FAKEY: Fake Hashed Key Attack on Payment Channel Networks

Alvi Ataur Khalil*, Mohammad Ashiqur Rahman*, and Hisham A. Kholidy†
*Department of Electrical and Computer Engineering, Florida International University, USA
†Department of Network and Computer Security, SUNY Polytechnic Institute, USA
*{akhal042,marahman}@fiu.edu, †kholidh@sunypoly.edu

***Abstract*—Although blockchain's immutability and decentralized capabilities make monetary transactions more secure than ever, its inherent scalability problem hinders its utilization in myriad applications. Payment channel networks (PCNs), one of the prominent solutions to blockchain's scalability issue, reduce the load on the blockchain by performing the transactions off-chain. However, malicious PCN participants can delay the block delivery by slightly misbehaving in the initialization phase. In this work, we introduce FAKEY, a fake hashed key-based attack on PCN that can block up a whole set of channels carrying a transaction for a certain period (via collateral lock), depending on the hashed timelock contract (HTLCs) in the attacked payment path. For the targeted attack setup, the effect of the FAKEY is more severe, as PCN throughput can be reduced significantly. Moreover, the attacker can gain direct monetary benefit by carefully choosing one or more victim nodes. We utilize one of the state-of-the-art PCN simulators, $PCNsim$, to perform multiple experiments with different attack strategies and calculate profit accumulation scenarios. To demonstrate the attack impacts in detail, we perform a couple of case studies using the simulator. Finally, to validate the attack effect and adversarial benefits in a real-world PCN, we utilize the Bitcoin Lightning Network snapshots spanning over a period of 18 months and calculate the exact monetary impact in satoshi units.**

***Index Terms*—Blockchain; payment channel networks; hashed timelock contract.**

I. INTRODUCTION

Blockchain technology has been widely embraced in the payment industry since the launch of Bitcoin in 2009 [1], [2]. While Bitcoin and Ethereum [3] offer decentralization and pseudonymity, their limitations in terms of throughput and scalability have driven researchers to explore scalable alternatives. One such solution is the payment channel network (PCN), which enables off-chain transactions through a network of interconnected payment channels [4]. The PCN facilitates faster transactions, lower fees, and enhanced anonymity compared to the blockchain network, due to the off-chain nature of transactions. To improve privacy, PCN utilizes methods like onion routing and balance concealment [5].

The concept of a "network" of payment channels arises when two parties without a direct payment channel between them need to transact. In such cases, a network of payment channels connecting the participants is employed, enabling multi-hop payments, i.e., payments that pass through one or more intermediary channels. The payer must identify a path that leads him to the payee in the case of a multi-hop payment. Payment is made with atomic updates (AU) of the channels' balances, which can be achieved by hashed timelock contracts (HTLCs). The process starts with the payee generating the hash of a secret (called the "preimage") value to lock each forward payment, and the payment proceeds if the payee reveals the secret within a timeout. Various types of HTLCs exist, enabling atomic swaps [6], contingent payments [7], high-frequency payment channels [8], and vaults [9].

Unfortunately, HTLCs are susceptible to incentive manipulation attacks [10], [11]. Winzer et al. [12] demonstrated how a receiver could pay miners to disregard transactions until the timeout expires by employing specific smart contracts. Harris et al. [13] showed how the receiver could prevent transactions from being confirmed by overloading the system with his own transactions. Some other works directly leverage the loopholes of HTLC-based payments. For instance, Robinson [14] illustrated an attack where the receiver locks the collaterals of the participating nodes in the payment path by not revealing the preimage. Malavolta et al. [15] proposed another kind of attack where two intermediate hops collude together to deprive one or more hops of the fees for forwarding a PCN transaction. None of these attacks, however, exploit the vulnerability from the manipulated hashed key used for creating hashlock of the HTLCs, which is a crucial aspect for PCNs.

Upon careful examination of the protocol, we discover that simple misbehavior from the sender can incur a significant adversarial impact on the PCN system. Once the transaction route is established, the receiver generates a key by hashing the secret preimage, which is then shared with the sender. The sender is expected to use this key for creating the hashlock of the HTLC, which is subsequently copied by intermediate hops for creating hashlocks of subsequent HTLCs. However, if the sender misbehaves by sending a different hashed key, the transaction will not be completed.

In this work, we propose FAKEY attack for PCNs, where the sender, instead of using the hashed key (i.e., hashed preimage) shared by the receiver, creates a fake key by hashing a forged preimage ("fake-preimage") and uses it to produce hashlock of the HTLCs. The same fake key is used for all the subsequent HTLCs, and when the receiver tries to claim the payment amount from its HTLC by using its preimage, the HTLC remains locked. All the collaterals invested by the intermediate nodes are locked for the timelock period, and they are deprived of their expected fees. By strategically selecting a receiver based on the network's topology and its channel balance, the attacker can directly benefit financially from this attack.

Our primary contributions in this work are fourfold:

- We identify an existing vulnerability of the HTLC protocol and show how it can be leveraged to launch a new kind of attack. We name it the FAKEY attack.
- We analyze the attack impact and define three kinds of adversarial effects on the PCN, one of which incurs direct monetary benefit for the attacker.
- We perform different experiments and case studies, utilizing the state-of-the-art PCN simulator called $PCNsim$ [16], to validate the effectiveness and impact of the proposed attack.
- We leverage the Bitcoin Lightning Network snapshots from an interval of 18 months (from Bitcoin's blocks at blockheight 503.816 to blockheight 585.844) to further validate the attack impact in a real-world PCN.

We discuss necessary preliminary information in Section II. The related works are discussed in Section III. We introduce our proposed FAKEY attack in Section IV. In Section V, we discuss the technical details of the attack. In Section VI, we describe the evaluation setup and explain the empirical analysis and findings. We briefly discuss potential defense techniques in Section VII. At last, we conclude the work in Section VIII.

II. BACKGROUND

In this section, we will provide some preliminary information about PCNs and HTLCs.

A. Payment Channel Networks

A payment channel is a method of conducting cryptocurrency transactions without recording each transaction on the blockchain [17]. In a payment channel, only two transactions are added to the blockchain: the opening and closing transactions. However, numerous transactions can take place between the participants within an active channel.

To establish a payment channel, participants need to invest a certain amount of cryptocurrency as a deposit, which determines the channel's liquidity. Hence, the number of channels a party can open is constrained. The network of payment channels, or PCN, has been proposed to facilitate transactions between parties that do not have a direct payment channel. Payments are conducted through a series of pairwise transactions along a path connecting the parties. For a successful payment, the chosen path must have sufficient liquidity (remaining deposit). When multiple routes are available, the shortest path is typically selected as the payment route. Various routing algorithms, such as Dijkstra's algorithm or Onion routing used in the Lightning Network, are employed based on the specific PCN protocols [18].

