Digital Forensic Implications of Collusion Attacks on the Lightning Network

Dmytro Piatkivskyi, Stefan Axelsson, Mariusz Nowostawski

Chapter 8

DIGITAL FORENSIC IMPLICATIONS OF COLLUSION ATTACKS ON THE LIGHTNING NETWORK

Dmytro Piatkivskyi, Stefan Axelsson and Mariusz Nowostawski

Abstract The limited size of a block in the Bitcoin blockchain produces a scaling bottleneck. The transaction scalability problem can be addressed by performing smaller transactions off-chain and periodically reporting the results to the Bitcoin blockchain. One such solution is the Lightning Network.

Bitcoin is employed by lawful users and criminals. This requires crimes against lawful users as well as the use of Bitcoin for nefarious purposes to be investigated. However, unlike Bitcoin, the Lightning Network enables collusion attacks involving intermediate nodes and recipients. In such an attack, regardless of a sender's actions, money is received by an intermediate node that colludes with a dishonest recipient. Since the dishonest recipient does not "actually" receive the money, it does not provide the goods/service to the sender. Thus, the sender pays for the unprovided goods/service, but the recipient can prove that the payment was not received.

This chapter discusses the forensic implications of collusion attacks with regard to lawful users because no discernible traces of attacks remain, as well as for law enforcement, where the attacks can target parties as a form of forfeiture, analogous to law enforcement "sting" operations. This chapter also discusses the potential of the Lightning Network to be used for money laundering activities.

Keywords: Bitcoin, Lightning Network, audit trail

1. Introduction

Digital currencies are increasingly being leveraged by criminal entities. Therefore, it is important for digital forensic investigators to have detailed knowledge of how these currencies work and how they can be exploited.

Bitcoin has emerged as the *de facto* standard for peer-to-peer value exchange in decentralized systems. However, a key problem with the Bitcoin blockchain technology is its scalability. Several solutions have been proposed to address the scalability problem. One solution is the Lightning Network, a peer-to-peer payment system that performs smaller transactions off-chain and periodically reports the results to the Bitcoin blockchain. This chapter discusses the design of the Lightning Network and demonstrates a fundamental flaw that facilitates collusion attacks. Such an attack enables money to go astray between a sender and a dishonest recipient who colludes with an intermediate to claim non-receipt of funds. This chapter discusses the forensic implications of collusion attacks with regard to lawful users and law enforcement, along with the potential of the Lightning Network to be used for money laundering activities.

2. Related Work

Decentralized crypto-currencies is a new research field. Off-chain transactions, as used in the Lightning Network, is an emergent trend that has not been investigated adequately. However, there is no published research on the security of the Lightning Network nor is there any discussion of the digital forensic implications of its use.

The concept of a collusion attack is not new. Conspiracies involving actors in a system have been investigated before. For reasons of space, it is not possible to discuss the topic in detail; instead, a few examples are presented from the literature.

Distributed systems such as wireless sensor networks rely heavily on their key management infrastructures. If the keys are not managed properly, network nodes can collude and reveal the keys [5].

In the case of fingerprinting digital data, when users collude, fingerprints can be removed and the data can be distributed freely [1]. Another example is a collusion attack on an Android device where two applications can collaborate to escalate their access rights [2].

In the financial sector, collusion can be used to manipulate stock prices or to secure loans despite having bad credit. With so many varied examples of collusion attacks, it is important that designers of new methods of collaboration, such as the Lightning Network, understand and guard against collusion.

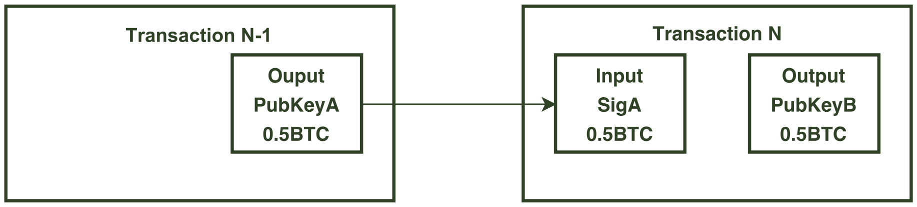

Figure 1. Bitcoin transaction.

