USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

Thunderdome: Timelock-Free Rationally-Secure Virtual Channels

Zeta Avarikioti, *TU Wien & Common Prefix;* Yuheng Wang, *TU Wien;*
Yuyi Wang, *CRRC Zhuzhou Institute & Tengen Intelligence Institute*

https://www.usenix.org/conference/usenixsecurity25/presentation/avarikioti

This paper is included in the Proceedings of the
34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the
34th USENIX Security Symposium is sponsored by USENIX.

Thunderdome: Timelock-Free Rationally-Secure Virtual Channels

Zeta Avarikioti
TU Wien & Common Prefix

Yuheng Wang
TU Wien

Yuyi Wang
CRRC Zhuzhou Institute & Tengen Intelligence Institute

Abstract

Payment channel networks (PCNs) offer a promising solution to address the limited transaction throughput of deployed blockchains. However, several attacks have recently been proposed that stress the vulnerability of PCNs to timelock and censoring attacks. To address such attacks, we introduce Thunderdome, the first timelock-free PCN. Instead, Thunderdome leverages the design rationale of virtual channels to extend a timelock-free payment channel primitive, thereby enabling multi-hop transactions without timelocks. Previous works either utilize timelocks or do not accommodate transactions between parties that do not share a channel.

At its core, Thunderdome relies on a committee of *non-trusted* watchtowers, known as wardens, who ensure that no honest party loses funds, even when offline, during the channel closure process. We introduce tailored incentive mechanisms to ensure that all participants follow the protocol's correct execution. Besides a traditional security proof that assumes an honest majority of the committee, we conduct a formal game-theoretic analysis to demonstrate the security of Thunderdome when all participants, including wardens, act rationally. We implement a proof of concept of Thunderdome on Ethereum to validate its feasibility and evaluate its costs. Our evaluation shows that deploying Thunderdome, including opening the underlying payment channel, costs approximately $15 (0.0089 ETH), while the worst-case cost for closing a channel is about $7 (0.004 ETH).

1 Introduction

Blockchains have introduced a groundbreaking decentralized financial paradigm through cryptocurrencies, eliminating the need for trusted intermediaries [42]. However, blockchains face major scalability limitations, primarily due to the requirement that each transaction be validated by all nodes in the network [15]. As a result, popular cryptocurrencies like Bitcoin [31] and Ethereum suffer from relatively slower processing speeds and much lower transaction throughput compared to centralized payment systems like Visa, hindering their potential for mass adoption. While solutions such as more efficient consensus protocols [13, 16, 36] can increase throughput, they typically require changes to existing protocols, often causing a hard fork. To overcome these scalability challenges, Payment Channel Networks (PCNs) have emerged as a promising alternative [23].

A payment channel functions as a secure off-chain "joint account" between two parties, utilizing the blockchain only for the opening and closing phases. This approach substantially reduces the number of on-chain transactions, effectively addressing the scalability issues of the underlying blockchain. Payment channels involve three primary operations: *Open*, *Update*, and *Close*. The channel is opened by locking a certain amount of coins on-chain, referred to as the channel balance. The parties can then transact off-chain by updating their channel balance through the exchange of digitally signed messages that specify the new distribution of coins. Finally, the channel can be closed, settling all off-chain transactions with the final agreed-upon state posted on-chain. To prevent fraud, payment channel protocols often include a dispute period with a *timelock*, allowing a counterparty to penalize a potentially dishonest party. For example, in the Bitcoin Lightning Network [33], a party can claim the entire channel balance if the other party posts an outdated update.

An overlay Payment Channel Network (PCN) operates as a Layer 2 solution on top of a single blockchain (Layer 1), primarily functioning off-chain to address the scalability limitations of the base network. Notably, a PCN allows parties with at least one direct payment channel to make payments across the network to other parties, even if they do not share a direct channel. To facilitate multi-hop payments from sender to receiver, a path of sufficiently funded channels is identified. All channels along this path are then updated using lock contracts with a *timelock* to ensure the atomic execution of the payment. Techniques such as Hash Timelock Contracts (HTLCs), used in the Bitcoin Lightning Network [33], and methods like adaptor signatures [3, 38] or Verifiable Timed Signatures (VTS) [39], are commonly employed in PCNs.