B. Hashed Timelock Contract

The HTLC is a prominent smart-contract design pattern setup for two participants, incorporating hashlocks and timelocks. It requires the recipient of a payment to provide cryptographic proof of payment before a deadline, or else forfeit the right to claim the payment, returning it to the payer [19].

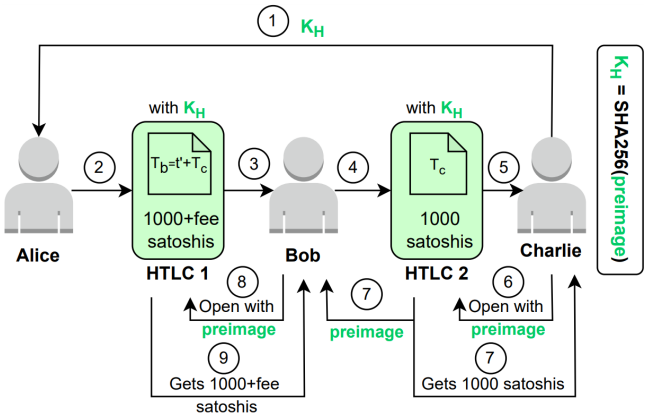


Fig. 1. Payment channel network workflow.

Timelocks are already utilized in payment channels, and adding hashlocks to them is relatively straightforward. HTLCs enable the routing of payments across multiple payment channels, which is a valuable advantage offered by PCNs [5]. This conditional payment mechanism relies on the knowledge of the preimage of a hash value for its success. Atomicity is a crucial feature of multi-hop payment protocols, ensuring that payment channel balances are either fully updated or terminated. Atomicity prevents fund loss issues, and HTLCs are utilized in PCNs to achieve this property.

C. General PCN Transaction Workflow

The standard workflow of a PCN transaction is presented in Figure 1, where Alice wants to buy something from Charlie but lacks a direct payment channel. They find a path through Bob as an intermediary. Charlie generates the preimage, produces its SHA256 hash, and sends it to Alice, who includes it in her payment to Bob along with a condition: in order for Bob to claim the payment, he has to provide the data (i.e., preimage) that was used to produce that hash. Bob, in turn, pays Charlie and attaches the same condition. Charlie has the preimage to finalize the payment and receives it from Bob. By doing so, Charlie reveals the preimage to Bob, who uses it to complete his payment from Alice. Formally, in the first step, Charlie shares the hash (K_H) of the preimage with sender Alice, who uses K_H to create a hashlock for the first HTLC with Bob. The timing constraint for this HTLC ($HTLC\ 1$) is Bob's deadline (T_b) to finalize the payment from Alice. T_b is the summation of Charlie's deadline (T_c) and some extra time t'. Although Alice intended to send 1,000 satoshis to Charlie, she locks additional satoshis as a fee for Bob's service, which is determined by Bob. Bob then uses the same K_H to hash the HTLC ($HTLC\ 2$) with Charlie, setting the timing constraint T_c ($< T_b$) and locking 1,000 satoshis intended for Charlie. Charlie can open '$HTLC\ 2$' by using the preimage before T_c and claim the 1,000 satoshis locked by Bob. This action reveals the preimage to Bob, who can then open '$HTLC\ 1$'. Even if Charlie takes the maximum allowed time (T_c), Bob still has at least t' time to open '$HTLC\ 1$' and claim the 1,000 satoshis plus the fee, resulting in his profit.

TABLE I
RELATED WORKS

Attack	Minimum Node Requirement	Congestion	Loss of Fee	Hashed Key Manipulation
Griefing Attack [14]	1	✓	✓	×
Wormhole Attack [15]	2	×	✓	×
LockDown [20]	1	×	✓	×
General Congestion Attack [21]	Many (Sybil)	✓	✓	×
Flood and Loot [13]	2	×	✓	×
Time-Dilation Attack [22]	Many (Sybil)	✓	×	×
FAKEY	1	✓	✓	✓

III. RELATED WORKS

As PCNs gain popularity for addressing blockchain scalability issues, researchers have focused on analyzing their security vulnerabilities. Various attack techniques have been proposed, each with different impacts on PCNs. One notable attack is the griefing attack proposed by Robinson [14], where the receiver (despite possessing the preimage) intentionally refuses to respond to the HTLC, causing the locked amount to remain inaccessible until the contract's time period expires. Another attack is the wormhole attack [15] proposed by Malavolta et al., where an adversary controls two intermediate hop nodes in a payment route. The adversarial node closer to the receiver obtains the preimage but does not reveal it to the previous neighboring node, leading to the withdrawal of locked amounts by previous nodes. When the other adversarial node is reached (closer to the sender), it reveals the preimage, allowing it to open the HTLC with its previous node. This deprives the nodes between the two adversarial nodes of their fees.

Another attack, known as "LockDown" [20], involves an attacker locking all available balances of a node with its neighbors. The attacker first establishes a channel with the victim and then routes a specific amount of payment through the victim to itself, depleting the victim's channel balances and rendering them unable to conduct further transactions. Lu et al. propose a generalized method for congestion attacks in PCNs [21], where the attacker creates Sybil nodes to initiate multiple multi-hop payments and grieves these payments, causing network congestion and disrupting transactions. "Flood and Loot" is an attack where the attacker sets up sender and receiver roles, establishes channels with victim nodes, and refuses to open HTLCs [13]. Due to blockchain's block size restrictions, many of the channels closing requests by victims will be discarded, which means the attacker will gain the fee, as well as payment amount from myriad failed transactions. Riard et al. proposed the "Time-Dilation Attack" [22], that involves the creation of multiple Sybil PCN nodes to eclipse (isolate) victims from the network and delay block delivery.

None of the previous attacks exploited the manipulation of hashed keys in HTLCs, which is a simple yet significant vulnerability. Among existing attacks, only the "Griefing" and "General Congestion" attacks cause both congestion and loss of fees (presented in Table I). However, the general congestion attack relies on sybil nodes, making it impractical for attackers with limited resources. While the griefing attack shares some similarities with FAKEY in terms of requirements and impact, it overlooks direct financial gain, which is a crucial incentive for rational attackers. The FAKEY attack only requires one colluding node and causes both affects with financial benefits.

IV. FAKEY ATTACK: KEY IDEA

In this section, we will explain the main theoretical idea of the proposed FAKEY attack, along with effects of the attack on the PCN throughput and the attacker's primary gains.

