FastPay: A Secure Fast Payment Method for Edge-IoT Platforms using Blockchain

Zijiang Hao
College of William and Mary
Williamsburg, Virginia
hebo@cs.wm.edu

Raymond Ji
College of William and Mary
Williamsburg, Virginia
mengxi25@gmail.com

Qun Li
College of William and Mary
Williamsburg, Virginia
liqun@cs.wm.edu

ABSTRACT

Blockchain-based cryptocurrency systems such as Bitcoin and Ethereum have attracted much attention during the last decade. In recent years, a trend of combining Internet of Things (IoT) devices with blockchain technology has emerged. Digital payments can be made on front-end IoT devices, while a back-end blockchain serves as a distributed ledger to ensure the validity of payments across the system. Nevertheless, fast payments are usually on demand in such a scenario, but an open problem still remains on how to protect blockchain-based systems from double-spending attacks in the context of fast payment. Off-chain techniques, such as Lightning Network and Raiden Network, act as countermeasures to this problem, but they all suffer from the hidden transactions problem. To combat this problem, we propose FastPay, a solution for achieving secure fast payments in blockchain-backed edge-IoT systems. Preliminary evaluation on our prototype demonstrates the effectiveness of FastPay.

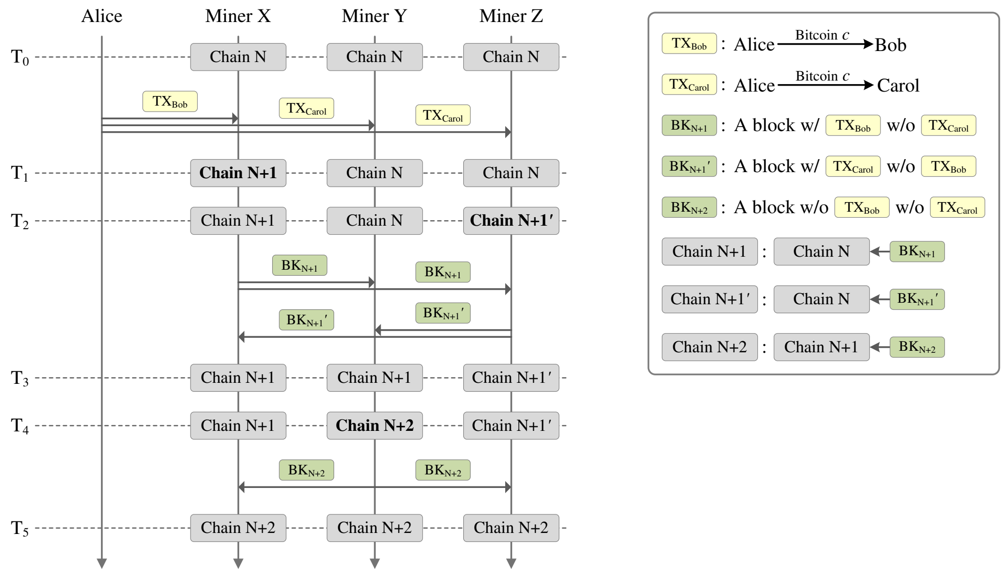
1 INTRODUCTION

The Internet of Things (IoT) has grown significantly in recent years. It is believed that the global market value of the IoT will reach as much as 7.1 trillion dollars by 2020 [9]. The basic idea of the IoT concept is that smart devices, including RFID (radio-frequency identification) tags, mobile devices, sensors and actuators, are connected to and managed by a global network infrastructure based on standard and interoperable communication protocols [20]. Since IoT devices usually possesses limited hardware resources, edge computing is viewed as the best enabler for IoT systems by both industry and academia, as it can serve the IoT devices with low network latency [1, 2, 17].

We observe that a new trend has emerged in the IoT field, i.e., employing the blockchain technique to either manage the data collected from the IoT devices or enhance the security of the IoT system [3, 5, 7, 8, 10, 15]. Blockchain provides a way for IoT systems to achieve consensus on the critical data within the system, even in presence of malicious IoT devices. This broadens the range of the IoT applications, which in turn pushes the advancement of the IoT technique. In particular, making digital payments via blockchain-backed IoT devices has been envisioned as a popular IoT application in the near future [16, 18]. Through this application, customers can make payments using a range of devices connected to the IoT, such as smart cars, household appliances, smartphones, and most recently, wearables. A blockchain serves as a distributed ledger behind the IoT devices, establishing a solid foundation for ensuring the validity of the payments in the IoT system.