These techniques rely on timelocks to ensure that parties can recover their coins in the event of a payment failure. Thus, *timelocks are essential for the security of both payment channel primitives and multi-hop payment protocols in PCNs.*

Recent research reveals that timelocks can introduce new attack vectors, notably censorship attacks such as timelock bribing. These attacks show that PCNs relying on timelock contracts for multi-hop payments might allow an adversary to steal the entire channel balance by delaying the on-chain inclusion of a specific transaction [14, 22, 30, 41]. Similar vulnerabilities are present in the payment channel primitives, as noted in [8,9], these issues arise from the design rationale of timelock-based channel protocols: Lighting [33] and Blitz use a "revoke until time T, else execute" logic, while DMC [17] employs a "revoke within time T, else execute" logic. In these cases, censoring an honest party's transaction prevents them from revoking misbehavior. Conversely, Thora [2] uses an "execute until time T, else revoke" logic, and Perun [18] follows a "reply within time T, else get punished" logic. In these designs, censoring an honest party's transaction blocks them from executing the committed transactions, resulting in a loss of funds. These censorship attacks, while difficult to trace as excluded transactions are not reported on-chain, can be practically executed using TxWithhold Smart Contracts, as highlighted by BitMEX Research [1]. Additionally, the cost of such attacks can be significantly reduced since multiple contracts can be targeted simultaneously (e.g., as many as fit in a block) [7, 41].

A common approach to lowering the success probability of censorship attacks is to use longer timelocks [30]. However, this strategy worsens another significant drawback of timelocks, known as the *griefing attack* [26]. In such attacks, the receiver initiates a payment path within a PCN but then aborts the payment, causing intermediaries to incur an *opportunity cost* as they lock their assets for the timelock duration without receiving any routing fee. Given the inherent instability of real-world networks and the potential for network outages, relying solely on timelocks for secure and efficient PCN implementation proves to be a flawed approach. This raises a critical question: *Can secure PCNs be designed without the reliance on timelocks?*

1.1 Related work

Various solutions have been proposed to counter the rising threat of censorship attacks on Payment Channel Networks (PCNs), many of which exploit miners' incentives, as miners often control censorship. Nadahalli et al. [30] were among the first to identify the safety vulnerabilities of Hash Timelock Contracts (HTLCs) under censorship, analyzing parameters like timelock duration and transaction fees to determine when HTLCs remain secure. Building on this, Tsabary et al. [41] introduced MAD-HTLC, a modified structure that incentivizes miners to act as natural enforcers by penalizing malicious actors. However, the counter-bribing vulnerabilities in MAD-HTLC were later identified, leading to the development of He-HTLC [22] and Rapidash [14]. Although these approaches provide secure channel primitives, they rely on specific assumptions, such as mining power distribution, and crucially require parties to remain online and responsive.

Another approach involves introducing third-party entities, known as watchtowers, to manage disputes on behalf of payment channel participants in a timely manner [6, 10, 24, 27]. However, watchtowers are themselves susceptible to censorship attacks [8] unless they also operate as miners, ensuring independent inclusion of transactions. Thus, the security of this approach depends on additional assumptions, such as the watchtower's mining power and ability to fulfill dual roles. Furthermore, while watchtowers help secure the payment channel primitive, they do not address the execution of multi-hop payments, leaving a gap in overall network protection.

Brick [8] introduced a pioneering payment channel primitive that eliminated the need for timelocks by internalizing dispute resolution within the channel itself. This was accomplished by establishing a committee of watchtowers, known as wardens, who were incentivized to act honestly through the collateral they had locked in the payment channel contract. These wardens were authorized to unilaterally close the Brick channel at the latest update state, removing the need for timelocks. However, extending Brick to support multi-hop payments introduced challenges, as current multi-hop execution mechanisms (e.g., HTLCs) still require timelocks, irrespective of the underlying channel primitive.

Recently, Ersoy et al. extended Brick to multi-hop scenarios by proposing a multi-hop payment protocol based on warden committees [20]. While promising, this protocol still has notable limitations. First, like traditional multi-hop payment protocols, Ersoy's approach requires the active participation of intermediary parties in each transaction, leading to higher costs and increased latency. Although the protocol claims to support virtual channels, a detailed design is not provided. Second, the protocol lacks a thorough game-theoretic analysis to formally evaluate the security implications of its incentive mechanisms.

Overall, while recent research has made significant progress in addressing timelock-based attacks in PCNs, current solutions remain constrained by specific assumptions (e.g., mining power distribution, intermediary participation) or inherent design limitations (e.g., limited applicability to multi-hop payments). These constraints underscore the need for an alternative approach that overcomes these challenges.