A. Theoretical Definition of FAKEY

In the proposed FAKEY attack, the sender manipulates the hashed key used for creating hashlocks in PCN transactions. By fabricating a "fake-preimage" and generating a corresponding fake hashed key, the sender initiates the attack. This fake key is used in the HTLCs along the payment route, starting from the first hop node. The intermediate hop nodes, unaware of the authenticity of the key, use the fake key in the subsequent HTLCs. As a result, the receiver's attempts to unlock the final HTLC with the genuine preimage fail. The receiver continues attempting to unlock the HTLC until the timelock duration expires. At that point, the neighboring hop of the receiver withdraws the collateral it invested in the transaction. Since the collateral was locked for the timelock duration and the payment was not completed, the hop node is deprived of the expected fee. This incident repeats for all intermediate nodes, and the longer the payment route, the greater the loss due to collateral lock and fee deprivation. The attacker (i.e., the sender) can withdraw the funds it locked in the HTLC after the timelock duration elapses, avoiding any monetary loss from launching the FAKEY attack.

B. FAKey Transaction Workflow

In Figure 2, an example PCN transaction is shown with the FAKEY attack. Alice wants (pretends) to buy something from Charlie for 1,000 satoshis, however, instead of using the hashed key shared by Charlie, she uses a fake-preimage to generate a fake hashed key. Bob unknowingly uses the same fake key, resulting in Charlie being unable to finalize the payment. The preimage is also not revealed to Bob, so he has to withdraw the amount and is denied the fee.

More formally, Charlie shares the hashed key (K_H) with adversarial sender Alice, who ignores it and creates a fake key (K_F). Alice creates hashlock of '$HTLC\ 1$' with this fake key, and consequently, Bob has to use the same K_F to hash the HTLC ($HTLC\ 2$) with receiver Charlie. Despite Charlie possessing the original preimage, he fails to open '$HTLC\ 2$' since its hashlock was created with K_F. Thus, Bob is forced to wait until the deadline T_c to withdraw the collateral and cannot claim the amount from '$HTLC\ 1$' or receive the fees.

C. Attack Effects

There are several motivations for the attacker to launch the FAKEY attack. The primary attack effects are as follows:

- With just random transactions being selected by the attacker to launch the attack, the throughput of the network will be compromised since all the participating hop nodes (in the FAKEY transaction) will have their collaterals

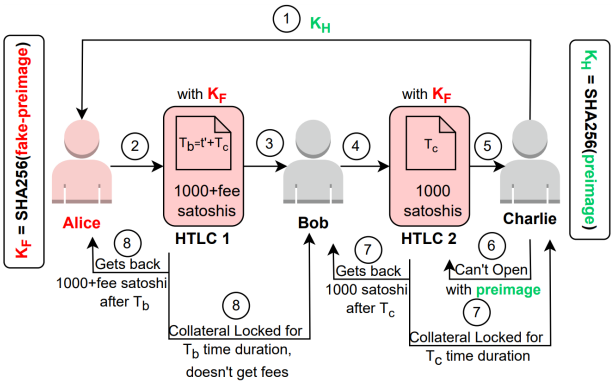


Fig. 2. FAKEY attack in payment channel network.

locked for certain periods and a lot of transactions will fail due to insufficient liquidity in their payment routes.

- By strategically choosing the receiver, the attacker can force non-participating nodes (i.e., not engaged in FAKEY transaction) to pay higher fees and experience transaction delays, indirectly impacting network throughput.
- The attacker can directly benefit financially by eclipsing (i.e., isolating) specific nodes from the network, forcing transactions from those nodes to be routed through the attacker node and gaining control over the associated fees.

All of these attack impacts will be discussed in detail, along with case studies in Section V.

V. TECHNICAL DETAILS

In this section, we will present the technical details of the FAKEY attack by first formally defining a PCN, followed by the attack technique. Later, we will present several case studies to demonstrate the attack and its impact on the PCN.

A. Formally Defining PCN

Since PCNs have users and channels resembling vertices and edges, respectively, we represent it by $P = (N, C)$, a directed graph, where N is the set of nodes representing users and C is the set of channels. The sender and the receiver node are represented by N_s and N_r. If N_s wants to send α satoshis to N_r and there is no direct channel between them, N_s will find a route to N_r and send α_s to its immediate neighbor hop in the chosen payment path, where,

$$\alpha_s = \alpha + \sum_{k=1}^{n} F_k \tag{1}$$

Here, F_k presents the amount node N_k charges for forwarding a payment, assuming that the selected route R_{sr} from N_s to N_r has n intermediate hops (i.e., $R_{sr} = N_s \rightarrow N_1 \rightarrow ... \rightarrow N_n \rightarrow N_r$). The length of any route R is represented by l. In this particular case, $l = n+1$. If, there is a direct channel between N_s and N_r, then l=1 (i.e., $R_{sr} = N_s \rightarrow N_r$).

Each channel C_k ($\in C$) between two nodes (N_i and N_j) in PCN has two directions for forwarding payments. The channels have deposit amounts in each direction, representing the liquidity of the channel in the respective direction. Initially, the deposits are equal in both directions. However, the deposits will update through the continuous transactions between N_i and N_j. The remaining deposit amount at any given time is denoted as D_{ij} and D_{ji}, representing the amounts that N_i can pay N_j and vice versa, respectively. All these notations are summarized in Table II.

The HTLC set by N_i towards N_j is denoted as $HTLC(N_i, N_j, \alpha_i, t_i)_{\mathcal{K}_\mathcal{H}}$, where $\mathcal{K}_\mathcal{H}$ is the key used to create the hashlock of the HTLC, α_i is the amount locked by N_i in the HTLC, and t_i is the time limit for N_j to open the HTLC and claim α_i. Hence, if there is n number of intermediate nodes, then the time-limits of the HTLCs in the payment path have the following relation:

$$t_s > t_1 > ... > t_n \tag{2}$$

where, t_s and t_n are the time-limits set by the sender N_s and the last hop node N_n respectively. Every HTLC's hashlock, along the payment path, has to be created with the same cryptographic key $\mathcal{K}_\mathcal{H}$ generated by N_r, who shares it with N_s, and the key is passed through the intermediate hop nodes to create subsequent hashlocks.

The collateral lock state of the channel between N_i and N_j in the '$N_i \rightarrow N_j$' direction is represented by LK_{N_i,N_j}. The '0' (Zero) value means the channel is no longer locked, while the '1' (One) value means the channel is still locked (HTLC still in place). LK_{N_i,N_j} can be describes as follows:

$$LK_{N_i,N_j} = \begin{cases} 0 & \text{if } \eta^1 \text{ or } (\eta^4 \text{ and } \eta^5) \\ 0/1 & \text{if } \eta^2 \text{ and } \eta^5 \\ 1 & \text{if } \eta^3 \text{ and } \eta^6 \end{cases}$$

Here, η^i represent i-th conditions. The conditions are:

η^1: $t_{curr} > t_i$ $\quad$ η^4: $t_{curr} < t_j (< t_i)$
η^2: $t_j < t_{curr} < t_i$ $\quad$ η^5: $LK_{N_j,N_{j+1}} = 0$
η^3: $t_{curr} = t_j$ $\quad$ η^6: $LK_{N_j,N_{j+1}} = 1$

Here, t_{curr} is the current elapsed time, and N_{j+1} is the next hop after node N_j in the payment path.

