IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 27 June 2025, accepted 22 July 2025, date of publication 28 July 2025, date of current version 1 August 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3592816

RESEARCH ARTICLE

LIoTning: A Peer-Supervised Payment Channel Approach for Secure Micropayments in IoT Ecosystem

ERFAN BAHRAMI, AMIR M. AGHAPOUR, AND MORTEZA AMINI
Department of Computer Engineering, Sharif University of Technology, Tehran 14588-89694, Iran
Corresponding author: Morteza Amini (amini@sharif.edu)

ABSTRACT The rapid proliferation of the Internet of Things has heightened the demand for efficient, low-cost micropayments to support seamless service provisioning—a requirement fundamentally at odds with the inherent scalability issues of blockchain technology. Digital assets like Bitcoin, despite offering a decentralized and cryptographically secure foundation for the machine-to-machine (M2M) economy, struggle to accommodate high-frequency transactions. While payment channel networks (PCNs) propose an innovative off-chain solution to bypass blockchain bottlenecks, they remain impractical for resource-constrained IoT devices, where stringent storage, memory, and processing restrictions make PCN architectures infeasible. Previous studies have attempted to integrate off-chain payments with IoT devices while addressing threats in this domain—particularly broadcasting old states. However, their reliance on third-party entity/entities, software modifications, trusted hardware, or overhead-intensive protocols leaves critical gaps unresolved. To bridge this gap, we propose LIoTning, a peer-supervised payment channel protocol with $N + 2$ participants operating under the honest majority assumption: two devices (sender and receiver) and N peers, including resource-efficient IoT devices and blockchain nodes—that collaboratively track and maintain full-state commitment transactions. To minimize storage and communication overhead, each full-state commitment embeds intermediate transactions. We formally prove the security of the LIoTning protocol in the Universal Composability (UC) framework, analyze its resilience against broadcasting revoked states, collusion, and ransom attacks, and implement the proposed model to evaluate its performance under different scenarios. Our evaluation results show that the model enables fast payments over low-bandwidth wireless networks. It demonstrates that, with practical parameters, the proposed model can achieve a throughput of at least 200 transactions per second with a network bandwidth of 300 kBps, which is fairly within the capabilities of IoT devices.

INDEX TERMS Blockchain, Internet of Things, lightning network, micropayment, payment channel.

I. INTRODUCTION

The Internet of Things (IoT) encompasses a network of diverse interconnected electronic devices [1]. Over the past decade, IoT has been rapidly adopted across various domains [2], including smart cities, smart homes, smart grids, and e-healthcare [3], [4], [5]. Enabling data and service sharing is crucial to establishing a buoyant and effective IoT ecosystem, which requires device-to-device payments [6]. Since these payments are small in amount with a generally high frequency, they are referred to as micropayments [7]. The concept of micropayments is fundamental in the digital realm, where larger payments for small, recurring services may not be acceptable or efficient.

Blockchain technology has revolutionized the transfer of digital assets by shifting transactions from centralized entities to a decentralized network. Based on this, some users prefer cryptocurrencies over traditional methods due

The associate editor coordinating the review of this manuscript and approving it for publication was Chakchai So-In.

to their privacy, anonymity, or personal preference. Thus, integrating IoT devices with cryptocurrencies offers a viable and effective solution for such payments [8]. Bitcoin stands out among all cryptocurrencies for its significant potential in IoT micropayments, driven by its widespread adoption—with over 170 million users[1]—and its market dominance—58.4% of the total cryptocurrency market capitalization[2]—at the time of writing this paper.

Despite blockchain's innovative features, such as security and immutability, distributed ledger-based digital currencies face challenges like high transaction fees and low throughput, which limit their *scalability* and make them less appealing for IoT micropayments [9]. For instance, Bitcoin and Ethereum only support a throughput of 7 and almost 15 transactions per second, respectively [10], [11], which is far lower than what Visa[3] or MasterCard[4] can process. As numerous works [12], [13], [14] have shown, the core challenge of blockchain scalability lies in the efficiency of consensus protocols.

Layer 2 scaling solutions [15] have been introduced to address blockchain scalability problems. These solutions, particularly payment channel networks (PCNs) [16], act as an additional layer on the blockchain, creating off-chain communication channels that bypass publishing transactions in the main blockchain and eliminate the need for direct verification through consensus protocols. This approach significantly reduces payment delays and blockchain interactions. For example, Lightning Network (LN) [17] has been introduced as a PCN solution over the Bitcoin blockchain. However, nodes in these networks must meet specific requirements, including robust availability, sufficient storage capacity to load and store blocks (e.g., 400 GB storage is required for running a full Bitcoin node), and relatively high computing power [18]. Such assumptions are challenging to guarantee for IoT devices due to their resource-constrained nature to operate as a full node. Based on these considerations, it is essential to design a solution that enables resource-constrained IoT devices to make off-chain transactions as IoT micropayments while satisfying the security requirements of both channels and devices.

Given the widespread adoption and reputation of the Lightning Network, our proposed solution draws inspiration from its design and extends it by utilizing available devices for securing micropayments among IoT devices. To this end, our proposed model utilizes full-state commitment transactions and intermediate transactions within a peer-supervised payment channel consisting of N peers and two devices under the *honest majority assumption*. Full-state commitment transactions act as channel state updates involving $N + 2$ participants, and each full-state commitment transaction may encompass multiple intermediate transactions. The specific contributions of this work are as follows:

- We propose a peer-supervised payment channel protocol that operates under the honest majority assumption, which does not require either the sender or receiver devices to be constantly online to monitor the blockchain. Moreover, it removes the need to outsource this task to a single third party (whether trusted or untrusted) by distributing it across peers. As a result, it eliminates the need for constantly online and available third-party services.
- We employ a novel script that enables a device to claim its funds immediately—without waiting for a timelock—when the latest channel state is broadcast, provided that a majority of peers give their approval.
- We introduce intermediate transactions within full-state commitment transactions to reduce the overhead of creating new full-state commitment transactions and to mitigate the impact of network latency. Additionally, we prevent the loss of a device's funds when it is offline by pre-signed revocation transactions and sharing them with the channel's peers.
- We analyze the security of our protocol against broadcasting revoked states, collusion, and ransom attacks, and show that it is secure against those attacks. Furthermore, we formally prove the security of our LIoTning protocol in the Universal Composability (UC) [19] framework.
- To assess the effectiveness of the proposed protocol, we implement a proof-of-concept of it. The implementation uses the Bitcoin test network to create and close the channel with a designed locking script associated with the funds. In this way, we can ensure the practicality of the solution.

The rest of the paper is organized as follows: Section II provides the background and relevant preliminaries for this work. Section III discusses related work in this area and highlights their limitations. Section IV presents our proposed peer-supervised payment channel protocol, which enables IoT micropayments. In Section V, we analyze the security of the proposed protocol and give a proof sketch in the UC framework. Section VI presents the implementation of the proposed protocol and evaluation results. Finally, Section VII concludes the paper and outlines future research directions.

II. PRELIMINARIES

This section provides an overview of the preliminaries used in this paper, including vertically and horizontally integrated IoT, types of payments, payment channel networks, and especially the Bitcoin Lightning Network.

A. VERTICALLY AND HORIZONTALLY INTEGRATED IOT

IoT environments mainly consist of three components: sensors, computation, and actuators. Their implementations vary significantly based on the application and use case. In a vertically integrated IoT, also referred to as a vertical silo [20], each application operates independently with its

[1] https://blog.chainalysis.com
[2] https://coinmarketcap.com
[3] https://annualreport.visa.com
[4] https://investor.mastercard.com

TABLE 1. Payment types and their characteristics.

Type	Volume	Frequency	Recipient presence	Posting time	Example
Wholesale Payment	Very High / High	Low	×	Delayed	Transactions between two factories involving large-scale purchase of raw materials
Mass Payment	High/Medium	Medium / Low	×	Generally delayed	Salary paying across multiple bank accounts
Retail Payment	Medium / Low	Generally high	×	Generally instant	Grocery shopping at the supermarket
Micro Payment	Very low / Low	Generally high	✓	Instant	1- Charging electric vehicles 2- Tolling pay
Streaming Payment	Very low / Low	Very high (continuously or periodically)	✓	Instant	1- Pay per second of video content 2- Weather station payments to soil moisture sensor

own components, network connectivity, and computation/-control algorithm [21]. In contrast, a horizontal integration platform facilitates resource and data sharing across verticals at a reduced cost while enabling IoT devices from different administrative domains to collaborate dynamically and create new applications. For instance, a local smart irrigation system could utilize data from an air quality station to optimize water usage based on humidity and particulate matter levels. In this research, we adopt a horizontally integrated IoT approach to support micropayments between devices belonging to different administrative entities. Although developing a fully interoperable, horizontally integrated IoT environment involves numerous technical challenges, this paper focuses specifically on implementing transactions between IoT devices.

B. PAYMENT TYPES

Payments can be categorized based on various factors, including volume, frequency, recipient presence, and posting time. Table 1 provides a comprehensive comparison of different payment types. Notably, this research focuses on micropayments and streaming payments, which have low volume and generally high frequency.

C. PAYMENT CHANNEL NETWORKS

Payment channel networks (PCNs) are a solution to the blockchain scalability problem. Payment channels are a specific type of state channels in which the traded assets are coins [22]. The concept was first introduced by Spilman [23], and later DMC [24] proposed the first duplex payment channel on Bitcoin. In the payment channel, an initial on-chain transaction is used to deposit funds into a multi-signature address jointly controlled by both parties. After this on-chain step, subsequent transactions occur off-chain by exchanging off-chain transactions and signatures that reflect the updated deposit state. Finally, the channel is closed with another on-chain transaction, which settles the funds based on the last recorded state. Therefore, a payment channel enables the transfer of coins between two users without settling every transaction on the blockchain. This characteristic significantly lowers payment latency by making transaction speed dependent on users' communication latency. The fundamental concept behind PCNs is that if two parties lack a direct payment channel between each other, they can still send and receive payments through the network, which is formed by connecting multiple payment channels between intermediate parties. Another key point is that a payment channel's security and non-custodial properties rely on the main chain's consensus protocol [15].

1) LIGHTNING NETWORK

The Bitcoin Lightning Network (LN) [17], proposed in 2016, is a bidirectional payment channel protocol that operates in three phases: (i) *opening channel*: To open a channel, two parties create a funding transaction that locks funds into a 2-of-2 multi-signature address controlled by their private keys. (ii) *exchanging payments*: Once the channel is established, channel participants exchange payments through commitment transactions, which are signed updates to the channel's balance. (iii) *closing channel*: The channel can be closed either cooperatively via a settlement transaction that directly spends the funding transaction or unilaterally by broadcasting the commitment transactions to the Bitcoin blockchain. For example, consider two parties, A and B, initiating a channel Ch_{AB}. After creating the funding transaction tx_{Fu}, they exchange their first commitment transactions: A holds $tx^{A}_{\text{Cm},1}$ which is signed by the signature $\sigma^{B,A}_{\text{Cm},1}$, while B holds $tx^{B}_{\text{Cm},1}$ which is signed by the signature $\sigma^{A,B}_{\text{Cm},1}$. As the channel state advances, they exchange updated commitment transactions (e.g., $tx^{A}_{\text{Cm},2}$ and $tx^{B}_{\text{Cm},2}$) and revoke previous states by sharing revocation keys $sk^{A,B}_{\text{Rv},1}$, $sk^{B,A}_{\text{Rv},1}$. This process continues until the channel is closed, either mutually or unilaterally.

2) REVOKED STATES

The transaction flow of commitment transactions in the Lightning Network is illustrated in Fig. 1. During unilateral closure, if party A broadcasts the latest commitment transaction (e.g., $tx^{A}_{\text{Cm},k}$), A can claim its funds after a timelock ΔT expires, and B can claim its share immediately as shown in

the first and third paths. However, if A maliciously broadcasts a revoked state (e.g., $tx^{A}_{\mathrm{Cm},i}$, $1 \leq i < k$), two scenarios may occur. If B is online, it can scan the blockchain, use the corresponding revocation key $sk^{A,B}_{\mathrm{Rv},i}$ to revoke the old state and claim both its own share (υ_B) and A's channel funds (υ_A) as a penalty. If B is offline during the timelock period, it irrevocably loses its funds. This highlights two critical limitations of the Lightning Network: (1) offline participants are at risk of losing funds due to their inability to respond during timelocks, and (2) even honest parties must wait to access their funds after broadcasting the latest state. Addressing these limitations is a part of our contribution to this work.

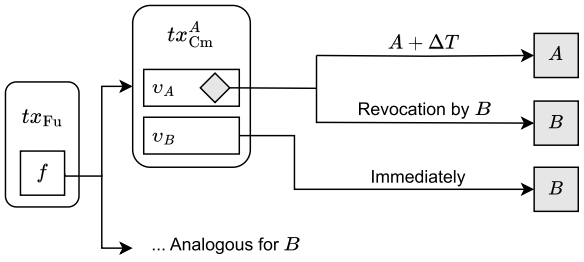

FIGURE 1. Transaction flow of Lightning Network commitments. Rounded boxes represent transactions, and the rectangles within them denote the outputs, where $\upsilon_{d_1} + \upsilon_{d_2} = f$. Incoming arrows represent transaction inputs, while outgoing arrows represent how an output can be spent. When multiple spending paths exist, a rhombus is used to indicate conditional branches. In this case, only one transaction from one path can be accepted. A single outgoing line from a transaction output, without passing through a rhombus, indicates that the output is a single-party address. The timelock (ΔT) associated with a transaction is written over the corresponding arrow.

III. RELATED WORK

We categorized the previous solutions proposed for enabling IoT micropayments into three categories inspired by [1] and [44]: payment channels, scalable digital currencies, and payment hubs. Table 2 provides a comparative overview of these approaches, highlighting their key features, advantages, and limitations. Additionally, at the end of this section, we include a comparative analysis of our protocol with existing general-purpose watchtower-based solutions.

A. PAYMENT CHANNELS

Payment channel solutions are divided into direct integration and hardware-based solutions.

1) DIRECT INTEGRATION

In this type of solution, IoT devices are integrated with payment channel networks through the use of gateways or lightweight clients. Gateways act as intermediaries or watchtowers, facilitating transactions and ensuring security. Hannon et al. [25] propose a protocol involving an *IoT gateway* and two groups of third parties: one for publishing transactions on behalf of the IoT device and preventing collusion, and another comprising *watchdogs* for detecting the broadcast of old states. However, the protocol only supports payments from IoT devices to gateways and does not enable direct payments between two IoT devices. Furthermore, their setup introduces considerable overhead and network latency, which is not discussed in the paper. Pouraghily and Wolf [26] introduce a ticket-based verification protocol (TBVP), where two entities are defined: *gateway* and *thing*. The protocol employs a single smart contract for a group of devices within an administrative domain rather than assigning a separate smart contract to each device. However, this approach raises security concerns due to the shared secret key of the blockchain account among devices. Additionally, the protocol requires a *ticket manager* and a *transaction/ticket verifier* for its implementation.

Robert et al. [20] propose an *IoT marketplace* connecting data providers, analysts, and consumers. Consumers request data or services, and the marketplace generates an LN-based invoice token from the *LN module*. After payment confirmation, access is granted via an API. Dependence on a trusted third party, the requirement for marketplace registration, and the inability to exchange payments between devices on different platforms are limitations of this work. Profentzas et al. [27] present the Tiny Ethereum Virtual Machine (TinyEVM) to enable off-chain payments on resource-constrained IoT devices with memory capacities of only a few kilobytes. A node publishes a smart contract as a *template* on the blockchain. Then, nodes utilize this template to deploy new off-chain payment channels (local smart contracts) identified by a unique monotonic counter. However, relying on a challenge period to allow the submission of a transaction with a higher sequence number in the event of broadcasting a revoked state introduces a notable limitation.

Tapas et al. [28] introduce a distributed framework for *IoT patch updates*, leveraging the Lightning Network for incentivized payments and decentralized file-sharing systems for patch delivery. However, its reliance on external distributors and the need for continuous network connectivity present challenges, especially for resource-constrained IoT devices. Rebello et al. [29] propose a hybrid PCN comprising a static and reliable core alongside peripheral unreliable devices. Full nodes are interconnected through core payment channels, while light nodes connect to full nodes via edge payment channels. This research introduces a minimum time window, requiring the terminating party to wait before retrieving tokens. Consequently, this approach is unsuitable for the devices that remain offline for long periods.

Kurt et al. [30] propose a 3-of-3 multi-signature channel involving the IoT device, *LN gateway*, and bridge LN node, allowing the device to participate in all operations. However, offline issues, the requirement for modified LN software on nodes, and changes to Basis of Lightning Technology (BOLT) #2 to prevent revoked state broadcasts are the limitations of this approach. Building on previous work, Kurt et al. in Lngate^2 [31] introduce a threshold client-server setup using (2,2)-threshold cryptography. This approach limits the

TABLE 2. Summary of related studies.

Category			Research	Approach	Mechanism for revoked state	Major limitation
Payment channels	Direct integration	Intermediaries (e.g., gateway or watchdog)	Hannon *et al.* [25]	IoT payment gateway	Watchdog	Supports only IoT device-to-gateway payments; not device-to-device payments
			Pouraghily *et al.* [26]	Administrative domain	Contract manager	Shared secret key of the blockchain account across devices
			Robert *et al.* [20]	IoT marketplace	LN module	Dependence on a trusted third party
			TinyEVM [27]	Tiny Ethereum Virtual Machine	Sequence number	Risk of losing money after challenge period
			P4UIoT [28]	LN and decentralized file-sharing	Patch delivery	Dependency on continuous network connectivity
			Rebello *et al.* [29]	Full and light nodes	Minimum time window	Risk of token theft if the minimum time window is exceeded
			Kurt *et al.* [30]	LN gateway	LN gateway	Requires modifications to LN software and BOLT #2
			LNGate^2 [31]	LN gateway	LN gateway	Requires modifications to BOLT #2
		Light clients	Neutrino [32]	Bloom filters & Golomb-Rice coding	Similar to LN	Requires constant online synchronization
			Phoenix [33]	Automated management with splicing	Similar to LN	Not designed specifically for IoT devices
	Hardware-based		Teechain [34]	Asynchronous blockchain access	Treasury Committee	Requirement of a TEE per device, Weak support for multi-hop payments
			Hyperchannel [35]	Third parties and Hyperledger fabric	Emergency withdrawal platform	Hardware overhead
Scalable digital currencies			IOTA, Nano, IoTeX, ITC, and Walton Chain [36]–[40]	Consensus algorithms	N/A	Dependency on specific algorithms or platforms
Payment hubs			Perun, Magma, Garou, and HyperPay [1], [41]–[43]	Star network topology	Centralized operational hub	Trust and performance bottlenecks

device's role to signing and key generation and eliminates the need for the LN gateway and bridge LN node to run modified LN software. However, it still inherits the other limitations mentioned earlier.

Unlike full nodes, light clients do not need to store the whole heavy-weight blockchain. Simplified Payment Verification (SPV) allows these clients to use only block headers, which are requested from untrusted full nodes. BIP37 [45] allows light clients to request only relevant transactions using Bloom filters, thereby reducing resource usage. However, it suffers from security and privacy issues. To address these concerns, BIPs 157/158 [46], [47] were introduced, defining client-side block filtering to obtain compact probabilistic filters from full nodes and specifying the creation of these compact filters, respectively. Neutrino project [32] is a Bitcoin light client for the Lightning Network that reduces resource requirements by synchronizing only block headers and filters, encoded with Golomb-Rice coding. While suitable for some resource-constrained IoT devices, it requires to remain online for synchronization, which not always be practical. Similarly, Phoenix [33] represents another effort to create lighter versions of LN. However, most IoT devices cannot run these solutions as they are not specifically designed for them.

2) HARDWARE-BASED

Trusted Execution Environments (TEEs) are predominantly used in the hardware-based category to ensure the secure execution of sensitive operations within a protected environment. They achieve this through mechanisms like attestation, runtime isolation, memory encryption, and controlled interfaces [34]. Lind et al., in their work on TEEchain [48], leverage TEEs as the root of trust instead of the underlying blockchain, enabling *asynchronous* blockchain access. To address TEE failures and rollback attacks, it employs a *committee of treasuries* and a *force-freeze replication* protocol. However, TEEchain faces limitations, including poor performance in multi-hop transactions, increased latency with larger committee sizes, the need for three network round trips per payment, reliance on a TEE for each device or domain, and reduced channel throughput due to committee chains. Wang et al. introduce HyperChannel [35], which consists of three main entities: clients, including IoT device owners and IoT service providers; service nodes, which are groups of TEEs managed by selfish third parties to execute payments using the *Hyperledger Fabric* platform; and an *emergency withdrawal platform*, which enables clients to withdraw their funds when the majority of service nodes crash or shut down. The risk of service nodes crashing or shutting down and the hardware overhead required to deploy an emergency enclave for each IoT device are notable limitations.

B. SCALABLE LAYER ONE CURRENCIES

The second category is distributed ledger-based currencies designed to improve network scalability. Some approaches focus on optimizing or developing consensus algorithms and transaction recording, while others aim to minimize storage needs and communication overhead among network members. Notable examples of such currencies include IOTA, Nano, IoTeX, ITC, and Walton Chain [36], [37], [38], [39], [40].

C. PAYMENT HUBS

A payment Hub (PH) constructs a star network topology to facilitate multi-party payments directly. In this model, a centralized operational node, or hub, processes transactions, simplifying the routing process and reducing setup and maintenance costs compared to PCNs. Users connected to the same PH can conduct off-chain payments through the hub without requiring direct payment channels between every pair of users. While PH offers simplicity and scalability by addressing routing challenges, its reliance on a centralized hub introduces potential trust and performance bottlenecks.

Examples of PH implementation include Perun, Magma, Garou, and HyperPay [1], [41], [42], [43]. However, not all of these are well-suited for IoT environments, as their assumptions often conflict with the limitations of typical IoT devices. The last two categories—scalable Layer 1 currencies and payment hubs—fall outside the scope of this paper, but we include them for the sake of a comprehensive comparison.

D. DISCUSSION ON WATCHTOWERS

For the problem of broadcasting revoked states in payment channel networks, several watchtower protocols have been proposed to outsource channel monitoring and protect users from fraud attempts [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59]. These works primarily focus on general-purpose designs and are not specifically tailored to IoT ecosystems. Some of these protocols rely on trusted watchtowers that are expected to act honestly without financial guarantees, while others require the watchtower to lock collateral as a form of accountability. Additionally, hybrid approaches combine both trust and collateral mechanisms to incentivize honest behavior and penalize misconduct. In contrast, our design eliminates the need for dedicated watchtowers by decentralizing monitoring responsibilities across a majority of peers—specifically, resource-efficient IoT devices and general-purpose blockchain full nodes. These peers are not explicitly designated as monitoring entities but instead participate implicitly as part of the protocol's normal operation. In prior designs, watchtowers are typically implemented as standalone third parties, introducing single points of failure and raising concerns about system robustness. Even in models where funds are protected via collateral, the watchtower itself may become a target of Denial-of-Service (DoS) attacks, potentially preventing timely action. By distributing monitoring duties, our approach mitigates these vulnerabilities.

IV. PROPOSED MODEL

The most prominent characteristics of payments in the IoT ecosystem are the presence of resource-constrained devices and the high volume of low-value transactions. These two properties stand in direct contrast to those of Bitcoin, which is characterized by high fees and long settlement times. While some existing solutions address the high-frequency requirement and provide scalability, they generally overlook the low-value nature of payments, rendering the solution unworthy of use. Other solutions account for the hardware and connectivity limitations of IoT devices but do not explicitly support high-frequency transactions. This work aims to bridge the gap between solutions by designing a model that not only supports high-frequency payments but also does so without relying on a single device's hardware or connectivity, or on specialized third parties. In this section, we first provide an overview of the proposed model for micropayments tailored to the IoT ecosystem. We then describe the threat model and the lifecycle of a payment channel. Finally, we conclude the section by providing an example of how the model can be applied.

A. OVERVIEW

LIoTning is a peer-supervised payment channel of $N + 2$ participants: two devices as sender and receiver of payments, and N peers as supervisor nodes. An overview of this setup is shown in Fig. 2, where the two devices aim to perform bidirectional micropayments, while the N peers are responsible for tracking and maintaining full-state commitment transactions under the honest majority assumption. This assumption guarantees that in all scenarios, at least a majority of the peers (i.e., $\lfloor N/2 \rfloor + 1$) behave rationally and do not engage in fraudulent activities. The two devices interact with the N peers to manage full-state commitment transactions. In addition to full-state commitment transactions, the devices exchange intermediate transactions that involve only the two devices, and the N peers are not aware of them. In the following, after explaining the threat model, the three phases of the protocol—(i) opening channel, (ii) sending and receiving payments, and (iii) closing channel—will be described in detail. The formal description of the protocol is presented in Fig. 5. Transactions in our model are represented as $tx := (inputs, outputs, amounts)$, where: *inputs* are references to previous transactions; *outputs* specify locking scripts and destination addresses for the funds; *amounts* indicate the values transferred by the transaction. In addition to these three components, signatures act as *witnesses*, providing the unlocking data required to authorize the spending of funds. The exact meaning of each symbol used in this section is provided in Table 3.

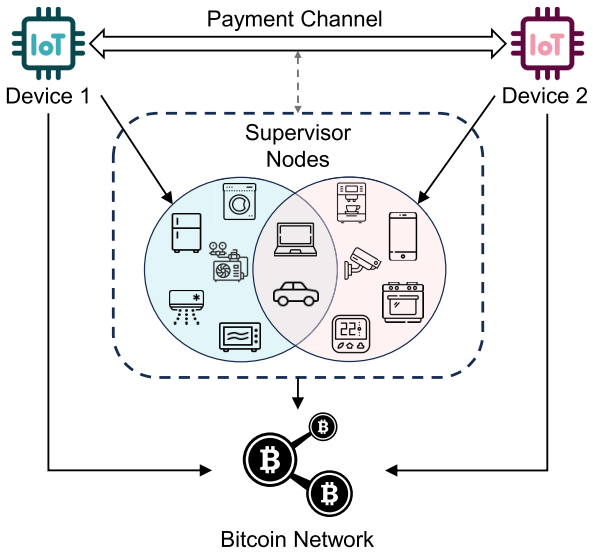

FIGURE 2. Usage overview of LIoTning.

B. THREAT MODEL

We assume that communications between the IoT devices and peers are secured using TLS-like mechanisms. However, due to the nature of our proposed model and its application,

devices and peers can potentially attempt to cheat. The possible attacks on our model are categorized as follows:

- **Broadcasting revoked states**. One significant threat is that a device might cheat the other party by broadcasting old channel states to the ledger, attempting to claim more funds.
- **Collusion attacks**. One of the IoT devices and some peer nodes may collude against the other IoT device. The first device broadcasts an old channel state and collaborates with specific peers, asking them to sign the invalid state to gain the second device's funds. Another type of collusion occurs when peers collude with each other to delay the channel's functionality.
- **Ransom attacks.** Any peer can ask the IoT device to pay ransom for each commitment transaction or instant closure of the channel, exploiting the necessity of its signature to fraudulently claim funds.

TABLE 3. Notations and their definitions.