Nevertheless, fast payments are usually on demand in such a scenario. Unfortunately, it is well-known that blockchain-based cryptocurrency systems are prone to double-spending attacks in the context of fast payment [14]. In order to ensure that a payment will not be invalidated afterwards, the payee must wait until enough subsequent blocks have appeared in the blockchain, which may take tens of minutes. This prevents the blockchain-based IoT payment application from being widely adopted in the real world. Consider the following example: two attackers share the same blockchain wallet. One attacker makes a purphase on fast food via an IoT device in a restaurant, while the other purchases some goods simultaneously via another IoT device in a supermarket. The restaurant and the supermarket are geographically far from each other, and each attacker spends all the fund in the wallet, meaning that the fund is spent twice. In such a case, either the restaurant or the supermarket cannot receive money from the wallet.

Notably, some off-chain techniques, such as Lightning Network [11] and Raiden Network [19], have been proposed as countermeasures to this problem. Nevertheless, they all suffer from the hidden transaction problem, i.e., the payments are batched and combined, and the blockchain only records the combined payments, losing lots of information about the raw payments. This is not desirable in the IoT scenario, because such information can be used to infer the trend of user behaviors and should be made public. To this end, we propose our solution, FastPay, which provides the security guarantee for fast payments in blockchain-backed edge-IoT systems. FastPay is built atop the smart contract mechanism supported by many well-known blockchain systems, such as Ethereum [21]. FastPay secures fast payments by forcing payers along with their guarantors to provide payment proofs before the payments take place. With these proofs, the payers and their guarantors will receive severe penalties if they dare to launch double-spending attacks. To the best of our knowledge, we are the first to build a secure fast payment method for edge-IoT platforms using blockchain. Preliminary evaluation on our prototype system reveals that FastPay works effectively with acceptable overhead. More specifically, FastPay confirms a payment in about 9 seconds,



Alice
Miner X
Miner Y
Miner Z
T0
T1
T2
T3
T4
T5
Chain N
Chain N+1
Chain N+1′
Chain N+2
TXBob
TXCarol
BKN+1
BKN+1′
BKN+2
TXBob : Alice —Bitcoin c→ Bob
TXCarol : Alice —Bitcoin c→ Carol
BKN+1 : A block w/ TXBob w/o TXCarol
BKN+1′ : A block w/ TXCarol w/o TXBob
BKN+2 : A block w/o TXBob w/o TXCarol
Chain N+1 : Chain N ← BKN+1
Chain N+1′ : Chain N ← BKN+1′
Chain N+2 : Chain N+1 ← BKN+2

Figure 1: Double-spending prevention in Bitcoin.

much faster than many widely-adopted blockchain-based cryptocurrency systems such as Bitcoin [4].

2 BLOCKCHAIN & DOUBLE-SPENDING

Blockchain systems are built on the peer-to-peer (P2P) basis [14, 21]. Every peer in the system is called a *miner*; they collaboratively maintain a global ledger called *blockchain* to keep track of the transactions acknowledged by the system. More specifically, each miner maintains a local replica of the blockchain, and the consensus on the blockchain is achieved through *proof-of-work* (POW). POW is organized in which all miners work on a generated computational puzzle, and the first one to solve the puzzle is allowed to append a new block to the blockchain. The new block contains a sequence of transactions seen by the puzzle-solver, so the blockchain is essentially a global ledger of the transactions. All miners in the system eventually agree on the same view of the blockchain, even with the presence of malicious miners, as long as the benign miners hold a majority of the CPU power.

Due to the long broadcast latency of the underlying P2P network and the unstable block generation rate, however, it is possible that multiple blocks generated from the same state of the blockchain simultaneously appear in the system [6], such as the case described in Figure 1. This is called a blockchain fork. The conflicts of the forked blockchains will be eliminated after a longer blockchain appears, due to the miners always treating the longest blockchain they have ever seen as the current state of the distributed blockchain. As such, consensus on the blockchain is achieved in an eventual manner, and it is possible that a transaction previously included in a block disappears in the blockchain due to a blockchain fork. Bitcoin suggests that a transaction should not be considered safely included by the blockchain until the block containing it has five or more subsequent blocks [4]. This rule is known as “6 confirmation”, and the transactions in accordance with this rule are considered to be “6-confirmed”.