The lock variable LK_{N_i,N_j} definitely has a value of '0' if t_{curr} is greater than the HTLC lock-time t_i (condition η^1) set by N_i, indicating that the channel is unlocked and the funds deposited by N_i (α_i) will be refunded. The other situation where the channel definitely becomes unlocked is when both conditions η^4 and η^5 are met. This condition demonstrates the recursive effect of channel locks, where if the current time (t_{curr}) is less than the HTLC lock-time (t_j) set by N_j for the subsequent HTLC with N_{j+1} and the lock state ($LK_{N_j,N_{j+1}}$) of that HTLC is '0', then the current HTLC's lock state LK_{N_i,N_j} can be '0'. To elaborate, condition η^5 means that the next hop HTLC was opened, and condition η^4 is making sure that it was not opened just by the elapsed time; instead, the preimage was used by node N_{j+1}, which in terms revealed the preimage to node N_j, who use it to open the HTLC with node N_i, thus setting LK_{N_i,N_j} to '0'.

On the other hand, LK_{N_i,N_j} definitely has a value of '1' if t_{curr} already equals to t_j and the lock state ($LK_{N_j,N_{j+1}}$) of the subsequent HTLC is '1'. These conditions are represented by η^3 and η^6. The value of LK_{N_i,N_j} remains '1' as long as the current elapsed time (t_{curr}) has not exceeded t_i. If the next hop HTLC is not opened before the timelock deadline, the current

TABLE II
LIST OF NOTATIONS

Term	*Definition*
P	Payment channel network
C	Set of channels in $P(i.e., N \times N)$
N	Set of nodes
N_s	Sender node
$N_s^{\mathcal{A}}$	Adversarial sender node
N_r	Receiver node
α	Payment amount
R_{sr}	Route from N_s to N_r
l	Length of route R
$\mathcal{M}$	Monetary loss for a node
LK_{N_i,N_j}	Lock-state of the channel between N_i and N_j
$\mathcal{K}_{\mathcal{H}}$	Honest hashed key generated by N_r to be used for HTLCs which is calculated by, $\mathcal{H}ash$(preimage)
$\mathcal{K}_{\mathcal{F}}$	Fake hashed key generated by N_s to be used for HTLCs which is calculated by, $\mathcal{H}ash$(fake-preimage)
t_i	Lock-time for the HTLC set by N_i
F_k	Fee that k-th Node (N_k) charges for forwarding a payment
D_{ij}	Deposit remaining in channel between N_i and N_j, in N_i to N_j direction (Amount N_i can pay N_j)
α_s	Initial payment amount by sender N_s

HTLC will remain locked until η^1 occurs, indicating that the elapsed time has passed for the current HTLC.

Finally, the lock state LK_{N_i,N_j} can have either '0' or '1' if t_{curr} has already passed t_j and has not passed t_i, and the lock state $LK_{N_j,N_{j+1}}$ of the subsequent HTLC has value '0'. That is because the unlocking of the next-hop channel can happen either by the opening of the corresponding HTLC using the preimage by node N_{j+1} before t_j (and preimage is revealed to N_j) or after t_j, the channel will get unlocked, and the corresponding HTLC will automatically refund node N_j. If condition η^2 stands, i.e., t_j has been passed, and t_i has not, and next hop lock sate $LK_{N_j,N_{j+1}}$ is '0', then there is no way to certainly find the current hop lock state LK_{N_i,N_j}.

B. FAKey Attack

The proposed FAKEY attack is initiated by an adversarial sender, which we will denote by $N_s^{\mathcal{A}}$. The receiver N_r first hashes the random secret preimage to generate the cryptographic key $\mathcal{K}_{\mathcal{H}}$ and sends it to $N_s^{\mathcal{A}}$, who discards $\mathcal{K}_{\mathcal{H}}$ and generates a fake key $\mathcal{K}_{\mathcal{F}}$ by hashing a random fake-preimage. Now, $N_s^{\mathcal{A}}$ uses $\mathcal{K}_{\mathcal{F}}$ to create hashlock of the HTLC with the first hop node N_1 in the payment path to N_r, i.e., $HTLC(N_s^{\mathcal{A}}, N_1, \alpha_s, t_s)_{\mathcal{K}_{\mathcal{F}}}$. Then N_1 uses $\mathcal{K}_{\mathcal{F}}$ to produce hashlock of the next HTLC, and the same incident happens with all the subsequent hop nodes. If the last hop node is N_n, then the HTLC it creates with the receiver N_r is $HTLC(N_n, N_r, \alpha, t_n)_{\mathcal{K}_{\mathcal{F}}}$ and N_r has time-limit of t_n to open this HTLC by revealing the preimage, that was used to generate $\mathcal{K}_{\mathcal{H}}$. By opening this HTLC, N_r was supposed to claim the α payment and, simultaneously, reveal preimage to N_n. However, since the HTLC's hashlock was created using $\mathcal{K}_{\mathcal{F}}$, rather than $\mathcal{K}_{\mathcal{H}}$, N_r will not be able to open it. As a result, α amount of collateral of the channel C_{n+1} between N_n and N_r will be locked in the '$N_n \rightarrow N_r$' direction for the duration t_n and that channel will not be able to accommodate any other transactions for the time being if $\alpha = D_{nr}$. After time t_n, α will be refunded back to N_n, and N_r will not be able to claim it. Thus, for N_r, the only attack result is the locking of collateral in its channel with N_n, which can be presented as:

$$t_{curr} < t_n \implies LK_{N_n,N_r} = 1 \tag{3}$$

However, for all the intermediate nodes, there is monetary loss (presented by $\mathcal{M}$) along with the collateral locks. For example, node N_n was suppose to claim α_{n-1}. However, it needed the preimage to claim α_{n-1} (which includes the fee F_n) from $HTLC(N_{n-1}, N_n, \alpha_{n-1}, t_{n-1})_{\mathcal{K}_{\mathcal{F}}}$. Since N_n does not have the preimage, α_{n-1} collateral will be locked in its channel with node N_{n-1} for t_{n-1} time. Again, the first hop node N_1 was suppose to claim α_s (Equation 1). However, N_1 will not be exposed to the preimage to open this HTLC because the next hop HTLC with node N_2 will not be opened either (i.e., $HTLC(N_1, N_2, \alpha_1, t_1)_{\mathcal{K}_{\mathcal{F}}}$). Hence, the overall loss for any intermediate hop node N_i will be as follows:

$$\begin{gathered} LK_{N_{i-1},N_i} = 1 \textit{ while } t_{curr} < t_{i-1} \\ LK_{N_i,N_{i+1}} = 1 \textit{ while } t_{curr} < t_i \\ \mathcal{M}_i = F_i \text{ satoshis} \end{gathered} \tag{4}$$

Here, $\mathcal{M}_i$ presents the monetary loss for node N_i. Equation 4 holds only when $l \geq 2$. However, Equation 3 holds even if $l = 1$. The total monetary loss caused by a single FAKEY transaction will be:

$$\mathcal{M}_{total} = \sum_{k=1}^{n} F_k \text{ satoshis} \tag{5}$$

The total loss $\mathcal{M}_{total}$ is positively correlated with the transaction path lentdh l, i.e.:

$$\mathcal{M}_{total} \underset{\sim}{\propto} l \tag{6}$$

This is apparent since the more lengthy the path, the more intermediate hops, resulting in more lost fees.