3. Bitcoin Blockchain

Blockchain technology enables novel decentralized applications ranging from simple digital tokens that represent currency, through digital assets management and audit trails, to establishing decentralized institutions [7]. Decentralization eliminates the need for a trusted third party in many scenarios. The first large-scale deployment of blockchain technology was in the Bitcoin crypto-currency and peer-to-peer payment system [6].

Bitcoin allows fast cross-border monetary transfers, on average within ten minutes, for a low transaction fee. In addition to providing pseudonymity, the system offers several advantages. Since Bitcoin is decentralized, no one holds custody over it. Moreover, if its secret keys are kept secure, no entity can steal or seize money intended for another entity.

A payment in the Bitcoin blockchain is a transfer of a numerical value from one public address to another. Bitcoins, the crypto-currency, are just numbers that belong to a public address (i.e., a public key). The ownership of bitcoins is claimed by demonstrating the associated private key. An entity can generate as many public-private key pairs as desired. A good practice is to generate a new key pair for every new monetary transfer. Thus, the de-anonymization of an address to a user identity is difficult, albeit potentially possible; in any case, this is an active topic of research [11].

To make a payment, the sender creates a transaction – Transaction N in Figure 1. The transaction consists of two parts: (i) inputs; and (ii) outputs. There may be multiple inputs and multiple outputs in a transaction, but for the sake of simplicity, only one input and one output are illustrated in Figure 1. In the transaction, the sender references the output of a previous transaction (Transaction N-1) and can claim ownership of the coins from this output. The sender claims the ownership via a signature made with his private key. This particular signature must be specified in the input of the new transaction. In the output, the sender identifies the entity to whom ownership is transferred by specifying the public key of this entity (receiver). Finally, the sender broadcasts the created transaction to the Bitcoin network of validating nodes so that the transaction can be accepted into the system. The broadcasted transaction is verified by each node in the network and is eventually recorded in the shared database called the Bitcoin blockchain. The Bitcoin blockchain protocol timestamps all transactions, preventing double spending; the earliest transaction has precedence over more recent transactions.

The blockchain is a decentralized database that is secured from tampering and revision. It consists of blocks that are chained by embedding the hash of the previous block into the next block. The hash calculation is made intentionally difficult so that after a block is stamped with a hash it cannot be recomputed easily. This means that all the transactions that get on the blockchain remain on it forever because the blocks cannot be changed.

The Bitcoin scalability problem arises because Bitcoin blocks can carry a limited amount of transactions. Since the blocks have a fixed size and new blocks are generated at fixed times, the Bitcoin payment system can only sustain a fixed transaction rate. The current limit is around seven transactions per second.

The scalability problem has attracted the attention of the research community and a number of potential solutions have been proposed. The most promising solution is the Lightning Network [9], which uses off-chain transactions.

4. Lightning Network

The Lightning Network [9] is a payment protocol built on top of the Bitcoin blockchain. It leverages off-chain transactions to provide a scalable solution to the problem of limited transaction throughput. The fundamental idea underlying the Lightning Network is not to log all the transactions directly on the blockchain, but to pass them between the participating nodes in a peer-to-peer fashion and log only the final balance of the accounts.

Transactions in the Lightning Network are processed within previously established payment channels. A channel is a set of two Bitcoin transactions created cooperatively by the channel participants. This work considers two participants to simplify the presentation. However, it is possible to emulate multi-party channels with slightly more elaborate protocols.