Symbol	Definition
s_i	Channel state i
r_i	Number of intermediate transaction in state i
g_{d_1}	Group of peers for device d_1
$\mathcal{P}$	The set of peers, defined as $\mathcal{P} = g_{d_1} \cup g_{d_2}$.
$Ch_{d_1 d_2}$	Funding address of channel between devices d_1 and d_2
tid_{d_1}	Transaction ID selected by d_1 for funding the channel
tx_{Fu}	Funding transaction
tx_{St}	Settlement transaction
$tx^{d_1}_{\mathrm{Cm},i}$	Full-state commitment transaction for state s_i owned by d_1
$tx^{d_1,d_2}_{\mathrm{IntSet},i,j}$	Set of j-th intermediate transactions in state i, initiated by d_1 as the payer
$tx^{d_1,d_2}_{\mathrm{Rv},i}$	Revocation transaction for state s_i, transferring d_1's funds to d_2
$tx^{d_1,d_2}_{\mathrm{PreRv},i}$	Pre-signed revocation transaction for state s_i, transferring d_1's funds to d_2
$tx^{d_1,d_1}_{\mathrm{Inst},i}$	Instant closure transaction for state s_i, transferring d_1's funds to d_1 from $tx^{d_1}_{\mathrm{Cm},i}$
$tx^{d_1,d_1}_{\mathrm{Lazy},i}$	Lazy closure transaction for state s_i, transferring d_1's funds to d_1 from $tx^{d_1}_{\mathrm{Cm},i}$
$tx^{d_1,d_2}_{\mathrm{Fwd},i}$	Fast withdrawal transaction for state s_i, transferring d_2's funds to d_2 from $tx^{d_1}_{\mathrm{Cm},i}$
$\sigma^{d_1}_{\mathrm{St}}$	Signature generated by d_1 on transaction tx_{St}
$\sigma^{d_1}_{tid_{d_1}}$	Signature generated by d_1 on transaction tid_{d_1}
$\sigma^{d_1,d_2}_{\mathrm{Cm},i}$	Signature generated by d_1 on the $tx^{d_2}_{\mathrm{Cm},i}$ for state s_i, provided to d_2
$\sigma^{d_1,d_2}_{\mathrm{IntSet},i,j}$	Set of signatures on $tx^{d_1,d_2}_{\mathrm{IntSet},i,j}$, generated by d_1 as the payer
$\sigma^{d_1,d_2}_{\mathrm{Rv},i}$	Signature generated by d_2 with $sk^{d_1,d_2}_{\mathrm{Rv},i}$ on transaction $tx^{d_1,d_2}_{\mathrm{Rv},i}$ for state s_i
$\sigma^{d_1,p}_{\mathrm{PreRv},i}$	Signature generated by d_1 with $sk^{d_2,d_1}_{\mathrm{Rv},i}$ on transaction $tx^{d_2,d_1}_{\mathrm{PreRv},i}$ for state s_i
$sk^{d_1,d_2}_{\mathrm{Rv},i}$	Secret revocation key for state s_i generated by d_1 and shared with d_2
$v_{d_1,i}$	Funds held by d_1 in state s_i
$v^j_{d_1,i}$	Funds held by d_1 in the j-th intermediate transaction at state s_i

C. OPENING CHANNEL

Initially, each device selects a group of peers, g_{d_1} and g_{d_2}, which include α and β number of peers for each device, respectively, with the total number of peers given by $N = \alpha + \beta$. These peers might be chosen from available resource-efficient IoT devices, blockchain nodes, or dedicated service providers. The selection is based on factors such as known nodes, historical payment interactions, network reputation, channel capacity, channel balances, or ultimately, random selection. Members of these groups serve as peers who can be honest or malicious; trusted or untrusted. To open the channel, the two devices create a funding transaction and broadcast it to the blockchain. The funds are sent to a multi-signature address that requires signatures from both devices and at least $\lfloor N/2 \rfloor + 1$ of the N peers. This condition ensures that the signatures of the two devices, along with those of at least $\lfloor N/2 \rfloor + 1$ number of peers, are necessary as the transaction's witnesses for spending the funds. Additional signatures beyond the minimum requirement can provide increased security for the channel. As a result of this choice, the number of peers that are selected exclusively by one device and not shared with the other must not hold a majority; therefore, the following conditions must be satisfied:

$$|g_{d_1} - g_{d_2}| \leq \lfloor N/2 \rfloor , \quad |g_{d_2} - g_{d_1}| \leq \lfloor N/2 \rfloor .$$

D. SENDING AND RECEIVING PAYMENTS

The two devices exchange payments through two types of transactions: full-state commitment transactions and intermediate transactions. Full-state commitment transactions prevent cheating by a penalty mechanism and require the cooperation of supervisor nodes and exchange between the two channel sides. On the other hand, intermediate transactions are lightweight and can be rapidly made by either side of the channel. In the following sections, these two categories of payments, along with the role of pre-signed revocation transactions, are described, and Fig. 4 provides a more precise representation.

1) FULL-STATE COMMITMENT TRANSACTIONS

For each payment in the form of a full-state commitment transaction, both devices create their own version of the full-state commitment transaction and send it to the other party. As shown in Fig. 3, full-state commitment transactions can be spent in several paths. As illustrated in the first path, when a device broadcasts its latest full-state commitment transaction to the blockchain, it can be used either after a relative timelock ΔT determined by a specific block height, in which case there is no sleepy channel [60]. The second path occurs when one device broadcasts its latest full-state commitment transaction, and the transaction is settled immediately once at least $\lfloor N/2 \rfloor + 1$ peers from two groups sign it under the honest majority assumption, verifying that it represents the latest state. If a device (e.g., d_1) broadcasts a full-state commitment transaction representing an old channel state, two scenarios may arise.

- If the counterparty device (d_2) is online and able to monitor the blockchain, it detects the presence of an old channel state. In this case, as shown in the third path of Fig. 3, d_2 revokes the transaction using the corresponding revocation transaction, signed with the revocation key for that state, and claims the other device's funds as a penalty.
- However, if d_2 is offline, the peers who hold the pre-signed revocation transactions (as described later in the paper) for all previous states, detect the old full-state commitment and revoke it using the corresponding pre-signed revocation transaction. By this action, the majority of the funds are transferred to the offline device (d_2), while a small fraction is retained as a fee for peers. This fee serves as an incentive for peers to monitor the state of the channel and blockchain consistently. This is illustrated in the fourth path of Fig. 3.

A key consideration is the implementation of a very short timelock Δt for the pre-signed revocation transactions. This ensures that the other party, d_2 if online, is prioritized to revoke the transaction before the peers are permitted to take action. The other party can claim its funds immediately, as shown in the last path, whether in the case of broadcasting the latest or a revoked full-state commitment.

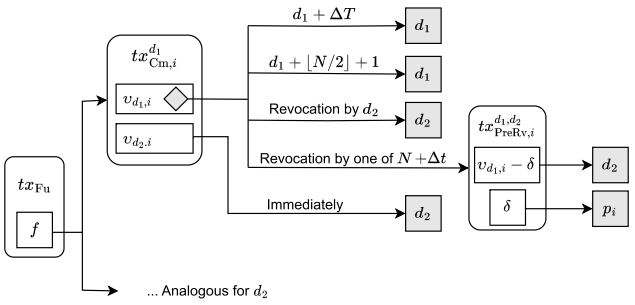

FIGURE 3. Transaction flow of full-state commitment transactions.

2) PRE-SIGNED REVOCATION TRANSACTIONS

As previously discussed, to enable peers to revoke an old broadcasted full-state commitment transaction, LIoTning employs pre-signed revocation transactions. These transactions are generated during each state transition using the `SIGHASH_SINGLE` flag[5] Each device constructs its own version of the pre-signed revocation transaction, signs it using the revocation key received from the other device for the corresponding state, and distributes it to its peers. Specifically, during a state transition from s_i to s_{i+1}, the devices exchange full-state commitment transactions for the new state s_{i+1} and revocation keys for the previous state s_i. At this point, the process is executed as follows: device d_1 uses the $\mathrm{toLocal}_{d_2}$ output from device d_2's full-state commitment transaction $tx^{d_2}_{\mathrm{Cm},i}$, representing $\upsilon_{d_2,i}$, as the input to a pre-signed revocation transaction. The output returns $\upsilon_{d_2,i} - \delta$ to d_1, while a service fee δ is allocated to the revoking peer (as illustrated in the fourth path of Fig. 3). After signing this transaction using $sk^{d_2,d_2}_{\mathrm{Rv},i}$, device d_1 distributes the transaction $tx^{d_2,d_1}_{\mathrm{PreRv},i}$ and its signature $\sigma^{d_1,p}_{\mathrm{PreRv},i}$ among the peers. Device d_2 performs the symmetric process, generating the transaction $tx^{d_1,d_2}_{\mathrm{PreRv},i}$ and signature $\sigma^{d_2,p}_{\mathrm{PreRv},i}$ for revoking d_1's old state. Alternatively, Bitcoin covenants, as described in [61], can be used to enforce output constraints directly within the transaction logic. However, such mechanisms are not yet practically available for deployment.

The order of signatures plays a crucial role in this protocol. In each state, the device first generates its full-state commitment transaction and distributes it among the available peers, including both the first and second groups. After this step, each available peer signs the commitment. Once the device receives the signed commitment from the peers, it then signs the full-state commitment transaction itself and sends it to the other party. The other party follows the exact process. Additionally, in each state, the devices generate and distribute corresponding pre-signed revocation transactions to the previous state among the available peers in both groups.

3) INTERMEDIATE TRANSACTIONS

Peers are not involved in intermediate transactions, as these occur exclusively between the two devices. In this process, the payer device creates a set of three versions of intermediate transactions, denoted by $tx^{\mathrm{payer,payee}}_{\mathrm{IntSet},i,j}$, where i is the state index and j represents the j-th intermediate transaction within that state. These three versions are constructed as follows:

1) The first version uses as input the payer's funds from its instant closure transaction.
2) The second version uses as input the payer's funds from its lazy closure transaction.
3) The third version uses as input the payer's funds from its fast withdrawal transaction, which originates from the payee's latest full-state commitment transaction.

Each version has an output that reflects the updated fund balances for both devices. In bidirectional payments, the roles of sender and receiver may switch over time, resulting in corresponding changes to the transaction inputs and roles in the intermediate transaction set.

Depending on the situation, each full-state commitment transaction may include multiple intermediate transactions. However, these transactions are sequentially numbered and cannot exceed a predefined threshold n_{th}. Furthermore, each intermediate transaction is associated with a timelock, where the timelock of each new transaction must be shorter than that of the previous one, and the cumulative duration of all timelocks must remain below a predefined time threshold, t_{th}. Thresholds n_{th} and t_{th} apply across all states. Thus, in the state s_i if the timelocks $t_{\mathrm{Int},i,1}, t_{\mathrm{Int},i,2}, \ldots, t_{\mathrm{Int},i,r_i}$ correspond to intermediate transactions $tx_{\mathrm{Int},i,1}, tx_{\mathrm{Int},i,2}, \ldots tx_{\mathrm{Int},i,r_i}$, respec-

[5] `SIGHASH_SINGLE` flag signs the input being signed and the output that shares the same index. If that output is changed, that signature becomes invalidated.

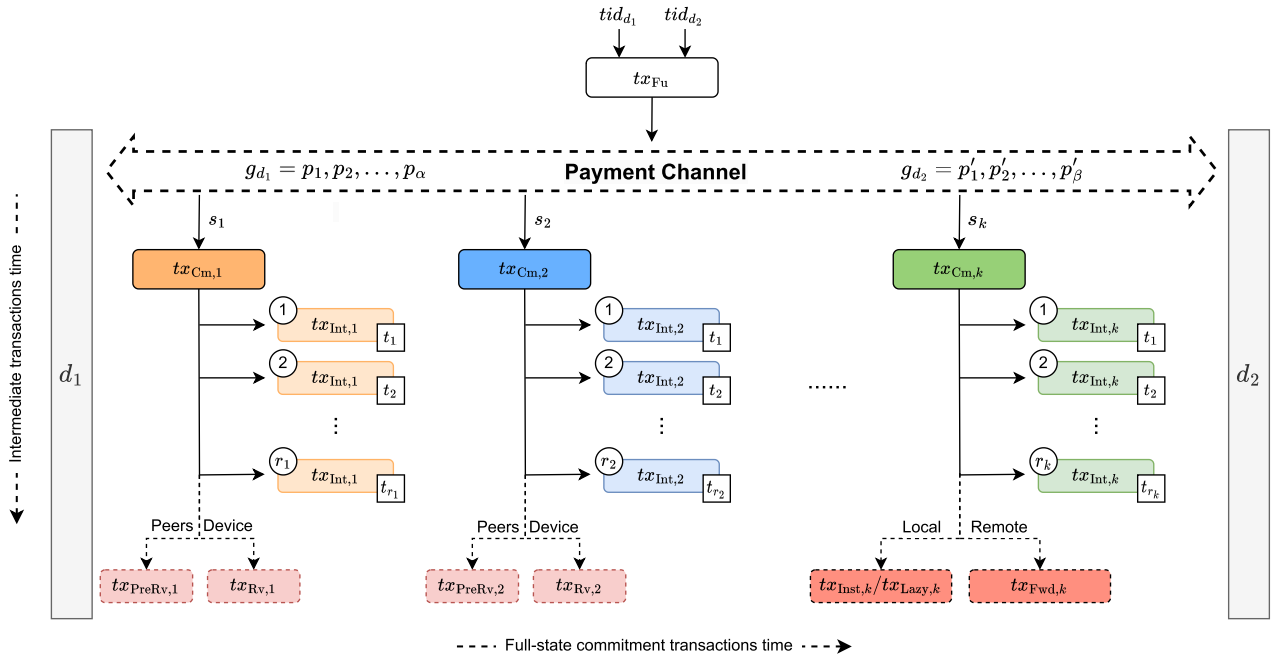

FIGURE 4. Protocol flow of LIoTning. Members of groups $g_{d_1} = \{p_1, p_2, \ldots, p_\alpha\}$ and $g_{d_2} = \{p'_1, p'_2, \ldots, p'_\beta\}$ serve as peers for devices d_1 and d_2, respectively. Full-state commitment transactions ($tx_{\text{Cm},i}$) are numbered from 1 to k. Each state s_i may include up to r_i intermediate transactions ($tx_{\text{Int},i,j}$), where $1 \leq j \leq r_i$ and $r_i \leq n_{\text{th}}$, represented as a sequence of circles. Each $tx_{\text{Int},i,j}$ contains a timelock, depicted by squares labeled from t_1 to t_{r_i}, such that $t_{r_i} < t_{r_i-1} < \cdots < t_2 < t_1$. At the end of each state, a revocation transaction (for the devices) and a pre-signed revocation transaction (for the peers) are also included.

tively, then the following conditions must hold:

$$\sum_{j=1}^{r_i} t_{\text{Int},i,j} \leq t_{th} \tag{1}$$

$$t_{\text{Int},i,r_i} < t_{\text{Int},i,r_i-1} < \cdots < t_{\text{Int},i,2} < t_{\text{Int},i,1} \tag{2}$$

$$\text{For each state } s_i,\ r_i \leq n_{\text{th}} \tag{3}$$

The primary objective of intermediate transactions is to minimize overhead by reducing computational requirements and network latency compared to full-state commitment transactions. This approach is based on the assumption that the likelihood of devices being online during the short interval for sending and receiving payments is high. Additionally, it accounts for scenarios where multiple rapid transactions occur within a short period (e.g., burst payments). The transition from intermediate transactions to a full-state commitment transaction occurs under three conditions. The first two conditions are when either the time threshold t_{th} or the number threshold n_{th} is exceeded. The third condition arises when a device detects the need to refresh the state. Thus, reaching the time threshold t_{th} or the number threshold n_{th} is not mandatory in this case. It is important to note that the first transaction after channel creation is always a full-state commitment transaction. In essence, full-state commitment transactions act as checkpoints for intermediate transactions.

E. CLOSING CHANNEL

Channel closure can be performed either mutually or unilaterally. Mutual closing, also known as cooperative closing, occurs when both devices agree to close the channel. In this case, a special transaction called a settlement transaction is created and signed by both parties. The funds are released once the transaction is signed by a majority of the peers, verifying that the amounts are consistent with the agreement between both devices.

A unilateral close, or uncooperative closing, occurs when one of the devices publishes a full-state commitment transaction into the blockchain. If the published full-state commitment transaction reflects the latest channel state, the broadcaster can claim its funds under one of two conditions: (i) immediately, by publishing an instant closure transaction signed by a majority of the peers. (ii) After a timelock (ΔT), by publishing a lazy closure transaction. These steps may optionally involve associated intermediate transactions. If such transactions exist in the current state, the devices must publish the most recent intermediate transaction following the lazy closure, instant closure, or fast withdrawal transaction.

If the published transaction does not represent the latest state of the channel and the counterparty device is online, it can revoke the transaction using the corresponding revocation key. In the event the counterparty is offline, the peers holding the pre-signed revocation transactions for the prior states detect and revoke the old channel state.

F. EXAMPLE

In this part, we aim to clarify the concept with an example. Assume devices d_1 and d_2 want to exchange micropayments. The process is described as follows:

- First of all, they choose their group of peers: $g_{d_1} = p_1, p_2, \ldots, p_\alpha$ and $g_{d_2} = p'_1, p'_2, \ldots, p'_\beta$, where α and β denote the number of peers in each group, respectively. The combined set of peers is denoted by $\mathcal{P} = g_{d_1} \cup g_{d_2}$ with cardinality $|\mathcal{P}| = N$.
- Next, d_1 and d_2 create a funding transaction to open the channel by sending funds to a multi-signature address (funding script address) that requires signatures from both of them and at least $\lfloor N/2 \rfloor + 1$ of the N peers to unlock the funds.
- Once the funding transaction is completed and confirmed on the blockchain, they begin creating their first full-state commitment transaction and send it to the other device. At this point, d_1 holds $tx^{d_1}_{\text{Cm},1}$ and d_2 holds $tx^{d_2}_{\text{Cm},1}$.
- Subsequently, they may begin to create intermediate transactions within a predefined number threshold n_{th} and time threshold t_{th}. For instance, at this state, if d_1 is the payer, they create $tx^{d_1,d_2}_{\text{IntSet},1,1}, tx^{d_1,d_2}_{\text{IntSet},1,2}, \ldots, tx^{d_1,d_2}_{\text{IntSet},1,r_1}$.
- Afterward, they proceed with creating the second full-state commitment transaction and exchange it with each other. As a result, d_1 possesses $tx^{d_1}_{\text{Cm},2}$, and d_2 possesses $tx^{d_2}_{\text{Cm},2}$. Simultaneously, they exchange the revocation keys $sk^{d_1,d_2}_{\text{Rv},1}$ and $sk^{d_2,d_1}_{\text{Rv},1}$ for the previous state (s_1), and use them to create the pre-signed revocation transactions corresponding to the first state of the channel. These pre-signed revocation transactions, denoted as $tx^{d_1,d_2}_{\text{PreRv},1}$ and $tx^{d_2,d_1}_{\text{PreRv},1}$, are then distributed to all peers from both groups.

This process continues iteratively. For example, after k full-state commitment transactions, device d_1 holds the transactions $tx^{d_1}_{\text{Cm},1}, tx^{d_1}_{\text{Cm},2}, \ldots, tx^{d_1}_{\text{Cm},k}$, along with the revocation keys $sk^{d_2,d_1}_{\text{Rv},1}, sk^{d_2,d_1}_{\text{Rv},2}, \ldots, sk^{d_2,d_1}_{\text{Rv},k-1}$ received from d_2. Correspondingly, d_2 holds the full-state commitment transactions $tx^{d_2}_{\text{Cm},1}, tx^{d_2}_{\text{Cm},2}, \ldots, tx^{d_2}_{\text{Cm},k}$, and the revocation keys $sk^{d_1,d_2}_{\text{Rv},1}, sk^{d_1,d_2}_{\text{Rv},2}, \ldots, sk^{d_1,d_2}_{\text{Rv},k-1}$ received from d_1. The peers hold the pre-signed revocation transactions $tx^{d_1,d_2}_{\text{PreRv},1}, tx^{d_1,d_2}_{\text{PreRv},2}, \ldots, tx^{d_1,d_2}_{\text{PreRv},k-1}$ and $tx^{d_2,d_1}_{\text{PreRv},1}$, $tx^{d_2,d_1}_{\text{PreRv},2}, \ldots, tx^{d_2,d_1}_{\text{PreRv},k-1}$, which enable the revocation of previous states in case of misbehavior. Each full-state commitment transaction can accommodate up ton_{th}intermediate transactions. After completing a series of payments in exchange for services, the devices proceed to close the channel either mutually or unilaterally.

V. SECURITY ANALYSIS

In the previous section, the operation of the LIoTning protocol was discussed. In this section, we specifically analyze the security of the protocol. To do so, we first discuss how the protocol mitigates the potential threats. We then model and formally prove the security of the proposed protocol using rigorous methods.

A. SECURITY AGAINST THREATS

As mentioned in Section IV-B, the threats addressed in this work fall into three categories: broadcasting revoked states, collusion attacks, and ransom attacks. In the following three subsections, we analyze the potential impact of each threat on the proposed protocol and present the corresponding mitigation strategies.

1) BROADCASTING REVOKED STATES

At the beginning of the sending payments and service provision process, the two devices initiate a channel as described in the previous section. They start with one full-state commitment transaction ($tx^{d_1}_{\text{Cm},1}$ and $tx^{d_2}_{\text{Cm},1}$) and then proceed to trade intermediate transactions based on the authorized number and the time threshold ($tx^{\text{payer,payee}}_{\text{IntSet},1,j}$, $1 \leq j \leq r_1$). Once one of the conditions expires, they refresh and update their state with a new full-state commitment transaction ($tx^{d_1}_{\text{Cm},1}$ and $tx^{d_2}_{\text{Cm},1}$). After this step, they begin sending and receiving the second set of intermediate transactions corresponding to the second full-state commitment transaction ($tx^{\text{payer,payee}}_{\text{IntSet},1,j}$, $1 \leq j \leq r_2$). This process continues until the devices mutually or unilaterally decide to terminate the payments and close the channel. In a mutual closing, both devices cooperate to make a settlement transaction tx_{St} using the direct output from the funding transaction. In a unilateral closing, two situations might occur:

1) If the full-state commitment transaction at state s_k is the latest state agreed upon by both devices, the device wishing to close the channel (e.g., d_1) broadcasts its own latest commitment ($tx^{d_1}_{\text{Cm},k}$). Consequently, it can claim the funds either immediately via an instant closure transaction if more than half of the channel peers sign it, or after the relative timelock ΔT expires via a lazy closure transaction. If a device (e.g., d_1) attempts to cheat by broadcasting an old full-state commitment transaction (e.g., $tx^{d_1}_{\text{Cm},i}$, $1 \leq i < k$), the other device (d_2) can take action if it is online. It can monitor the blockchain, revoke $tx^{d_1}_{\text{Cm},i}$ by using the corresponding revocation key $sk^{d_1,d_2}_{\text{Rv},i}$ to construct a revocation transaction $tx^{d_1,d_2}_{\text{Rv},i}$, and claim both its own funds and the cheating device's funds as a penalty. If the counterparty device (d_2) is offline, one of the available peers steps in and uses the pre-signed revocation transaction corresponding to state s_i ($tx^{d_1,d_2}_{\text{PreRv},i}$) to prevent cheating. This action transfers the funds to the offline device and provides the peer with a service fee for its participation.
2) If the full-state commitment transaction at state s_k and the intermediate transaction set r_k represent the latest state and intermediate transactions agreed upon by both devices, then the transactions ($tx^{d_1}_{\text{Cm},k}$, $tx^{d_2}_{\text{Cm},k}$, and $tx^{\text{payer,payee}}_{\text{IntSet},k,r_k}$) are considered the most up-to-date channel records. The device wishing to close the channel (e.g., d_1) broadcasts its own latest commitment

Channel opening

Devices d_1 and d_2 open the payment channel with peer groups g_{d_1} and g_{d_2}, having a total channel balance $\upsilon_{d_1,0}+\upsilon_{d_2,0}$, which originates from tid_{d_1} and tid_{d_2}, respectively.

1) d_1 and d_2 select their peer groups: $g_{d_1}=\{p_1,p_2,\ldots,p_\alpha\}$ and $g_{d_2}=\{p'_1,p'_2,\ldots,p'_\beta\}$, such that the total set of peers is $\mathcal{P}=g_{d_1}\cup g_{d_2}$ with cardinality $|\mathcal{P}|$.
2) d_1 and d_2 construct the funding script address $Ch_{d_1d_2}$ (Algorithm 1) and then create a funding transaction by sending $\upsilon_{d_1,0}+\upsilon_{d_2,0}$ to it. $tx_{\text{Fu}}\coloneqq tx\left([tid_{d_1},tid_{d_2}],Ch_{d_1d_2},\upsilon_{d_1,0}+\upsilon_{d_2,0}\right)$
3) Then, d_1 and d_2 generate the signatures $\sigma^{d_1}_{tid_{d_1}}$ and $\sigma^{d_2}_{tid_{d_2}}$ on transactions tid_{d_1} and tid_{d_2}, respectively.
4) They post $\left(tx_{\text{Fu}},\{\sigma^{d_1}_{tid_{d_1}},\sigma^{d_2}_{tid_{d_2}}\}\right)$ on $\mathbb{B}$. If tx_{Fu} is accepted by $\mathbb{B}$, then the channel is created. Otherwise, the transaction is rejected, and the channel is not opened.

***i*-th Full-state commitment transaction**

Devices d_1 and d_2, along with their respective peer groups g_{d_1} and g_{d_2}, update the channel to a new valid state s_i by creating the i-th full-state commitment transaction, transferring the funds to the updated balances $\upsilon_{d_1,i}$ and $\upsilon_{d_2,i}$.

Generate transactions:

1) d_1 and d_2 construct the commitment script addresses (Algorithm 2) toLocal_{d_1} and toLocal_{d_2}, and generate full-state commitment transactions. $tx^{d_1}_{\text{Cm},i}\coloneqq tx(Ch_{d_1d_2},[\text{toLocal}_{d_1},\text{toRemote}_{d_1}],[\upsilon_{d_1,i},\upsilon_{d_2,i}])$ and $tx^{d_2}_{\text{Cm},i}\coloneqq tx(Ch_{d_1d_2},[\text{toLocal}_{d_2},\text{toRemote}_{d_2}],[\upsilon_{d_2,i},\upsilon_{d_1,i}])$
2) Generate revocation transactions $tx^{d_1,d_2}_{\text{Rv},i}\coloneqq tx(\text{toLocal}_{d_1},pk_{\text{Rv},d_2},\upsilon_{d_1,i})$ and $tx^{d_2,d_1}_{\text{Rv},i}\coloneqq tx(\text{toLocal}_{d_2},pk_{\text{Rv},d_1},\upsilon_{d_2,i})$.
3) Generate pre-signed revocation transactions $tx^{d_1,d_2}_{\text{PreRv},i}\coloneqq tx(\text{toLocal}_{d_1},pk_{\text{Rv},d_2},\upsilon_{d_1,i}-\delta)$ and $tx^{d_2,d_1}_{\text{PreRv},i}\coloneqq tx(\text{toLocal}_{d_2},pk_{\text{Rv},d_1},\upsilon_{d_2,i}-\delta)$.

Generate signatures:

1) Devices d_1 and d_2 distribute $tx^{d_1}_{\text{Cm},i},tx^{d_2}_{\text{Cm},i}$ among peers in $\mathcal{P}$.
2) Peers verify and generate signatures $\sigma^{p,d_1}_{\text{Cm},i}$ on $tx^{d_1}_{\text{Cm},i}$ and $\sigma^{p,d_1}_{\text{Cm},i}$ on $tx^{d_2}_{\text{Cm},i}$ and send them to devices.
3) Device d_1 generates signature $\sigma^{d_1,d_2}_{\text{Cm},i}$ on transaction $tx^{d_2}_{\text{Cm},i}$ and d_2 generates signature $\sigma^{d_2,d_1}_{\text{Cm},i}$ on transaction $tx^{d_1}_{\text{Cm},i}$ send them to other device along with the revocation key for previous state s_{i-1}. Therefore, d_1 holds $\left(tx^{d_1}_{\text{Cm},i},\sigma^{d_2,d_1}_{\text{Cm},i},sk^{d_2,d_1}_{\text{Rv},i-1}\right)$ and d_2 holds $\left(tx^{d_2}_{\text{Cm},i},\sigma^{d_1,d_2}_{\text{Cm},i},sk^{d_1,d_2}_{\text{Rv},i-1}\right)$
4) d_1 generates signature $\sigma^{d_1,p}_{\text{PreRv},i-1}$ by $sk^{d_2,d_1}_{\text{Rv},i-1}$ on transaction $tx^{d_2,d_1}_{\text{PreRv},i-1}$ and d_2 generates signature $\sigma^{d_2,p}_{\text{PreRv},i-1}$ by $sk^{d_1,d_2}_{\text{Rv},i-1}$ on transaction $tx^{d_1,d_2}_{\text{PreRv},i-1}$ send them to all peers in $\mathcal{P}$.

Intermediate transactions

Devices d_1 and d_2, in their latest full-state commitment transactions $tx^{d_1}_{\text{Cm},i}$ and $tx^{d_2}_{\text{Cm},i}$, with balances $\upsilon_{d_1,i}$ and $\upsilon_{d_2,i}$, generate lightweight intermediate transactions while considering the number threshold n_{th} and time threshold t_{th}.

1) **While** refresh is not requested **and** t_{th} is not exceeded **and** n_{th} is not exceeded

   If d_1 is the payer, it creates three versions of intermediate transactions with timelock $t_{\text{Int},i,j}$.
   $tx^{d_1,d_2}_{\text{IntSet},i,j}\coloneqq\left\{tx(pk_{\text{Lazy},d_1},[pk_{d_1},pk_{d_2}],[\upsilon^j_{d_1,i},\upsilon^j_{d_2,i}]),tx(pk_{\text{Inst},d_1},[pk_{d_1},pk_{d_2}],[\upsilon^j_{d_1,i},\upsilon^j_{d_2,i}]),tx(pk_{\text{Fwd},d_1},[pk_{d_1},pk_{d_2}],[\upsilon^j_{d_1,i},\upsilon^j_{d_2,i}])\right\}$
   and then generate the signatures $\sigma^{d_1,d_2}_{\text{IntSet},i,j}$ on $tx^{d_1,d_2}_{\text{IntSet},i,j}$ and send them to the d_2.

   Else if d_2 is the payer, it creates three versions of intermediate transactions with timelock $t_{\text{Int},i,j}$.
   $tx^{d_2,d_1}_{\text{IntSet},i,j}\coloneqq\left\{tx(pk_{\text{Lazy},d_2},[pk_{d_1},pk_{d_2}],[\upsilon^j_{d_1,i},\upsilon^j_{d_2,i}]),tx(pk_{\text{Inst},d_2},[pk_{d_1},pk_{d_2}],[\upsilon^j_{d_1,i},\upsilon^j_{d_2,i}]),tx(pk_{\text{Fwd},d_2},[pk_{d_1},pk_{d_2}],[\upsilon^j_{d_1\,i},\upsilon^j_{d_2,i}])\right\}$
   and then generate the signatures $\sigma^{d_2,d_1}_{\text{IntSet},i,j}$ on $tx^{d_2,d_1}_{\text{IntSet},i,j}$ and send them to the d_1.

   End if

   End while
2) Devices generate new full-state commitment transactions $tx^{d_1}_{\text{Cm},i+1},tx^{d_2}_{\text{Cm},i+1}$.