Figure 1 demonstrates how double-spending is prevented in Bitcoin. Miner X, Miner Y and Miner Z are Bitcoin miners. Alice is a Bitcoin user, who intends to double-spend *c*, the only bitcoin in her wallet, by transferring it to both Bob and Carol. At time T_0, the system blockchain is in the state of Chain N. At some time after T_0, Alice sends transaction TX_{Bob} to Miner X, and transaction TX_{Carol} to Miner Y and Miner Z, hoping that the system blockchain will record both transactions. In other words, Alice has launched a double-spending attack.

At time T_1, Miner X solves the puzzle and generates a new block BK_{N+1}, which contains TX_{Bob} that Miner X has received from Alice. Similarly, at time T_2, Miner Z solves the puzzle and generates another block BK_{N+1}', which contains TX_{Carol}. T_1 and T_2 are so close that Miner Z has no chance to learn the existence of BK_{N+1} before generating BK_{N+1}'. Miner X and Miner Z further update their local blockchain to Chain N+1 and Chain N+1′, respectively, and broadcast the new block to other miners.

Miner Y receives BK_{N+1} first, and then BK_{N+1}'. Since both blocks are generated by solving the same puzzle, Miner Y accepts only the first one it has received, i.e., BK_{N+1}, and updates its local blockchain at time T_3 to Chain N+1, the same as that of Miner X. At time T_4, Miner Y solves the next puzzle and generates a new block BK_{N+2}. It then updates the local blockchain from Chain N+1 to Chain N+2 using BK_{N+2}, and broadcasts BK_{N+2} to other miners. Upon receiving BK_{N+2}, Miner X and Miner Z update their local blockchain to Chain N+2. Note that Chain N+2 is derived from Chain N+1, so it is in conflict with Chain N+1′. Since Chain N+2 is longer

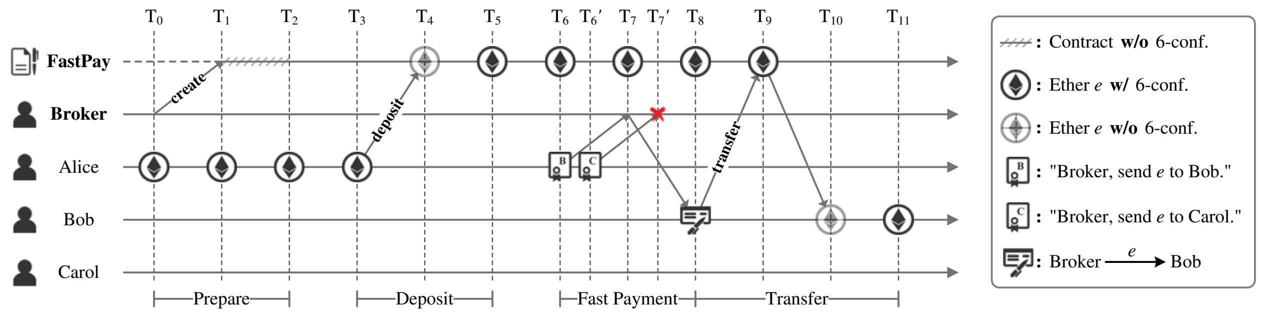


Figure 2: The FastPay protocol.

than Chain N+1′, Miner Z must desert Chain N+1′ and adopt Chain N+2 according to the blockchain protocol.

There are surely other miners existing in the system, but we only consider the three shown in Figure 1 for the simplicity of discussion. Clearly, consensus on the blockchain is reached at time T_5 among the three miners, because their local blockchains are all in the state of Chain N+2. Because Chain N+2 contains only TX_{Bob} but not TX_{Carol}, Bob is the only one who receives *c* from Alice. In other words, Alice has failed to double-spend her bitcoin.

Figure 1 shows only a simple case of how double-spending is prevented in Bitcoin. Nevertheless, it reveals an important characteristic of the blockchain protocol, i.e., consensus on the blockchain is achieved in an eventual manner, and divergence on the blockchain may temporarily occur in the system. In essence, any payee can never assert that he has received the money from a payer in such a system, because it is always possible that a longer, conflicting blockchain emerges and replaces the existing one. For this reason, Bitcoin suggests that the users adopt 6-confirmation [4], meaning that a transaction should not be confirmed until the block containing the transaction is followed by 5 or more blocks in the blockchain.