A funding transaction spends channel participants' funds while a commitment transaction returns funds to the channel participants. The

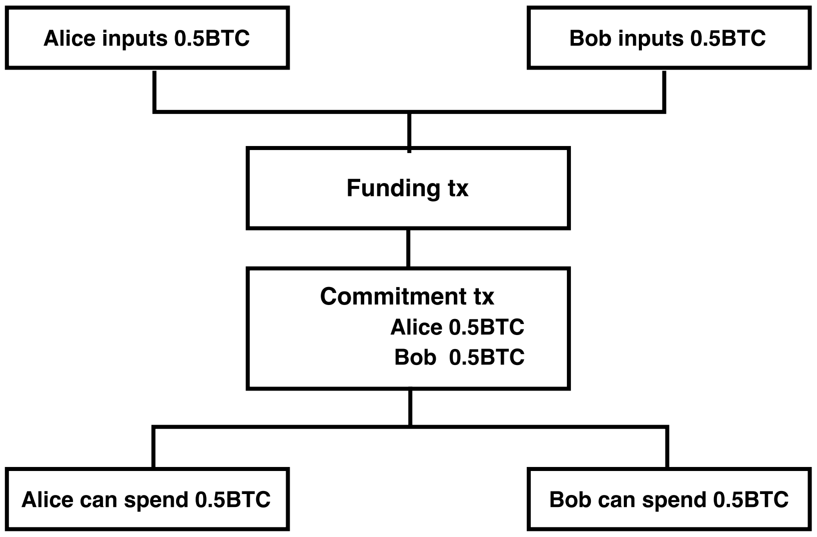

Figure 2. Funding and commitment transactions.

commitment transaction is kept off the blockchain during the time that the channel is open. After a channel is opened, the participants can send arbitrarily small payments to each other up to the channel capacity. The number of transactions possible in a channel is nearly infinite and the transaction speed is only limited by the direct connections between the channel participants, which typically means nearly instantaneous delivery.

The processed transactions are not broadcast to the Bitcoin network. Instead, the transactions are passed in a peer-to-peer fashion between the channel participants. The only transactions that get advertised and, consequently, recorded on the Bitcoin blockchain are channel funding and commitment transactions. At any time, the commitment transaction reflects the channel state. When a channel is to be closed, the commitment transaction is published on the blockchain, which returns the funds to the channel participants. The balance is established at the moment of channel closure. The commitment transaction is the guarantee that a channel participant can get the funds back at any point in time with the agreed balance.

Figure 2 shows funding and commitment transactions. In the case of a bi-directional channel, both channel participants create inputs to the channel funding transaction that define the channel capacity. A funding transaction has a single two-of-two multi-signature output. In other

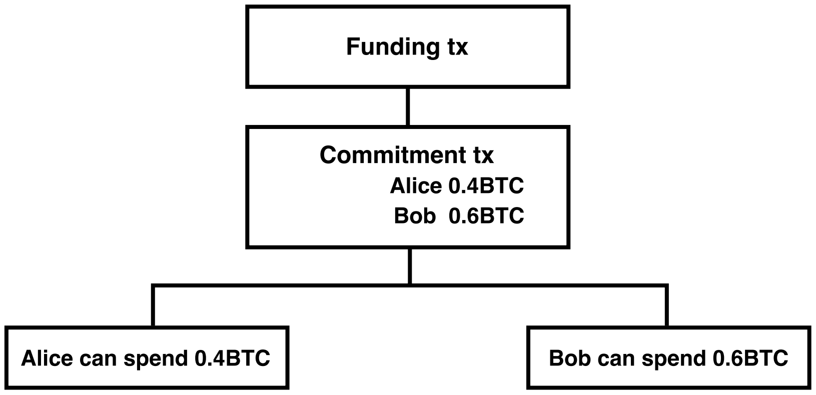

Figure 3. Commitment transaction balances are updated after transacting 0.1BTC.

words, it takes two signatures, one belonging to each channel participant, to spend the output.