From the delaying perspective, the neighbor of the adversarial sender $N_s^{\mathcal{A}}$ in the payment path suffers the most because the timelock t_s of the HTLC between $N_s^{\mathcal{A}}$ and the first node is the maximum one (Equation 2). Thus, by selecting a targeted node as the neighbor, $N_s^{\mathcal{A}}$ can maximize its collateral lock duration, which might serve the attacker's purpose.

There are several motivations for the attacker to commit the FAKEY attack (as briefly discussed in Section IV): making the network throttled, forcing regular nodes to spend more, or receiving direct monetary gain. In the following, we discuss these three attack scenarios in detail.

1) Compromise Network Throughput: In the simplistic form of the attack, the attacker $N_s^{\mathcal{A}}$ randomly selects a node as the receiver and initiates a transaction to launch the attack. The attacker's focus is not on direct monetary gain in this case, so the selection of intermediate nodes is not a concern. Instead, the attacker strategically chooses N_r to maximize the impact by selecting the node with the maximum shortest route length l (based on the routing algorithm used, some mentioned in Section II). The targeted node $N_r^{\mathcal{T}}$ and corresponding maximum route length l_{max} from $N_s^{\mathcal{A}}$ can be presented as:

$$\begin{aligned} (N_r^{\mathcal{T}}, l_{max}) = \{(N_i, l_i) | (N_i, l_i) \in \{(N_1, l_1), ..., (N_z, l_z)\} \\ and \quad l_i \geq l_k; \text{where k = 1 to z}\} \end{aligned} \tag{7}$$

Here, it is assumed that z number of nodes are reachable from $N_s^{\mathcal{A}}$ (i.e., $N_1, ..., N_z$), and the shortest distance to each of those nodes is represented by $(l_1, ..., l_z)$. By selecting $N_r^{\mathcal{T}}$, the attacker can lock collaterals at the maximum number of

channels for the timelock duration and deprive the intermediate nodes of gaining fees from executing other regular transactions.

2) Coercing Higher Fees and Delay: In the second type of attack, $N_s^{\mathcal{A}}$ does not directly benefit monetarily, but it forces non-participating neighboring nodes of intermediate hops to incur significantly higher costs for their transactions. By disrupting the channels among the intermediate hops, the attacker prevents the neighbors from using those channels whose collaterals are locked due to the attack. Suppose, length of the shortest route R_{ij} between N_i and N_j, who are not involved in the FAKEY transaction, is l_{min}. However, if one intermediate channel's collateral is locked due to FAKEY attack, the above nodes are forced to select another alternative route R'_{ij}, whose length $l' \geq l_{min}$. For the cases when $l' > l_{min}$, the increased number of hops leads to higher transaction fees and delays in processing time due to the additional HTLCs that need to be created and opened.

3) Monetary Benefit for the Attacker: In the most sophisticated form of the attack, $N_s^{\mathcal{A}}$ directly benefits monetarily. By strategically selecting N_r in accordance with the network topology, $N_s^{\mathcal{A}}$ can eclipse specific nodes, preventing them from using their immediate neighbors' channels. As a result, these eclipsed nodes are forced to route their legitimate payments through $N_s^{\mathcal{A}}$. The attacker then charges fees for forwarding these payments, generating additional profits. It is important to note that the victim nodes are not directly included in the payment path but are neighbors of the intermediate hops.

In the next subsection, we will discuss two case studies, which will help understand the *Coercing Higher Fees and Delay* (CHFD) effect and the *Monetary Benefit for the Attacker* (MBA) effect in a more comprehensive manner.

C. Case Studies

In two case studies presented in Figure 3, we illustrate the attack techniques and their impacts. The first case demonstrates the CHFD effect, where the attacker causes a user to pay a higher fee and experience transaction delays without gaining direct benefits. The second case showcases the MBA effect, where an eclipsed node is compelled to route its transaction through the attacker, resulting in the attacker receiving fees that would not be obtained otherwise. Figure 3 displays a small PCN represented by a directed graph with six nodes, where regular nodes (numbered from N_2 to N_6) are presented with green circles, while the adversarial node N_1 is marked using a red circle. The channels, labeled C_1 to C_8, are represented by black lines with arrows indicating payment directions. The remaining deposit amount in each channel in either direction, denoted as D_{ij}, reflects the capacity for N_i to pay N_j.

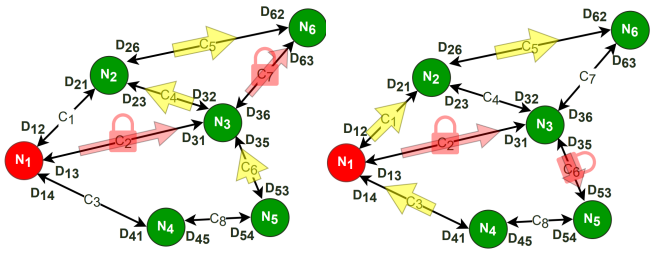
1) Case 1: The CHFD effect is demonstrated in Figure 3(a), where the adversarial node N_1 initiates a transaction with N_6, taking the shortest route R_{16} through N_3. By launching the FAKEY attack, N_1 locks collaterals in the intermediate channels. If the HTLCs set up by N_1 and N_3 are $HTLC(N_1, N_3, \alpha_1, t_1)_{\mathcal{K}_{\mathcal{F}}}$ and $HTLC(N_3, N_6, \alpha_3, t_3)_{\mathcal{K}_{\mathcal{F}}}$ respectively, then C_2's and C_7's collaterals will be locked till t_1 and t_3 passes respectively. At this point, N_5 tries to send α' to N_6, and the shortest route R_{56} for the transaction would have been: $(N_5 \rightarrow N_3 \rightarrow N_6)$, via channel C_6 and C_7. However, if $t_{curr} < t_3$, and $\alpha' > D_{36} - \alpha_3$, then C_7 cannot be used, and an alternative shortest route $(N_5 \rightarrow N_3 \rightarrow N_2 \rightarrow N_6)$ is taken. Thus, although N_5 was only supposed to pay F_3 satoshis as the fee for forwarding the payment, now he has to pay $F_3 + F_2$ satoshis. Hence the extra expenditure is:

$$\mathcal{M}_5 = F_2$$

The monetary loss of a node who is not even involved in the actual FAKEY transaction, not to confuse with the monetary loss of the involved intermediate hop's (N_3) fee deprivation for $HTLC(N_1, N_3, \alpha_1, t_1)_{\mathcal{K}_{\mathcal{F}}}$:

$$\mathcal{M}_3 = F_3$$

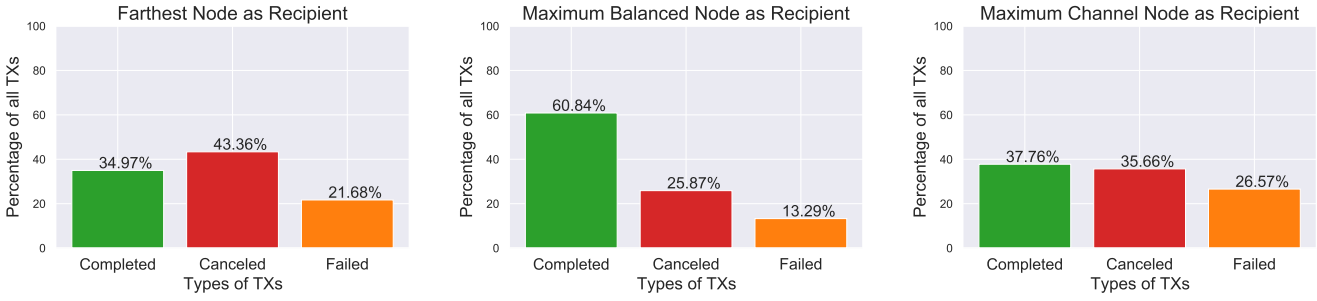
This monetary loss was discussed through Equation 4, and it is out of the scope of the CHFD effect discussion. In this type of attack, adversary N_1 does not directly benefit monetarily; however, the attacker forces some of the neighboring nodes (N_5) of the intermediate hops (N_3) to spend substantially more (F_2) to perform a payment. Moreover, the payment execution time is also increased, which is positively correlated with the alternative route length.



(a) (b)

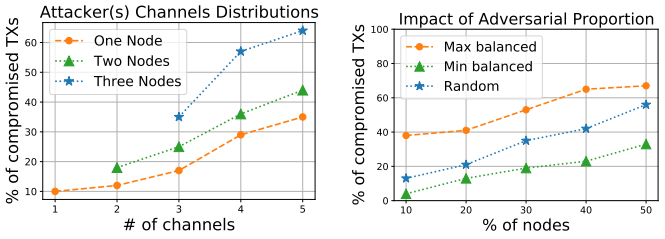
Fig. 3. Case studies to present the (a) Coercing higher fees & delay effect and (b) Monetary benefit for the attacker effect.

2) Case 2: The MBA effect is demonstrated in Figure 3(b), where adversarial N_1 initiates a transaction with N_5, taking the shortest route R_{15} through N_3. If the HTLCs set up by N_1 and N_3 are $HTLC(N_1, N_3, \alpha_1, t_1)_{\mathcal{K}_{\mathcal{F}}}$ and $HTLC(N_3, N_5, \alpha_3, t_3)_{\mathcal{K}_{\mathcal{F}}}$ respectively, then C_2's and C_6's collaterals will be locked till t_1 and t_3 passes respectively. When N_4 attempts to send α' to N_6, there are two alternative shortest routes: $(N_4 \rightarrow N_5 \rightarrow N_3 \rightarrow N_6)$ and $(N_4 \rightarrow N_1 \rightarrow N_2 \rightarrow N_6)$. If $\alpha' > D_{53}$ and N_4 can wait till D_{53} increases (via transactions), it may choose the first option. However, if $\alpha' > D_{35} - \alpha_3$ and N_4 needs to send it before t_3 elapses, the second route involving N_1 becomes the only option. Thus, even if N_1 charges substantially more for forwarding the payment than the first alternative, N_4 is forced to use the attacker as an intermediate hop. Thus, the attacker is gaining some monetary benefit it was not supposed to receive. Moreover, for another scenario where N_5 wants to transact with N_6 before t_3 passes, the route length would have also increased. This scenario is not discussed since the monetary benefit for the attacker would have been the same. In this attack, the adversary N_1 can choose the receiver (N_5), the payment path to whom will eclipse particular nodes (N_4 and N_5), who will be forced to route



(a) (b) (c)

Fig. 4. The average percentage of completed, canceled, and failed transactions with FAKEY attack in a network of 10 nodes (3 adversarial nodes) having (a) The farthest node as the recipient node, (b) The maximum balanced node as the recipient node, and (c) The maximum channel node as the recipient node. The average was calculated from 10 unique topologies.



(a) (b)

Fig. 5. Percentage of transactions that are compromised (incomplete) due to FAKEY attack with (a) different distribution of adversarial channels, and (b) different percentage of adversarial nodes having alternative node selections.

their payment through the attacker node for a particular time interval (till t_3).

VI. EVALUATION

In this section, we conduct experimental analysis using the state-of-the-art simulator called $PCNsim$ [16] to validate the attack impact on PCN. We define various terminal states of PCN transactions and select different target nodes as recipients to observe the attack's effect. Additionally, we experiment with varying collateral investments and measure the profit gained from launching the FAKEY attack. Then we compare the attacker's costs for launching different attacks. Finally, we validate the attack impact by considering the Bitcoin Lightning Network snapshots from January 2018 to July 2019.

A. Transaction Types

***Canceled Transactions*:** Transactions directly obstructed by the FAKEY attack are categorized as $Canceled$ payments.

***Failed Transactions*:** Transactions that cannot be completed due to insufficient liquidity in the payment path are considered $Failed$ transactions. If N_r is reachable from N_s, the transaction can only proceed if the minimum remaining deposit of all channels in the route is greater than the α. This can happen even without the FAKEY attack, however, the attack affects the number of $Failed$ transactions as the remaining deposit of channels depends on the successful execution of transactions.

***Completed Transactions*:** The transaction that are successfully executed, without being $Canceled$ or $Failed$, are referred to as completed transaction

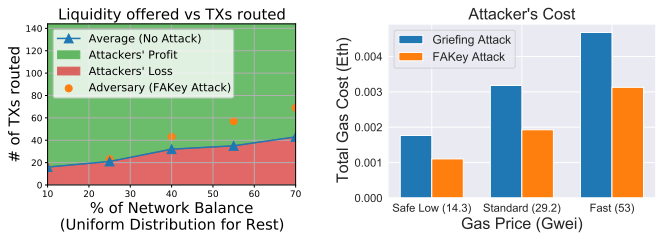
B. Attack Impact w.r.t. Target Recipient

In this section, we analyze the impact of the FAKEY attack on incomplete transactions ($Canceled$ and $Failed$) with different target nodes. We consider three types of target nodes as recipients based on the PCN's topology: the farthest node from the attacker, the maximum balanced node, and the maximum channeled node (presented in Figure 4). Using the workload and topology files of the $PCNsim$ simulator, we determine the target nodes and obtain average results from 10 experiments.