Channel closing

Both devices can close the channel mutually or unilaterally. To do this:

Mutual closing:

1) If both devices agree, they create settlement transaction $tx_{\text{St}}\coloneqq tx\left(Ch_{d_1d_2},[pk_{d_1},pk_{d_2}],[\upsilon_{d_1,k},\upsilon_{d_2,k}]\right)$ then send it to peers.
2) Peers verify and generate signatures σ^p_{St} on it, and then send signatures to devices.
3) Devices generate signature $\{\sigma^{d_1}_{\text{St}},\sigma^{d_2}_{\text{St}}\}$ and post $\left(tx_{\text{St}},\{\sigma^{d_1}_{\text{St}},\sigma^{d_2}_{\text{St}},\sigma^p_{\text{St}}\text{ for }\lfloor N/2\rfloor+1\text{ of peers}\}\right)$ on $\mathbb{B}$.

Unilateral closing:

1) Device d_1 posts $\left(tx^{d_1}_{\text{Cm},i},\sigma^{d_2,d_1}_{\text{Cm},i}\right)$ on $\mathbb{B}$. This is followed by one of the two cases:
   a) Instant closure: Device d_1 creates instant closure transaction $tx^{d_1,d_1}_{\text{Inst},i}\coloneqq tx\left(\text{toLocal}_{d_1},pk_{\text{Inst},d_1},\upsilon_{d_1,i}\right)$ and send it to peers. The peers generate signatures $\sigma^{p,d_1}_{\text{Inst},i}$ and return them to d_1. Then, d_1 posts $\left(tx^{d_1,d_1}_{\text{Inst},i},\{\sigma^{d_1,d_1}_{\text{Inst},i},\sigma^{p,d_1}_{\text{Inst},i}\text{ for }\lfloor N/2\rfloor+1\text{ of peers}\}\right)$ on $\mathbb{B}$. This allows d_1 to claim its funds immediately. This can be followed by posting the corresponding intermediate transaction from $tx^{d_2,d_1}_{\text{IntSet},i,j}$ along with its signature from $\sigma^{d_2,d_1}_{\text{IntSet},i,j}$.
   b) Lazy closure: Device d_1 creates lazy closure transaction $tx^{d_1,d_1}_{\text{Lazy},i}\coloneqq tx(\text{toLocal}_{d_1},pk_{\text{Lazy},d_1},\upsilon_{d_1,i})$ and generate signature $\sigma^{d_1,d_1}_{\text{Lazy},i}$ on it. Then, d_1 post $\left(tx^{d_1,d_1}_{\text{Lazy},i},\sigma^{d_1,d_1}_{\text{Lazy},i}\right)$ on $\mathbb{B}$ after relative timelock ΔT. In this case, d_1 can claim the funds after the timelock. This may also be followed by posting the corresponding intermediate transaction from $tx^{d_2,d_1}_{\text{IntSet},i,j}$ along with its signature from $\sigma^{d_2,d_1}_{\text{IntSet},i,j}$.
2) Analogously, if device d_2 posts $\left(tx^{d_2}_{\text{Cm},i},\sigma^{d_1,d_2}_{\text{Cm},i}\right)$, on $\mathbb{B}$, then the corresponding steps described above can be similarly performed by d_2.

Revocation

1) If an old full-state commitment $tx^{d_1}_{\text{Cm},z}$ appears on $\mathbb{B}$, and device d_2 is online, then it posts $\left(tx^{d_1,d_2}_{\text{Rv},z},\sigma^{d_1,d_2}_{\text{Rv},z}\right)$ to revoke the old state. If d_2 is offline, peers post $\left(tx^{d_1,d_2}_{\text{PreRv},z},\sigma^{d_2,p}_{\text{PreRv},z}\right)$ to revoke the old state after Δt.
2) If the latest full-state commitment $tx^{d_1}_{\text{Cm},i}$ appears on $\mathbb{B}$ along with one of the instant or lazy closure transactions $\left(tx^{d_1,d_1}_{\text{Inst},i}\text{ or }tx^{d_1,d_1}_{\text{Lazy},i}\right)$, and one of the devices posts an old intermediate transaction $tx_{\text{Int},i,j}$ on $\mathbb{B}$, where $1\le j<r_i$, the other device broadcasts tx_{Int,i,r_i} during $t_{\text{Int},i,j}$.

Analogously, the corresponding steps can be described in a similar manner for d_2.

FIGURE 5. LIoTning Protocol. Setup, payments, closing, and revocation.

transaction $tx^{d_1}_{\mathrm{Cm},k}$ along with either an instant or a lazy closure transaction. After that, d_1 broadcasts the appropriate transaction from the intermediate set, specifically $tx^{\mathrm{payer,payee}}_{\mathrm{IntSet},k,r_k}$. In this scenario, if a device (e.g., d_1) attempts to cheat by broadcasting:

- a revoked full-state commitment transaction $tx^{d_1}_{\mathrm{Cm},i}$ where $1 \leq i < k$,
- a revoked full-state commitment along with an intermediate transaction $tx^{d_1}_{\mathrm{Cm},i}$, $tx^{\mathrm{payer,payee}}_{\mathrm{Int},i,j}$ where $1 \leq i < k$, $1 \leq j \leq r_i$, or
- the latest full-state commitment with an intermediate transaction other than the last one, $tx^{d_1}_{\mathrm{Cm},k}$, $tx^{\mathrm{payer,payee}}_{\mathrm{Int},k,j}$ where $1 \leq j < r_k$,

then the following applies: in the first and second cases, if the counterparty device (d_2) is online, it detects the fraud and revokes the broadcasted full-state commitment transaction using the corresponding revocation key $sk^{d_1,d_2}_{\mathrm{Rv},i}$, constructing a revocation transaction $tx^{d_1,d_2}_{\mathrm{Rv},i}$ to claim both its own funds and those of the cheating device as a penalty. If the counterparty is offline, one of the peers uses the pre-signed revocation transaction $tx^{d_1,d_2}_{\mathrm{PreRv},i}$ to reclaim the offline device's funds and penalize the cheating device, earning a small service fee. In the third case, the timelock conditions $t_{\mathrm{Int},i,1} > t_{\mathrm{Int},i,2} > \cdots > t_{\mathrm{Int},i,r_k}$ take effect. These ensure that each device has a defined time window to verify the correctness of the broadcasted intermediate transactions. If fraud is detected, the honest device has enough time to broadcast the correct intermediate transaction from its own set. All possible scenarios are illustrated in Fig. 6.

The introduction of intermediate transactions creates an opportunity for a party to cheat by withholding the broadcast of the latest intermediate transaction after the most recent commitment. However, we argue that this does not benefit the cheating party for three main reasons:

1) **Loss of reputation**: The defrauded party has proof of the misbehavior, which can be shared with peers. These peers may subsequently choose to refrain from participating in channels initiated by the fraudulent party.
2) **Low value of transactions**: The fee associated with posting intermediate transactions is usually much higher than the value transferred in them. The fee disincentivizes misbehavior of any party.
3) **Fee competition**: The defrauded device can broadcast the intermediate transaction with a higher fee than that of the fraudulent device to reclaim its loss. This is feasible because the fee required for an intermediate transaction is lower than that of a full-state commitment transaction.

2) COLLUSION ATTACKS

As described earlier, one type of collusion occurs when one of the IoT devices and some peer nodes collude against the other IoT device. The first device broadcasts an old channel state and collaborates with specific peers, requesting them to sign the invalid state to gain the second device's funds. Another type of collusion occurs when peers collaborate to delay the channel's functionality. Both cases violate the honest majority assumption, as they require the number of colluding or non-functional peers to exceed half of the total channel peers. Consequently, collusion attacks can be mitigated and are practically impossible to execute. Another deterrent against collusion attacks—where peers might sign a revoked channel state—is the incentive for them to earn higher fees by using pre-signed revocation transactions rather than supporting the old channel state.

Readers may wonder whether collusion attacks could occur when peer nodes independently collude against both devices with the primary goal of stealing funds from other parties, which would require channel peers to broadcast full-state

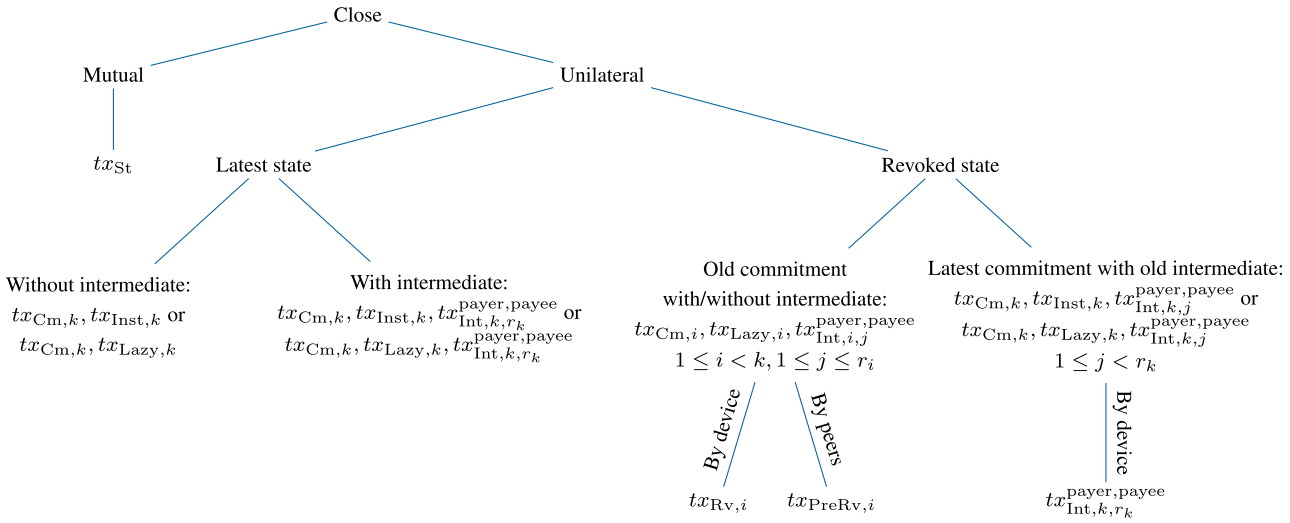

FIGURE 6. Tree of channel closing and revocation across all possible scenarios.

commitment transactions. However, no peer can broadcast any valid full-state commitment transaction, as the final signature required to validate the transaction is exclusively completed by the devices, as specified in the signature order.

3) RANSOM ATTACKS

In ransom attacks, a peer demands a ransom from an IoT device in exchange for signing a message. The message to be signed depends on the scenario. Suppose in a payment channel protocol, the signature of a peer is required to revoke a broadcasted old commitment. This scenario is an example of a severe case because it can cause a significant loss to the device. While this attack is possible in some of the previous works [31], it can not be done in our system because revocation does not require the cooperation of peers. Even when the defrauded device is offline, peers cannot claim more money than the agreed-upon value δ because the pre-signed revocation transaction limits the maximum fee that a peer can claim.

In scenarios where a peer's signature is required—specifically, for full-state commitments and the immediate closure of the channel—a device can refuse ransom payment without losing funds. We argue that this ability for the victim to reject the payment safely further discourages the ransom demand.

- Each device can unilaterally close the channel by broadcasting the latest commitment and waiting for the timelock to claim its funds.
- The fraudulent peer loses the trust of devices and is subsequently less likely to be included in future channels. The lack of cooperation from a peer can also be monitored by users outside the channel through the blockchain, resulting in further damage to the peer's reputation.
- More than half of the peers should agree on demanding the ransom, which is improbable due to the honest majority assumption.

Therefore, the ransom is expected to be rejected, and the peer asking for the ransom is expected to lose its reputation; consequently, no incentive exists to engage in this kind of attack in our proposed system.

B. FORMAL SECURITY PROOF

Our formal security proof is based on the Global Universal Composability (GUC) framework [62] and closely follows the models used in prior works [60], [63]. In this framework, security is proven by demonstrating the indistinguishability between the execution traces of the real world and ideal versions of the protocol. The ideal model of the protocol captures the basic expected functionality of the protocol when all parties are honest and cooperative. Due to space constraints, the complete formal description of the protocol, its ideal functionality, and the full proof of the following theorem are included in the Appendix. Below, we present the main security theorem of the proposed protocol.

Theorem 1: The LIoTning protocol Π UC-realizes the ideal functionality $\mathcal{F}$.

We now present a sketch of the proof. In the GUC framework, proving security involves three main steps. First, the protocol and its corresponding ideal functionality $\mathcal{F}$ must be defined. Second, for each case regarding the honesty of the parties involved, a simulator $\mathcal{S}$ must be constructed that, given access to the ideal functionality, produces an execution trace indistinguishable from that of the real-world protocol. Finally, it must be shown that, from the viewpoint of the environment $\mathcal{E}$, the execution trace resulting from the interaction between the simulator $\mathcal{S}$ and the ideal functionality $\mathcal{F}$ is equal to and indistinguishable from the execution trace produced by the interaction between the real-world protocol Π and arbitrary adversary $\mathcal{A}$. In other words, $EXEC_{\Pi,\mathcal{A},\mathcal{E}} = EXEC_{\mathcal{F},\mathcal{S},\mathcal{E}}$

VI. IMPLEMENTATION AND EVALUATION

Since our main target is the IoT ecosystem, we have implemented a proof-of-concept for the proposed solution to show its practicality and suitability by measuring performance metrics. The implementation is done using the Python programming language and the Bitcoin Core 28.0 test network. The measurements are made using a personal machine with Intel® Core™ i7-12700H with Ubuntu® 22.04 operating system. We have modified the locking scripts used in Lightning Network to meet our needs. The locking script for the funding transaction output is presented in Algorithm 1, and the locking script for the local output of the full-state commitment transactions is shown in Algorithm 2.

Algorithm 1 Locking Script of Funding Output

Input: Public keys of the devices (pk_{d_1}, pk_{d_2}), public key of supervisor peers ($pk_{p_1}, pk_{p_2}, \ldots, pk_{p_N}$).
Result: Script used for locking funding output.

1: $\langle pk_{d_1} \rangle$ OP_CHECKSIGVERIFY
2: $\langle pk_{d_2} \rangle$ OP_CHECKSIGVERIFY
3: $\langle \lfloor N/2 \rfloor + 1 \rangle$ $\langle \mathrm{pk}_{p_1}, \ldots, \mathrm{pk}_{p_N} \rangle$ $\langle N \rangle$ OP_CHECKMULTISIG

We start off by comparing the throughput (number of transactions per second) of the channel with different numbers of supervisor peers and intermediate transactions. We consider the full life cycle of a channel, including submission of funding and closing transactions to the blockchain, for computing the throughput. The closing is done instantly using the majority of peers' signatures. The result of our experiment is shown in Fig. 7. It is evident that throughput can be significantly improved by introducing intermediate transactions. For example, with 16 peers, increasing the number of intermediate transactions in each full-state commitment from 0 to 5 can raise the throughput from 80 tx/s to 300 tx/s, and increasing it further from 5 to 20 can raise the throughput to approximately 600 tx/s. Throughput improvements for other numbers of peers and intermediate transactions are shown in this figure. For creating a channel, a device can choose an appropriate combination of the number of peers

Algorithm 2 Locking Script for Local Commitment Output

Input: The set of public keys for the devices ($pkey_{\text{set}}^{d_1}$, $pkey_{\text{set}}^{d_2}$), public key of supervisor peers ($pk_{p_1}, pk_{p_2}, \ldots, pk_{p_N}$), state s_i's public key for revocation ($pk_{\text{Rv},i}^{d_1,d_2}$), and small and large relative timelock (Δt, ΔT).
Result: Script used for locking local output of full-state commitment transaction.

```
⟨pk_{d_1}⟩ OP_CHECKSIG
OP_IF
    OP_IF                                          /* Instant closure */
        ⟨⌊N/2⌋ + 1⟩ ⟨pk_{p_1}, ..., pk_{p_N}⟩ ⟨N⟩ OP_CHECKMULTISIG
    OP_ELSE                                        /* Lazy closure */
        ⟨ΔT⟩ OP_CHECKSEQUENCEVERIFY
    OP_ENDIF
OP_ELSE
    ⟨pk_{Rv,i}^{d_1,d_2}⟩ OP_CHECKSIGVERIFY
    OP_IF                                          /* Revoke by device */
        ⟨pk_{d_2}⟩ OP_CHECKSIG
    OP_ELSE                                        /* Revoke by peers */
        ⟨pk_{d_2}⟩ OP_CHECKSIGVERIFY
        ⟨Δt⟩ OP_CHECKSEQUENCEVERIFY
    OP_ENDIF
OP_ENDIF
```

and intermediate transactions in each state based on the required throughput and acceptable level of risk.

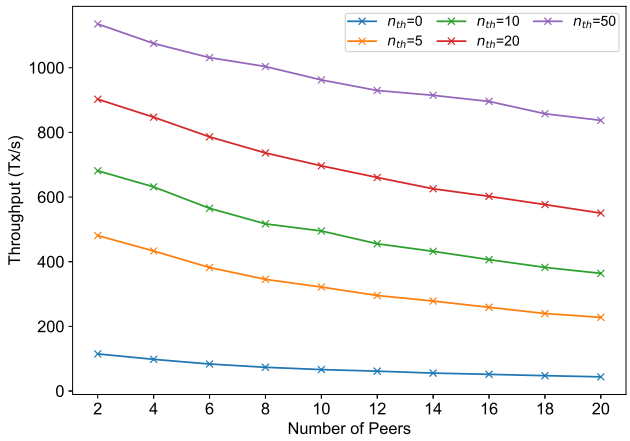

FIGURE 7. Throughput of channel for different numbers of peers and intermediate transactions.

An important factor in deciding the suitability of a system for use in an IoT ecosystem is its network overhead. We studied the network utilization of our proposed protocol by measuring the transfer size for each phase of the channel. The results of an experiment with 500 transactions using a channel created with 16 peers and closed by broadcasting a full-state commitment are shown in Fig. 8. The figure demonstrates the total transferred data in each phase. The chosen values for the parameters of this experiment are meant to illustrate the trend in network overhead clearly. Experimenting with various numbers of transactions and peers shows consistent trends, with results scaling linearly along both the vertical and horizontal axes. Additionally, the choice of 16 peers closely aligns with the typical federation sizes used in systems such as Rootstock [64] and Liquid [65] sidechains.

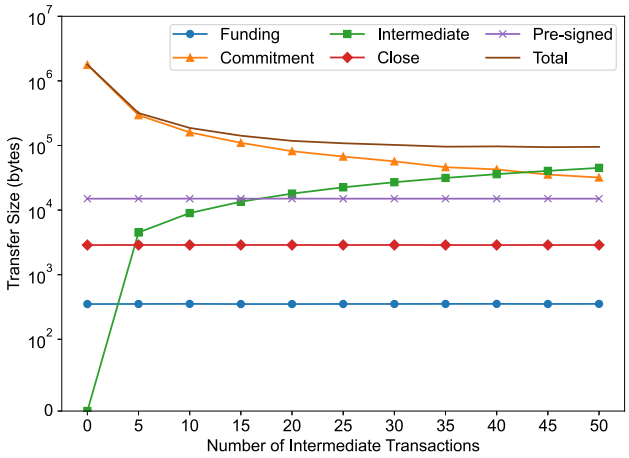

FIGURE 8. The transfer size (network usage) of a single device for making a fixed number of transactions using different intermediate transaction counts.

The first notable point is the constant size of funding and closing, as they do not depend on the number of intermediate transactions. Another interesting point is the significant decrease in total transferred size by using only five intermediate transactions rather than not using them at all (note the logarithmic scale of the y-axis). This decrease can be explained by the larger size of full-state commitment transactions compared to intermediate transactions. The decrease in the total size of transferred data with more intermediate transactions continues until it reaches a plateau. This observation shows that increasing intermediate transactions helps decrease network overhead up to some point, after which the network overhead of intermediate transactions is more than that of full-state commitment transactions. This optimal point depends on other channel parameters, such as the number of peers, the total number of transactions, and the amount of payments. We should note here that the increase in the number of intermediate transactions may add to the risk of the payment channel and should be set carefully.

We also studied the total fee paid for opening and closing a channel under different closure scenarios. The fee in the Bitcoin network depends on the size of the transaction and the fee rate, which is chosen by the submitter. The measured size of transactions and corresponding fee by considering a fee rate of 1.5×10^{-8} BTC/byte is shown in Fig. 9. We consider five scenarios:

1) **Mutual closure**: All parties of the channel agree to close the channel. The closure is done by spending the funding output and giving each participant their share.
2) **Lazy closure**: A party closes the channel unilaterally after a specific time ΔT has passed since the broadcast of the latest full-state commitment transaction.

3) **Instant closure**: Similar to the previous scenario, except that the closure is done instantly with the help of the majority of peers.
4) **Revoke by device**: An old full-state commitment is broadcast, and the cheated device revokes a broadcasted old channel state using its revocation key.
5) **Revoke by peers**: Similar to the previous scenario, with the difference that the cheated party is not online; therefore, the revocation is done after Δt time has passed since the broadcast of the full-state commitment by one of the peers using the pre-signed revocation transaction.

The fee for funding the channel is the same in all the scenarios. If the devices in the channel agree to close the channel mutually, the fee for broadcasting commitment transactions is not paid, and therefore, the total fee is the least among other scenarios. In the case of unilateral closure, the closing device can close the channel by waiting for the timelock or close it instantly by gathering the signatures of the majority of peers—the latter case results in paying the highest fee. In case of cheating, the fee paid for closing the channel, whether the cheated device is online or offline, is almost the same. This fee is also close to the paid fee in the lazy closure scenario because both of them involve the submission of a commitment transaction and spending it without the signature of peers. A possible consequence of this closeness is that the devices are urged to keep the channel open as much as possible and close it only mutually.

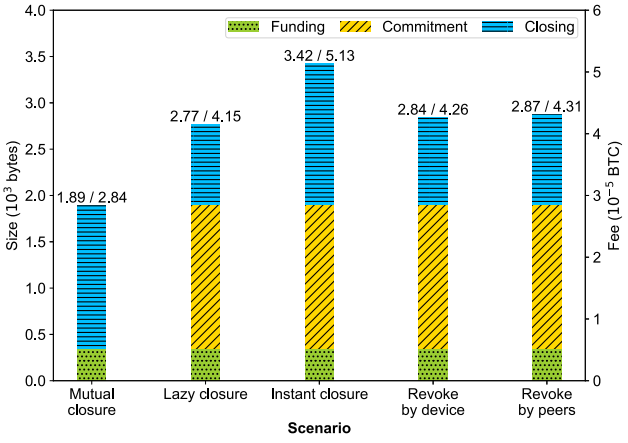

FIGURE 9. Size and submission fee for all transactions in a channel's lifetime under different scenarios.

Finally, we sum up the implementation findings by calculating the exact formula for performance metrics of all stages and scenarios in the LIoTning protocol. The results are reported in Table 4. Each column of this table is derived as follows:

- **On-chain transactions**: The only transactions that need to be submitted on-chain are the funding (opening) and closing transactions. Therefore, full-state commitments and intermediate transactions are not published on-chain. Typically, a single closing transaction is sufficient for cooperative channel closure. In the case of unilateral closure, the party must broadcast a full-state commitment transaction, followed by either a lazy or instant closure transaction, resulting in a total of two on-chain transactions. In case of cheating, the revocation transaction by the device or a pre-signed revocation transaction by the peers must be submitted after the old full-state commitment transaction, increasing the total number of on-chain transactions by one.
- **Off-chain messages**: Assuming that peer selection is handled through pre-communication, the funding phase requires only one off-chain message between the two devices. For each full-state commitment, the initiating device must contact its counterpart once for the update, once for the revocation transaction, and $|\mathcal{P}|$ times for both the update and the pre-signed revocation transactions–totaling $2(1 + |\mathcal{P}|)k$ messages. The secondary device needs to send one message for the update, one for the revocation transaction, and $|\mathcal{P}|$ messages for the pre-signed revocation transactions–totaling $(2 + |\mathcal{P}|)k$ messages. For intermediate transactions, no peers are contacted, and one message is exchanged per transaction. In the case of mutual closure, each party can sign the transaction in turn, and the initiating device must exchange $|\mathcal{P}|$ messages with the peers. For instant closure, only the initiating device needs to contact the peers. For lazy closure and revocation by the device or peers, no off-chain messages are required.
- **Signature count by device**: For funding the channel, each device signs the input to the channel (assuming a single input). For each full-state commitment, each device signs the new state, the revocation transaction, and the pre-signed revocation transaction–resulting in three signatures. For intermediate transactions, each device must sign three versions of this type of transaction. In the case of mutual channel closure, a single signature is sufficient when no cheating occurs. To revoke a broadcasted old state by the device, one signature using the device's own private key is required. However, in revocation by peers, no signature from the device is needed, as the signature for the pre-signed revocation transaction has already been provided during the update phase.
- **Transaction size**: The size of each transaction consists of two parts: variable and constant. The constant part of the transaction includes parts that do not depend on the number of peers, such as headers, fixed parts of the script, and the public key or signature of the devices. On the other hand, variable parts of the transaction depend on the number of public keys and signatures included in the transaction, which are proportional to the number of peers or the majority of peers in the channel. For all transactions, except the funding, the size depends on the number of peers' public keys. Additionally, the size of the transactions that require the signature of the majority of peers depends on the number of signatures made by them.

TABLE 4. Fee and communication costs in different procedures in LIoTning.

Procedure	On-chain transactions	Off-chain messages[1]		Signature count by device	Transaction size (bytes)[2]
		By devices[3]	By each peer		
Funding[4]	1	1 / 1	0	1	352
Full-state commitment	0	$2(1+\|\mathcal{P}\|)k$ / $(2+\|\mathcal{P}\|)k$[5]	$3k$	$3k$	$34\|\mathcal{P}\|+72\lfloor\|\mathcal{P}\|/2+1\rfloor+352$
Intermediate	0	$\sum_{i=1}^{k} r_i$ / 0	0	$3\sum_{i=1}^{k} r_i$	$34\|\mathcal{P}\|+356$
Mutual closure	1	$1+\|\mathcal{P}\|$ / 1	1	1	$34\|\mathcal{P}\|+72\lfloor\|\mathcal{P}\|/2+1\rfloor+352$
Lazy closure	2	0 / 0	0	1	$34\|\mathcal{P}\|+322$
Instant closure	2	$\|\mathcal{P}\|$ / 0	1	1	$34\|\mathcal{P}\|+72\lfloor\|\mathcal{P}\|/2+1\rfloor+324$
Revoke by device	2	0 / 0	0	1	$34\|\mathcal{P}\|+394$
Revoke by peers	2	0 / 0	0	0	$34\|\mathcal{P}\|+428$

[1] The maximum number is reported considering all peers are contacted for signatures.
[2] We assume each public key is encoded with 34 bytes and each signature is encoded with 72 bytes. Due to the use of DER encoding for the signature, its size might vary by one byte.
[3] The number on the left of / corresponds to the initiating device, and the number on the right corresponds to the secondary device.
[4] We assume the funds used for opening the channel are locked with P2PKH scripts.
[5] The symbols used in this table are defined in Table 3, and k represents the latest channel state.

Notable observations in the table are the constant size of the funding transaction, the linear relationship between the number of peers and the size of the transaction, and the independence of the network overhead of peers from the number of intermediate transactions.

VII. CONCLUSION AND FUTURE WORK

In this paper, we proposed LIoTning, a peer-supervised payment channel protocol operating under the honest majority assumption, inspired by the Bitcoin Lightning Network, to enable micropayments for resource-constrained IoT devices within a horizontally integrated ecosystem. The primary objective is to address key limitations in previous works–specifically, the requirement for devices to be constantly online and monitor the blockchain, or delegating this task to trusted third parties (e.g., gateways). Another motivation for our work is mitigating hardware overhead (e.g., TEEs) while leveraging the availability of resource-efficient IoT devices and other blockchain nodes. Furthermore, we introduced intermediate transactions between full-state commitment transactions to reduce latency and communication overhead. We also designed new locking scripts that allow devices to immediately retrieve their funds upon broadcasting the latest channel state, eliminating the strict need to wait for a timelock.

A theoretical security analysis was provided to prove that IoT devices do not lose money in cases of cheating, collusion, or ransom attacks. A formal security proof of the LIoTning protocol is outlined within the Universal Composability (UC) framework, providing security and efficiency guarantees under realistic adversarial models. We implemented a proof-of-concept, and the evaluation results showed that the proposed protocol can be used in practice by resource-constrained devices. The measures of performance metrics in our experiments indicate that the proposed solution supports over 200 transactions per second between two devices with 16 supervisor nodes. Multiplying this number by the commitment and intermediate transaction size yields a network bandwidth usage of 300 kilobytes per second, which is fairly below the usual network bandwidth available for IoT devices. The system can therefore easily keep up with transaction throughput demands in IoT ecosystems while having a low communication and processing footprint.