In contrast, a commitment transaction spends the output of the funding transaction and has two outputs. Each of these outputs returns back to an investor exactly the amount of funds invested in the channel. The benefit of establishing the channel is that funds can be moved within the channel capacity by simply updating the commitment transaction.

For example, if Alice wishes to send 0.1BTC to Bob, then the commitment transaction is updated so that it returns 0.4BTC to Alice and 0.6BTC to Bob (Figure 3). Within the updated channel Bob now can send to Alice up to 0.6BTC and Alice can send to Bob only up to 0.4BTC. Such payment channels allow nearly unlimited transactions within a channel. Note however, that this simple scheme requires that an entity has to open a channel with every entity with which it has ever interacted. This is an expensive proposition. A solution to this problem is to route payments through existing channels.

4.1 Payment Routing

The Lightning Network extends the idea of payment channels by routing payments over multiple entities that have pre-existing channels between them. For example, if Alice has a channel with Bob and Bob has a channel with Charlie, then Alice can send funds to Charlie through Bob. Because the two money transfers – from Alice to Bob and from Bob to Charlie – are independent, there must be a way to bind them so that the execution of one depends on execution of the other. Otherwise, Bob can send funds to Charlie, but Alice does not send funds to Bob, leaving Bob defrauded. Another scenario involves Alice sending funds

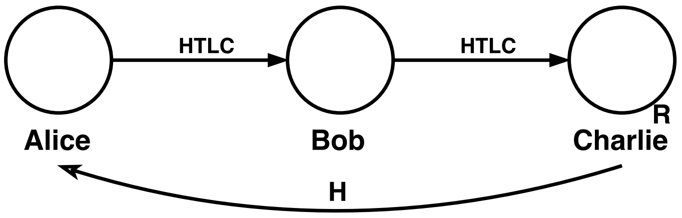

Figure 4. Chain of hashed timelock contracts in a channel.

to Bob, but Bob not sending funds to Charlie, leaving Alice defrauded. The binding has to account for the untrusted nature of Bitcoin and the Lightning Network. Specifically, network nodes do not know anything about their peers and do not rely on established trust relationships.

The Lightning Network protocol relies on a method for binding the execution of transactions without any custodial trust. This solution is called a hashed timelock contract (HTLC). A hashed timelock contract enables Bob to pull funds from Alice only after Charlie has pulled the funds from Bob. The basic idea is that Charlie, the recipient, tells Alice, the sender, a riddle. They agree that, in order to pull the funds, Charlie must give the answer to the riddle. Thus, Alice makes a contract with Bob that, if Bob knows the answer (i.e., the secret), then Alice pays the funds to Bob. Bob cannot know the secret unless Bob pays Charlie. Thus, two contracts are created – between Alice and Bob and between Bob and Charlie. Each contract says "'I will pay you if you give me the answer to the riddle." Only Charlie knows the answer, so he gives the answer to Bob and Bob gives the money to Charlie. Since Bob knows the answer, he can give the answer to Alice and Alice gives him the money.

The scheme works the same when there are more than three participants on the route. The answer to the riddle is passed through all the nodes from the recipient to the sender. At the end, the sender is the only node that does not pull the funds from any other node; the sender just spends the funds whereas everyone else along the path pulls the funds from their respective senders and pushes the funds to their respective recipients. The recipient is the only node that just pulls the funds, so it is the only entity to ultimately receive funds.

A hashed timelock contract riddle is "What value hashes to hash H?" Nobody knows the answer except the entity that generated the hash. Thus, Charlie, the recipient, generates a random secret value R and calculates its hash $H = h(R)$. Then, he sends the hash H to Alice. Based on the hash value H, Alice creates a hashed timelock contract with Bob and Bob creates a hashed timelock contract with Charlie (Figure 4).