Theoretically speaking, 6-confirmation can only reduce, but not eliminate, the chance that a confirmed transaction becomes invalid. However, it works effectively in practice, because it is extremely unlikely that a block followed by 5 or more blocks will be invalidated by a longer, conflicting blockchain. Despite its effectiveness, 6-confirmation comes with a cost: the latency between when a transaction is issued and when it is confirmed is quite long. It is known that Bitcoin miners produce one block every 10 minutes on average [13], so roughly speaking, the latency of 6-confirmation is one hour. This might be fine for many existing scenarios, but is unacceptable when fast payment is on demand, such as those involving IoT payments.

3 THE FASTPAY PROTOCOL

To protect the blockchain-based IoT payment systems from double-spending attacks, we design a protocol, called FastPay, based on the smart contract mechanism [12] provided by many popular blockchain systems such as Ethereum [21]. A smart contract is a piece of code stored on the blockchain. Users can invoke a smart contract by submitting transactions calling its public functions to the miners. After a miner has accepted such a transaction, it executes the code of the smart contract, and stores the current state of the smart contract on the blockchain. Based on the blockchain mechanism, smart contracts achieve consensus among the miners, and hence can be used to apply many secure computations.

The FastPay protocol works on the basis that a special user, called Broker, exists in the system. The Broker acts as the intermediate of the payer and the payee in a fast payment process. By carefully coordinating the behavior of the payer and the payee using the smart contract approach, the Broker achieves secure fast payments for IoT systems.

Figure 2 illustrates a simple example of how the protocol works. The protocol includes four phases: the Prepare phase, the Deposit phase, the Fast Payment phase, and the Transfer phase. The underlying system in Figure 2 is Ethereum, as we need to choose a smart-contract-supporting system. Therefore, Alice tries to double-spend her ether *e*, rather than the bitcoin *c* used in Figure 1's example. In what follows, we will describe the four phases in detail.

3.1 The Prepare Phase

Before providing the fast payment service, the Broker needs to establish a "FastPay" smart contract on the blockchain. This FastPay contract essentially encodes all the actions that should be performed upon receiving messages from the system users. Such actions include:

- **Deposit**: A user transfers money to the smart contract as its deposit.
- **Withdraw**: A user retrieves a specified amount of money from its deposit in the smart contract.
- **Transfer**: A user asks the smart contract to transfer the payment from another user's deposit to its own account.
- **Add Security Deposit**: The Broker transfers money to the smart contract as available security deposit.
- **Retrieve Security Deposit**: The Broker retrieves a specified amount of money from the available security deposit.
- **Increase Insurance**: A user asks the smart contract to take a specified amount of money from the Broker's available security deposit and use the money as the insurance

for the subsequent fast payments in which the user acts as the payee. This action should be done in advance, e.g., a restaurant using FastPay should have acquired enough insurance before it starts selling food.

- **Decrease Insurance**: A user asks the smart contract to revoke the insurance on its account by a specified amount.
- **Cancel**: A user asks the smart contract to cancel a payment it has made to another user, with the confirmation from that user.

The first three actions listed above are the basic operations for the fast payment service. The other five actions are used to manage the security deposit, which ensure that the Broker will get punished if it violates the rules of the protocol. More details will be discussed later.

After creating the FastPay contract on the blockchain, the Broker must wait until the smart contract is 6-confirmed. When the smart contract is 6-confirmed, the Prepare phase ends, and the Broker can provide the fast payment service to the system users. In Figure 2, the Prepare phase occurs during $[T_0, T_2]$.

3.2 The Deposit Phase

Any user intending to make fast payments via the FastPay protocol must have enough deposit on the Broker's side. In other words, the user must have transferred enough money (i.e., cryptocurrency) to the Broker, and the transaction(s) transferring the money must have been 6-confirmed before the payment is made. This is done during the Deposit phase.

In Figure 2, Alice initiates a Deposit phase at T_3, transferring the ether e to the FastPay contract via a deposit message. At T_5, the transaction is 6-confirmed, and the Deposit phase thus ends.

3.3 The Fast Payment Phase

Any user with enough deposit in the FastPay contract can make a fast payment to another user, by initiating a Fast Payment phase. The user is supposed to generate a signed request, indicating the payer, the payee, and the amount to transfer in the fast payment, and send the request to the Broker. The Broker verifies whether the request contains valid data and has been signed correctly, and whether the payer has enough deposit. If both answers are "yes", the Broker generates a digital cheque for the request, signs it, and sends it to the payee. All the communication is done through specified network links, not via the system blockchain or the FastPay contract.