In the case of the farthest node (Figure 4(a)), approximately 43% of the transactions were canceled, and around 21% of the transactions failed. This resulted in over 65% incomplete transactions, the highest among the three cases. It is apparent that choosing the farthest node ensured the involvement of numerous channels in the FAKEY transaction led to collateral locks, causing many transactions to fail. For the maximum balanced node case (Figure 4(b)), over 60% of the transactions were completed, the highest completion rate among the cases. Canceled transactions decreased to less than 26%, and failed transactions reduced to 13%. However, our hypothesis that targeting the maximum balanced node would yield favorable results proved to be invalid based on the experimental results. In the case of the maximum channeled node (Figure 4(c)), nearly 63% of the transactions were incomplete, comparable to the farthest node case. In the incomplete portion, more than 43% contribution is from the failed transactions (more than 26% of all the transactions). Targeting this node, which serves as a junction point for different sub-regions, resulted in a significant number of both canceled and failed transactions. Although it had a slightly lower rate of incomplete transactions compared to the farthest node case, it was a stealthier choice due to the higher proportion of failed transactions.

C. Performance Analysis of Different Attack Strategies

In this section, we examine the attack impacts (presented in Figure 5) based on the capabilities of the attackers. In Figure 5(a), we investigate the influence of the number of channels set up by adversarial nodes. The orange dashed line represents the case of a 'single' adversarial node. As the number of adversarial channels increases, the percentage of compromised (incomplete) transactions shows a linear growth. With 'one' adversarial channel, approximately 10% of transactions are compromised, while with five channels, over 35% of transactions are compromised. Next, we distribute channels among 'two' adversarial nodes, as shown by the green dotted



(a) (b)

Fig. 6. (a) Calculation of profit for a FAKEY attacker with respect to the balance invested by comparing with a non-attacking regular node, where the balance for the rest of the network is distributed uniformly. (b) Contrasting Attacker's cost for deploying the griefing attack and FAKEY attack.

line. The total number of adversarial nodes ranges from two to five. The line starts from value 2 on the x-axis because each node must have at least one channel with another node in the network. The steeper slope compared to the single node case demonstrates the higher impact of two adversarial nodes. With five channels from the two adversarial nodes, nearly 45% transactions are compromised. This suggests that distributing channels among multiple adversarial nodes is a more effective strategy. To firmly establish this assumption, we experiment with 'three' nodes (blue dotted line), and we observe an even improved result, as with 5 channels of the three adversarial nodes, close to 67% of the transactions are compromised.

In Figure 5(b), we explore the impact of different percentages of adversarial nodes and their balance. The green dotted line represents the choice of minimum balanced nodes as attackers. With 10% of the nodes having the minimum balance selected as attackers, approximately 2% of transactions are compromised. As the number of adversarial nodes increases, the percentage of compromised transactions escalates to 32%. The orange dashed line represents the choice of the most balanced nodes as adversarial nodes. Even with only 10% of nodes being adversarial, nearly 38% of transactions are compromised. However, with 50% attackers, 67% of transactions are compromised, indicating that the increase in incomplete transactions is not as pronounced as in the minimum balanced case. Randomly selecting nodes as attackers showed a steeper progress in attackers' success, reaching approximately 55% compromised transactions with 50% attackers, similar to the maximum balanced case. In a real-world scenario, the random choice of attackers is more probable due to varying balances.

D. Impact of Balance on Profit

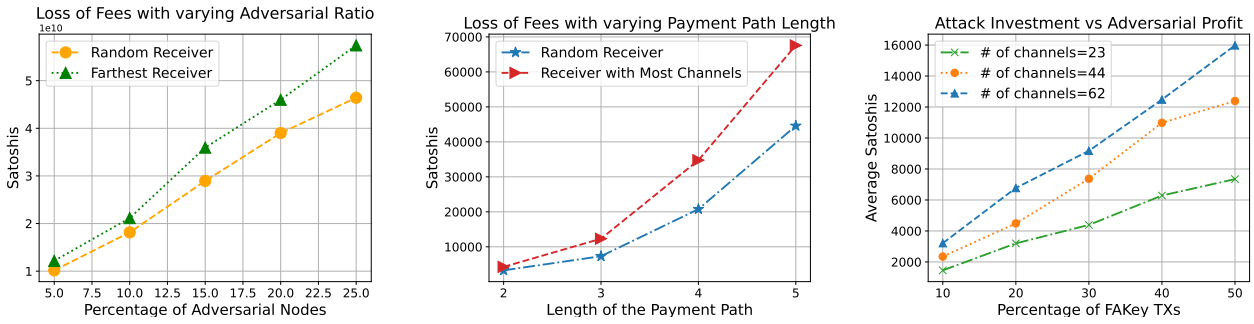
In this section, we examine the relationship between invested channel deposits and the profit earned from transaction fees via those channels. We also analyze the impact of the FAKEY attack in augmenting the profit. We measure the number of transactions routed through a node, and the actual profit amount can be calculated by multiplying it by the exact fee amount per transaction. In Figure 6(a), the blue line represents the number of transactions routed through an average node that does not launch any attack. We allocate 10% to 70% of the network's total balance to this node, distributing the remaining balance 'uniformly' among other nodes. We observe that with a 70% balance share, this node handles over 40 transactions out of the 144 transactions in the simulation. We divide the regions into two parts: the upper region denotes profit, and the lower region denotes loss for an attacker. We grant an attacker with a similar amount of balance share. With a 10% balance share, the number of transactions is similar for both the attacker and the average node. However, with a 70% balance share, the attacker has approximately 70 transactions routed through their node.

E. Contrasting Attacker's Cost

In this section, we compute the gas cost required for the attacker to launch the FAKEY attack, and then compare it with the Griefing Attack [14] since this is the only existing attack that causes both the "congestion" and "loss of fee" affects, with only one adversarial party. We implemented the attacks using Solidity language on the Remix Online IDE [23] and deployed them on the Sepolia testnet. We set the gas limit to 30k per transaction, and the gas price is set as 14.3 (Safe Low), 29.2 (Standard), or 53 (Fast) Gwei according to different execution speeds [24]. In Figure 6(b), we observe that FAKEY required substantially less Ether to launch compared to the Griefing attack, regardless of the gas price. This is because Griefing attacker needs to execute the protocol smart contracts (generating, hashing, and sending of key), while FAKEY attacker can just ignore the smart contracts (receiving the key) and simply create a fake-key and forward that, i.e., much less computation. Consequently, the gas requirements for the Griefing attack ranged from 88k to 126k, while for the FAKEY attack, the range was 59k to 77k.