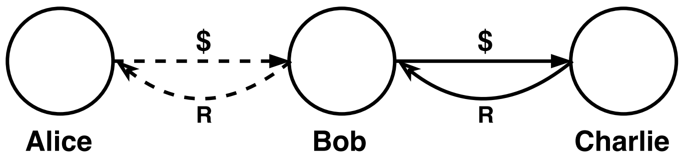

Figure 5. All the hashed timelock contracts are executed after R is revealed.

To complete the transaction, Charlie reveals R to Bob, Bob checks that R hashes to H and Bob pays the promised funds. Then, Bob reveals R to Alice and receives his funds from her (Figure 5).

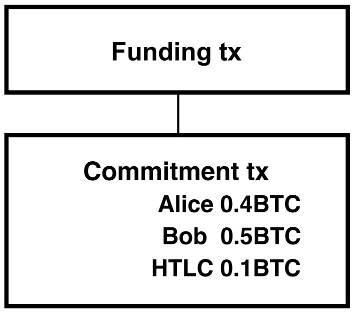

Figure 6. Commitment transaction with a hashed timelock contract.

A hashed timelock contract is realized as one additional output in a commitment transaction (last output in Figure 6). There are two ways to spend the output. Bob can spend this output by providing R (hashed timelock contract execution delivery). Alice can spend this output after some timeout t (hashed timelock contract timeout). Needless to say, only one of these transactions can be published because they spend the same output. They are kept by the channel participants as guarantees that a counterparty will not misbehave analogous to the commitment transaction itself. Unlike the commitment transaction, which eventually gets recorded on the blockchain when the channel is closed, hashed timelock contract execution delivery and timeout transactions may never get on the blockchain. Before closing a channel the parties may cooperatively cancel or execute all the hashed timelock contracts in the channel.

In order to execute a hashed timelock contract, Bob sends R to Alice. Alice knows that, if the commitment transaction gets on the blockchain, Bob will spend the hashed timelock contract output, so she agrees to update the commitment transaction, removing the hashed timelock contract output in Bob's favor. If Alice does not agree to update the commitment transaction, then Bob simply publishes it on the blockchain and right after it is confirmed, he publishes the execution delivery transaction that sends the hashed timelock contract output to himself. Bob does the same if Alice is unresponsive for any reason.

On the other hand, Alice may wish to cancel the hashed timelock contract if Bob does not provide the secret value R for an extended period of time or Alice may simply wish to close the channel and release the funds. In this case, Alice asks Bob to cancel the hashed timelock contract in her favor. If Bob does not agree or does not respond, Alice publishes the commitment transaction on the blockchain and after the timeout t, she publishes the timeout transaction, which sends the hashed timelock contract funds to her. Note that, during the timeout t, Bob can get to know the secret value R and publish the execution delivery transaction to get his funds. In such a case, Alice considers the payment completed. If Alice is an intermediate node on the route, she can execute the contract on the other side and pull her funds.

The Lightning Network uses hashed timelock contracts to provide a secure way to route payments through untrusted nodes. The problem is that the system relies on the recipient being honest and keeping R secret. If the recipient is dishonest and colludes with a node along the route, it is possible to steal money from the sender or use the scheme for money laundering purposes.

4.2 Lightning Network Topology

Several researchers have speculated about the topology of the Lightning Network [8]. A Lightning channel keeps the funds locked within the channel. Unless an entity uses the channel, the time value of the money locked in the channel is wasted. Therefore, channel management is very important. The intuition is that it should dictate the topology taken by the Lightning Network.

Two likely topologies are the hub-and-spoke topology and the organic topology. A hub-and-spoke topology assumes the emergence of bank-like operators (hubs) that would process and route large numbers of transactions. The concerns regarding this topology are centralization, privacy and money locking. There is a fear of large hubs growing larger, which would lead to centralization. Since hubs process transactions, they could aggregate knowledge about many transactions and, thus, pose a threat to privacy (anonymity). Finally, although money locking is a contextual notion, hubs could lock the money in open channels because they do not intend to spend money, only route money. The vast amount of money locked in the topology could impact the viability of the topology and result in high transaction fees.