1.2 Our contribution

In this work, we introduce Thunderdome, *the first timelock-free virtual channel protocol*, providing an answer to the previously posed challenge. The core concept of Thunderdome revolves around a committee of wardens appointed by the

channel parties, who store the channel states and publish the most recent state on-chain when parties are unresponsive, whether due to being offline or censored. These wardens are incentivized with rewards or penalties based on their actions, ensuring the security of Thunderdome. As a result, Thunderdome effectively neutralizes all blockchain liveness attacks, including censorship attacks, within its payment channel network. To enable multi-hop payments, Thunderdome leverages the design principles of virtual channels. First introduced by Dziembowski et al. [18], virtual channels allow direct off-chain payments without requiring intermediary involvement in every transaction, as seen in multi-hop payment protocols like Blitz. Central to this concept is the mechanism that allows Alice and Bob, who do not share a direct payment channel, to collaborate with Ingrid, who does, to create a virtual channel. This setup enables Alice to transact directly with Bob, bypassing the need for Ingrid's participation. In essence, a virtual channel is a channel built atop two existing payment channels. Thunderdome adopts a similar architectural design, constructed over two Brick channels, rather than relying on timelock-based payment channel primitives.

However, constructing a secure virtual channel atop two Brick channels presents several challenges due to Brick's design and inherent limitations. Specifically, a Brick channel relies on complex incentive mechanisms for both parties and the warden committee to maintain balance security (i.e., safety) and prevent hostage situations (i.e., liveness). These incentive mechanisms vary between Brick channels with different warden committees, making them not directly composable. As a result, the virtual channel cannot function as a standard payment channel without additional considerations.

Specifically, three major challenges arise when transitioning from Brick to Thunderdome. First, Brick employs proofs-of-fraud that allow a party to claim the wardens' collateral. However, since the virtual channel comprises two separate Brick channels, a party in one channel cannot directly access the collateral locked in the other channel's smart contract. Furthermore, in a virtual channel, the intermediary party remains unaware of the virtual channel's state, as transactions are exclusively exchanged between the other two parties. This leaves the intermediary party vulnerable to potential fraud by the other participants or wardens during the channel closure. Therefore, it is uncertain whether the proof-of-fraud mechanism can still ensure the security of all virtual channel participants, including the intermediary. Second, since the virtual channel is built directly on payment channels without requiring additional on-chain deposits for its opening, ensuring compatibility with the underlying Brick payment channels presents a challenge. Third, although Brick is claimed to be secure in the rational setting, no formal framework or analysis has been provided. Thus, the third challenge is to develop a formal model that can prove Thunderdome's security when all participants act rationally.

To address the first challenge, we modify the protocols involved in the update and close operations of Thunderdome. We ensure the correctness of channel closure while allowing parties to penalize only those wardens with collateral in the relevant payment channel's smart contract. Additionally, we achieve closing security for the intermediary party by ensuring they either have Thunderdome knowledge before closing the channel or the closing party is indistinguishable in terms of possessing such knowledge. For the second challenge, we draw inspiration from [4], offloading the virtual channel to an on-chain payment channel only in pessimistic cases with the blockchain's assistance. This allows us to distribute wardens' collateral according to each channel's balance, ensuring Thunderdome remains securely compatible with the underlying payment channel. To address the third challenge, we model our protocol using Extensive Form Games (EFG), drawing on recent work that introduced the first game-theoretic analysis suitable for off-chain protocols like the Bitcoin Lightning Network [34]. Specifically, we model Thunderdome 's closing operation as an EFG and prove the game-theoretic security of the protocol by analyzing all possible strategies during the channel closure.

Summary of Contribution. Our contributions can be summarized as follows:

- We introduce Thunderdome, the first timelock-free virtual channel protocol [18]. Thunderdome builds upon the asynchronous payment channel primitive, Brick [8], enabling secure payments between parties that do not share a direct channel, all without the need for timelocks (Section 5). We begin by presenting the design and security analysis of Thunderdome with a single intermediary and then extend the protocol to support multi-hop payments involving more than two hops (Section 5.4).

- We formalize the security properties of Thunderdome and conduct a security analysis within the honest/Byzantine model (Section 6). Specifically, we demonstrate that our protocol is secure under the assumption of distrusting participants, where f out of $3f+1$ wardens in each payment channel are Byzantine, and the remainder are honest.

- We design the incentive mechanisms of Thunderdome and introduce a formal game-theoretic model to represent the protocol (Section 7). We prove that following the Thunderdome closing protocol honestly constitutes a Subtree Perfect Nash Equilibrium (SPNE) strategy, ensuring the protocol's game-theoretic security in the rational model. To our knowledge, this is the first off-chain protocol to include a formal game-theoretic analysis.

- We evaluate the practicality of Thunderdome by fully implementing its on-chain functions on the Ethereum blockchain using Solidity (Section 8). We assess the gas costs for each procedure with 10 wardens in each committee. While the cost of opening Thunderdome is zero,

deploying and opening the underlying payment channel requires 444,4861 gas (approximately 14.83 USD). In the pessimistic scenario, closing Thunderdome along with the underlying payment channel costs 2,079,766 gas (around 6.94 USD). We compare these costs with the timelock-based virtual channel protocol Perun: opening Thunderdome's underlying channel costs 1.5 times more than Perun, while the pessimistic closing costs are 3 times higher. This shows that the additional cost of eliminating timelocks is not excessive. Additionally, we evaluate how gas fees scale as the number of wardens per committee increases from 10 to 25.

