the number of devices it is applicable for. Also, the block time of our light client is on average 15 seconds meaning that there is a transaction delay for data [11]. This is superior to Bitcoin's block time which is closer to 10 minutes [11]. Therefore, devices requiring real-time data transfer should not use this protocol. Another consideration is that the average cost to send an Ethereum transaction using our light client implementation is 10 cents for gas [11]. This is considerably less expensive compared to Bitcoin-based NeuroMesh which costs an average of 75 cents per transaction [7]. However, for IoT use cases that require constant transactions, this could become very costly over time. While our Ethereum light client is both faster and less expensive than NeuroMesh, we acknowledge that it is not optimal for all types of use cases.

Another advantage of our IoT blockchain compared with IoTex, IoT chain and the tangle is that the light client resides on a major blockchain. Ethereum is widely used - over 10,000 transactions are sent every hour and within a 24 hour time period, around 200,000 addresses will be active [11]. Because of the high volume of users, it would be difficult for an attacker to gain control of over 51% of the network, unlike other IoT blockchain implementations.

To date, we have tested the light client on less than 100 devices. The transactions have reliably been sent and received within the expected constraints of the Ethereum network. We plan to install this communication protocol on more devices to test the light client at scale and under failure conditions, such as when a large portion of the network crashes or a node's transaction is continually rejected. To achieve scale, we will be publicly release the light client to the research community for testing.

VI. Conclusion and Future Work

Our research in developing a light client, hopes to address the problems with communication integrity for IoT devices. Public blockchains such as Ethereum have given us the ability to disseminate data in a scalable and distributed fashion. Our future work on the light client will include further reducing its size and establishing optimal conditions for its function on smart city IoT devices. We also aim to develop an agent that will interpret data from the light client and implement commands on the IoT endpoint. Building an agent for the light client will be the basis for an integrity-driven approach to performing updates for IoT devices at scale.

References

[1] Iotex: A decentralized network for internet of things. 2018. https://iotex.io/white-paper.

[2] Our response to 'a cryptocurrency without a blockchain has been built to outperform bitcoin. 2018.

[3] *Ethereum Blockchain App Platform*, (Accessed: 2019-02-13). https://www.ethereum.org.

[4] *Shodan*, (accessed April, 2018). www.shodan.io/.

[5] Iotchain: A blockchain security architecture for the internet of things. *In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2018)*, pages 1–6, April 2018.

[6] Riccardo Bonetto, Nicola Bui, Vishwas Lakkundi, Alexis Olivereau, Alexandru Serbanati, and Michele Rossi. Secure communication for smart iot objects: Protocol stacks, use cases and practical examples. *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2012.

[7] David Easley, Maureen O'Hara, and Soumya Basu. From mining to markets: The evolution of bitcoin transaction fees. (May 1, 2018). https://ssrn.com/abstract=3055380.

[8] Gregory Falco, Caleb Li, Pavel Fedorov, Carlos Caldera, Rahul Arora, and Kelly Jackson. Neuromesh: Iot security enabled by a blockchain powered botnet vaccine. *ACM Proceedings: International Conference on Omni-Layer Intelligent Systems (COINS)*, 2019.

[9] Gregory Falco, Arun Viswanathan, Carlos Caldera, and Howard Shrobe. A master attack methodology for an ai-based automated attack planner for smart cities. *IEEE Access*, pages 48360–48373, August 28, 2018.

[10] Igor Fovino, Andrea Carcano, Thibault Murel, and Alberto Trombetta. Modbus/dnp3 state-based intrusion detection system. *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010.

[11] Adam Gencer, Soumya Basul, Ittay Eyall, Robert van Renessel, and Emin Sirer. Decentralization in bitcoin and ethereum networks. *arXiv preprint arXiv:1801.03998*, 2018.

[12] Jiyong Han, Minkeun Ha, and Daeyoung Kim. Practical security analysis for the constrained node networks: Focusing on the dtls protocol. *2015 5th International Conference on the Internet of Things (IOT)*, 2015.

[13] Andrew Minteer. *Analytics for the Internet of Things (IoT)*.

[14] Satoshi Nakamotoi. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf.

[15] Popov Sergui. The tangle. 2015. https://iota.org/IOTAWhitepaper.pdf.

[16] Zhengguo Sheng, Shusen Yang, Yifan Yu, Athanasios Vasilakos, Julie McCann, and Kin Leung. A survey on the ietf protocol suite for the internet of things: standards, challenges, and opportunities. *IEEE Wireless Communications*, pages 91–98, 2013.

[17] Dinesh Thangavel, Xiaoping Ma, Alvin Valera, Hwee-Xian Tan, and Colin Tan. Performance evaluation of mqtt and coap via a common middleware. *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2014.

[18] Gavin Wood. Ethereum (eth) – whitepaper. 2015. http://gavwood.com/paper.pdf.