Organic routing may lock less money because channels are opened on demand. Two limitations with organic routing are route finding and supporting the needed route capacity in the network. Another problem is that the number of on-chain transactions are expected to be much higher than in the hub-and-spoke topology. Detailed evaluations of these topologies is a good topic for future research.

Onion routing has been proposed as a mechanism to alleviate the threat to anonymity in the Lightning Network [10]. It limits the knowledge about nodes in the network only to their neighbor nodes. While the principal advantage of onion routing is final destination masking, this type of routing can impede network analysis and hinder forensic investigations. The implications of onion routing are discussed later in this chapter.

5. Collusion Attack on the Lightning Network

The Lightning Network design relies on the recipient being honest and keeping R secret. At the same time, it is designed to operate in an absolutely untrusted environment and to provide a good level of anonymity to all its participants. The latter assumes an adaption of onion routing, where every node in the network only knows its neighbor nodes. In such conditions, the system must be perfectly secure and flawless. While the Lighting Network is cryptographically secure, it does not take into account the misbehavior of its users.

The original Lightning Network article [9] states that the only way of acknowledging successful transactions is "knowing R is proof of funds sent." However, this does not necessary hold. The sender Alice considers a transaction to be completed when her hashed timelock contract is executed. The recipient Dave considers a transaction to be completed when he receives the funds (Dave is added to the channel for a more explanatory scenario). If the system is used as intended, these two events occur together.

However, there is a situation in which Alice executes her hashed timelock contract, but Dave does not receive the funds. This can only happen if a node on the route (e.g., Bob) knows the pre-image of the hash H – the secret value R. It could be that Bob just guessed it; this is very unlikely, but it is still a possibility. A much more likely scenario involves Dave secretly sending R to Bob, enabling Bob to defraud Alice.

Figure 7 illustrates the simple collusion attack. The recipient Dave generates a secret value R upon which depends the execution of each hashed timelock contract on the route. Dave calculates the hash $H = h(R)$ and sends it to Alice. All this is according to the protocol.

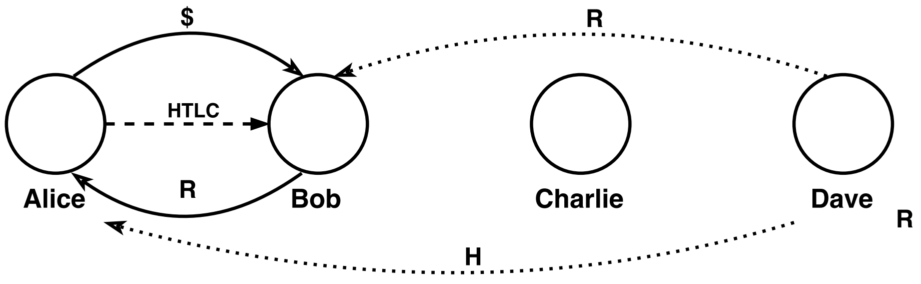

Figure 7. Simple collusion attack.

However, the attack occurs when Dave also shares the secret value R with Bob, which is a breach of the protocol. Dave and Bob are now in collusion. Alice, who is unaware of the collusion against her, creates a hashed timelock contract with Bob promising to pay money upon him revealing R. At this point, Bob already knows R, so he executes the contract. Alice still does not suspect anything and considers the transaction to be completed. She expects Dave to deliver the goods/services for which she has paid. Dave "rightfully" claims not to have received the funds. As a result, Alice loses her money to Dave and Bob.