LIoTning protocol, introduced by this research, has some limitations, which pave the way for future work:

- **Scaling the multi-hop protocol**. An interesting direction for future work could be to answer the question of how to make scaling multi-hop payments feasible. While it is practical to employ peers from existing channels (e.g., $Ch_{d_1d_2}$ between d_1 and d_2) for establishing new channels (e.g., $Ch_{d_3d_4}$ between d_3 and d_4) in LIoTning, it is currently not feasible for devices to utilize already deployed channels in a multi-hop manner for their payments. This is an extension to prevent the opening of channels that devices do not necessarily need for their payments. We argue that this is achievable by defining extended peer roles and new scripts. Specifically, this requires (i) a complicated selection algorithm at the opening phase and (ii) a dynamic coordination mechanism that enables devices to securely engage with pre-existing channels.
- **Inclusion and exclusion of peers**. Due to the availability and presence of peers, an approach to enhance the efficiency of LIoTning can involve the exclusion of a set of peers voluntarily or forced (e.g., $E \subset \mathcal{P}$) and inclusion of a set of new peers that meet the necessary requirements to participate in the channel.
- **Privacy-preserving protocol**. In our peer-supervised payment channel approach, the privacy of channel states is not protected from the peers. Designing a privacy-preserving protocol that prevents peers from accessing these states while still allowing them to

verify their validity is not straightforward. Nevertheless, exploring such a solution could be a promising direction for future work.

- **Extend to multi-peer channel**. In our current peer-supervised protocol, peers assist in maintaining and tracking states for payments between two devices. This implies that a separate payment channel must be created for each pair of devices that need to make payments to each other. Extending this protocol to allow more than two devices to participate in payments and making their liquidity fully available inside a multi-peer channel is a very practical enhancement. One potential approach is to generalize funding and commitment transactions to include multiple parties. However, this introduces the challenge of ensuring fund security in the presence of more than two devices.

APPENDIX
FORMAL SECURITY PROOF

In this section, we present a formal version of the protocol in a manner compatible with the Universal Composability (UC) framework [19]. Specifically, we utilize the Global Universal Composability (GUC) framework [62], an extension of the standard UC framework, to formally assess the security of our proposed payment channel. Our model is inspired by the model used to prove the security of SleepyChannel [60]. Our model captures the same security and efficiency notions, except that we additionally achieve support for multi-peer coordination, intermediate micropayments under timing and count thresholds, and revocation mechanisms tailored for IoT environments.

To enhance the readability of the protocol, we omit trivial checks that an honest user would naturally perform. These checks include verifying that the parameters provided by the environment are well-formed, that both users possess the appropriate amount of funds, and that the channels to be updated or closed exist. Additionally, the checks ensure that the new state is valid and confirm that a channel is not currently undergoing update or closure. These checks can be systematically managed by employing a protocol wrapper [63] that performs validation on messages from the environment and discards any invalid ones. We first informally outline the most important security and efficiency notions of interest that a LIoTning payment channel should provide:

- **Consensus on creation**: A payment channel γ is successfully created only if both devices in γ.devices, along with their selected peers in γ.peers, agree with the creation. Moreover, parties in γ.devices and γ.peers agree on whether the channel is created or not within an a priori bounded number of rounds.
- **Consensus on update**: A payment channel is successfully updated only if both devices in γ.devices and the majority of peers in γ.peers agree with the update within an a priori bounded number of rounds.
- **Conditional consensus on closure**: A payment channel γ can be closed with pre-committed balances for each party when a device in γ.devices requests the closure. The closure may carry on either after a specific amount of time has passed since the request without any party disputing it or when the majority of peers in γ.peers agree to close the channel. Alternatively, the channel may also be closed with corresponding balances when all parties, including devices and peers, agree on closure.
- **Revocability of cheating**: In the event of a cheating attempt, i.e. broadcasting an old state of the channel, the cheated device $d_i \in \gamma$.devices can enforce a state in which it receives the entire channel balance γ.cash if it is online. If d_i is offline, any peer $p_i \in \gamma$.peers can broadcast a pre-signed revocation transaction on its behalf.
- **Validity of intermediate transactions**: The intermediate transactions made between two state updates are guaranteed to be valid transactions that can be submitted to the blockchain. Additionally, if one side of the channel decides to close the channel or broadcast a full-state commitment, the other side of the channel can take its share of the money according to the latest intermediate transaction.

For the remainder of this section, we first introduce the notation used in the proof. We then model the LIoTning protocol and its corresponding ideal functionality, which is assumed to be secure. Next, we demonstrate the existence of simulators for each phase of the protocol that produce the same observable effects on the environment as the real protocol, thereby completing the UC security proof by presenting a series of supporting lemmas.

A. NOTATION

The LIoTning protocol Π is run between two devices, d_1 and d_2, and a set of peers $\mathcal{P}$. The protocol is executed in the presence of an adversary $\mathcal{A}$, who may corrupt any party/parties at the beginning of the execution. Devices, peers, and the adversary receive their inputs from an entity called the environment $\mathcal{E}$, which represents anything "external" to the current protocol execution. We assume a synchronous communication network, meaning that protocol execution happens in rounds, formalized via a global ideal functionality $\mathcal{F}_{clock}$.

A LIoTning channel γ is defined as an attribute tuple $\gamma := (\gamma.\text{id}, \gamma.\text{devices}, \gamma.\text{cash}, \gamma.\text{st}, \gamma.\text{peers}, \gamma.\text{params})$, where $\gamma.\text{id} \in \{0,1\}^*$ is the unique channel identifier, $\gamma.\text{devices} = \{d_1, d_2\}$ defines the two devices of the channel, and $\gamma.\text{cash} \in \mathbb{R}^{\geq 0}$ denotes the total funds locked in the channel. $\gamma.\text{st} = (\theta_1, \ldots, \theta_n)$ is the state of γ composed of a list of outputs. Each output θ_i has two attributes: (1) the value $\theta_i.\text{cash} \in \mathbb{R}^{\geq 0}$ indicating a balance, and (2) the condition $\theta_i.\varphi : \{0,1\}^* \to \{0,1\}$ representing the spending condition. Additionally, $\gamma.\text{peers} = g_{d_1} \cup g_{d_2}$ denotes the collective set

of peer devices supporting the channel. Protocol parameters such as timeouts ΔT, Δt, intermediate thresholds (n_{th}, t_{th}), and revocation keys are included in γ.params. We denote by $m \hookrightarrow\!\!\leftarrow \tau P$ the output of message m to party P in round τ. Similarly, $m \hookrightarrow \tau P$ denotes the input of message m from party P in round τ. A message m typically consists of a pair (MESSAGE−ID, *parameters*). For simplicity, we omit session identifiers unless necessary. In our communication model, messages sent between the two devices are delivered with a one-round delay. That is, if device d_1 sends a message to device d_2 in round τ, it will be received by d_2 in round $\tau + 1$. In contrast, messages sent to the environment $\mathcal{E}$, the simulator $\mathcal{S}$, or the ideal functionality $\mathcal{F}$ are received within the same round.

B. PROTOCOL AND IDEAL FUNCTIONALITY

Using the introduced notation above and in Table 3, the LIoTning protocol Π is formally defined in D. We also capture the desired functionality of Π as an ideal functionality $\mathcal{F}$, which closely mirrors the functionality outlined in Fig. 5. To do so, we describe the steps taken by $\mathcal{F}$ for each phase of the protocol. It should be noted that $\mathcal{F}$ closely follows Π, except that in the ideal setting, all parties behave honestly and cooperatively.

1) CREATE

When both devices of the channel γ, namely d_1 and d_2, send a message (CREATE, γ, tid_{d_p}, g_{d_p}) to the ideal functionality $\mathcal{F}$ within T_p rounds, the functionality expects a funding transaction to appear on the ledger $\mathbb{B}$ within Δ rounds, using both inputs tid_{d_1}, tid_{d_2}, and locking a total amount of γ.cash coins under the funding script address. The channel funding address $\mathrm{Ch}_{d_1 d_2}$ is stored in Γ, and a CREATED message is sent to both devices.

2) UPDATE FULL-STATE COMMITMENT

The paying device $d_p \in \{d_1, d_2\}$ initiates the update by sending the message (UPDATE, γ.id, $\vec{\theta}$, t_{stp}), where γ.id is the channel identifier, $\vec{\theta}$ denotes the proposed new state, and t_{stp} denotes the time needed to setup anything that is built on top of the channel. First, d_p and peers agree on the new state. Then, the ideal functionality $\mathcal{F}$ is informed of a vector of k transactions representing the intended update, which includes a new state and revocation of the previous state. The same procedure is carried out between d_p and the other device. $\mathcal{F}$ proceeds with the update only if the following three conditions are satisfied: (i) all peers send a SETUP−OK message within T_p rounds; (ii) each device sends a confirmation message UPDATE−OK; and (iii) all parties subsequently send a REVOKE message. If all these conditions are met, $\mathcal{F}$ outputs UPDATED to all parties. In case of an error in any step, the ForceClose sub-procedure is executed, which initiates the unilateral closure of the channel.

3) UPDATE INTERMEDIATE

For making intermediate transactions, the procedure is similar to updating full-state commitments. Other than the change in the identifiers of the exchanged messages, the main difference is that peers are not contacted for the update.

4) CLOSE

The closure, where both devices agree to close the channel, is initiated by $\mathcal{F}$ after receiving CLOSE message. The initiating device gathers the signature on the settlement transaction from the majority of peers by contacting them with peerSign message. For each peer that signed the closure transaction, $\mathcal{F}$ sends PEER−CLOSED to the environment. Next, both devices sign the settlement transaction by interacting with each other. Finally, after submitting the settlement transaction on $\mathbb{B}$, $\mathcal{F}$ informs the environment about the closure by sending CLOSED message to it.

5) FORCECLOSE

The ForceClose sub-procedure is used when a device closes the channel unilaterally. This generally occurs when the counterparty device or the majority of peers are not responsive or when an error occurs while creating signatures or submitting a transaction on $\mathbb{B}$. The procedure starts by trying to gather signatures on the closure transaction spending from the latest commitment from most of the peers by sending peerSign message to them. $\mathcal{F}$ sends PEER−CLOSED to the environment for each successful peer signature. If the device succeeds in gathering signatures from the majority of peers, it closes the channel γ instantly. Otherwise, the device submits the latest commitment with only its own signatures and waits for ΔT to submit the closure transaction. At the end of either case, $\mathcal{F}$ informs the environment about the closure by sending a CLOSED message to it.

6) REVOKE

Revoking an old state can be done by a device or a peer. Both procedures follow similar steps. It starts with $\mathcal{F}$ receiving REVOKE message. The revoking party then scans $\mathbb{B}$ to check if any revoked state of any channel is broadcasted on it. If so, the corresponding signed or pre-signed revocation transaction is submitted. After successful revocation $\mathcal{F}$ outputs REVOKED message to the environment.

C. SIMULATORS

To demonstrate that the protocol Π is a universally composable (UC) realization of the functionality $\mathcal{F}$, we begin by defining simulators for each phase of the protocol. For each phase, the simulator, which is detailed in D, interacts with the ideal functionality $\mathcal{F}$ to generate sequences of messages that are intended to be indistinguishable from those produced in the real-world execution of protocol Π under an arbitrary adversary $\mathcal{A}$. We assume that the simulator has the capability to generate valid signatures on behalf of honest parties. Additionally, the simulator interacts with supporting functionalities beyond $\mathcal{F}$, such as digital signature and ledger interfaces, to simulate realistic execution traces.

D. SIMULATION PROOF

In this subsection, we present a series of lemmas to establish the equivalence between the execution ensembles of the real-world scenario ($EXEC_{\Pi,\mathcal{A},\mathcal{E}}$) and the ideal-world scenario ($EXEC_{\mathcal{F},\mathcal{S},\mathcal{E}}$). That is, we show that for every adversary $\mathcal{A}$ in the real world, there exists a simulator $\mathcal{S}$ such that no environment $\mathcal{E}$ can distinguish between the real and ideal executions. To formalize this, we define $m[t]$ to represent the message m observed by the environment in round t. Messages directed to parties other than the environment are considered adversarially controlled and thus delayed by one round before being observed. We also define obsSet(a, t) as the set of all observable side effects triggered by action a in round t, allowing us to precisely capture the effects visible to $\mathcal{E}$. Finally, we assume the presence of a ledger $\mathbb{B}$ that supports transaction authorization and absolute time-locks, which are necessary for accurately simulating on-chain behaviors in both the real and ideal worlds.

Lemma 1: The Create phase of Π UC-realizes the Create phase of $\mathcal{F}$.

Proof: We consider the case where device d_1 is honest and d_2 is corrupted. Note that the reverse case is symmetric.

Real world: After receiving `CREATE` in round t_0, d_1 sends message createInfo to d_2 in t_0. If d_1 receives also createInfo in $t_0 + 1$ from d_2, d_1 will perform first the action a_0 := "run address generation" in round $t_0 + 1$ and on success, create the transactions for the channel followed by a_1 := "create signatures" in round t_0+1+t_{addr}. If this is successful, d_1 generates the signature for the funding transaction tx_{Fu} and sends the signature via createFund to d_2 in $t_0 + 1 + t_{\text{addr}} + t_{\text{sign}}$. If d_1 receives also createFund from d_2 in round $t_0 + 2 + t_{\text{addr}} + t_{\text{sign}}$, it performs action a_2 := "Post tx_{Fu} on $\mathbb{B}$". If it is accepted in round $t_1 \leq t_0 + 2 + t_{\text{addr}} + t_{\text{sign}} + \Delta$, finally d_1 outputs `CREATED`. Thus, the execution ensemble is $EXEC^{\text{create}}_{\Pi,\mathcal{A},\mathcal{E}}$:= {createFund$[t_0 + 1]$, obsSet$(a_0, t_0 + 1)$, obsSet$(a_1, t_0 + 1 + t_{\text{addr}})$, createFund$[t_0 + 2 + t_{\text{addr}} + t_{\text{sign}}]$, obsSet$(a_2, t_0 + 2 + t_{\text{addr}} + t_{\text{sign}})$, `CREATED`$[t_1]$}.

Ideal world: After d_1 sends `CREATE` in round t_0 to $\mathcal{F}$, the simulator sends message createInfo to d_2. If d_2 sends createInfo to d_1, the simulator informs $\mathcal{F}$ and performs a_0 in round $t_0 + 1$. Upon success, $\mathcal{S}$ creates the transactions for the channel and performs a_1 in round $t_0 + 1 + t_{\text{addr}}$. If this was successful, the simulator on behalf of d_1 generates the signature of tx_{Fu} and sends createFund to d_2 in round $t_0 + 1 + t_{\text{addr}} + t_{\text{sign}}$. If d_2 also sends createFund to d_1 received in $t_0 + 2 + t_{\text{addr}} + t_{\text{sign}}$, performs $a_2 := t_0 + 2 + t_{\text{addr}} + t_{\text{sign}} + \Delta$. If the funding transtion is accepted in round $t_1 \leq t_0 + 2 + t_{\text{addr}} + t_{\text{sign}} + \Delta$, $\mathcal{F}$ (which expects it after being informed by $\mathcal{S}$) outputs `CREATED` in $t_1 \leq t_0 + 2 + t_{\text{addr}} + t_{\text{sign}} + \Delta$. Thus, the execution ensemble is $EXEC^{\text{create}}_{\mathcal{F},\mathcal{S},\mathcal{E}}$:= {createFund$[t_0 + 1]$, obsSet$(a_0, t_0 + 1)$, obsSet$(a_1, t_0 + 1 + t_{\text{addr}})$, createFund$[t_0 + 2 + t_{\text{addr}} + t_{\text{sign}}]$, obsSet$(a_2, t_0 + 2 + t_{\text{addr}} + t_{\text{sign}})$, `CREATED`$[t_1]$}. ■

Lemma 2: The Update Full-state Commitment phase of Π UC-realizes the Update Full-state Commitment phase of $\mathcal{F}$.

Proof: We first consider the case where device d_1 along with $N_h \geq \lfloor N/2 \rfloor + 1$ peers are honest and d_2 along with $N - N_h$ peers are corrupted. In all of the subsequent proofs, we only consider an execution path that completes the phase. Branches leading to the execution of other sub-protocols are regarded as secure based on the other proofs presented in this section.

Real world: d_1 upon receiving `UPDATE` in round t_0, first informs d_2 of the update request. It then generates the updated commitment transactions for the new state and signs them. The same procedure is done with the peers in round t_1. Finally, the revocation for d_2 is signed in round $t_2 + 1 + t_{\text{peer}} + t_{\text{sign}}$, followed by the revocation for d_1 in round $t_2 + 2 + t_{\text{peer}} + t_{\text{sign}} + t_{\text{rev}}$. The execution ensemble $EXEC^{\text{updateCommit}}_{\Pi,\mathcal{A},\mathcal{E}}$ follows as a list for better readability.

1) updateReq$[t_0 + 1]$
2) `SETUP`$[t_0 + 2]$
3) updateInfo$[t_1 + 1](t_1 \leq t_0 + 2 + t_{\text{stp1}})$
4) `PEER-SETUP`$[t_1 + 1]$
5) obsSet("sign commitment with peers", $t_1 + 1 + t_{\text{stp2}}$)
6) updateCom$[t_2 + 1 + t_{\text{peer}}](t_2 \leq t_1 + 1 + t_{\text{stp2}})$
7) obsSet("SignTxs", $t_2 + 1 + t_{\text{peer}}$)
8) `UPDATE-OK`$[t_2 + 1 + t_{\text{peer}} + t_{\text{sign}}]$
9) obsSet("sign revocation of d_2", $t_2 + 1 + t_{\text{peer}} + t_{\text{sign}}$)
10) revoke$[t_2 + 2 + t_{\text{peer}} + t_{\text{sign}} + t_{\text{rev}}]$
11) peerRevoke$[t_2 + 2 + t_{\text{peer}} + t_{\text{sign}} + t_{\text{rev}}]$
12) obsSet("sign revocation of d_1", $t_2+2+t_{\text{peer}}+t_{\text{sign}}+t_{\text{rev}}$)
13) peerRevoked$[t_2 + 3 + t_{\text{peer}} + t_{\text{sign}} + t_{\text{rev}}]$
14) `UPDATED`$[t_2 + 3 + t_{\text{peer}} + t_{\text{sign}} + 2t_{\text{rev}}]$
15) `PEER-UPDATED`$[t_2 + 3 + t_{\text{peer}} + t_{\text{sign}} + 2t_{\text{rev}}]$
16) peerRevoked$[t_2 + 4 + t_{\text{peer}} + t_{\text{sign}} + 2t_{\text{rev}}]$

Ideal world: In an ideal world, the simulator would take similar steps to have the same effect on the environment as the real protocol. The execution ensemble $EXEC^{\text{updateCommit}}_{\mathcal{F},\mathcal{S},\mathcal{E}}$ follows as a list for better readability. The actor of each item in the execution is mentioned before the colon mark. The equality of the execution ensemble in the real and ideal world completes the proof.

1) $\mathcal{S}$: updateReq$[t_0 + 1]$
2) $\mathcal{F}$: `SETUP`$[t_0 + 2]$
3) $\mathcal{S}$: updateInfo$[t_1 + 1](t_1 \leq t_0 + 2 + t_{\text{stp1}})$
4) $\mathcal{F}$: `PEER-SETUP`$[t_1 + 1]$
5) $\mathcal{S}$: obsSet("sign commitment with peers", $t_1 + 1 + t_{\text{stp2}}$)
6) $\mathcal{S}$: updateCom$[t_2 + 1 + t_{\text{peer}}](t_2 \leq t_1 + 1 + t_{\text{stp2}})$
7) $\mathcal{S}$: obsSet("SignTxs", $t_2 + 1 + t_{\text{peer}}$)
8) $\mathcal{F}$: `UPDATE-OK`$[t_2 + 1 + t_{\text{peer}} + t_{\text{sign}}]$
9) $\mathcal{S}$: obsSet("sign revocation of d_2", $t_2+1+t_{\text{peer}}+t_{\text{sign}}$)
10) $\mathcal{S}$: revoke$[t_2 + 2 + t_{\text{peer}} + t_{\text{sign}} + t_{\text{rev}}]$
11) $\mathcal{S}$: peerRevoke$[t_2 + 2 + t_{\text{peer}} + t_{\text{sign}} + t_{\text{rev}}]$
12) $\mathcal{S}$: obsSet("sign revocation of d_1", $t_2+2+t_{\text{peer}}+t_{\text{sign}}+t_{\text{rev}}$)
13) $\mathcal{S}$: peerRevoked$[t_2 + 3 + t_{\text{peer}} + t_{\text{sign}} + t_{\text{rev}}]$
14) $\mathcal{F}$: `UPDATED`$[t_2 + 3 + t_{\text{peer}} + t_{\text{sign}} + 2t_{\text{rev}}]$
15) $\mathcal{F}$: `PEER-UPDATED`$[t_2 + 3 + t_{\text{peer}} + t_{\text{sign}} + 2t_{\text{rev}}]$

16) $\mathcal{S}$: peerRevoked$[t_2 + 4 + t_{\text{peer}} + t_{\text{sign}} + 2t_{\text{rev}}]$

We now consider the case where device d_2 along with $N_h \geq \lfloor N/2 \rfloor + 1$ peers are honest and d_1, along with $N - N_h$ peers are corrupted.

Real world: d_2 upon receiving updateReq message in round τ_0, first generates the transactions for the new state and informs the environment. It then sends updateInfo message to d_1 and waits for the message updateCom from it in round τ_1. At this point, d_2 signs the update transactions. Finally, the revocation for d_2 is signed, followed by the revocation for d_1. The execution ensemble $EXEC^{\text{updateCommit}}_{\Pi,\mathcal{A},\mathcal{E}}$ follows as a list for better readability.

1) UPDATE−REQ$[\tau_0]$
2) updateInfo$[\tau_0 + 1]$
3) PEER−SETUP$[\tau_0 + 2 + t_{\text{stp1}}]$
4) obsSet("sign commitment with peers", $\tau_0 + 2 + t_{\text{stp1}} + t_{\text{stp2}}$)
5) SETUP−OK$[\tau_1](\tau_1 \leq \tau_0 + 3 + t_{\text{stp1}} + t_{\text{stp2}} + t_{\text{peer}})$
6) obsSet("SignTxs", τ_1)
7) obsSet("sign revocation of d_2", $\tau_1 + t_{\text{sign}}$)
8) REVOKE − REQ$[\tau_1 + 1 + t_{\text{sign}} + t_{\text{rev}}]$
9) obsSet("sign revocation of d_1", $\tau_1 + 1 + t_{\text{sign}} + t_{\text{rev}}$)
10) peerRevoked$[\tau_1 + 2 + t_{\text{sign}} + t_{\text{rev}}]$
11) revoke$[\tau_1 + 2 + t_{\text{sign}} + 2t_{\text{rev}}]$
12) peerRevoke$[\tau_1 + 2 + t_{\text{sign}} + 2t_{\text{rev}}]$
13) PEER−UPDATED$[\tau_1 + 2 + t_{\text{sign}} + 2t_{\text{rev}}]$
14) peerRevoked$[\tau_1 + 3 + t_{\text{sign}} + 2t_{\text{rev}}]$
15) UPDATED$[\tau_1 + 3 + t_{\text{sign}} + 2t_{\text{rev}}]$

Ideal world: Similar to the previous case, in an ideal world, the simulator would take appropriate steps to have the same effect on the environment as the real protocol. The execution ensemble $EXEC^{\text{updateCommit}}_{\mathcal{F},\mathcal{S},\mathcal{E}}$ follows as a list for better readability. The actor of each item in the execution is mentioned before the colon mark. The equality of the execution ensemble in the real and ideal worlds completes the proof.

1) $\mathcal{F}$: UPDATE−REQ$[\tau_0]$
2) $\mathcal{S}$: updateInfo$[\tau_0 + 1]$
3) $\mathcal{F}$: PEER−SETUP$[\tau_0 + 2 + t_{\text{stp1}}]$
4) $\mathcal{S}$: obsSet("sign commitment with peers", $\tau_0 + 2 + t_{\text{stp1}} + t_{\text{stp2}}$)
5) $\mathcal{F}$: SETUP−OK$[\tau_1](\tau_1 \leq \tau_0 + 3 + t_{\text{stp1}} + t_{\text{stp2}} + t_{\text{peer}})$
6) $\mathcal{S}$: obsSet("SignTxs", τ_1)
7) $\mathcal{S}$: obsSet("sign revocation of d_2", $\tau_1 + t_{\text{sign}}$)
8) $\mathcal{F}$: REVOKE−REQ$[\tau_1 + 1 + t_{\text{sign}} + t_{\text{rev}}]$
9) $\mathcal{S}$: obsSet("sign revocation of d_1", $\tau_1 + 1 + t_{\text{sign}} + t_{\text{rev}}$)
10) $\mathcal{S}$: peerRevoked$[\tau_1 + 2 + t_{\text{sign}} + t_{\text{rev}}]$
11) $\mathcal{S}$: revoke$[\tau_1 + 2 + t_{\text{sign}} + 2t_{\text{rev}}]$
12) $\mathcal{S}$: peerRevoke$[\tau_1 + 2 + t_{\text{sign}} + 2t_{\text{rev}}]$
13) $\mathcal{F}$: PEER−UPDATED$[\tau_1 + 2 + t_{\text{sign}} + 2t_{\text{rev}}]$
14) $\mathcal{S}$: peerRevoked$[\tau_1 + 3 + t_{\text{sign}} + 2t_{\text{rev}}]$
15) $\mathcal{F}$: UPDATED$[\tau_1 + 3 + t_{\text{sign}} + 2t_{\text{rev}}]$

■

Lemma 3: The Update Intermediate phase of Π UC-realizes the Update Intermediate phase of $\mathcal{F}$.

Proof: We first consider the case where device d_1 along with $N_h \geq \lfloor N/2 \rfloor + 1$ peers are honest and d_2 along with $N - N_h$ peers are corrupted.

Real world: d_1 upon receiving UPDATE−INT in round t_0, first informs d_2 of the update request. It then generates the updated intermediate transactions and signs them in round t_1. Finally, d_1 lets the environment know about the update. The execution ensemble $EXEC^{\text{updateInter}}_{\Pi,\mathcal{A},\mathcal{E}}$ follows as a list for better readability.

1) updateReqInt$[t_0 + 1]$
2) SETUP−INT$[t_0 + 2]$
3) obsSet("sign transactions", t_1)$(t_1 \leq t_0 + 2 + t_{\text{stp}})$
4) UPDATED−INT$[t_1 + t_{\text{sign}}]$

Ideal world: In an ideal world, the simulator would take similar steps to have the same effect on the environment as the real protocol. The execution ensemble $EXEC^{\text{updateInter}}_{\mathcal{F},\mathcal{S},\mathcal{E}}$ follows as a list for better readability. The actor of each item in the execution is mentioned before the colon mark. The equality of the execution ensemble in the real and ideal worlds completes the proof.

1) $\mathcal{S}$: updateReqInt$[t_0 + 1]$
2) $\mathcal{F}$: SETUP−INT$[t_0 + 2]$
3) $\mathcal{S}$: obsSet("sign transactions", t_1)$(t_1 \leq t_0 + 2 + t_{\text{stp}})$
4) $\mathcal{F}$: UPDATED−INT$[t_1 + t_{\text{sign}}]$

We now consider the case where device d_2 along with $N_h \geq \lfloor N/2 \rfloor + 1$ peers are honest and d_1 along with $N - N_h$ peers are corrupted.

Real world: d_2 upon receiving updateReqInt message in round τ_0, generates the updated intermediate transactions and signs them in round τ_1. Finally, d_2 lets the environment know about the update. The execution ensemble $EXEC^{\text{updateInter}}_{\Pi,\mathcal{A},\mathcal{E}}$ follows as a list for better readability.

1) UPDATE−REQ−INT$[\tau_0]$
2) updateInfoInt$[\tau_0 + 1]$
3) SETUP−OK−INT$[\tau_1](\tau_1 \leq \tau_0 + 1 + t_{\text{stp}})$
4) obsSet("sign transactions", τ_1)
5) UPDATED−INT$[\tau_1 + t_{\text{sign}}]$

Ideal world: Similar to the previous case, in an ideal world, the simulator would take appropriate steps to have the same effect on the environment as the real protocol. The execution ensemble $EXEC^{\text{updateInter}}_{\mathcal{F},\mathcal{S},\mathcal{E}}$ follows as a list for better readability. The actor of each item in the execution is mentioned before the colon mark. The equality of the execution ensemble in the real and ideal worlds completes the proof.

1) $\mathcal{F}$: UPDATE−REQ−INT$[\tau_0]$
2) $\mathcal{S}$: updateInfoInt$[\tau_0 + 1]$
3) $\mathcal{F}$: SETUP−OK−INT$[\tau_1](\tau_1 \leq \tau_0 + 1 + t_{\text{stp}})$
4) $\mathcal{S}$: obsSet("sign transactions", τ_1)
5) $\mathcal{F}$: UPDATED−INT$[\tau_1 + t_{\text{sign}}]$

■

Lemma 4: The Close phase of Π UC-realizes the Close phase of $\mathcal{F}$.

Proof: We consider the case where d_1 and $N_h \geq \lfloor N/2 \rfloor + 1$ of peers are honest and d_2 and $N - N_h$ peers are corrupted. Note that the reverse case is symmetric.