F. Validation of Attack Impact using Snapshots

In this section, we analyze the impact of the FAKEY attack using real-world PCN network data from the Bitcoin Lightning Network. The dataset includes snapshots of the network's routing table taken every 15 minutes over 18 months, from January 12th, 2018, at blockheight 503.816, to July 17th, 2019, at blockheight 585.844 [25]. Figure 7 provides observations from experiments conducted on the Lightning snapshots. In Figure 7(a), the average loss of intermediate hops in a FAKEY transactions is calculated. We randomly choose a percentage of nodes as attackers and execute one FAKEY transaction from each. Two cases are compared: randomly selecting a victim receiver and selecting the farthest node as the victim. The results show that choosing the farthest node yields significant benefits compared to random selection. Figure 7(b) examines the average loss of intermediate hops based on different path lengths in FAKEY transactions, where one randomly chosen attacker is considered. Comparing the cases where the victim is randomly chosen or the node with maximum channels at the same distance is selected, a polynomial trend is observed, indicating that choosing a farther node is preferable. Furthermore, selecting a victim with more channels has a greater impact on the network, as it leads to more failed transactions. In Figure 7(c), the attackers' monetary profit is plotted against the invested "effort" in launching FAKEY transactions. Attacker nodes are chosen based on the number of channels they



(a) (b) (c)

Fig. 7. Experimentation with the Bitcoin Lightning Network snapshots (Average from ten iterations to smoothen the curves): (a) Average loss of fees in the network with variable percentage of adversarial nodes, (b) Average loss of fees in the network with variable payment path length for the compromised transaction, and (c) Adversarial profit given the attack investment with respect to the percentage of transactions compromised.

possess. The results show that attackers with the maximum number of channels can gain the highest profit, as they can eclipse a larger number of victim nodes and force them to route transactions through the attacker.

VII. Potential Defense Technique

One candidate solution could have been through asymmetric cryptography, where the hashed key will be signed with N_r's private key for verification. However, it will break the anonymity property of PCN. The potential solution could be sharing a secret γ from N_s to N_r, which will be used to encrypt the $\mathcal{K}_\mathcal{H}$ to generate ${\mathcal{K}_\mathcal{H}}^\gamma$. N_r will send ${\mathcal{K}_\mathcal{H}}^\gamma$ in the reverse direction of the payment path, through the hops and upon reaching the end, $\mathcal{K}_\mathcal{H}$ will be revealed to N_s, who has to add the γ information in the HTLC. In that way, the intermediate hops can verify the authenticity of $\mathcal{K}_\mathcal{H}$ since they already possess ${\mathcal{K}_\mathcal{H}}^\gamma$. In our future work, we will validate the effectiveness of the proposed defense technique.

VIII. Conclusion

In this work, we introduced a novel attack technique, which allows an adversarial sender to throttle the PCN transaction throughput. We conducted a comprehensive formal analysis of the attack and its impacts, highlighting the differences from existing attacks. Through case studies and state-of-the-art simulations, we demonstrated the specific impacts of the attack, including monetary loss and transaction compromise. We found that targeting the farthest node as the recipient had the maximum impact. Furthermore, we explored different attack approaches and observed that distributing the adversarial channels among multiple nodes compromised more transactions. We also investigated the relationship between balance investment and profit for adversarial nodes, revealing that FAKEY enables them to gain more profit. Finally, we validated the attack's impact using real-world Bitcoin Lightning Network snapshots, analyzing the monetary benefits and fee losses for victim nodes due to the FAKEY attack.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[2] A. A. Khalil, J. Franco, I. Parvez, S. Uluagac, H. Shahriar, and M. A. Rahman, "A literature review on blockchain-enabled security and operation of cyber-physical systems," in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2022, pp. 1774–1779.

[3] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, 2014.

[4] L. Gudgeon *et al.*, "Sok: Layer-two blockchain protocols," in *Financial Cryptography and Data Security*. Springer, 2020.

[5] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[6] M. H. Miraz and D. C. Donald, "Atomic cross-chain swaps: development, trajectory and potential of non-monetary digital token swap facilities," *arXiv preprint arXiv:1902.04471*, 2019.

[7] S. Bursuc and S. Kremer, "Contingent payments on a public ledger: models and reductions for automated verification," in *European Symposium on Research in Computer Security*. Springer, 2019, pp. 361–382.

[8] M. Green and I. Miers, "Bolt: Anonymous payment channels for decentralized currencies," in *ACM SIGSAC CCS*, 2017.

[9] I. Tsabary *et al.*, "Mad-htlc: because htlc is crazy-cheap to attack," in *IEEE SP*. IEEE, 2021.

[10] J. Bonneau, "Why buy when you can rent?" in *International Conference on Financial Cryptography and Data Security*. Springer, 2016.

[11] P. McCorry *et al.*, "Smart contracts for bribing miners," in *International Conference on Financial Cryptography and Data Security*. Springer.

[12] F. Winzer, B. Herd, and S. Faust, "Temporary censorship attacks in the presence of rational miners," in *EuroS&PW*. IEEE, 2019.

[13] J. Harris and A. Zohar, "Flood & loot: A systemic attack on the lightning network," in *Proceedings of the 2nd ACM AFT*, 2020.

[14] D. Robinson, "Htlcs considered harmful," in *Proc. Stanford Blockchain Conf.*, 2019.

[15] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability," *Cryptology ePrint Archive*, 2018.

[16] G. A. F. Rebello, G. F. Camilo, M. Potop-Butucaru, M. E. M. Campista, M. D. de Amorim, and L. H. M. Costa, "Pcnsim: A flexible and modular simulator for payment channel networks," in *IEEE INFOCOM 2022*.

[17] "Bitcoin wiki pcn," https://en.bitcoin.it/wiki/Payment_channels.

[18] S. P. Kumble, D. Epema, and S. Roos, "How lightning's routing diminishes its anonymity," in *The 16th International Conference on Availability, Reliability and Security*, 2021, pp. 1–10.

[19] "Bitcoin wiki," https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts.

[20] Pérez-Sola *et al.*, "Lockdown: Balance availability attack against lightning network channels," in *FCDS*. Springer, 2020.

[21] Z. Lu, R. Han, and J. Yu, "General congestion attack on htlc-based payment channel networks," *Cryptology ePrint Archive*, 2020.

[22] A. Riard and G. Naumenko, "Time-dilation attacks on the lightning network," *arXiv preprint arXiv:2006.01418*, 2020.

[23] "Remix project," https://remix-project.org/.

[24] "Gas price," https://ethgasstation.info/.

[25] J.-H. Lin, E. Marchese, C. J. Tessone, and T. Squartini, "The weighted bitcoin lightning network," *Chaos, Solitons & Fractals*, 2022.