This collusion attack is due to a fundamental flaw in the Lightning Network. Poon and Dryja [9], the authors of the protocol, mention the problem, but do not analyze or attempt to mitigate it:

> "*In the event that R gets disclosed to the participants halfway through expiry along the path, then it is possible for some parties along the path to be enriched. The sender will be able to know R, so due to [the] Pay to Contract, the payment will have been fulfilled even though the receiver did not receive the funds. Therefore, the receiver must never disclose R.*"

This scenario described by Poon and Dryja has the same dependencies and consequences as the collusion attack discussed above. The entire operation of the system relies on the assumption that the receiver is not interested in revealing R. However, as discussed below, there are certain incentives for a receiver to prematurely reveal R.

6. Collusion Attack Implications

The collusion attack has several potential consequences for digital forensics. These consequences are passive or active. The entity making the payment is either the victim of criminal activity or a subject of interest to law enforcement.

6.1 Fraud

A straightforward use of the collusion attack is fraud. After the attack steps are performed as described above, the attacking nodes may become non-responsive. This attack requires the recipient to collude with a node on the route. Of particular interest are the traces left by the attack that remain in the Lightning Network. Unfortunately, the protocol does not impose a requirement to save any information of value. This is one of the fundamental points of the Lightning Network, namely to summarize many smaller transactions into a fewer large ones for communication with and entry into the shared ledger that is the Bitcoin blockchain. Because it is decentralized, the Lightning Network also cannot dictate a particular implementation with sound logging.

6.2 Money Laundering

Another possible use case is to pass money to a recipient on the route via a regular, apparently legitimate, payment system. In this scenario, the sender, the final recipient and an intermediary node collude in order to pretend that a legitimate payment has been made and lost to a rogue intermediary node along the route. The intention is to pass funds to an "unknown" intermediary node under the false pretext of making a legitimate payment. The sender can claim the loss of the funds that were paid but not received by the final intended recipient. This is analogous to the ever popular playing-poker-badly method of money laundering.

6.3 Forfeiture

It is possible for an illegal service to claim no wrongdoing and blame intermediary nodes for the loss of sender funds. For example, in the case of law enforcement sting operations, the police could use the mechanism to intercept illegal funds from a criminal at one of the intermediary nodes as a form of forfeiture [4]; the destination never receives the funds because they have been intercepted by law enforcement.

Although this may be problematic from the law enforcement jurisdictional and procedural law perspectives, it is by no means an impossible scenario. Specifically, an anonymous payment routing protocol renders a traditional sting operation, whose goal is to identify the perpetrator, impossible; as a result, law enforcement can only attempt to disrupt illegal activity instead of prosecuting a suspect. In this scenario the ability of a recipient to show "clean hands" by legitimately claiming that the funds never arrived provides plausible deniability and postpones the recipient from being flagged as a "fake" supplier by a reputation-based system.

7. Attack Mitigation

A straightforward way to mitigate a collusion attack is to require (e.g., via a contract) that the knowledge of R constitutes a proof of payment. One method is via a pay-to-contract scheme [3] that is mentioned by the inventors of the Lightning Network [9]. The obvious problem with this approach – which is not implemented in the Lightning Network at this time – is that it relies heavily on a public-key infrastructure. The same is true of all similar mitigation strategies.

Forensic readiness can be implemented by the verbose logging of transactions; the logs would serve as evidence if something goes wrong. However, logging Lightning Network transactions is not enough. Hashed timelock contracts only appear in commitment transactions and they are signed with temporary keys that are not bound to physical-world identities. This is where a public-key infrastructure is needed. If a Lightning Network node has a SSL certificate, it can sign the commitment transactions that it processes. This enables an entity to prove the obligations of its neighbor nodes. Specifically, the completion of a monetary transfer can be proven by following the route and checking the commitment transactions with the corresponding hashed timelock contracts. All the nodes on the route would have to follow forensically-sound logging procedures, which may be enforced by appropriate regulations. A regulated node can also check that all the neighbor nodes fall in the appropriate jurisdiction. However, this may require the network to be partitioned into regulated and unregulated segments. While the regulated segment would have the desired properties, the presence of an unregulated segment would raise the risk of abuse if the two network segments were to interact.