Real world: After receiving CLOSE in round t_0, d_1 creates a settlement transaction tx_{St} from the latest state of the channel and sends peerSign to peers. d_1 then performs action $a_0 :=$ "create the signature for tx_{St} with peers" in round t_0+1. The process proceeds as follows, depending on the situation. Either: (i) In case of success–meaning valid signatures from at least $\lfloor N/2 \rfloor + 1$ distinct peers are collected–d_1 performs action $a_1 :=$ "create the signature for tx_{St} with d_2" in round t_0+1+t_{peer}. In case of success, it performs action $a_2 :=$ "post $(tx_{\text{St}}, \{\sigma_{\text{St}}, \sigma_{\text{St}}^p$ for $\lfloor N/2 \rfloor + 1$ peers $p \in \mathcal{P}\})$ on $\mathbb{B}$" in round $t_0 + 1 + t_{\text{peer}} + t_{\text{sign}}$. If it appears on $\mathbb{B}$ in round $t_1 \leq t_0 + 1 + t_{\text{peer}} + t_{\text{sign}} + \Delta$, it sends CLOSED. If the signature generation with d_2 is unsuccessful in round $t_2 \geq t_0 + 1 + t_{\text{peer}}$, it runs $a_3 :=$ "ForceClose". Otherwise (ii) it runs action a_3 in round $t_3 \geq t_0 + 1$. Thus, the execution ensemble is $EXEC^{\mathsf{close}}_{\Pi,\mathcal{A},\mathcal{E}} :=$ {peerSign[t_0], obsSet($a_0, t_0 + 1$), obsSet($a_1, t_0 + 1 + t_{\text{peer}}$), obsSet($a_2, t_0 + 1 + t_{\text{peer}} + t_{\text{sign}}$), CLOSED[$t_1$]} or $EXEC^{\mathsf{close}}_{\Pi,\mathcal{A},\mathcal{E}} :=$ {peerSign[t_0], obsSet($a_0, t_0 + 1$), obsSet($a_1, t_0+1+t_{\text{peer}}$), obsSet($a_3, t_2$)}or$EXEC^{\mathsf{close}}_{\Pi,\mathcal{A},\mathcal{E}} :=$ {peerSign[t_0], obsSet($a_0, t_0 + 1$), obsSet(a_3, t_3)}

Ideal world: After d_1 sends CLOSE in round t_0, the simulator handles creating the settlement transaction tx_{St} and sending peerSign to peers. $\mathcal{S}$ then performs action a_0 in round $t_0 + 1$ and proceeds based on the following situations. Either: (i) In case of success–meaning valid signatures from at least $\lfloor N/2 \rfloor + 1$ distinct peers are collected–$\mathcal{S}$ performs action a_1 in round $t_0 + 1 + t_{\text{peer}}$ and action a_2 in round $t_0+1+t_{\text{peer}}+t_{\text{sign}}$, while $\mathcal{F}$ sends CLOSED if the settlement transaction tx_{St} appears on $\mathbb{B}$ in round $t_1 \leq t_0 + 1 + t_{\text{peer}} + t_{\text{sign}} + \Delta$. If the signature generation is unsuccessful in round $t_2 \geq t_0 + 1 + t_{\text{peer}}$, the simulator performs a_3 and instructs $\mathcal{F}$ to do the same (by not sending CLOSE on behalf of d_2). Otherwise (ii) the simulator runs action a_3 in round $t_3 \geq t_0 + 1$. Thus, the execution ensemble is $EXEC^{\mathsf{close}}_{\mathcal{F},\mathcal{S},\mathcal{E}} :=$ {peerSign[t_0], obsSet($a_0, t_0 + 1$), obsSet($a_1, t_0 + 1 + t_{\text{peer}}$), obsSet($a_2, t_0 + 1 + t_{\text{peer}} + t_{\text{sign}}$), CLOSED[$t_1$]} or $EXEC^{\mathsf{close}}_{\mathcal{F},\mathcal{S},\mathcal{E}} :=$ {peerSign[t_0], obsSet($a_0, t_0 + 1$), obsSet($a_1, t_0+1+t_{\text{peer}}$), obsSet($a_3, t_2$)} or $EXEC^{\mathsf{close}}_{\mathcal{F},\mathcal{S},\mathcal{E}} :=$ {peerSign[t_0], obsSet($a_0, t_0 + 1$), obsSet(a_3, t_3)} ■

Lemma 5: The Revoke by device phase of Π UC-realizes the Revoke by device phase of $\mathcal{F}$.

Proof: We consider the case where d_1 is honest and d_2 is corrupted. Note that the reverse case is symmetric.

Real world: After d_1 receives REVOKE from $\mathcal{E}$ in round t_0, d_1 checks if there is a full-state commitment transaction on $\mathbb{B}$ that belongs to an old state of one of its channels. If yes, using the corresponding revocation secret, d_1 performs action $a_0 :=$ "post revocation transaction $tx^{d_2,d_1}_{\text{Rv},i}$" in round t_0. After it is accepted in round $t_1 \leq t_0 + \Delta$, d_1 performs action $a_1 :=$ "post fast withdrawal transaction $tx^{d_2,d_1}_{\text{Fwd},i}$ in round t_1". If that is accepted in round $t_2 \leq t_1 + \Delta$, d_1 outputs message REVOKED. Simultaneously, after d_1 receives REVOKE from $\mathcal{E}$ in round t_0, d_1 checks if there is a full-state commitment transaction on $\mathbb{B}$ that belongs to the latest state of one of its channels and has some intermediate transactions. If yes, d_1 performs action $a_2 :=$ "post appropriate intermediate transaction and corresponding signature" in round $\tau_1 \leq \tau_0 + t_{\text{Int},i,j} + \Delta$ when transaction $tx^{d_2,d_2}_{\text{Inst},i}$ appears on $\mathbb{B}$. If that is accepted in round $\tau_m := \tau_2 \leq \tau_1 + \Delta$, d_1 outputs message REVOKED−INT. d_1 performs action a_2 in round $\tau_3 \leq \tau_0 + \Delta T + t_{\text{Int},i,j} + \Delta$ when transaction $tx^{d_2,d_2}_{\text{Lazy},i}$ appears on $\mathbb{B}$. If that is accepted in round $\tau_m := \tau_4 \leq \tau_3 + \Delta$, d_1 outputs message REVOKED−INT. Thus, the execution ensemble is $EXEC^{\mathsf{revokeDevice}}_{\Pi,\mathcal{A},\mathcal{E}} :=$ {obsSet(a_0, t_0), obsSet(a_1, t_1), REVOKED[t_2]} or $EXEC^{\mathsf{revokeDevice}}_{\Pi,\mathcal{A},\mathcal{E}} :=$ {$o \in$ {obsSet(a_2, τ_1), obsSet(a_2, τ_3)}, REVOKED−INT[τ_m]}

Ideal world: The ideal functionality checks at the end of every round t_0 whether a transaction that spends the funding transaction tx_{Fu}, which is not the most recent state, is on B. If it does, and the other party is honest, it expects a revocation transaction to appear in round $t_1 \leq t_0 + \Delta$. Additionally, it expects the corresponding fast withdrawal transaction from that party to appear in round$t_2 \leq t_1 + \Delta$. If both appear,$\mathcal{F}$ outputs REVOKED in round t_2. Meanwhile, the simulator is responsible for performing actionsa_0 and $a_1$in rounds t_0 and t_1, respectively. Simultaneously, the ideal functionality checks at the end of every round t_0 whether a transaction that spends the funding transaction tx_{Fu}, which is the most recent state, is on $\mathbb{B}$ and whether it includes an intermediate transaction. If it does, and the other party is honest, it expects an intermediate transaction to appear in round $\tau_m := \tau_2 \leq \tau_1 + \Delta$, where$\tau_1 \leq \tau_0 + t_{\text{Int},i,j} + \Delta$denotes the round in which the related instant closure transaction appears on $\mathbb{B}$. Additionally, it expects another intermediate transaction to appear in round$\tau_m := \tau_4 \leq \tau_3 + \Delta$, where $\tau_3 \leq \tau_0 + \Delta T + t_{\text{Int},i,j} + \Delta$ denotes the time of appearance of the corresponding lazy closure transaction. If an intermediate transaction appears, $\mathcal{F}$ outputs REVOKED−INT in round τ_m. Meanwhile, the simulator is responsible for performing action a_2 in roundτ_1 or τ_3, respectively. Thus, the execution ensemble is $EXEC^{\mathsf{revokeDevice}}_{\mathcal{F},\mathcal{S},\mathcal{E}} :=$ {obsSet(a_0, t_0), obsSet(a_1, t_1), REVOKED[t_2]} or $EXEC^{\mathsf{revokeDevice}}_{\mathcal{F},\mathcal{S},\mathcal{E}} :=$ {$o \in$ {obsSet(a_2, τ_1), obsSet(a_2, τ_3)}, REVOKED−INT[τ_m]} ■

Lemma 6: The Revoke by Peer phase of Π UC-realizes the Revoke by Peer phase of $\mathcal{F}$.

Proof: We assume an honest peer p. Note that since a single peer executes this phase, the case where the peer is malicious is not secure.

Real world: After p receives REVOKE from $\mathcal{E}$ in round t_0, p checks if there is a full-state commitment transaction on $\mathbb{B}$ that belongs to an old state of one of the monitored channels. If yes, using the corresponding pre-signed revocation transaction and its signature, p performs action $a_0 :=$ "post pre-signed revocation transaction $tx^{d_2,d_1}_{\text{PreRv},i}$" in round

t_0. After the trasnaction is accepted in round $t_1 \leq t_0 + \Delta$, p outputs message REVOKED. Thus, the execution ensemble is $EXEC^{\mathsf{revokePeer}}_{\Pi,\mathcal{A},\mathcal{E}} := \{\text{obsSet}(a_0, t_0), \texttt{REVOKED}[t_1]\}$

Ideal world: The ideal functionality $\mathcal{F}$ outputs REVOKE at the end of every round t_0 to signal the simulator to check if a transaction spending the output of the funding transaction tx_{Fu} of a monitored channel, which is not the most recent state, is on $\mathbb{B}$. If it does, the simulator performs action a_0. When the pre-signed revocation transaction is accepted by $\mathbb{B}$ in round t_1, and the peer is honest and expects a revocation transaction to appear in round $t_1 \leq t_0 + \Delta$. If both appear, $\mathcal{F}$ outputs REVOKED in round t_2. Meanwhile, the simulator is responsible for performing actions and a_1 in rounds t_0 and t_1, $\mathcal{F}$ outputs REVOKED. Thus, the execution ensemble is $EXEC^{\mathsf{revokePeer}}_{\mathcal{F},\mathcal{S},\mathcal{E}} := \{\text{obsSet}(a_0, t_0), \text{obsSet}(a_1, t_1), \texttt{REVOKED}[t_2]\}$ or $EXEC^{\mathsf{revokePeer}}_{\mathcal{F},\mathcal{S},\mathcal{E}} := \{\text{obsSet}(a_0, t_0), \texttt{REVOKED}[t_1]$ which equals the real-world ensemble. ■

Lemma 7: The ForceClose sub-protocol of Π UC-realizes the ForceClose sub-procedure of $\mathcal{F}$.

Proof: We consider the case where d_1 and $N_h \geq \lfloor N/2 \rfloor + 1$ of peers are honest and d_2 and $N - N_h$ peers are corrupted. Note that the reverse case is symmetric.

Real world: Taking the latest state, d_1 performs action $a_0 :=$ "post $(tx^{d_1}_{\text{Cm},i}, \sigma^{d_2,d_1}_{\text{Cm},i})$ on $\mathbb{B}$" in round t_0. After the transaction appears on $\mathbb{B}$ in round $t_1 \leq t_0 + \Delta$, d_1 sends peerSign to peers in t_1 and performs action $a_1 :=$ "create the signature for $tx^{d_1,d_1}_{\text{Inst},i}$ with peers" in round $t_1 + 1$. In case of success, d_1 do the following in round $t_1 + 1 + t_{\text{peer}}$ depending on the situations. Either (i) valid signatures from at least $\lfloor N/2 \rfloor + 1$ distinct peers are collected, it performs action $a_2 :=$ "post $(tx^{d_1,d_1}_{\text{Inst},i}, \{\sigma^{d_1,d_1}_{\text{Inst},i}, \sigma^{p,d_1}_{\text{Inst},i}$ for $\lfloor N/2 \rfloor + 1$ peers $p \in \mathcal{P}\})$" before ΔT followed by sending CLOSED in round $t_m := t_2 \leq t_1 + 1 + t_{\text{peer}} + \Delta$. Otherwise, (ii) it posts $(tx^{d_1,d_1}_{\text{Lazy},i}, \sigma^{d_1,d_1}_{\text{Lazy},i})$ after ΔT, which we denote as action a_3, followed by sending CLOSED in round $t_m := t_3 \leq t_1 + 1 + t_{\text{peer}} + \Delta T + \Delta$. Thus, the execution ensemble is $EXEC^{\mathsf{forceClose}}_{\Pi,\mathcal{A},\mathcal{E}} := \{\text{obsSet}(a_0, t_0), \text{peerSign}[t_1], \text{obsSet}(a_1, t_1 + 1), o \in \{\text{obsSet}(a_2, t_2), \text{obsSet}(a_3, t_3)\}, \texttt{CLOSED}[t_m]\}$.

Ideal world: Taking the latest state, the simulator mirrors the behavior of the real world. In round t_0, it performs action a_0. After the transaction appears on $\mathbb{B}$in round $t_1 \leq t_0 + \Delta$, it sendspeerSign to peers in t_1. It performs the action a_1 with peers in round $t_1 + 1$and then proceeds based on the following situations. Either (i) valid signatures from at least$\lfloor N/2 \rfloor + 1$ distinct peers are collected, it posts $(tx^{d_1,d_1}_{\text{Inst},i}, \{\sigma^{d_1,d_1}_{\text{Inst},i}, \sigma^{p,d_1}_{\text{Inst},i}$ for $\lfloor N/2 \rfloor + 1$ peers $p \in \mathcal{P}\})$ before ΔT, which we denote as action a_2. Otherwise, (ii) it posts $(tx^{d_1,d_1}_{\text{Lazy},i}, \sigma^{d_1,d_1}_{\text{Lazy},i})$ after ΔT, which we denote as action a_3. If this happens, either in round $t_m := t_2 \leq t_1 + 1 + t_{\text{peer}} + \Delta$ in case (i) or in round $t_m := t_3 \leq t_1 + 1 + t_{\text{peer}} + \Delta T + \Delta$ in case (ii), it outputs CLOSED. Thus, the execution ensemble is $EXEC^{\mathsf{forceClose}}_{\mathcal{F},\mathcal{S},\mathcal{E}} := \{\text{obsSet}(a_0, t_0), \text{peerSign}[t_1], \text{obsSet}(a_1, t_1 + 1), o \in \{\text{obsSet}(a_2, t_2), \text{obsSet}(a_3, t_3)\}, \texttt{CLOSED}[t_m]\}$. ■

LIoTning protocol

Create

Device d_1 upon $(\texttt{CREATE}, \gamma, tid_{d_1}) \xleftarrow{t_0} \mathcal{E}$:

(1) Set id $:= \gamma$.id, generate (pk_{d_1}, sk_{d_1}), $(pk_{\text{Lazy},d_1}, sk_{\text{Lazy},d_1})$, $(pk_{\text{Inst},d_1}, sk_{\text{Inst},d_1})$, $(pk_{\text{Fwd},d_1}, sk_{\text{Fwd},d_1})$ and $(pk_{\text{Rv},d_1}, sk_{\text{Rv},d_1})$. Let $pkey^{d_1}_{\text{set}}$ denote the set of corresponding public keys.

(2) Extract $\upsilon_{d_1,0}$ and $\upsilon_{d_2,0}$ from γ.st.

(3) Send $(\text{createInfo}, \text{id}, tid_{d_1}, pkey^{d_1}_{\text{set}}, pkey^{g_{d_1}}_{\text{set}}) \xhookrightarrow{t_0} d_2$.

(4) If $(\text{createInfo}, \text{id}, tid_{d_2}, pkey^{d_2}_{\text{set}}, pkey^{g_{d_2}}_{\text{set}}) \xhookleftarrow{t_0+1} d_2$, continue. Else, go idle.

(5) Using $pkey^{d_1}_{set}$, $pkey^{d_2}_{set}$, and $pkey^{\mathcal{P}}_{set}$, d_1 together with d_2 runs $\mathcal{F}_{\text{JKGen}}$ to generate the following set of addresses: $addr_{set} := \{Ch_{d_1 d_2}, \text{toLocal}_{d_1}, \text{toLocal}_{d_2}, \text{toRemote}_{d_1}, \text{toRemote}_{d_2}\}$ which takes t_{addr} rounds. In case of failure, abort.

(6) Generate $tx_{\text{Fu}} := tx([tid_{d_1}, tid_{d_2}], Ch_{d_1 d_2}, [\upsilon_{d_1,0} + \upsilon_{d_2,0}])$.

(7) Let $tx_{\text{set}_0} \leftarrow \texttt{GenerateTxs}(addr_{\text{set}}, pkey^{d_1}_{\text{set}}, pkey^{d_2}_{\text{set}}, \upsilon_{d_1,0}, \upsilon_{d_2,0})$.

(8) Let $sig^{d_1}_{\text{set}_0} \leftarrow \texttt{SignTxs}^{d_1}(tx_{set_0}, addr_{\text{set}}, pkey^{d_1}_{\text{set}} \cup pkey^{d_2}_{\text{set}})$.

(9) Device d_1 generates the signature $\sigma^{d_1}_{tid_{d_1}}$ for the output tid_{d_1} and sends $(\text{createFund}, \text{id}, \sigma_{tid_{d_1}}) \xhookrightarrow{t_0+1+t_{\text{addr}}+t_{\text{sign}}} d_2$.

(10) If $(\text{createFund}, \text{id}, \sigma^{d_2}_{tid_{d_2}}) \xhookleftarrow{t_0+2+t_{\text{addr}}+t_{\text{sign}}} d_2$, post $(tx_{\text{Fu}}, \{\sigma^{d_1}_{tid_{d_1}}, \sigma^{d_2}_{tid_{d_2}}\})$ to $\mathbb{B}$.

(11) If tx_{Fu} is accepted by $\mathbb{B}$ in round $t_1 \leq t_0 + 2 + t_{\text{addr}} + t_{\text{sign}} + \Delta$, store $\Gamma^{d_1}(\text{id}) := (tx_{Fu}, tx_{set_0}, sig^{d_1}_{set_0}, pkey^{d_1}_{\text{set}}, pkey^{d_2}_{\text{set}}, pkey^{\mathcal{P}}_{\text{set}}$, and send $(\texttt{CREATED}, \text{id}) \xhookrightarrow{t_1} \mathcal{E}$.

Update full-state commitment

Device d_1 upon $(\texttt{UPDATE}, \text{id}, \vec{\theta}, t_{\text{stp}_1}, t_{\text{stp}_2}) \xhookleftarrow{t_0} \mathcal{E}$

(1) $(\text{updateReq}, \text{id}, \vec{\theta}, t_{\text{stp}_1}) \xhookrightarrow{t_0} d_2$

Device d_2 upon $(\text{updateReq}, \text{id}, \vec{\theta}, t_{\text{stp}_1}) \xhookleftarrow{T_0} d_1$

(2) Retrieve $(tx_{\mathrm{Fu}}, tx_{set_{i-1}}, sig^{d_2}_{set_{i-1}}, addr_{set}, pkey^{d_1}_{\mathrm{set}}, pkey^{d_2}_{\mathrm{set}}, pkey^{\mathcal{P}}_{\mathrm{set}}) = \Gamma^{d_2}(\mathrm{id})$
(3) Extract $v_{d_1,i}$ and $v_{d_2,i}$ from $\vec{\theta}$.
(4) Let $tx_{set_i} \leftarrow \mathtt{GenerateTxs}(addr_{set}, pkey^{d_1}_{set}, pkey^{d_2}_{set}, v_{d_1,i}, v_{d_2,i})$.
(5) Let $\vec{tid} := (tx^{d_1}_{Cm,i}.\mathrm{id}, tx^{d_2}_{Cm,i}.\mathrm{id})$ be a tuple of the transaction ids of transaction $tx^{d_1}_{Cm,i}$ and $tx^{d_2}_{Cm,i}$.
(6) $(\mathtt{UPDATE\text{-}REQ}, \mathrm{id}, \vec{\theta}, t_{\mathrm{stp}_1}, \vec{tid}) \xhookrightarrow{\tau_0} \mathcal{E}$.
(7) $(\mathrm{updateInfo}, \mathrm{id}) \xhookrightarrow{\tau_0} d_1$.

Device d_1 upon $(\mathrm{updateInfo}, \mathrm{id}) \xhookleftarrow{t_0+2} d_2$

(8) Retrieve $(tx_{Fu}, tx_{set_{i-1}}, sig^{d_1}_{set_{i-1}}, addr_{set}, pkey^{d_1}_{set}, pkey^{d_2}_{set}, pkey^{\mathcal{P}}_{\mathrm{set}}) = \Gamma^{d_1}(\mathrm{id})$.
(9) Extract $v_{d_1,i}$ and $v_{d_2,i}$ from $\vec{\theta}$.
(10) Let $tx_{set_i} \leftarrow \mathtt{GenerateTxs}(addr_{set}, pkey^{d_1}_{set}, pkey^{d_2}_{set}, v_{d_1,i}, v_{d_2,i})$.
(11) Let $\vec{tid} := (tx^{d_1}_{Cm,i}.\mathrm{id}, tx^{d_2}_{Cm,i}.\mathrm{id})$ be a tuple of the transaction ids of transaction $tx^{d_1}_{Cm,i}$ and $tx^{d_2}_{Cm,i}$.
(12) $(\mathtt{SETUP}, \mathrm{id}, \vec{tid}) \xhookrightarrow{t_0+2} \mathcal{E}$.
(13) If $(\mathtt{SETUP\text{-}OK}, \mathrm{id}) \xhookleftarrow{t_1 \le t_0+2+t_{\mathrm{stp}_1}} \mathcal{E}$, go to the next step, else go idle.
(14) Send $(\mathrm{updateInfo}, \mathrm{id}, tx^{d_1}_{\mathrm{Cm},i}, tx^{d_2}_{\mathrm{Cm},i}, t_{\mathrm{stp}_2}) \xhookrightarrow{t_1}$ each peer $p \in \mathcal{P}$.
(15) Wait $1 + t_{\mathrm{stp}_2}$ rounds.
(16) d_1 together with each peer p runs the interactive protocol $\mathcal{F}_{Sign}$ simultaneously to generate the signature $\sigma^{p,d_1}_{\mathrm{Cm},i}$ on the commitment transaction $tx^{d_1}_{\mathrm{Cm},i}$ in t_{peer} rounds.

Each peer $p \in \mathcal{P}$ upon $(\mathrm{updateInfo}, \mathrm{id}, tx^{d_1}_{\mathrm{Cm},i}, tx^{d_2}_{\mathrm{Cm},i}, t_{\mathrm{stp}_2}) \xhookleftarrow{t_1+1} d_1$

(17) Let $\vec{tid} := (tx^{d_1}_{Cm,i}.\mathrm{id}, tx^{d_2}_{Cm,i}.\mathrm{id})$ be a tuple of the transaction ids of transaction $tx^{d_1}_{Cm,i}$ and $tx^{d_2}_{Cm,i}$.
(18) $(\mathtt{PEER\text{-}SETUP}, \mathrm{id}, \vec{tid}, t_{\mathrm{stp}_2}) \xhookrightarrow{t_1+1} \mathcal{E}$.
(19) If $(\mathtt{PEER\text{-}SETUP\text{-}OK}, \mathrm{id}) \xhookleftarrow{t_2 \le t_1+1+t_{\mathrm{stp}_2}} \mathcal{E}$, go to the next step, else go idle.
(20) Peer p together with d_1 runs the interactive protocol $\mathcal{F}_{Sign}$ to generate the signature $\sigma^{p,d_1}_{\mathrm{Cm},i}$ on the full-state commitment transaction $tx^{d_1}_{\mathrm{Cm},i}$. This takes t_{peer} rounds.

Device d_1 in round $t_2 + t_{\mathrm{peer}}$

(21) If valid signatures $\sigma^{p,d_1}_{\mathrm{Cm},i}$ from at least $\lfloor N/2 \rfloor + 1$ distinct peers $p \in \mathcal{P}$ are collected, continue. Otherwise, send $(\mathtt{UPDATE\text{-}PEER\text{-}FAIL}, \mathrm{id}) \xhookrightarrow{t_2+t_{\mathrm{peer}}} \mathcal{E}$ and execute $\mathtt{ForceClose}(\mathrm{id})$.
(22) Send $(\mathrm{updateCom}, \mathrm{id}, \sigma^{p,d_1}_{\mathrm{Cm},i}$ from each $p \in \mathcal{P}) \xhookrightarrow{t_2+t_{\mathrm{peer}}} d_2$.
(23) Wait one round.
(24) $\mathtt{SignTxs}^{d_1}(tx_{set_i}, addr_{set}, pkey^{d_1}_{\mathrm{set}} \cup pkey^{d_2}_{\mathrm{set}})$.

Device d_2 upon $(\mathrm{updateCom}, \mathrm{id}) \xhookleftarrow{\tau_1 \le \tau_0+3+t_{\mathrm{stp}_1}+t_{\mathrm{stp}_2}+t_{\mathrm{peer}}} d_1$

(25) $(\mathtt{SETUP\text{-}OK}, \mathrm{id}) \xhookrightarrow{\tau_1} \mathcal{E}$.
(26) If not $(\mathtt{UPDATE\text{-}OK}, \mathrm{id}) \xhookleftarrow{\tau_1} \mathcal{E}$, go idle.
(27) $\mathtt{SignTxs}^{d_1}(tx_{set_i}, addr_{set}, pkey^{d_1}_{\mathrm{set}} \cup pkey^{d_2}_{\mathrm{set}})$.

Device d_1 in round $t_2 + 1 + t_{\mathrm{peer}} + t_{\mathrm{sign}}$

(28) If $sig^{d_1}_{set_i}$ is returned from $\mathtt{SignTxs}^{d_1}$, $(\mathtt{UPDATE\text{-}OK}, \mathrm{id}) \xhookleftarrow{t_2+1+t_{\mathrm{peer}}+t_{\mathrm{sign}}} \mathcal{E}$. Else, execute $\mathtt{ForceClose}(\mathrm{id})$ and go idle.
(29) If not $(\mathtt{REVOKE}, \mathrm{id}) \xhookleftarrow{t_2+1+t_{\mathrm{peer}}+t_{\mathrm{sign}}} \mathcal{E}$, go idle.
(30) Device d_1 together with d_2 runs the interactive protocol $\mathcal{F}_{Sign}$ to generate the signature $\sigma^{d_1,d_2}_{\mathrm{Rv},i-1}$ on the revocation transaction $tx^{d_1,d_2}_{\mathrm{Rv},i-1}$ and $\sigma^{d_1,p}_{\mathrm{PreRv},i-1}$ on the pre-signed revocation transaction $tx^{d_2,d_1}_{\mathrm{PreRv},i-1}$. This takes t_{rev} rounds. In case of failure, execute $\mathtt{ForceClose}(\mathrm{id})$.
(31) $(\mathrm{revoke}, \mathrm{id}, \sigma^{d_1,d_2}_{\mathrm{Rv},i-1}) \xhookrightarrow{t_2+1+t_{\mathrm{peer}}+t_{\mathrm{sign}}+t_{\mathrm{rev}}} d_2$.
(32) $(\mathrm{peerRevoke}, \mathrm{id}, tx^{d_2,d_1}_{\mathrm{PreRv},i-1}, \sigma^{d_1,p}_{\mathrm{PreRv},i-1}) \xhookrightarrow{t_2+1+t_{\mathrm{peer}}+t_{\mathrm{sign}}+t_{\mathrm{rev}}}$ each peer $p \in \mathcal{P}$.

Device d_2 in round $\tau_1 + t_{\mathrm{sign}}$

(33) If $sig^{d_2}_{set_i}$ is not returned from $\mathtt{SignTxs}^{d_1}$, execute $\mathtt{ForceClose}(\mathrm{id})$ and go idle.
(34) Participate in the signing of $tx^{d_1,d_2}_{\mathrm{Rv},i-1}$ and $tx^{d_2,d_1}_{\mathrm{PreRv},i-1}$.
(35) Upon $(\mathrm{revoke}, \mathrm{id}, \sigma^{d_1,d_2}_{\mathrm{Rv},i-1}) \xhookleftarrow{\tau_1+1+t_{\mathrm{sign}}+t_{\mathrm{rev}}} d_1$, continue. Else, execute $\mathtt{ForceClose}(\mathrm{id})$ and go idle.
(36) $(\mathtt{REVOKE\text{-}REQ}, \mathrm{id}) \xhookleftarrow{\tau_1+1+t_{\mathrm{sign}}+t_{\mathrm{rev}}} \mathcal{E}$.
(37) If not $(\mathtt{REVOKE}, \mathrm{id}) \xhookleftarrow{\tau_1+1+t_{\mathrm{sign}}+t_{\mathrm{rev}}} \mathcal{E}$, go idle.

(38) Device d_2 together with d_1 runs the interactive protocol $\mathcal{F}_{Sign}$ to generate the signature $\sigma^{d_2,d_1}_{\mathrm{Rv},i-1}$ on the revocation transaction $tx^{d_2,d_1}_{\mathrm{Rv},i-1}$ and $\sigma^{d_2,p}_{\mathrm{PreRv},i-1}$ on the pre-signed revocation transaction $tx^{d_1,d_2}_{\mathrm{PreRv},i-1}$. This takes t_{rev} rounds. In case of failure, execute `ForceClose`(id).