After receiving the digital cheque, the payee first verifies whether it is valid. If the digital cheque passes the verification, the payee further checks if he has enough insurance on his account. If so, it is good for the payee to serve the payer. This process implies that the protocol achieves safety through the insurance mechanism, which works as follows.

- All security deposit belongs to the Broker. The FastPay contract maintains a pool of the available security deposit. The Broker can add/retrieve money to/from the pool.
- A user can ask the Broker to assign a specified amount of security deposit to its account as the insurance. Upon receiving such a request, the Broker generates a confirmation, indicating the target account (i.e., the user's account) and the amount of security deposit requested. It then signs the confirmation and sends it to the user. The user then sends the signed confirmation to the FastPay contract. Note that only the Broker needs to sign, because the FastPay contract will verify whether the confirmation has been sent by the target account. After the FastPay contract has ensured that the confirmation is valid, it takes the specified amount of security deposit from the pool, and assigns it to the user's account.
- After the security deposit has been assigned to the user's account as an insurance, it becomes unavailable, i.e., it cannot be retrieved by the Broker or assigned to another user. On the other hand, the Broker may periodically charge the user for the assigned security deposit. The more security deposit has been assigned, the more charge is imposed. The FastPay contract may automatically transfer a charge from the user's account to the Broker's account when it is time for the user to pay for it.
- The user can freely decrease the insurance assigned to its account. The decreased part of the insurance will be put back to the pool of the available security deposit. By doing this, the user can reduce the charges on its insurance.
- When a user sends a valid digital cheque to the FastPay contract, the FastPay contract checks whether the payer indicated by the digital cheque has enough deposit. If not, the Broker has violated the rules, probably colluding with the payer to launch a double-spending attack. In such a case, the FastPay contract converts all the insurance assigned to the user's account to the user's property, i.e., it transfers the security deposit previously belonging to the Broker to the user. By doing this, the FastPay contract punishes the Broker and compensates the user.
- Any payee is responsible for tracking the states of the insurance on his account. For example, an insurance cannot be used as the guarantee for any fast payment until the transaction assigning the insurance is 6-confirmed, as the transaction can be invalidated by a blockchain fork. More importantly, a payee should track the fast payments made to him, because the insurance on his account is used to guarantee all the fast payments that have been made to him but are not 6-confirmed yet. The payee must ensure that the total amount of the outstanding fast payments is less than that of the insurance he has received. Otherwise, he may still suffer an economic loss even though he receives all the insurance as the compensation in case he has experienced a double-spending attack.
- If a payee has decided to reject a fast payment due to the lack of the insurance on his account, he cancels the fast payment by signing a confirmation and sending it to the payer. The payer can contact the FastPay contract at any time with the confirmation to get her money back.

We would highlight that the FastPay contract has been established on the blockchain before the protocol is put into action, which means that the FastPay contract is public and immutable during the execution of the protocol. For this reason, it cannot be modified by attackers, and if the Broker dare collude with a malicious payer to launch a double-spending attack, it will receive a severe penalty as long as the victim payee sends the FastPay contract the digital cheque it has received from the Broker, as the proof of the attack. Moreover, the Fast Payment phase is initiated by the payee through a client device. More specifically, the client device sends a fast payment request to a nearby IoT device, which accomplishes the request by interacting with the FastPay contract on the blockchain. All resource-intensive computations, such as generating and verifying digital signatures, are done on an edge server close to the IoT device.

In Figure 2, a Fast Payment phase occurs during [T_6, T_8]. At T_6, Alice sends a request to the Broker, asking the Broker to transfer the ether *e* to Bob. In order to double-spend *e*, later at T_6', Alice sends another request to the Broker, asking it to transfer *e* to Carol. The Broker, however, accepts only the first request at T_7, while rejects the second one at T_7', according to the protocol rules. It then generates the corresponding digital cheque for the first request and sends it to Bob. Bob receives the digital cheque at T_8, which closes the Fast Payment phase.

3.4 The Transfer Phase

After receiving the digital cheque from the Broker, the payee can initiate a Transfer phase at any time to retrieve the payment, by sending a transfer message with the digital cheque to the FastPay contract. The FastPay contract then verifies the digital cheque and submits the embedded transaction to the miners. If the transaction has been invalidated due to a blockchain fork, the payee can resend the transfer message, asking the FastPay contract to re-submit the transaction. It can keep doing so until the transaction is 6-confirmed.