To conclude, a public-key infrastructure could solve a number of fundamental problems with the Lightning Network and crypto-currencies in general. An example problem is transaction acknowledgement in a decentralized environment. Unless the receiver is a legally recognized electronic entity, the receiver can always claim to have not received funds even if it did receive the funds. Such problems arise due to the nature of a virtual identity and the fact that it is distinct from a physical identity. A public-key infrastructure does bridge the gap between the virtual and physical worlds. However, in the domain of decentralized applications, anonymity and freedom are paramount. Therefore, the mitigations described in this section are not adequate and an appropriate solution remains elusive.

8. Conclusions

Off-chain transactions in the Bitcoin-based Lightning Network increase the likelihood of collusion attacks. These attacks enable payment recipients or merchants to collude with intermediaries to ensure payments to the intermediaries, while claiming that payments were not received by the ultimate receivers. This does not meet the guarantee made by the Lightning Network, where the end state of an initiator of a transaction is that it should end up with no funds and the goods/services or with funds and no goods/services.

Collusion attacks have forensic implications because they enable fraud with very little traceability. Additionally, they enable law enforcement to intercept funds used in illegal transactions for the purpose of forfeiture. In the two scenarios, the fraudulent entities and law enforcement can claim innocence when the (unidentifiable) initiators of the transactions complain that their funds were lost and they did not receive any goods/services.

Digital currencies are increasingly becoming the targets of crime and vehicles for furthering criminal activities. It is hoped that this research will stimulate increased efforts in this new and important area of research.

References

[1] D. Boneh and J. Shaw, Collusion-secure fingerprinting for digital data, *IEEE Transactions on Information Theory*, vol. 44(5), pp. 1897–1905, 1998.

[2] S. Bugiel, L. Davi, R. Dmitrienko, T. Fischer, A. Sadeghi and B. Shastry, Towards taming privilege-escalation attacks on Android, *Proceedings of the Nineteenth Annual Network and Distributed System Security Symposium*, 2012.

[3] I. Gerhardt and T. Hanke, Homomorphic Payment Addresses and the Pay-to-Contract Protocol, arXiv:1212.3257v1 [cs.CR], Cornell University Library, Cornell University, Ithaca, New York (`arxiv.org/pdf/1212.3257v1.pdf`), 2012.

[4] B. Hay, Sting operations, undercover agents and entrapment, *Missouri Law Review*, vol. 70(2), pp. 387–432, 2005.

[5] M. Moharrum, M. Eltoweissy and R. Mukkamala, Dynamic combinatorial key management scheme for sensor networks, *Wireless Communications and Mobile Computing*, vol. 6(7), pp. 1017–1035, 2006.

[6] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System (`bitcoin.org/bitcoin.pdf`), 2008.

[7] M. Nowostawski and C. Frantz, Blockchain: The emergence of distributed autonomous institutions, *Proceedings of the Sixth International Conference on Social Media Technologies, Communication and Informatics*, pp. 29–35, 2016.

[8] C. Pacia, Lightning Network skepticism (`chrispacia.wordpress.com/2015/12/23/lightning-network-skepticism`), December 23, 2015.

[9] J. Poon and T. Dryja, The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments, Draft Version 0.5.9.2 (`lightning.network/lightning-network-paper.pdf`), 2016.

[10] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy and O. Osuntokun, Flare: An Approach to Routing in the Lightning Network, White Paper (`bitfury.com/content/5-white-papers-research/whitepaper_flare_an_approach_to_routing_in_lightning_network_7_7_2016.pdf`), 2016.

[11] F. Reid and M. Harrigan, An analysis of anonymity in the Bitcoin system, *Proceedings of the Third IEEE International Conference on Privacy, Security, Risk and Trust/Social Computing/Workshop on Security and Privacy in Social Networks*, pp. 1318–1326, 2011.