(39) $(\mathrm{revoke}, \mathrm{id}, \sigma^{d_2,d_1}_{\mathrm{Rv},i-1}) \xhookrightarrow{\tau_1+1+t_{\mathrm{sign}}+2t_{\mathrm{rev}}} d_1$.

(40) $(\mathrm{peerRevoke}, \mathrm{id}, tx^{d_1,d_2}_{\mathrm{PreRv},i-1}, \sigma^{d_2,p}_{\mathrm{PreRv},i-1}) \xhookrightarrow{\tau_1+1+t_{\mathrm{sign}}+2t_{\mathrm{rev}}}$ each peer $p \in \mathcal{P}$.

(41) Upon $(\mathrm{peerRevoked}, \mathrm{id}) \xhookleftarrow{\tau_1+3+t_{\mathrm{sign}}+2t_{\mathrm{rev}}}$ from at least $\lfloor N/2 \rfloor + 1$ distinct peers $p \in \mathcal{P}$, continue, Else, execute `ForceClose`(id) and go idle.

(42) $\Theta^{d_2}(\mathrm{id}) \coloneqq \Theta^{d_2} \cup \left\{ \left(tx_{\mathrm{set}_{i-1}}, sig^{d_2}_{\mathrm{set}_{i-1}}, \sigma^{d_1,d_2}_{\mathrm{Rv},i-1} \right) \right\}$.

(43) $\Gamma^{d_2}(\mathrm{id}) \coloneqq (tx_{\mathrm{Fu}}, tx_{set_i}, sig^{d_2}_{set_i}, addr_{set}, pkey^{d_1}_{\mathrm{set}}, pkey^{d_2}_{\mathrm{set}}, pkey^{\mathcal{P}}_{\mathrm{set}})$.

(44) $(\texttt{UPDATED}, \mathrm{id}) \xhookrightarrow{\tau_1+3+t_{\mathrm{sign}}+2t_{\mathrm{rev}}} \mathcal{E}$.

Device d_1 in round $t_2 + 2 + t_{\mathrm{peer}} + t_{\mathrm{sign}} + t_{\mathrm{rev}}$

(45) Upon $(\mathrm{peerRevoked}, \mathrm{id}) \xhookleftarrow{t_2+2+t_{\mathrm{peer}}+t_{\mathrm{sign}}+t_{\mathrm{rev}}}$ from at least $\lfloor N/2 \rfloor + 1$ distinct peers $p \in \mathcal{P}$, continue. Else, execute `ForceClose`(id) and go idle.

(46) Participate in the signing of $tx^{d_2,d_1}_{\mathrm{Rv},i-1}$ and $tx^{d_1,d_2}_{\mathrm{PreRv},i-1}$.

(47) If $(\mathrm{revoke}, \mathrm{id}, \sigma^{d_1,d_2}_{\mathrm{Rv},i-1}) \xhookleftarrow{t_2+3+t_{\mathrm{peer}}+t_{\mathrm{sign}}+2t_{\mathrm{rev}}} d_2$ and the signature is valid, go to next step. Else, execute `ForceClose`(id).

(48) $\Theta^{d_1}(\mathrm{id}) \coloneqq \Theta^{d_1} \cup \left\{ \left(tx_{\mathrm{set}_{i-1}}, sig^{d_1}_{\mathrm{set}_{i-1}}, \sigma^{d_2,d_1}_{\mathrm{Rv},i-1} \right) \right\}$.

(49) $\Gamma^{d_1}(\mathrm{id}) \coloneqq (tx_{\mathrm{Fu}}, tx_{set_i}, sig^{d_1}_{set_i}, addr_{set}, pkey^{d_1}_{\mathrm{set}}, pkey^{d_2}_{\mathrm{set}}, pkey^{\mathcal{P}}_{\mathrm{set}})$.

(50) $(\texttt{UPDATED}, \mathrm{id}) \xhookrightarrow{t_2+3+t_{\mathrm{peer}}+t_{\mathrm{sign}}+2t_{\mathrm{rev}}} \mathcal{E}$.

Each peer $p \in \mathcal{P}$ in round $t_2 + 2 + t_{\mathrm{peer}} + t_{\mathrm{sign}} + t_{\mathrm{rev}}$

(51) If $(\mathrm{peerRevoke}, \mathrm{id}, tx^{d_2,d_1}_{\mathrm{PreRv},i-1}, \sigma^{d_1,p}_{\mathrm{PreRv},i-1}) \xhookleftarrow{t_2+2+t_{\mathrm{peer}}+t_{\mathrm{sign}}+t_{\mathrm{rev}}} d_1$ and the signatures are valid, go to next step. Else, go idle.

(52) $(\mathrm{peerRevoked}, \mathrm{id}) \xhookrightarrow{t_2+2+t_{\mathrm{peer}}+t_{\mathrm{sign}}+t_{\mathrm{rev}}} d_1$.

(53) If $(\mathrm{peerRevoke}, \mathrm{id}, tx^{d_1,d_2}_{\mathrm{PreRv},i-1}, \sigma^{d_2,p}_{\mathrm{PreRv},i-1}) \xhookleftarrow{t_2+3+t_{\mathrm{peer}}+t_{\mathrm{sign}}+2t_{\mathrm{rev}}} d_2$, and the signatures are valid, go to next step. Else, go idle.

(54) $(\mathrm{peerRevoked}, \mathrm{id}) \xhookrightarrow{t_2+3+t_{\mathrm{peer}}+t_{\mathrm{sign}}+2t_{\mathrm{rev}}} d_2$.

(55) $\Theta^{p,d_1}(\mathrm{id}) \coloneqq \Theta^{p,d_1} \cup \left\{ (tx^{d_1}_{\mathrm{Cm},i-1}, tx^{d_1,d_2}_{\mathrm{PreRv},i-1}, \sigma^{d_2,p}_{\mathrm{Rv},i-1}) \right\}$.

(56) $\Theta^{p,d_2}(\mathrm{id}) \coloneqq \Theta^{p,d_2} \cup \left\{ (tx^{d_2}_{\mathrm{Cm},i-1}, tx^{d_2,d_1}_{\mathrm{PreRv},i-1}, \sigma^{d_2,p}_{\mathrm{Rv},i-1}) \right\}$.

(57) $(\texttt{PEER-UPDATED}, \mathrm{id}) \xhookrightarrow{t_2+3+t_{\mathrm{peer}}+t_{\mathrm{sign}}+2t_{\mathrm{rev}}} \mathcal{E}$.

Update intermediate

Device d_1 upon $(\texttt{UPDATE-INT}, \mathrm{id}, \vec{\theta'}, t_{\mathrm{stp}}) \xhookleftarrow{t_0} \mathcal{E}$

(1) $(\mathrm{updateReqInt}, \mathrm{id}, \vec{\theta'}, t_{\mathrm{stp}}) \xhookrightarrow{t_0} d_2$.

Device d_2 upon $(\mathrm{updateReqInt}, \mathrm{id}, \vec{\theta'}, t_{\mathrm{stp}}) \xhookleftarrow{\tau_0} d_1$

(2) Extract $(tx_{Fu}, tx_{set_i}, sig^{d_1}_{set_i}, addr_{set}, pkey^{d_1}_{set}, pkey^{d_2}_{set}, pkey^{\mathcal{P}}_{\mathrm{set}}) = \Gamma^{d_2}(\mathrm{id})$.

(3) Extract $v^j_{d_1,i}$ and $v^j_{d_2,i}$ from $\vec{\theta'}$.

(4) Create transactions $tx^{d_1,d_2}_{\mathrm{IntSet},i,j} \coloneqq \{tx(pk_{\mathrm{Lazy},d_1}, [pk_{d_1}, pk_{d_2}], [v^j_{d_1,i}, v^j_{d_2,i}]), tx(pk_{\mathrm{Inst},d_1}, [pk_{d_1}, pk_{d_2}], [v^j_{d_1,i}, v^j_{d_2,i}]),$ $tx(pk_{\mathrm{Fwd},d_1}, [pk_{d_1}, pk_{d_2}], [v^j_{d_1,i}, v^j_{d_2,i}])\}$ which all of them can be spent after timelock $t^{d_1,d_2}_{\mathrm{Int},i,j}$.

(5) Let $\vec{tid}$ be the tuple of the transaction ids of transactions in $tx^{d_1,d_2}_{\mathrm{IntSet},i,j}$.

(6) $(\mathrm{updateInfoInt}, \mathrm{id}, \vec{tid}, t^{d_1,d_2}_{\mathrm{Int},i,j}) \xhookrightarrow{\tau_0} d_1$.

(7) $(\texttt{UPDATE-REQ-INT}, \mathrm{id}, \vec{\theta'}, t_{\mathrm{stp}}, \vec{tid}) \xhookrightarrow{\tau_0} \mathcal{E}$.

Device d_1 upon $(\mathrm{updateInfoInt}, \mathrm{id}, \vec{tid}, t^{d_1,d_2}_{\mathrm{Int},i,j}) \xhookleftarrow{t_0+2} d_2$

(8) Extract $(tx_{Fu}, tx_{set_i}, sig^{d_1}_{set_i}, addr_{set}, pkey^{d_1}_{set}, pkey^{d_2}_{set}, pkey^{\mathcal{P}}_{\mathrm{set}}) = \Gamma^{d_1}(\mathrm{id})$.

(9) Extract $v^j_{d_1,i}$ and $v^j_{d_2,i}$ from $\vec{\theta'}$.

(10) Create transactions $tx^{d_1,d_2}_{\mathrm{IntSet},i,j} \coloneqq \{tx(pk_{\mathrm{Lazy},d_1}, [pk_{d_1}, pk_{d_2}], [v^j_{d_1,i}, v^j_{d_2,i}]), tx(pk_{\mathrm{Inst},d_1}, [pk_{d_1}, pk_{d_2}], [v^j_{d_1,i}, v^j_{d_2,i}]),$ $tx(pk_{\mathrm{Fwd},d_1}, [pk_{d_1}, pk_{d_2}], [v^j_{d_1,i}, v^j_{d_2,i}])\}$ which all of them can be spent after timelock $t^{d_1,d_2}_{\mathrm{Int},i,j}$.

(11) Let $\vec{tid}$ be the tuple of the transaction ids of transactions in $tx^{d_1,d_2}_{\mathrm{IntSet},i,j}$.

(12) $(\texttt{SETUP-INT}, \mathrm{id}, \vec{tid}) \xhookrightarrow{t_0+2} \mathcal{E}$.

(13) If $(\texttt{SETUP-OK-INT}, \text{id}) \xleftarrow{t_1 \le t_0+2+t_{\text{stp}}} \mathcal{E}$, run the interactive protocol $\mathcal{F}_{Sign}$ to generate the signatures $\sigma^{d_1,d_2}_{\text{IntSet},i,j}$ on $tx^{d_1,d_2}_{\text{IntSet},i,j}$ in t_{sign} rounds. In case of failure, execute $\texttt{ForceClose}(\text{id})$.

(14) $\Theta^{d_1}_{\text{Int},i}(\text{id}) := \Theta^{d_1}_{\text{Int},i} \cup \left\{ \left(tx^{d_1,d_2}_{\text{IntSet},i,j}, \sigma^{d_1,d_2}_{\text{IntSet},i,j} \right) \right\}$.

(15) $(\texttt{UPDATED-INT}, \text{id}) \xleftarrow{t_1+t_{\text{sign}}} \mathcal{E}$.

Device d_2 in round $\tau_1 \le \tau_0 + 1 + t_{\text{stp}}$

(16) $(\texttt{SETUP-OK-INT}, \text{id}) \xleftarrow{\tau_1} \mathcal{E}$.

(17) If not $(\texttt{UPDATE-OK-INT}, \text{id}) \xleftarrow{\tau_1} \mathcal{E}$, go idle.

(18) Participate in the signing of $tx^{d_1,d_2}_{\text{IntSet},i,j}$ to generate $\sigma^{d_1,d_2}_{\text{IntSet},i,j}$.

(19) $\Theta^{d_2}_{\text{Int},i}(\text{id}) := \Theta^{d_2}_{\text{Int},i} \cup \left\{ \left(tx^{d_1,d_2}_{\text{IntSet},i,j}, \sigma^{d_1,d_2}_{\text{IntSet},i,j} \right) \right\}$.

(20) $(\texttt{UPDATED-INT}, \text{id}) \xleftarrow{\tau_1+t_{\text{sign}}} \mathcal{E}$.

Close

Device d_1 upon $(\texttt{CLOSE}, \text{id}) \xleftarrow{t_0} \mathcal{E}$:

(1) Extract $(tx_{Fu}, tx_{set_i}, sig^{d_1}_{set_i}, addr_{set}, pkey^{d_1}_{set}, pkey^{d_2}_{set}, pkey^{\mathcal{P}}_{\text{set}}) = \Gamma^{d_1}(\text{id})$.

(2) Extract $\upsilon_{d_1,i}$ and $\upsilon_{d_2,i}$ from $tx^{d_1}_{\text{Cm},i} \in tx_{set_i}$.

(3) Create transaction $tx_{\text{St}} := tx(Ch_{d_1 d_2}, [pk_{d_1}, pk_{d_2}], [\upsilon_{d_1,i}, \upsilon_{d_2,i}])$.

(4) Send $(\text{peerSign}, \text{id}, tx_{\text{St}}) \xleftarrow{t_0}$ each peer $p \in \mathcal{P}$.

(5) Wait one round.

(6) d_1 together with each peer p runs the interactive protocol $\mathcal{F}_{Sign}$ simultaneously to generate the signature $\sigma^{p,d_1}_{\text{St},i}$ on the settlement transaction tx_{St} in t_{peer} rounds.

Each peer $p \in \mathcal{P}$ upon $(\text{peerSign}, \text{id}, tx_{\text{St}}) \xleftarrow{\upsilon_0 \top \lambda} d_1$

(7) $(\texttt{PEER-CLOSED}, \text{id}) \xleftarrow{t_0+1} \mathcal{E}$.

(8) Peer p together with d_1 runs the interactive protocol $\mathcal{F}_{Sign}$ to generate the signature $\sigma^{p,d_1}_{\text{St},i}$ on the settlement transaction tx_{St}. This takes t_{peer} rounds.

Device d_1 in round $t_0 + 1 + t_{\text{peer}}$:

(9) If valid signatures σ^{p}_{St} from at least $\lfloor N/2 \rfloor + 1$ distinct peers $p \in \mathcal{P}$ are collected, continue. Otherwise, execute $\texttt{ForceClose}(\text{id})$.

(10) Device d_1 together with d_2 runs the interactive protocol $\mathcal{F}_{Sign}$ to generate the following signature, σ_{St} on the transaction tx_{St}. This takes t_{sign} rounds.

(11) In case the signature generation was successful, post $(tx_{\text{St}}, \{\sigma_{\text{St}}, \sigma^{p}_{\text{St}}$ for $\lfloor N/2 \rfloor + 1$ peers $p \in \mathcal{P}\})$ on $\mathbb{B}$. Else, execute $\texttt{ForceClose}(\text{id})$.

(12) If tx_{St} appears on $\mathbb{B}$ in round $t_1 \le t_0 + 1 + t_{\text{peer}} + t_{\text{sign}} + \Delta$, set $\Theta^{d_1}(\text{id}) := \perp$, $\Gamma^{d_1}(\text{id}) := \perp$, and send $(\texttt{CLOSED}, \text{id}) \xleftarrow{t_1} \mathcal{E}$.

Revoke by device

Device d_1 upon $\texttt{REVOKE} \xleftarrow{t_0} \mathcal{E}$:

For each $\text{id} \in \{0,1\}^*$ such that $\Theta^{d_1}(\text{id}) \neq \perp$:

(1) Iterate over all elements $(tx_{\text{set}_i}, sig^{d_1}_{\text{set}_i}, \sigma^{d_2,d_1}_{\text{Rv},i})$ in $\Theta^{d_1}(\text{id})$.

(2) If an old full-state commitment transaction $tx^{d_2}_{\text{Cm},i} \in tx_{\text{set}_i}$ is on $\mathbb{B}$, post $(tx^{d_2,d_1}_{\text{Rv},i}, \sigma^{d_2,d_1}_{\text{Rv},i})$ on $\mathbb{B}$ before the timelock ΔT.

(3) Let $tx^{d_2,d_1}_{\text{Rv},i}$ be accepted by $\mathbb{B}$ in round $t_1 \le t_0 + \Delta$, post $(tx^{d_2,d_1}_{\text{Fwd},i}, \sigma^{d_1,d_1}_{\text{Fwd},i} \in sig^{d_1}_{\text{set}_i})$ on $\mathbb{B}$.

(4) After $tx^{d_2,d_1}_{\text{Fwd},i}$ is accepted by $\mathbb{B}$ in round $t_2 \le t_1 + \Delta$, set $\Theta^{d_1}(\text{id}) := \perp$, $\Gamma^{d_1}(\text{id}) := \perp$, and send $(\texttt{REVOKED}, \text{id}) \xleftarrow{t_2} \mathcal{E}$.

For each $\text{id} \in \{0,1\}^*$ such that $\Gamma^{d_1}(\text{id}) \neq \perp$:

(1) Retrieve tx_{set_i} from $\Gamma^{d_1}(\text{id})$.

(2) If the latest full-state commitment transaction $tx^{d_2}_{\text{Cm},i} \in tx_{\text{set}_i}$ is on $\mathbb{B}$ go to next step. Else, go idle.

(3) Extract $tx^{d_2,d_2}_{\text{Lazy},i}, tx^{d_2,d_2}_{\text{Inst},i} \in tx_{\text{set}_i}$.

(4) Let $\tau_0 \le t_0 + \Delta$ be the round in which $tx^{d_2}_{\text{Cm},i}$ is accepted by $\mathbb{B}$.

(5) If $\Theta^{d_1}_{\text{Int},i}(\text{id}) \neq \perp$ go to next step. Else, go idle.

(6) Extract $\left(tx^{d_1,d_2}_{\text{IntSet},i,j}, \sigma^{d_1,d_2}_{\text{IntSet},i,j} \right)$ from $\Theta^{d_1}_{\text{Int},i}(\text{id})$.

(7) If $tx^{d_2,d_2}_{\text{Inst},i}$ appears on $\mathbb{B}$ at or after round $\tau_1 \le \tau_0 + t_{\text{Int},i,j} + \Delta$, post appropriate transaction and corresponding signature from $\left(tx^{d_1,d_2}_{\text{IntSet},i,j}, \sigma^{d_1,d_2}_{\text{IntSet},i,j} \right)$.

(8) After that transaction appears on $\mathbb{B}$ in round $\tau_2 \leq \tau_1 + \Delta$, set $\Theta^{d_1}(\mathrm{id}) := \perp$, $\Gamma^{d_1}(\mathrm{id}) := \perp$, and send $(\texttt{REVOKED-INT}, \mathrm{id}) \xhookrightarrow{\tau_2} \mathcal{E}$.

(9) If $tx^{d_2,d_2}_{\mathrm{Lazy},i}$ appears on $\mathbb{B}$ at or after round $\tau_3 \leq \tau_0 + \Delta T + t_{\mathrm{Int},i,j} + \Delta$, post appropriate transaction and corresponding signature from $\left(tx^{d_1,d_2}_{\mathrm{IntSet},i,j}, \sigma^{d_1,d_2}_{\mathrm{IntSet},i,j}\right)$.

(10) After that transaction appears on $\mathbb{B}$ in round $\tau_4 \leq \tau_3 + \Delta$, set $\Theta^{d_1}(\mathrm{id}) := \perp$, $\Gamma^{d_1}(\mathrm{id}) := \perp$, and send $(\texttt{REVOKED-INT}, \mathrm{id}) \xhookrightarrow{\tau_4} \mathcal{E}$.

Revoke by peers

Peer p upon $\texttt{REVOKE} \xhookleftarrow{t_0} \mathcal{E}$:

For each $\mathrm{id} \in \{0,1\}^*$ such that $\Theta^{p,d_1}(\mathrm{id}) \neq \perp$:

(1) Iterate over all elements $(tx^{d_1}_{\mathrm{Cm},i}, tx^{d_2,d_1}_{\mathrm{PreRv},i}, \sigma^{d_1,p}_{\mathrm{PreRv},i})$ in $\Theta^{p,d_1}(\mathrm{id})$ and $(tx^{d_2}_{\mathrm{Cm},i}, tx^{d_1,d_2}_{\mathrm{PreRv},i}, \sigma^{d_2,p}_{\mathrm{PreRv},i})$ in $\Theta^{p,d_2}(\mathrm{id})$.

(2) If an old full-state commitment transaction $tx^{d_1}_{\mathrm{Cm},i}$ is on $\mathbb{B}$, prepare the pre-signed revocation transaction $tx^{d_1,d_2}_{\mathrm{PreRv},i}$ by inserting the additional output with balance δ.

(3) Post $(tx^{d_1,d_2}_{\mathrm{PreRv},i}, \sigma^{d_2,p}_{\mathrm{PreRv},i})$ on $\mathbb{B}$ after Δt and before the timelock ΔT.

(4) After $tx^{d_1,d_2}_{\mathrm{PreRv},i}$ is accepted by $\mathbb{B}$ in round $t_1 \leq t_0 + \Delta$, set $\Theta^{p,d_1}(\mathrm{id}) := \perp$ and send $(\texttt{REVOKED}, \mathrm{id}) \xhookrightarrow{t_1} \mathcal{E}$.

$\texttt{ForceClose}(\mathrm{id})$:

Let t_0 be the current round

(1) Extract $(tx_{Fu}, tx_{set_i}, sig^{d_1}_{set_i}, addr_{set}, pkey^{d_1}_{set}, pkey^{d_2}_{set}, pkey^{\mathcal{P}}_{\mathrm{set}}) = \Gamma^{d_1}(\mathrm{id})$.

(2) Extract $tx^{d_1,d_1}_{\mathrm{Inst},i}, tx^{d_1}_{\mathrm{Cm},i} \in tx_{set_i}$ and $\sigma^{d_1,d_1}_{\mathrm{Inst},i}, \sigma^{d_2,d_1}_{\mathrm{Cm},i} \in sig^{d_1}_{\mathrm{set}_i}$.

(3) Device d_1 post $(tx^{d_1}_{\mathrm{Cm},i}, \sigma^{d_2,d_1}_{\mathrm{Cm},i})$ on $\mathbb{B}$.

(4) Let $t_1 \leq t_0 + \Delta$ be the round in which $tx^{d_1}_{\mathrm{Cm},i}$ is accepted by $\mathbb{B}$.

(5) Send $(\mathrm{peerSign}, \mathrm{id}, tx^{d_1,d_1}_{\mathrm{Inst},i}) \xhookrightarrow{t_1}$ each peer $p \in \mathcal{P}$.

(6) Wait one round.

(7) d_1 together with each peer p runs the interactive protocol $\mathcal{F}_{Sign}$ simultaneously to generate the signature $\sigma^{p,d_1}_{\mathrm{Inst},i}$ on the instant closure transaction $tx^{d_1,d_1}_{\mathrm{Inst},i}$ in t_{peer} rounds.

Each peer $p \in \mathcal{P}$ upon $(\mathrm{peerSign}, \mathrm{id}, tx^{d_1,d_1}_{\mathrm{Inst},i}) \xhookleftarrow{t_1+1} d_1$

(8) $(\texttt{PEER-CLOSED}, \mathrm{id}) \xhookrightarrow{t_1+1} \mathcal{E}$.

(9) Peer p together with d_1 runs the interactive protocol $\mathcal{F}_{Sign}$ to generate the signature $\sigma^{p,d_1}_{\mathrm{Inst},i}$ on the instant closure transaction $tx^{d_1,d_1}_{\mathrm{Inst},i}$. This takes t_{peer} rounds.

Device d_1 in round $t_1 + 1 + t_{\mathrm{peer}}$

(10) If valid signatures $\sigma^{p,d_1}_{\mathrm{Inst},i}$ from at least $\lfloor N/2 \rfloor + 1$ distinct peers $p \in \mathcal{P}$ are collected, go to next step. Else post $(tx^{d_1,d_1}_{\mathrm{Lazy},i}, \sigma^{d_1,d_1}_{\mathrm{Lazy},i})$ on $\mathbb{B}$ after ΔT and send $(\texttt{CLOSED}, \mathrm{id}) \xhookrightarrow{t_3 \leq t_1+1+t_{\mathrm{peer}}+\Delta T+\Delta} \mathcal{E}$ and set $\Theta^{d_1}(\mathrm{id}) := \perp$, $\Gamma^{d_1}(\mathrm{id}) := \perp$.

(11) Post $(tx^{d_1,d_1}_{\mathrm{Inst},i}, \{\sigma^{d_1,d_1}_{\mathrm{Inst},i}, \sigma^{p,d_1}_{\mathrm{Inst},i}$ for $\lfloor N/2 \rfloor + 1$ peers $p \in \mathcal{P}\})$ on $\mathbb{B}$ before ΔT and send $(\texttt{CLOSED}, \mathrm{id}) \xhookrightarrow{t_2 \leq t_1+1+t_{\mathrm{peer}}+\Delta} \mathcal{E}$.

(12) Set $\Theta^{d_1}(\mathrm{id}) := \perp$, $\Gamma^{d_1}(\mathrm{id}) := \perp$.

$\texttt{GenerateTxs}(addr_{\mathrm{set}}, pkey^{d_1}_{\mathrm{set}}, pkey^{d_2}_{\mathrm{set}}, \upsilon_{d_1,i}, \upsilon_{d_2,i})$:

(1) Using the addresses in $addr_{\mathrm{set}}$, the public keys in $pkey^{d_1}_{\mathrm{set}}$ and $pkey^{d_2}_{\mathrm{set}}$, do the following.

(2) Generate full-state commitment transactions $tx^{d_1}_{\mathrm{Cm},i} := tx(Ch_{d_1 d_2}, [\mathrm{toLocal}_{d_1}, \mathrm{toRemote}_{d_1}], [\upsilon_{d_1,i}, \upsilon_{d_2,i}])$ and $tx^{d_2}_{\mathrm{Cm},i} := tx(Ch_{d_1 d_2}, [\mathrm{toLocal}_{d_2}, \mathrm{toRemote}_{d_2}], [\upsilon_{d_2,i}, \upsilon_{d_1,i}])$.

(3) Generate lazy closure transactions $tx^{d_1,d_1}_{\mathrm{Lazy},i} := tx(\mathrm{toLocal}_{d_1}, pk_{\mathrm{Lazy},d_1}, \upsilon_{d_1,i})$ and $tx^{d_2,d_2}_{\mathrm{Lazy},i} := tx(\mathrm{toLocal}_{d_2}, pk_{\mathrm{Lazy},d_2}, \upsilon_{d_2,i})$. Both are timelocked until time ΔT after submission of their inputs.

(4) Generate instant closure transactions $tx^{d_1,d_1}_{\mathrm{Inst},i} := tx(\mathrm{toLocal}_{d_1}, pk_{\mathrm{Inst},d_1}, \upsilon_{d_1,i})$ and $tx^{d_2,d_2}_{\mathrm{Inst},i} := tx(\mathrm{toLocal}_{d_2}, pk_{\mathrm{Inst},d_2}, \upsilon_{d_2,i})$. Both require the signatures of $\lfloor N/2 \rfloor + 1$ peers for successful submission.

(5) Generate revocation transactions $tx^{d_1,d_2}_{\mathrm{Rv},i} := tx(\mathrm{toLocal}_{d_1}, pk_{\mathrm{Rv},d_2}, \upsilon_{d_1,i})$ and $tx^{d_2,d_1}_{\mathrm{Rv},i} := tx(\mathrm{toLocal}_{d_2}, pk_{\mathrm{Rv},d_1}, \upsilon_{d_2,i})$. Both require signatures.

(6) Generate pre-signed revocation transactions $tx^{d_1,d_2}_{\mathrm{PreRv},i} := tx(\mathrm{toLocal}_{d_1}, pk_{\mathrm{Rv},d_2}, \upsilon_{d_1,i} - \delta)$ and $tx^{d_2,d_1}_{\mathrm{PreRv},i} := tx(\mathrm{toLocal}_{d_2}, pk_{\mathrm{Rv},d_1}, \upsilon_{d_2,i} - \delta)$. Both require signatures.

(7) Generate fast withdrawal transactions $tx^{d_1,d_2}_{\text{Fwd},i} := tx(\text{toRemote}_{d_1}, pk_{\text{Fwd},d_2}, \upsilon_{d_2,i})$ and $tx^{d_2,d_1}_{\text{Fwd},i} := tx(\text{toRemote}_{d_2}, pk_{\text{Fwd},d_1}, \upsilon_{d_1,i})$. Both require their corresponding owner's signature.

(8) Return $tx_{\text{set}} := \{tx^{d_1}_{\text{Cm},i}, tx^{d_2}_{\text{Cm},i}, tx^{d_1,d_1}_{\text{Lazy},i}, tx^{d_2,d_2}_{\text{Lazy},i}, tx^{d_1,d_1}_{\text{Inst},i}, tx^{d_2,d_2}_{\text{Inst},i}, tx^{d_1,d_2}_{\text{Rv},i}, tx^{d_2,d_1}_{\text{Rv},i}, tx^{d_1,d_2}_{\text{PreRv},i}, tx^{d_2,d_1}_{\text{PreRv},i}, tx^{d_1,d_2}_{\text{Fwd},i}, tx^{d_2,d_1}_{\text{Fwd},i}\}$.