In Figure 2, a Transfer phase occurs during [T_8, T_{11}]. At T_8, Bob sends a transfer message with the digital cheque he has received to the FastPay contract. At T_{10}, the miners receive the transactions. At T_{11}, the transaction is 6-confirmed, which indicates that Bob has successfully received the ether *e* from Alice's deposit.

4 EVALUATION

To examine whether FastPay works well in the real world, we have implemented a prototype and deployed it on a testbed. Evaluation on the prototype reveals that FastPay is as effective and efficient as expected.

4.1 Latency Comparison

We have first examined the minimum latency a payer must wait for until the payee ensures that the payment is safe and is about to provide his good/service to the payer. Two cases are considered. In the first case, the payment is done through the common practice, i.e., the payer transfers money to the payee by submitting a transaction to the miners and the payee waits until the transaction is 6-confirmed. The latency in this case is therefore the time difference of when the transaction is 6-confirmed and when the payer submits the transaction. In the second case, the payment is done through FastPay, and the latency is exactly the duration of the Fast Payment phase.

Two experiments are conducted for the two cases. In the first experiment, the miners receives money-transferring transactions from the payers, while in the second experiment, the Broker receives FastPay-style fast payment requests from the payers. In both experiments, the requests are generated by the four machines expect the one where the Broker resides, and the payer and the payee involved in each payment process are assumed to be located at the same machine. The requests are generated on each machine with the time intervals following a normal distribution, which has a mean of 40 seconds and a standard deviation of 10 seconds. Both experiments generate 1,000 requests in total.

Table 1: Metrics of the Secure Latency.

	Common Practice	FastPay
Minimum (sec)	39.45	8.89
Maximum (sec)	31389.89	9.68
Average (sec)	3452.01	9.11
Std. Dev. (sec)	3397.18	0.11

Table 1 shows the comparison results of the latency required by the common practice and FastPay. It can be seen that the average latency of FastPay is only about 9 seconds, and is quite stable, with a standard deviation of only 0.11 seconds. This result is quite satisfactory for the IoT payment scenario from our point of view. In contrast, the average latency of the common practice is about 58 minutes, 379 times of that of FastPay. Furthermore, the latency of the common practice varies a lot during the experiment, because the block intervals are exponentially distributed. These results are in accord with our analysis on the common practice of how payments are currently done in a blockchain-based cryptocurrency system, and are far from satisfactory for the IoT payment scenario. Note that we adopt the Bitcoin settings as the common practice, because it is the most widely-adopted blockchain-based cryptocurrency system, and is thus worth being adopted as the baseline. Moreover, although Bitcoin does not support smart contract at the current stage, we expect that it will introduce such a support in the near future.

4.2 The Safety of FastPay

To determine whether FastPay can indeed protect the payees from double-spending attacks, we have conducted the following experiment. We only run a one-thread miner on two of the machines, and execute a program on both machines so that the communication between them are periodically disabled and enabled. When the communication between the two miner machines is cut off, they mine independently, so that a blockchain fork is artificially generated. To guarantee that no blockchain fork invalidates more than 6 blocks, the program stops the miner after it has generated 3 blocks since the latest communication cut-off, and restarts it after the communication has been recovered.

Each payer generates a pair of fast payment requests, trying to transfer all its funds to different payees. As there are 500 payers in the system, totally 1,000 requests are generated. The requests are then sent to the Broker, which colludes with the payers to launch double-spending attacks. The Broker runs on a machine other than the two running miners, and we also use this machine to perform any processing for the payers and the payees. Fast payment requests are generated by this machine on behalf of the payers, and their time intervals follow a normal distribution with a mean of 10 seconds and a standard deviation of 10 seconds.

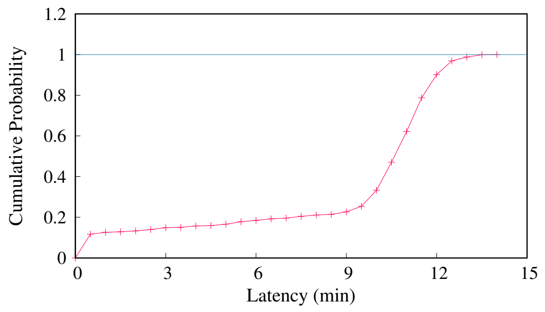


Figure 3: The cumulative probability of the successful requests.