$\texttt{SignTxs}^{d_1}(tx_{\text{set}}, addr_{\text{set}}, pkey^{d_1}_{\text{set}} \cup pkey^{d_2}_{\text{set}})$:

Device d_1 (specified by the superscript of the function) is the one that receives the signatures first. The following is executed upon agreement, i.e., d_1 and d_2 start executing this sub-protocol in the same round with the same parameters. Extracting the transactions, addresses, and public keys from the parameters, device d_1 together with d_2 runs $\mathcal{F}_{\text{Sign}}$ to sign the transactions as follows.

(1) Device d_1 generates signature $\sigma^{d_1,d_1}_{\text{Lazy},i}$ on the transaction $tx^{d_1,d_1}_{\text{Lazy},i}$ and device d_2 generates signature $\sigma^{d_2,d_2}_{\text{Lazy},i}$ on the transaction $tx^{d_2,d_2}_{\text{Lazy},i}$.

(2) Device d_1 generates signature $\sigma^{d_1,d_1}_{\text{Inst},i}$ on the transaction $tx^{d_1,d_1}_{\text{Inst},i}$ and device d_2 generates signature $\sigma^{d_2,d_2}_{\text{Inst},i}$ on the transaction $tx^{d_2,d_2}_{\text{Inst},i}$.

(3) Device d_1 generates signature $\sigma^{d_1,d_1}_{\text{Fwd},i}$ on the transaction $tx^{d_2,d_1}_{\text{Fwd},i}$ and device d_2 generates signature $\sigma^{d_2,d_2}_{\text{Fwd},i}$ on the transaction $tx^{d_1,d_2}_{\text{Fwd},i}$.

(4) Device d_1 receives signature $\sigma^{d_2,d_1}_{\text{Cm},i}$ on the transaction $tx^{d_1}_{\text{Cm},i}$ and device d_2 receives signature $\sigma^{d_1,d_2}_{\text{Cm},i}$ on the transaction $tx^{d_2}_{\text{Cm},i}$.

This takes t_{sign} rounds and in case of failure (i.e., a signature is not received or not valid for the specified transaction and output), execute the steps in close. In case of success:

(5) Returns to d_1 $sig^{d_1}_{\text{set}_i} := \left\{\sigma^{d_1,d_1}_{\text{Lazy},i}, \sigma^{d_1,d_1}_{\text{Inst},i}, \sigma^{d_1,d_1}_{\text{Fwd},i}, \sigma^{d_2,d_1}_{\text{Cm},i}\right\}$ and to d_2 $sig^{d_2}_{\text{set}_i} := \left\{\sigma^{d_2,d_2}_{\text{Lazy},i}, \sigma^{d_2,d_2}_{\text{Inst},i}, \sigma^{d_2,d_2}_{\text{Fwd},i}, \sigma^{d_1,d_2}_{\text{Cm},i}\right\}$.

Simulators

Simulator for create

Let $T_1 = \tau_0 + 2 + t_{\text{addr}} + t_{\text{sign}}$ be the total time of create phase.

Case d_1 is honest and d_2 is corrupted

Upon device d_1 sending $(\texttt{CREATE}, \gamma, tid_{d_1}) \overset{\tau_0}{\hookrightarrow} \mathcal{F}$, if d_2 does not send $(\texttt{CREATE}, \gamma, tid_{d_2}) \overset{\tau}{\hookrightarrow} \mathcal{F}$ where $|\tau - \tau_0| \leq T_1$, then distinguish the following cases:

(1) If d_2 sends $(\text{createInfo}, \text{id}, tid_{d_2}, pkey^{d_2}_{\text{set}}, pkey^{g_{d_2}}_{\text{set}}) \overset{\tau_0}{\hookrightarrow} d_1$, then send $(\texttt{CREATE}, \gamma, tid_{d_2}) \overset{\tau}{\hookrightarrow} \mathcal{F}$ on behalf of d_2.
(2) Otherwise, stop.

Do the following:

(1) Set $\text{id} := \gamma.\text{id}$, generate (pk_{d_1}, sk_{d_1}), $(pk_{\text{Lazy},d_1}, sk_{\text{Lazy},d_1})$, $(pk_{\text{Inst},d_1}, sk_{\text{Inst},d_1})$, $(pk_{\text{Fwd},d_1}, sk_{\text{Fwd},d_1})$ and $(pk_{\text{Rv},d_1}, sk_{\text{Rv},d_1})$. Let $pkey^{d_1}_{\text{set}}$ denote the set of corresponding public keys. Send $(\text{createInfo}, \text{id}, tid_{d_1}, pkey^{d_1}_{\text{set}}, pkey^{g_{d_1}}_{\text{set}}) \overset{\tau_0}{\hookrightarrow} d_2$.
(2) If you receive $(\text{createInfo}, \text{id}, tid_{d_2}, pkey^{d_2}_{\text{set}}, pkey^{g_{d_2}}_{\text{set}}) \overset{\tau_0+1}{\hookleftarrow} d_2$, do the following. Else, go idle.
(3) Using $pkey^{d_1}_{set}$, $pkey^{d_2}_{set}$, and $pkey^{\mathcal{P}}_{set}$, the simulator on behalf of d_1 together with d_2 runs $\mathcal{F}_{\text{JKGen}}$ to generate the following set of addresses: $addr_{set} := \{Ch_{d_1 d_2}, \text{toLocal}_{d_1}, \text{toLocal}_{d_2}, \text{toRemote}_{d_1}, \text{toRemote}_{d_2}\}$ which takes t_{addr} rounds. In case of failure, abort.
(4) Extract $\upsilon_{d_1,0}$ and $\upsilon_{d_2,0}$ from $\gamma.\text{st}$.
(5) Generate $tx_{\text{Fu}} := tx([tid_{d_1}, tid_{d_2}], Ch_{d_1 d_2}, [\upsilon_{d_1,0} + \upsilon_{d_2,0}])$.
(6) Let $tx_{\text{set}_0} \leftarrow \texttt{GenerateTxs}(addr_{\text{set}}, pkey^{d_1}_{\text{set}}, pkey^{d_2}_{\text{set}}, \upsilon_{d_1,0}, \upsilon_{d_2,0})$.
(7) Let $sig^{d_1}_{\text{set}_0} \leftarrow \texttt{SignTxs}^{d_1}(tx_{set_0}, addr_{\text{set}}, pkey^{d_1}_{\text{set}} \cup pkey^{d_2}_{\text{set}}, pkey^{\mathcal{P}}_{\text{set}})$.
(8) Generates the signature $\sigma_{tid_{d_1}}$ on behalf of d_1 for the output tid_{d_1} and sends $(\text{createFund}, \text{id}, \sigma_{tid_{d_1}}) \overset{\tau_0+1+t_{\text{addr}}+t_{\text{sign}}}{\longleftarrow} d_2$.
(9) If $(\text{createFund}, \text{id}, \sigma_{tid_{d_2}}) \overset{\tau_0+2+t_{\text{addr}}+t_{\text{sign}}}{\longleftarrow} d_2$, post $(tx_{\text{Fu}}, \{\sigma_{tid_{d_1}}, \sigma_{tid_{d_2}}\})$ to $\mathbb{B}$.
(10) If tx_{Fu} is accepted by $\mathbb{B}$ in round $\tau_1 \leq \tau_0 + 2 + t_{\text{addr}} + t_{\text{sign}} + \Delta$, store $\Gamma^{d_1}(\text{id}) := (tx_{Fu}, tx_{set_0}, sig^{d_1}_{set_0}, pkey^{d_1}_{\text{set}}, pkey^{d_2}_{\text{set}}, pkey^{\mathcal{P}}_{\text{set}})$.

Simulator for update full-state commitment

Case d_1 and $N_h \geq \lfloor N/2 \rfloor + 1$ of peers are honest and d_2 together with $N - N_h$ of peers are corrupted

Upon device d_1 sending $(\texttt{UPDATE}, \text{id}, \vec{\theta}, t_{\text{stp}_1}, t_{\text{stp}_2}) \xhookrightarrow{\tau_0} \mathcal{F}$, proceed as follows:

(1) $(\text{updateReq}, \text{id}, \vec{\theta}, t_{\text{stp}_1}) \xhookrightarrow{t_0} d_2$.
(2) Upon $(\text{updateInfo}, \text{id}) \xhookleftarrow{t_0+2} d_2$, do the following.
(3) Retrieve $(tx_{\text{Fu}}, tx_{set_{i-1}}, sig^{d_1}_{set_{i-1}}, addr_{set}, pkey^{d_1}_{\text{set}}, pkey^{d_2}_{\text{set}}, pkey^{\mathcal{P}}_{\text{set}}) = \Gamma^{d_1}(\text{id})$.
(4) Extract $\upsilon_{d_1,i}$ and $\upsilon_{d_2,i}$ from $\vec{\theta}$.
(5) Let $tx_{set_i} \leftarrow \texttt{GenerateTxs}(addr_{set}, pkey^{d_1}_{set}, pkey^{d_2}_{set}, \upsilon_{d_1,i}, \upsilon_{d_2,i})$.
(6) Let $\vec{tid} := (tx^{d_1}_{Cm,i}.\text{id}, tx^{d_2}_{Cm,i}.\text{id})$ be a tuple of the transaction ids of transaction $tx^{d_1}_{Cm,i}$ and $tx^{d_2}_{Cm,i}$. Inform $\mathcal{F}$ of $\vec{tid}$ in round $t_0 + 2$.
(7) If d_1 sends $(\texttt{SETUP-OK}, \text{id}) \xhookrightarrow{t_1 \le t_0+2+t_{\text{stp}_1}} \mathcal{F}$, send $(\text{updateInfo}, \text{id}, tx^{d_1}_{\text{Cm},i}, tx^{d_2}_{\text{Cm},i}, t_{\text{stp}_2}) \xhookrightarrow{t_1}$ each peer $p \in \mathcal{P}$.
(8) If each **corrupted** peer p starts $\mathcal{F}_{\text{Sign}}$ in round $t_1 + 1 + t_{\text{stp}_2}$, send $(\texttt{PEER-SETUP-OK}, \text{id}) \xhookrightarrow{t_2 \le t_1+1+t_{\text{stp}_2}} \mathcal{F}$ on behalf of p.
(9) The simulator on behalf of d_1 and **honest** peers who have sent $(\texttt{PEER-SETUP-OK}, \text{id}) \xhookrightarrow{t_2 \le t_1+1+t_{\text{stp}_2}} \mathcal{F}$ and **corrupted** peers who have started $\mathcal{F}_{\text{Sign}}$ in round $t_1 + 1 + t_{\text{stp}_2}$, runs the interactive protocol $\mathcal{F}_{Sign}$ to generate the signature $\sigma^{p,d_1}_{\text{Cm},i}$ on the full-state commitment transaction $tx^{d_1}_{\text{Cm},i}$ in t_{peer} rounds.
(10) If d_1 receives valid signatures $\sigma^{p,d_1}_{\text{Cm},i}$ from at least $\lfloor N/2 \rfloor + 1$ distinct peers $p \in \mathcal{P}$, send $(\text{updateCom}, \text{id}, \sigma^{p,d_1}_{\text{Cm},i}$ for each $p \in \mathcal{P}) \xhookrightarrow{t_2+t_{\text{peer}}}$ d_2. Otherwise, instruct $\mathcal{F}$ to $(\texttt{UPDATE-PEER-FAIL}, \text{id}) \xhookrightarrow{t_2+t_{\text{peer}}} \mathcal{E}$ via d_1 and execute $\texttt{ForceClose}^{d_1}(\text{id})$.
(11) If in round $t_2 + 1 + t_{\text{peer}}$ device d_2 starts executing $\texttt{SignTxs}^{d_1}(tx_{set_0}, addr_{\text{set}}, pkey^{d_1}_{\text{set}} \cup pkey^{d_2}_{\text{set}}, pkey^{\mathcal{P}}_{\text{set}})$, send $(\texttt{UPDATE-OK}, \text{id}) \xhookrightarrow{t_2+1+t_{\text{peer}}} \mathcal{F}$ on behalf of d_2.
(12) $\texttt{SignTxs}^{d_1}(tx_{set_i}, addr_{set}, pkey^{d_1}_{\text{set}} \cup pkey^{d_2}_{\text{set}})$.
(13) If $sig^{d_1}_{set_i}$ is returned from $\texttt{SignTxs}^{d_1}$, instruct $\mathcal{F}$ to $(\texttt{UPDATE-OK}, \text{id}) \xhookrightarrow{t_2+1+t_{\text{peer}}+t_{\text{sign}}} \mathcal{E}$ via d_1. Else, execute $\texttt{ForceClose}^{d_1}(\text{id})$ and go idle.
(14) If d_1 does not send $(\texttt{REVOKE}, \text{id}) \xhookrightarrow{t_2+1+t_{\text{peer}}+t_{\text{sign}}} \mathcal{F}$, go idle.
(15) The simulator on behalf of d_1 together with d_2 runs the interactive protocol $\mathcal{F}_{\text{Sign}}$ to generate the signature $\sigma^{d_1,d_2}_{\text{Rv},i-1}$ on the revocation transaction $tx^{d_1,d_2}_{\text{Rv},i-1}$ and $\sigma^{d_1,p}_{\text{PreRv},i-1}$ on the pre-signed revocation transaction $tx^{d_2,d_1}_{\text{PreRv},i-1}$ in t_{rev} rounds. In case of failure, execute $\texttt{ForceClose}^{d_1}(\text{id})$.
(16) $(\text{revoke}, \text{id}, \sigma^{d_1,d_2}_{\text{Rv},i-1}) \xhookrightarrow{t_2+1+t_{\text{peer}}+t_{\text{sign}}+t_{\text{rev}}} d_2$.
(17) $(\text{peerRevoke}, \text{id}, tx^{d_2,d_1}_{\text{PreRv},i-1}, \sigma^{d_2,p}_{\text{PreRv},i-1}) \xhookrightarrow{t_2+1+t_{\text{peer}}+t_{\text{sign}}+t_{\text{rev}}}$ each peer $p \in \mathcal{P}$.
(18) For each **honest** peer p, send $(\text{peerRevoked}, \text{id}) \xhookrightarrow{t_2+2+t_{\text{peer}}+t_{\text{sign}}+t_{\text{rev}}} d_1$.
(19) If d_2 starts $\mathcal{F}_{\text{Sign}}$ to sign transactions $tx^{d_2,d_1}_{\text{Rv},i-1}$ and $tx^{d_1,d_2}_{\text{PreRv},i-1}$ in round $t_2 + 2 + t_{\text{peer}} + t_{\text{sign}} + t_{\text{rev}}$, send $(\texttt{REVOKE}, \text{id}) \xhookrightarrow{t_2+2+t_{\text{peer}}+t_{\text{sign}}+t_{\text{rev}}} \mathcal{F}$ on behalf of d_2, and participate in the signing on behalf of d_1.
(20) If $(\text{peerRevoked}, \text{id}) \xhookleftarrow{t_2+3+t_{\text{peer}}+t_{\text{sign}}+t_{\text{rev}}} p$ from at least $\lfloor N/2 \rfloor + 1$ distinct peers $p \in \mathcal{P}$, continue. Otherwise, execute $\texttt{ForceClose}^{d_1}(\text{id})$.
(21) For each **honest** peer p, send $(\text{peerRevoked}, \text{id}) \xhookrightarrow{t_2+3+t_{\text{peer}}+t_{\text{sign}}+2t_{\text{rev}}} d_2$.
(22) For each **honest** peer p, set $\Theta^{p,d_1}(\text{id}) := \Theta^{p,d_1} \cup \left\{(tx^{d_1}_{\text{Cm},i-1}, tx^{d_1,d_2}_{\text{PreRv},i-1}, \sigma^{d_2,p}_{\text{Rv},i-1})\right\}$ and $\Theta^{p,d_2}(\text{id}) := \Theta^{p,d_2} \cup \left\{(tx^{d_2}_{\text{Cm},i-1}, tx^{d_2,d_1}_{\text{PreRv},i-1}, \sigma^{d_2,p}_{\text{Rv},i-1})\right\}$.
(23) If $(\text{revoke}, \text{id}, \sigma^{d_2,d_1}_{\text{Rv},i-1}) \xhookleftarrow{t_2+3+t_{\text{peer}}+t_{\text{sign}}+2t_{\text{rev}}} d_2$ and the signature is valid, go to next step. Else, execute $\texttt{ForceClose}^{d_1}(\text{id})$.
(24) $\Theta^{d_1}(\text{id}) := \Theta^{d_1} \cup \left\{\left(tx_{\text{set}_{i-1}}, sig^{d_1}_{\text{set}_{i-1}}, \sigma^{d_2,d_1}_{\text{Rv},i-1}\right)\right\}$.
(25) $\Gamma^{d_1}(\text{id}) := (tx_{\text{Fu}}, tx_{set_i}, sig^{d_1}_{set_i}, addr_{set}, pkey^{d_1}_{\text{set}}, pkey^{d_2}_{\text{set}}, pkey^{\mathcal{P}}_{\text{set}})$.

Case d_2 and $N_h \ge \lfloor N/2 \rfloor + 1$ of peers are honest and d_1 together with $N - N_h$ of peers are corrupted

Upon device d_1 sending $(\text{updateReq}, \text{id}, \vec{\theta}, t_{\text{stp}_1}) \xhookrightarrow{t_0} d_2$, send $(\texttt{UPDATE}, \text{id}, \vec{\theta}, t_{\text{stp}_1}, t_{\text{stp}_2}) \xhookrightarrow{t_0} \mathcal{F}$ on behalf of d_1, if d_1 has not already sent this message. Proceed as follows:

(1) Upon $(\text{updateReq}, \text{id}, \vec{\theta}, t_{\text{stp}_1}) \xhookleftarrow{\tau_0} d_1$ do the following:
(2) Retrieve $(tx_{\text{Fu}}, tx_{set_{i-1}}, sig^{d_2}_{set_{i-1}}, addr_{set}, pkey^{d_1}_{\text{set}}, pkey^{d_2}_{\text{set}}, pkey^{\mathcal{P}}_{\text{set}}) = \Gamma^{d_2}(\text{id})$.
(3) Extract $\upsilon_{d_1,i}$ and $\upsilon_{d_2,i}$ from $\vec{\theta}$.
(4) Let $tx_{set_i} \leftarrow \texttt{GenerateTxs}(addr_{set}, pkey^{d_1}_{set}, pkey^{d_2}_{set}, \upsilon_{d_1,i}, \upsilon_{d_2,i})$.
(5) Let $\vec{tid} := (tx^{d_1}_{Cm,i}.\text{id}, tx^{d_2}_{Cm,i}.\text{id})$ be a tuple of the transaction ids of transaction $tx^{d_1}_{Cm,i}$ and $tx^{d_2}_{Cm,i}$. Inform $\mathcal{F}$ of $\vec{tid}$.
(6) $(\text{updateInfo}, \text{id}) \xhookrightarrow{\tau_0} d_1$.
(7) Upon d_1 sending $(\text{updateInfo}, \text{id}, tx^{d_1}_{\text{Cm},i}, tx^{d_2}_{\text{Cm},i}, t_{\text{stp}_2}) \xhookrightarrow{\tau_0+1+t_{\text{stp}_1}}$ each **honest** peer $p \in \mathcal{P}$, send $(\texttt{SETUP-OK}, \text{id}) \xhookrightarrow{\tau_0+1+t_{\text{stp}_1}} \mathcal{F}$ on behalf of d_1.

(8) The simulator on behalf of d_1 and **honest** peers who have sent $(\texttt{PEER-SETUP-OK}, \text{id}) \xhookrightarrow{\tau_0+2+t_{\text{stp}_1}+t_{\text{stp}_2}} \mathcal{F}$, runs the interactive protocol $\mathcal{F}_{Sign}$ to generate the signature $\sigma^{p,d_1}_{\text{Cm},i}$ on the full-state commitment transaction $tx^{d_1}_{\text{Cm},i}$ which takes t_{peer} rounds.

(9) If d_2 sends $(\texttt{UPDATE-OK}, \text{id}) \xhookrightarrow{\tau_1} \mathcal{F}$, execute $\texttt{SignTxs}^{d_1}(tx_{set_i}, addr_{set}, pkey^{d_1}_{\text{set}} \cup pkey^{d_2}_{\text{set}})$.

(10) If $sig^{d_2}_{set_i}$ is not returned from $\texttt{SignTxs}^{d_1}$ in round $\tau_1 + t_{\text{sign}}$, execute $\texttt{ForceClose}^{d_2}(\text{id})$ and go idle.

(11) If d_1 starts $\mathcal{F}_{\text{Sign}}$ to sign transactions $tx^{d_1,d_2}_{\text{Rv},i-1}$ and $tx^{d_1,p}_{\text{PreRv},i-1}$ in round $\tau_1 + t_{\text{sign}}$, send $(\texttt{REVOKE}, \text{id}) \xhookrightarrow{\tau_1+t_{\text{sign}}} \mathcal{F}$ on behalf of d_1, and participate in the signing on behalf of d_2.

(12) Upon $(\text{revoke}, \text{id}, \sigma^{d_1,d_2}_{\text{Rv},i-1}) \xhookleftarrow{\tau_1+1+t_{\text{sign}}+t_{\text{rev}}} d_1$, continue. Else, execute $\texttt{ForceClose}^{d_2}(\text{id})$ and go idle.

(13) For each **honest** peer p, if $(\text{peerRevoke}, \text{id}, tx^{d_2,d_1}_{\text{PreRv},i-1}, \sigma^{d_2,p}_{\text{PreRv},i-1}) \xhookleftarrow{\tau_1+1+t_{\text{sign}}+t_{\text{rev}}} d_1$, send $(\text{peerRevoked}, \text{id}) \xhookrightarrow{\tau_1+1+t_{\text{sign}}+t_{\text{rev}}} d_1$.

(14) If d_2 does not send $(\texttt{REVOKE}, \text{id}) \xhookrightarrow{\tau_1+1+t_{\text{sign}}+t_{\text{rev}}} \mathcal{F}$, go idle.

(15) The simulator on behalf of d_2 together with d_1 runs the interactive protocol $\mathcal{F}_{\text{Sign}}$ to generate the signature $\sigma^{d_2,d_1}_{\text{Rv},i-1}$ on the revocation transaction $tx^{d_2,d_1}_{\text{Rv},i-1}$ and $\sigma^{d_2,p}_{\text{PreRv},i-1}$ on the pre-signed revocation transaction $tx^{d_1,d_2}_{\text{PreRv},i-1}$ in t_{rev} rounds. In case of failure, execute $\texttt{ForceClose}^{d_2}(\text{id})$.

(16) $(\text{revoke}, \text{id}, \sigma^{d_2,d_1}_{\text{Rv},i-1}) \xhookrightarrow{\tau_1+1+t_{\text{sign}}+2t_{\text{rev}}} d_1$.

(17) $(\text{peerRevoke}, \text{id}, tx^{d_1,d_2}_{\text{PreRv},i-1}, \sigma^{d_1,p}_{\text{PreRv},i-1}) \xhookrightarrow{\tau_1+1+t_{\text{sign}}+2t_{\text{rev}}}$ each peer $p \in \mathcal{P}$.

(18) For each **honest** peer p, send $(\text{peerRevoked}, \text{id}) \xhookrightarrow{\tau_1+2+t_{\text{sign}}+2t_{\text{rev}}} d_2$.

(19) For each **honest** peer p, set $\Theta^{p,d_1}(\text{id}) := \Theta^{p,d_1} \cup \left\{(tx^{d_1}_{\text{Cm},i-1}, tx^{d_1,d_2}_{\text{PreRv},i-1}, \sigma^{d_2,p}_{\text{Rv},i-1})\right\}$ and $\Theta^{p,d_2}(\text{id}) := \Theta^{p,d_2} \cup \left\{(tx^{d_2}_{\text{Cm},i-1}, tx^{d_2,d_1}_{\text{PreRv},i-1}, \sigma^{d_2,p}_{\text{Rv},i-1})\right\}$.

(20) If $(\text{peerRevoked}, \text{id}) \xhookleftarrow{\tau_1+3+t_{\text{sign}}+2t_{\text{rev}}} p$ from at least $\lfloor N/2 \rfloor + 1$ distinct peers $p \in \mathcal{P}$, continue. Otherwise, execute $\texttt{ForceClose}^{d_2}(\text{id})$.

(21) $\Theta^{d_2}(\text{id}) := \Theta^{d_2} \cup \left\{\left(tx_{\text{set}_{i-1}}, sig^{d_2}_{\text{set}_{i-1}}, \sigma^{d_1,d_2}_{\text{Rv},i-1}\right)\right\}$.

(22) $\Gamma^{d_2}(\text{id}) := (tx_{\text{Fu}}, tx_{set_i}, sig^{d_2}_{set_i}, addr_{set}, pkey^{d_1}_{\text{set}}, pkey^{d_2}_{\text{set}}, pkey^{\mathcal{P}}_{\text{set}})$.

Simulator for update intermediate

Case d_1 is honest and d_2 is corrupted

Upon device d_1 sending $(\texttt{UPDATE-INT}, \text{id}, \vec{\theta'}, t_{\text{stp}}) \xhookrightarrow{t_0} \mathcal{F}$, proceed as follows:

(1) $(\text{updateReqInt}, \text{id}, \vec{\theta'}, t_{\text{stp}}) \xhookrightarrow{t_0} d_2$.

(2) Extract $(tx_{Fu}, tx_{set_i}, sig^{d_1}_{set_i}, addr_{set}, pkey^{d_1}_{set}, pkey^{d_2}_{set}, pkey^{\mathcal{P}}_{\text{set}}) = \Gamma^{d_1}(\text{id})$.

(3) Extract $v^j_{d_1,i}$ and $v^j_{d_2,i}$ from $\vec{\theta'}$.

(4) Create transactions $tx^{d_1,d_2}_{\text{IntSet},i,j} := \{tx(pk_{\text{Lazy},d_1}, [pk_{d_1}, pk_{d_2}], [v^j_{d_1,i}, v^j_{d_2,i}]), tx(pk_{\text{Inst},d_1}, [pk_{d_1}, pk_{d_2}], [v^j_{d_1,i}, v^j_{d_2,i}]), tx(pk_{\text{Fwd},d_1}, [pk_{d_1}, pk_{d_2}], [v^j_{d_1,i}, v^j_{d_2,i}])\}$ which all of them can be spent after timelock $t^{d_1,d_2}_{\text{Int},i,j}$.

(5) Let $\vec{tid}$ be the tuple of the transaction ids of transactions in $tx^{d_1,d_2}_{\text{IntSet},i,j}$. Inform $\mathcal{F}$ of $\vec{tid}$ in round $t_0 + 2$.

(6) If in round $t_1 \leq t_0 + 2 + t_{\text{stp}}$, device d_2 started $\mathcal{F}_{\text{Sign}}$, send $(\texttt{UPDATE-OK-INT}, \text{id}) \xhookrightarrow{t_1} \mathcal{F}$ on behalf of d_2.

(7) If d_1 sends $(\texttt{SETUP-OK-INT}, \text{id}) \xhookrightarrow{t_1} \mathcal{F}$, execute $\mathcal{F}_{\text{Sign}}$ on behalf of d_1 and d_2 to generate signature $\sigma^{d_1,d_2}_{\text{IntSet},i,j}$ on $tx^{d_1,d_2}_{\text{IntSet},i,j}$ in t_{sign} rounds.

(8) $\Theta^{d_1}_{\text{Int},i}(\text{id}) := \Theta^{d_1}_{\text{Int},i} \cup \left\{\left(tx^{d_1,d_2}_{\text{IntSet},i,j}, \sigma^{d_1,d_2}_{\text{IntSet},i,j}\right)\right\}$.

Case d_2 is honest and d_1 is corrupted

Upon device d_1 sending $(\text{updateReqInt}, \text{id}, \vec{\theta'}, t_{\text{stp}}) \xhookrightarrow{t_0} d_2$, send $(\texttt{UPDATE-INT}, \text{id}, \vec{\theta'}, t_{\text{stp}}) \xhookrightarrow{t_0} \mathcal{F}$ on behalf of d_1, if d_1 has not already sent this message. Proceed as follows:

(1) Extract $(tx_{Fu}, tx_{set_i}, sig^{d_1}_{set_i}, addr_{set}, pkey^{d_1}_{set}, pkey^{d_2}_{set}, pkey^{\mathcal{P}}_{\text{set}}) = \Gamma^{d_2}(\text{id})$.

(2) Extract $v^j_{d_1,i}$ and $v^j_{d_2,i}$ from $\vec{\theta'}$.

(3) Create transactions $tx^{d_1,d_2}_{\text{IntSet},i,j} := \{tx(pk_{\text{Lazy},d_1}, [pk_{d_1}, pk_{d_2}], [v^j_{d_1,i}, v^j_{d_2,i}]), tx(pk_{\text{Inst},d_1}, [pk_{d_1}, pk_{d_2}], [v^j_{d_1,i}, v^j_{d_2,i}]), tx(pk_{\text{Fwd},d_1}, [pk_{d_1}, pk_{d_2}], [v^j_{d_1,i}, v^j_{d_2,i}])\}$ which all of them can be spent after timelock $t^{d_1,d_2}_{\text{Int},i,j}$.

(4) Let $\vec{tid}$ be the tuple of the transaction ids of transactions in $tx^{d_1,d_2}_{\text{IntSet},i,j}$. Inform $\mathcal{F}$ of $\vec{tid}$ in round τ_0.

(5) $(\text{updateInfoInt}, \text{id}, \vec{tid}, t^{d_1,d_2}_{\text{Int},i,j}) \xhookrightarrow{\tau_0} d_1$.

(6) If d_1 starts executing $\mathcal{F}_{\text{Sign}}$ in round $\tau_1 \leq \tau_0 + 1 + t_{\text{stp}}$, send $(\texttt{SETUP-OK-INT}, \text{id}) \xhookrightarrow{\tau_1} \mathcal{F}$ on behalf of d_1.

(7) If d_2 sends $(\texttt{UPDATE-OK-INT}, \text{id}) \xhookrightarrow{\tau_1} \mathcal{F}$, execute $\mathcal{F}_{\text{Sign}}$ on behalf of d_1 and d_2 to generate signature $\sigma^{d_1,d_2}_{\text{IntSet},i,j}$ on $tx^{d_1,d_2}_{\text{IntSet},i,j}$ in t_{sign} rounds.

(8) $\Theta^{d_2}_{\mathrm{Int},i}(\mathrm{id}) \coloneqq \Theta^{d_2}_{\mathrm{Int},i} \cup \left\{ \left(tx^{d_1,d_2}_{\mathrm{IntSet},i,j}, \sigma^{d_1,d_2}_{\mathrm{IntSet},i,j} \right) \right\}$.

Simulator for close

Case d_1 and $N_h \geq \lfloor N/2 \rfloor + 1$ of peers are honest and d_2 together with $N - N_h$ of peers are corrupted

Upon device d_1 sending $(\mathtt{CLOSE}, \mathrm{id}) \overset{t_0}{\hookrightarrow} \mathcal{F}$ do the following.

(1) Extract $(tx_{Fu}, tx_{set_i}, sig^{d_1}_{set_i}, addr_{set}, pkey^{d_1}_{set}, pkey^{d_2}_{set}, pkey^{\mathcal{P}}_{\mathrm{set}}) = \Gamma^{d_1}(\mathrm{id})$.
(2) Extract $\upsilon_{d_1,i}$ and $\upsilon_{d_2,i}$ from $tx^{d_1}_{\mathrm{Cm},i} \in tx_{set_i}$.
(3) Create transaction $tx_{\mathrm{St}} \coloneqq tx(Ch_{d_1 d_2}, [pk_{d_1}, pk_{d_2}], [\upsilon_{d_1,i}, \upsilon_{d_2,i}])$.
(4) Send $(\mathrm{peerSign}, \mathrm{id}, tx_{\mathrm{St}}) \overset{t_0}{\hookrightarrow}$ each peer $p \in \mathcal{P}$.
(5) Wait one round.
(6) The simulator on behalf of d_1 together with each peer (honest and corrupted) runs the interactive protocol $\mathcal{F}_{Sign}$ simultaneously to generate σ^p_{St} on tx_{St}. This takes t_{peer} rounds.
(7) If valid signatures σ^p_{St} from at least $\lfloor N/2 \rfloor + 1$ distinct peers are collected in round $t_0 + 1 + t_{\mathrm{peer}}$, continue. Otherwise, execute $\mathtt{ForceClose}^{d_1}(\mathrm{id})$.
(8) The simulator on behalf of d_1 together with d_2 runs the interactive protocol $\mathcal{F}_{Sign}$ to generate the signature $\sigma^{d_1 d_2}_{\mathrm{St}}$ on the transaction tx_{St}. This takes t_{sign} rounds.
(9) In case the signature generation was successful, post $(tx_{\mathrm{St}}, \{\sigma^{d_1 d_2}_{\mathrm{St}}, \sigma^p_{\mathrm{St}}$ for $\lfloor N/2 \rfloor + 1$ peers $p \in \mathcal{P}\})$ on $\mathbb{B}$ in round $t_0 + 1 + t_{\mathrm{peer}} + t_{\mathrm{sign}}$. Else, execute $\mathtt{ForceClose}^{d_1}(\mathrm{id})$.
(10) If tx_{St} appears on $\mathbb{B}$ in round $t_1 \leq t_0 + 1 + t_{\mathrm{peer}} + t_{\mathrm{sign}} + \Delta$, set $\Theta^{d_1}(\mathrm{id}) \coloneqq \perp$, $\Gamma^{d_1}(\mathrm{id}) \coloneqq \perp$.

Simulator for revoke by device

Case d_1 is honest and d_2 is corrupted

Upon d_1 sending $\mathtt{REVOKE} \overset{\tau_0}{\hookrightarrow} \mathcal{F}$, for each $\mathrm{id} \in \{0,1\}^*$ such that $\Theta^{d_1}(\mathrm{id}) \neq \perp$:

(1) Parse $\{(tx_{\mathrm{set}_i}, sig^{d_1}_{\mathrm{set}_i}, \sigma^{d_2,d_1}_{\mathrm{Rv},i})\}_{i \in m} \coloneqq \Theta^{d_1}(\mathrm{id})$ and extract γ from $\Gamma^{d_1}(\mathrm{id})$. If for some $i \in m$, there exist a transaction $tx^{d_2}_{\mathrm{Cm},i} \in tx_{\mathrm{set}_i}$ on $\mathbb{B}$ do the following.
(2) post $(tx^{d_2,d_1}_{\mathrm{Rv},i}, \sigma^{d_2,d_1}_{\mathrm{Rv},i})$ on $\mathbb{B}$ before the timelock ΔT.
(3) Let $tx^{d_2,d_1}_{\mathrm{Rv},i}$ be accepted by $\mathbb{B}$ in round $t_1 \leq t_0 + \Delta$. Post $(tx^{d_2,d_1}_{\mathrm{Fwd},i}, \sigma^{d_1,d_1}_{\mathrm{Fwd},i} \in sig^{d_1}_{\mathrm{set}_i})$ on $\mathbb{B}$.
(4) After $tx^{d_2,d_1}_{\mathrm{Fwd},i}$ is accepted by $\mathbb{B}$ in round $t_2 \leq t_1 + \Delta$, set $\Theta^{d_1}(\mathrm{id}) \coloneqq \perp$, $\Gamma^{d_1}(\mathrm{id}) \coloneqq \perp$.

For each $\mathrm{id} \in \{0,1\}^*$ such that $\Gamma^{d_1}(\mathrm{id}) \neq \perp$:

(1) Retrieve tx_{set_i} from $\Gamma^{d_1}(\mathrm{id})$.
(2) If the latest full-state commitment transaction $tx^{d_2}_{\mathrm{Cm},i} \in tx_{\mathrm{set}_i}$ is on $\mathbb{B}$ go to next step. Else, go idle.
(3) Extract $tx^{d_2,d_2}_{\mathrm{Lazy},i}, tx^{d_2,d_2}_{\mathrm{Inst},i} \in tx_{\mathrm{set}_i}$.
(4) Let $\tau_0 \leq t_0 + \Delta$ be the round in which $tx^{d_2}_{\mathrm{Cm},i}$ is accepted by $\mathbb{B}$.
(5) If $\Theta^{d_1}_{\mathrm{Int},i}(\mathrm{id}) \neq \perp$ go to next step. Else, go idle.
(6) Extract $\left(tx^{d_1,d_2}_{\mathrm{IntSet},i,j}, \sigma^{d_1,d_2}_{\mathrm{IntSet},i,j} \right)$ from $\Theta^{d_1}_{\mathrm{Int},i}(\mathrm{id})$.
(7) If $tx^{d_2,d_2}_{\mathrm{Inst},i}$ appears on $\mathbb{B}$ at or after round $\tau_1 \leq \tau_0 + t_{\mathrm{Int},i,j} + \Delta$, post appropriate transaction and corresponding signature from $\left(tx^{d_1,d_2}_{\mathrm{IntSet},i,j}, \sigma^{d_1,d_2}_{\mathrm{IntSet},i,j} \right)$.
(8) After that transaction appears on $\mathbb{B}$ in round $\tau_2 \leq \tau_1 + \Delta$, set $\Theta^{d_1}(\mathrm{id}) \coloneqq \perp$, $\Gamma^{d_1}(\mathrm{id}) \coloneqq \perp$.
(9) If $tx^{d_2,d_2}_{\mathrm{Lazy},i}$ appears on $\mathbb{B}$ at or after round $\tau_3 \leq \tau_0 + \Delta T + t_{\mathrm{Int},i,j} + \Delta$, post appropriate transaction and corresponding signature from $\left(tx^{d_1,d_2}_{\mathrm{IntSet},i,j}, \sigma^{d_1,d_2}_{\mathrm{IntSet},i,j} \right)$.
(10) After that transaction appears on $\mathbb{B}$ in round $\tau_4 \leq \tau_3 + \Delta$, set $\Theta^{d_1}(\mathrm{id}) \coloneqq \perp$, $\Gamma^{d_1}(\mathrm{id}) \coloneqq \perp$.

Simulator for revoke by peers

Upon honest peer p sending $\mathtt{REVOKE} \overset{t_0}{\hookrightarrow} \mathcal{F}$:

For each $\mathrm{id} \in \{0,1\}^*$ such that $\Theta^{p,d_1}(\mathrm{id}) \neq \perp$:

(1) Iterate over all elements $(tx^{d_1}_{\mathrm{Cm},i}, tx^{d_2,d_1}_{\mathrm{PreRv},i}, \sigma^{d_1,p}_{\mathrm{PreRv},i})$ in $\Theta^{p,d_1}(\mathrm{id})$ and $(tx^{d_2}_{\mathrm{Cm},i}, tx^{d_1,d_2}_{\mathrm{PreRv},i}, \sigma^{d_2,p}_{\mathrm{PreRv},i})$ in $\Theta^{p,d_2}(\mathrm{id})$.

(2) If an old full-state commitment transaction $tx^{d_1}_{\mathrm{Cm},i}$ is on $\mathbb{B}$, prepare the pre-signed revocation transaction $tx^{d_1,d_2}_{\mathrm{PreRv},i}$ by inserting the additional output with balance δ.
(3) Post $(tx^{d_1,d_2}_{\mathrm{PreRv},i}, \sigma^{d_2,p}_{\mathrm{PreRv},i})$ on $\mathbb{B}$ after Δt and before the timelock ΔT.
(4) After $tx^{d_1,d_2}_{\mathrm{PreRv},i}$ is accepted by $\mathbb{B}$ in round $t_1 \leq t_0 + \Delta$, set $\Theta^{p,d_1}(\mathrm{id}) := \perp$.

Simulator for $\mathtt{ForceClose}^{d_1}(\mathrm{id})$

Let t_0 be the current round

(1) Extract $(tx_{Fu}, tx_{set_i}, sig^{d_1}_{set_i}, addr_{set}, pkey^{d_1}_{set}, pkey^{d_2}_{set}, pkey^{\mathcal{P}}_{\mathrm{set}})$ from $\Gamma^{d_1}(\mathrm{id})$.
(2) Extract $tx^{d_1,d_1}_{\mathrm{Inst},i}, tx^{d_1}_{\mathrm{Cm},i} \in tx_{set_i}$ and $\sigma^{d_1,d_1}_{\mathrm{Inst},i}, \sigma^{d_2,d_1}_{\mathrm{Cm},i} \in sig^{d_1}_{\mathrm{set}_i}$.
(3) Post $(tx^{d_1}_{\mathrm{Cm},i}, \sigma^{d_2,d_1}_{\mathrm{Cm},i})$ on $\mathbb{B}$.
(4) Let $t_1 \leq t_0 + \Delta$ be the round in which $tx^{d_1}_{\mathrm{Cm},i}$ is accepted by $\mathbb{B}$.
(5) Send $(\mathrm{peerSign}, \mathrm{id}, tx^{d_1,d_1}_{\mathrm{Inst},i}) \xhookrightarrow{t_1}$ each peer $p \in \mathcal{P}$.
(6) Wait one round.
(7) The simulator on behalf of d_1 together with each peer (honest and corrupted) runs the interactive protocol $\mathcal{F}_{Sign}$ simultaneously to generate σ^{p}_{St} on tx_{St}. This takes t_{peer} rounds.
(8) If valid signatures $\sigma^{p,d_1}_{\mathrm{Inst},i}$ from at least $\lfloor N/2 \rfloor + 1$ distinct peers are collected, go to next step. Else post $(tx^{d_1,d_1}_{\mathrm{Lazy},i}, \sigma^{d_1,d_1}_{\mathrm{Lazy},i})$ to $\mathbb{B}$ in round $t_3 \leq t_1 + 1 + t_{\mathrm{peer}} + \Delta T + \Delta$ and set $\Theta^{d_1}(\mathrm{id}) := \perp$, $\Gamma^{d_1}(\mathrm{id}) := \perp$.
(9) Post $(tx^{d_1,d_1}_{\mathrm{Inst},i}, \{\sigma^{d_1,d_1}_{\mathrm{Inst},i}, \sigma^{p,d_1}_{\mathrm{Inst},i}$ for $\lfloor N/2 \rfloor + 1$ peers $p \in \mathcal{P}\})$ on $\mathbb{B}$ in round $t_2 \leq t_1 + 1 + t_{\mathrm{peer}} + \Delta$.
(10) Set $\Theta^{d_1}(\mathrm{id}) := \perp$, $\Gamma^{d_1}(\mathrm{id}) := \perp$.

REFERENCES

[1] J. He, W. Qiu, S. Zhuo, M. Xu, Q. Zhang, Z. Xiong, and Z. Zheng, "An efficient multiparty payment protocol for IoT micro-payments," *IEEE Internet Things J.*, vol. 11, no. 20, pp. 32854–32867, Oct. 2024, doi: 10.1109/JIOT.2024.3421245.
[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013, doi: 10.1016/j.future.2013.01.010.
[3] Z. Xu, W. Liang, K.-C. Li, J. Xu, and H. Jin, "A blockchain-based roadside unit-assisted authentication and key agreement protocol for Internet of Vehicles," *J. Parallel Distrib. Comput.*, vol. 149, pp. 29–39, Mar. 2021, doi: 10.1016/j.jpdc.2020.11.003.
[4] B. K. Sovacool and D. D. F. Del Rio, "Smart home technologies in europe: A critical review of concepts, benefits, risks and policies," *Renew. Sustain. Energy Rev.*, vol. 120, Mar. 2020, Art. no. 109663, doi: 10.1016/j.rser.2019.109663.
[5] P.-B. Wang, K.-C. Li, R.-H. Shi, and B.-S. Shao, "VC-DCPS: Verifiable cross-domain data collection and privacy-persevering sharing scheme based on lattice in blockchain-enhanced smart grids," *IEEE Internet Things J.*, vol. 10, no. 14, pp. 12449–12461, Jul. 2023, doi: 10.1109/JIOT.2023.3247487.
[6] S. Huckle, R. Bhattacharya, M. White, and N. Beloff, "Internet of Things, blockchain and shared economy applications," *Proc. Comput. Sci.*, vol. 98, pp. 461–466, Jan. 2016, doi: 10.1016/j.procs.2016.09.074.
[7] N. Papadis and L. Tassiulas, "Blockchain-based payment channel networks: Challenges and recent advances," *IEEE Access*, vol. 8, pp. 227596–227609, 2020, doi: 10.1109/ACCESS.2020.3046020.
[8] B. Radenković, M. Despotović-Zrakić, and A. Labus, "Digital payment systems: State and perspectives," in *Digital Transformation of the Financial Industry: Approaches and Applications*. Cham, Switzerland: Springer, 2023, pp. 203–216, doi: 10.1007/978-3-031-23269-5_12.
[9] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, and E. G. Sirer, "On scaling decentralized blockchains: (*a position* paper)," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2016, pp. 106–125, doi: 10.1007/978-3-662-53357-4_8.
[10] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: Mar. 21, 2025. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[11] V. Buterin. (2013). *Ethereum White Paper*. Accessed: Mar. 21, 2025. [Online]. Available: https://ethereum.org/en/whitepaper/
[12] S. Kim, Y. Kwon, and S. Cho, "A survey of scalability solutions on blockchain," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2018, pp. 1204–1207, doi: 10.1109/ICTC.2018.8539529.
[13] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "A survey on the scalability of blockchain systems," *IEEE Netw.*, vol. 33, no. 5, pp. 166–173, Sep. 2019, doi: 10.1109/MNET.001.1800290.
[14] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16440–16455, 2020, doi: 10.1109/ACCESS.2020.2967218.
[15] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "SoK: Layer-two blockchain protocols," in *Financial Cryptography and Data Security*. Cham, Switzerland: Springer, 2020, pp. 201–226, doi: 10.1007/978-3-030-51280-4_12.
[16] C. Sguanci, R. Spatafora, and A. Mario Vergani, "Layer 2 blockchain scaling: A survey," 2021, *arXiv:2107.10881*.
[17] J. Poon and T. Dryja. (2016). *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*. [Online]. Available: https://lightning.network/lightning-network-paper.pdf
[18] K. Kurimoto. (2021). *Lightning Network X IoT (LoT); Potential, Challenges and Solutions*. [Online]. Available: https://medium.com/nayuta-en/lightning-network-x-iot-lot-potential-challenges-and-solutions-6e4d8b4c252a
[19] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proc. 42nd IEEE Symp. Found. Comput. Sci.*, Oct. 2001, pp. 136–145, doi: 10.1109/SFCS.2001.959888.
[20] J. Robert, S. Kubler, and S. Ghatpande, "Enhanced lightning network (off-chain)-based micropayment in IoT ecosystems," *Future Gener. Comput. Syst.*, vol. 112, pp. 283–296, Nov. 2020, doi: 10.1016/j.future.2020.05.033.
[21] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with IoT. Challenges and opportunities," *Future Gener. Comput. Syst.*, vol. 88, pp. 173–190, Nov. 2018, doi: 10.1016/j.future.2018.05.046.
[22] S. Dziembowski, S. Faust, and K. Hostáková, "General state channel networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 949–966, doi: 10.1145/3243734.3243856.

[23] J. Spilman. (2013). *Anti Dos for Tx Replacement*. Bitcoin-Dev Mailing List. [Online]. Available: https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002433.html

[24] C. Decker and R. Wattenhofer, "A fast and scalable payment network with Bitcoin duplex micropayment channels," in *Stabilization, Safety, and Security of Distributed Systems*. Cham, Switzerland: Springer, 2015, pp. 3–18, doi: 10.1007/978-3-319-21741-3_1.

[25] C. Hannon and D. Jin, "Bitcoin payment-channels for resource limited IoT devices," in *Proc. Int. Conf. Omni-Layer Intell. Syst.*, May 2019, pp. 50–57, doi: 10.1145/3312614.3312629.

[26] A. Pouraghily and T. Wolf, "A lightweight payment verification protocol for blockchain transactions on IoT devices," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2019, pp. 617–623, doi: 10.1109/ICCNC.2019.8685545.

[27] C. Profentzas, M. Almgren, and O. Landsiedel, "TinyEVM: Off-chain smart contracts on low-power IoT devices," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Nov. 2020, pp. 507–518, doi: 10.1109/ICDCS47774.2020.00025.

[28] N. Tapas, Y. Yitzchak, F. Longo, A. Puliafito, and A. Shabtai, "P4UIoT: Pay-per-piece patch update delivery for IoT using gradual release," *Sensors*, vol. 20, no. 7, p. 2156, Apr. 2020, doi: 10.3390/s20072156.

[29] G. A. F. Rebello, M. Potop-Butucaru, M. D. de Amorim, and O. C. M. B. Duarte, "Securing wireless payment-channel networks with minimum lock time windows," in *Proc. IEEE Int. Conf. Commun.*, May 2022, pp. 2297–2302, doi: 10.1109/ICC45855.2022.9839064.

[30] A. Kurt, S. Mercan, E. Erdin, and K. Akkaya, "3-of-3 multisignature approach for enabling lightning network micro-payments on IoT devices," *ITU J. Future Evolving Technol.*, vol. 2, no. 5, pp. 53–67, Jul. 2021, doi: 10.52953/wzpc8083.

[31] A. Kurt, K. Akkaya, S. Yilmaz, S. Mercan, O. Shlomovits, and E. Erdin, "LNGate2: Secure bidirectional IoT micro-payments using Bitcoin's lightning network and threshold cryptography," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 6027–6044, May 2024, doi: 10.1109/TMC.2023.3317704.

[32] Lightning Labs. (2023). *Neutrino: Privacy-Preserving Bitcoin Light Client*. [Online]. Available: https://github.com/lightninglabs/neutrino

[33] ACINQ. (2023). *Phoenix is a Self-Custodial Bitcoin Wallet Using Lightning to Send/Receive Payments*. [Online]. Available: https://github.com/ACINQ/phoenix

[34] J. Lind, O. Naor, I. Eyal, F. Kelbert, E. G. Sirer, and P. Pietzuch, "Teechain: A secure payment network with asynchronous blockchain access," in *Proc. 27th ACM Symp. Operating Syst. Princ.*, Oct. 2019, pp. 63–79, doi: 10.1145/3341301.3359627.

[35] Q. Wang, C. Zhang, L. Wei, and Y. Xie, "HyperChannel: A secure Layer-2 payment network for large-scale IoT ecosystem," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6, doi: 10.1109/ICC42927.2021.9500288.

[36] S. Popov, "The tangle," *White Paper*, vol. 1, no. 3, p. 30, 2018.

[37] *Nano Whitepaper*. Accessed: Jul. 21, 2025. [Online]. Available: https://docs.nano.org/whitepaper/english/

[38] *IoTeX*. Accessed: Jul. 21, 2025. [Online]. Available: https://iotex.io/

[39] *IoT Chain*. Accessed: Jul. 21, 2025. [Online]. Available: https://iotchain.io/aboutus

[40] *Walton Chain*. Accessed: Jul. 21, 2025. [Online]. Available: https://www.waltonchain.org/en/

[41] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "Perun: Virtual payment hubs over cryptocurrencies," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 106–123, doi: 10.1109/SP.2019.00020.

[42] Z. Ge, Y. Zhang, Y. Long, and D. Gu, "Magma: Robust and flexible multi-party payment channel," *IEEE Trans. Depend. Secure Comput.*, vol. 20, no. 6, pp. 5024–5042, Nov. 2023, doi: 10.1109/TDSC.2023.3238332.

[43] Y. Ye, Z. Ren, X. Luo, J. Zhang, and W. Wu, "Garou: An efficient and secure off-blockchain multi-party payment hub," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 4450–4461, Dec. 2021, doi: 10.1109/TNSM.2021.3101808.

[44] S. Mercan, A. Kurt, K. Akkaya, and E. Erdin, "Cryptocurrency solutions to enable micropayments in consumer IoT," *IEEE Consum. Electron. Mag.*, vol. 11, no. 2, pp. 97–103, Mar. 2022, doi: 10.1109/MCE.2021.3060720.

[45] M. Hearn and M. Corallo. *Bip 37: Connection Bloom Filtering*. Accessed: Jul. 21, 2025. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0037.mediawiki

[46] O. Osuntokun, A. Akselrod, and J. Posen. *Bip 157: Client Side Block Filtering*. Accessed: Jul. 21, 2025. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0157.mediawiki

[47] O. Osuntokun and A. Akselrod. *Bip 158: Compact Block Filters for Light Clients*. Accessed: Jul. 21, 2025. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0158.mediawiki

[48] M. Li, Y. Yang, G. Chen, M. Yan, and Y. Zhang, "SoK: Understanding design choices and pitfalls of trusted execution environments," in *Proc. 19th ACM Asia Conf. Comput. Commun. Secur.*, Jul. 2024, pp. 1600–1616, doi: 10.1145/3634737.3644993.

[49] T. Dryja and S. B. Milano. (2016). *Unlinkable Outsourced Channel Monitoring*. [Online]. Available: https://diyhpl.us/wiki/transcripts/scalingbitcoin/milan/unlinkable-outsourced-channel-monitoring/

[50] G. Avarikioti, F. Laufenberg, J. Sliwinski, Y. Wang, and R. Wattenhofer, "Towards secure and efficient payment channels," 2018, *arXiv:1811.12740*.

[51] M. Khabbazian, T. Nadahalli, and R. Wattenhofer, "Outpost: A responsive lightweight watchtower," in *Proc. 1st ACM Conf. Adv. Financial Technol.*, Oct. 2019, pp. 31–40, doi: 10.1145/3318041.3355464.

[52] P. McCorry, S. Bakshi, I. Bentov, S. Meiklejohn, and A. Miller, "Pisa: Arbitration outsourcing for state channels," in *Proc. 1st ACM Conf. Adv. Financial Technol.*, Oct. 2019, pp. 16–30, doi: 10.1145/3318041.3355461.

[53] Z. Avarikioti, O. S. Thyfronitis Litos, and R. Wattenhofer, *Cerberus Channels: Incentivizing Watchtowers for Bitcoin*. Cham, Switzerland: Springer, 2020, pp. 346–366, doi: 10.1007/978-3-030-51280-4_19.

[54] B. Liu, P. Szalachowski, and S. Sun, "Fail-safe watchtowers and short-lived assertions for payment channels," in *Proc. 15th ACM Asia Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 506–518, doi: 10.1145/3320269.3384716.

[55] Y. Zhang, D. Yang, G. Xue, and R. Yu, "Counter-collusion smart contracts for watchtowers in payment channel networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2021, pp. 1–10, doi: 10.1109/INFOCOM42981.2021.9488831.

[56] A. Mirzaei, A. Sakzad, J. Yu, and R. Steinfeld, *Garrison: A Novel Watchtower Scheme for Bitcoin*. Cham, Switzerland: Springer, 2022, pp. 489–508, doi: 10.1007/978-3-031-22301-3_24.

[57] M. Du, P. Yang, W. Tian, and Z. Han, "Anti-collusion multiparty smart contracts for distributed watchtowers in payment channel networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3600–3614, Dec. 2022, doi: 10.1109/JSAC.2022.3213355.

[58] A. Mirzaei, A. Sakzad, J. Yu, and R. Steinfeld, "Formal treatment of watchtowers and FPPW: A fair and privacy-preserving Bitcoin watchtower," *IEEE Trans. Depend. Secure Comput.*, vol. 22, no. 3, pp. 2385–2397, May 2025, doi: 10.1109/TDSC.2024.3502689.

[59] M. Xu, Y. Zhang, and S. Zhong, "Towards payment channel watchtowers with collateral-free security and robustness," *IEEE Trans. Depend. Secure Comput.*, vol. 22, no. 2, pp. 1519–1536, Mar. 2025, doi: 10.1109/TDSC.2024.3445429.

[60] L. Aumayr, S. A. Thyagarajan, G. Malavolta, P. Moreno-Sanchez, and M. Maffei, "Sleepy channels: Bi-directional payment channels without watchtowers," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2022, pp. 179–192, doi: 10.1145/3548606.3559370.

[61] R. O'Connor and M. Piekarska, *Enhancing Bitcoin Transactions With Covenants*. Cham, Switzerland: Springer, 2017, pp. 191–198, doi: 10.1007/978-3-319-70278-0_12.

[62] R. Canetti, Y. Dodis, R. Pass, and S. Walfish, *Universally Composable Security With Global Setup*. Berlin, Germany: Springer, pp. 61–85, doi: 10.1007/978-3-540-70936-7_4.

[63] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sánchez, and S. Riahi, "Generalized channels from limited blockchain scripts and adaptor signatures," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2021, pp. 635–664, doi: 10.1007/978-3-030-92075-3_22.

[64] S. D. Lerner, J. Á. Cid-Fuentes, J. Len, R. Fernàndez-València, P. Gallardo, N. Vescovo, R. Laprida, S. Mishra, F. Jinich, and D. Masini, "RSK: A Bitcoin sidechain with stateful smart-contracts," Cryptology ePrint Arch., Int. Association Cryptologic Res (IACR), Tech. Rep. 2022/684, 2022. [Online]. Available: https://eprint.iacr.org/2022/684

[65] J. Nick, A. Poelstra, and G. Sanders. (2020). *Liquid: A Bitcoin Sidechain*. Liquid White Paper. [Online]. Available: https://blockstream.com/assets/downloads/pdf/liquid-whitepaper.pdf

ERFAN BAHRAMI was born in Isfahan, Iran, in 1998. He received the Diploma degree in mathematics and physics from the National Organization for Development of Exceptional Talents (NODET), in 2017, the B.S. degree in computer engineering from the Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran, in 2022, and the M.S. degree in computer engineering with a specialization in secure computing from the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, in 2025. His research interests include blockchain, distributed systems, applied cryptography, and the Internet of Things (IoT).

AMIR M. AGHAPOUR received the B.S. degree in computer engineering from the Department of Computer Science, Amirkabir University of Technology, Tehran, Iran, in 2022, and the M.S. degree in computer engineering from the Department of Computer Engineering, Sharif University of Technology, Tehran, in 2024. He is currently pursuing the Ph.D. degree in secure computing with the Sharif University of Technology. His research interests include the development of offline payment systems, fairness of cryptocurrencies, and the integration of programmability into traditional financial systems.

MORTEZA AMINI received the B.S. degree in computer software engineering from Shahid Beheshti University, Tehran, Iran, in 2001, and the M.S. degree in computer software engineering and the Ph.D. degree in software engineering (data security field) from the Sharif University of Technology, Tehran, in 2004 and 2010, respectively. He is currently an Associate Professor with the Department of Computer Engineering, Sharif University of Technology. He is also one of the department's directors of the Data and Network Security Laboratory (DNSL). Additionally, he is the Dean of Education at the Sharif University of Technology. His research interests include information security, database security and access control, intrusion and advanced persistent threat (APT) detection, alert/event correlation, the IoT security and privacy, and android security.