Figure 3 illustrates the cumulative percentage of successful requests. A request is considered successful when the corresponding payment or compensation the payee has received is 6-confirmed, i.e., when the payee holds the money for real. The latency shown on the X axis is the time difference of when request becomes successful and when the payee first submits the transfer transaction to one of the miners.

During the experiment, 381 transaction re-submissions have occurred. It can be seen from Figure 3 that when the latency is 9 minutes, the cumulative percentage of successful requests is only 22.7%. When the latency $>$ 9 minutes, however, the cumulative percentage increases dramatically with the increase of latency, partially because of the exponential distribution of the block intervals, and partially because more and more transactions previously invalidated by blockchain forks eventually become 6-confirmed. When the latency reaches 14 minutes, all requests become successful, i.e., all payees eventually receive their money.

5 CONCLUSION

We have introduced FastPay, a secure fast payment method for blockchain-backed edge-IoT platforms. Preliminary evaluation on our prototype reveals that FastPay works effectively and efficiently. It requires only 9 seconds to confirm a payment. Most importantly, it can work directly using the existing blockchain platforms, such as Ethereum, without any modification to the system.

ACKNOWLEDGMENT

The authors would like to thank all the reviewers for their helpful comments. This project was supported in part by US National Science Foundation grant CNS-1816399.

REFERENCES

[1] Bonomi, F., Milito, R., Natarajan, P., and Zhu, J. Fog computing: A platform for internet of things and analytics. In *Big data and internet of things: A roadmap for smart environments*. Springer, 2014, pp. 169–186.

[2] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing* (2012), MCC '12, pp. 13–16.

[3] Christidis, K., and Devetsikiotis, M. Blockchains and smart contracts for the internet of things. *IEEE Access 4* (2016), 2292–2303.

[4] Confirmation. https://en.bitcoin.it/wiki/Confirmation, 2017.

[5] Conoscenti, M., VetrÃš, A., and Martin, J. C. D. Blockchain for the internet of things: A systematic literature review. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications* (2016), AICCSA '16, pp. 1–6.

[6] Decker, C., and Wattenhofer, R. Information propagation in the bitcoin network. In *Proceedings of the 2013 IEEE International Conference on Peer-to-Peer Computing* (2013).

[7] Dorri, A., Kanhere, S. S., and Jurdak, R. Towards an optimized blockchain for iot. In *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation* (2017), IoTDI '17, pp. 173–178.

[8] Dorri, A., Kanhere, S. S., Jurdak, R., and Gauravaram, P. Blockchain for iot security and privacy: The case study of a smart home. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops* (2017), PerCom Workshops '17, pp. 618–623.

[9] Hsu, C.-L., and Lin, J. C.-C. An empirical examination of consumer adoption of internet of things services. *Comput. Hum. Behav. 62*, C (2016), 516–527.

[10] Huh, S., Cho, S., and Kim, S. Managing iot devices using blockchain platform. In *2017 19th International Conference on Advanced Communication Technology* (2017), ICACT '17, pp. 464–467.

[11] Lightning network. https://lightning.network/, 2018.

[12] Luu, L., Chu, D.-H., Olickel, H., Saxena, P., and Hobor, A. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016).

[13] Mining. https://en.bitcoin.it/wiki/Mining, 2017.

[14] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008.

[15] Samaniego, M., and Deters, R. Hosting virtual iot resources on edge-hosts with blockchain. In *2016 IEEE International Conference on Computer and Information Technology* (2016), ICCIT '16, pp. 116–119.

[16] Secure Technology Alliance. Iot and payments: Current market landscape. https://www.securetechalliance.org/wp-content/uploads/IoT-Payments-WP-Final-Nov-2017.pdf, 2018.

[17] Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. Edge computing: Vision and challenges. *IEEE Internet of Things Journal 3*, 5 (2016), 637–646.

[18] The internet of things (iot) in payment. https://www.optile.net/blog/internet-things-iot-payment/, 2018.

[19] The raiden network. https://raiden.network/, 2018.

[20] Van Kranenburg, R. *The Internet of Things: A Critique of Ambient Technology and the All-Seeing Network of RFID.* Institute of Network Cultures, 2008.

[21] Wood, G. Ethereum: A secure decentralised generalised transaction ledger. http://gavwood.com/paper.pdf, 2014.