2 Background

In this section, we provide a brief overview of the fundamental concepts of payment channels, with further details available in Gudgeon et al. [23]. We then explore the design of the asynchronous Brick payment channel [8], which serves as the foundation of our protocol. Finally, we discuss payment channel networks and virtual channels.

2.1 Payment channels

A payment channel allows two users to exchange arbitrary transactions off-chain, while only a constant number of transactions are recorded on-chain, such as a worst-case scenario of three transactions in Lightning [33]. To initiate a payment channel, parties deposit coins on the blockchain, which can only be spent when specific conditions are met, such as requiring both parties' signatures (*channel open*). Once the coins are locked on-chain, the parties can communicate off-chain and update the channel's balance or state (*channel update*). To claim their funds on-chain, the parties can close the payment channel by publishing the most recent agreed-upon balance (*channel close*). In the following, we discuss the key steps involved in these three operations in detail.

Channel open. Consider two parties, Alice and Bob, who wish to open a payment channel with initial deposits of x_A and x_B coins, respectively. For cryptocurrencies that support smart contracts, Alice and Bob can publish a *payment channel smart contract* C_L on the blockchain, containing a total balance of $x_A + x_B$ from both parties. These coins can only be spent with the signatures of both Alice (σ_A) and Bob (σ_B). Once C_L is published on-chain, the payment channel is officially open.

Channel update. Suppose Alice wants to pay Bob an amount $a \leq x_A$. Alice generates a new payment channel state, s_L, signs it with her private key, and sends $\{s_L, \sigma_A(s_L)\}$ to Bob as an update request. Bob then verifies the request. If he agrees with the new state, he signs it as well and returns the final state, $\{s_L, \sigma_A(s_L), \sigma_B(s_L)\}$, to Alice. If Bob disagrees, he simply ignores the request. Once a valid channel state, signed by both parties, is generated (along with any other predefined protocol data exchange, such as revocation keys in Lightning), the new off-chain transaction between Alice and Bob is considered successfully completed.

Channel close. A payment channel can be closed either collaboratively or unilaterally by one party. In a collaborative closure, both parties publish a state with their signatures on-chain and distribute the channel balance accordingly. In a unilateral closure, one party publishes the most recent payment channel state signed by both, $\{s_{latest}, \sigma_A(s_{latest}), \sigma_B(s_{latest})\}$. After verifying the signatures, the smart contract C_L closes the channel and distributes the coins according to the submitted state. The key challenge arises when a malicious party attempts to close the channel using an outdated state. Synchronous payment channel protocols typically address this by enforcing a timelock on the submitted state, which corresponds to the party posting it on-chain. This timelock, often combined with a secret exchanged during the channel update, enables the counterparty to punish a malicious party attempting to use an old state. Alternative techniques, such as verifiable timed signatures [40], can also be used, but they still rely on timing assumptions.

2.2 Brick channel

The Brick channel, introduced in [8], is an asynchronous payment channel primitive. To address the challenges of operating without timelocks, Brick incorporates a committee of third-party entities known as wardens in the channel operations. In essence, wardens are responsible for verifying and storing commitments of channel state updates, which can then be used by a channel party to unilaterally close the Brick channel. Importantly, wardens are not fully trusted; rather, they are incentivized to follow the protocol honestly. Compared to its synchronous counterparts, Brick exhibits the following key differences:

Channel structure. In addition to the two primary parties, Alice and Bob, a Brick channel includes a committee of $3f+1$ wardens, where up to f wardens can behave maliciously (Byzantine). The basic structure of the Brick channel is illustrated in Fig. 1.

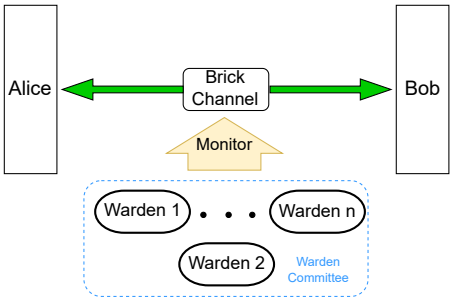

Figure 1: Brick payment channel

Brick open. Once Alice and Bob agree to open a Brick channel, they broadcast the channel information to the war-

dens. Each party can consider the channel open only after receiving at least $2f+1$ signed acknowledgments from the wardens – a quorum certificate. The threshold of $2f+1$ ensures safety in asynchronous communication networks.

Brick update. Each state generated by Alice and Bob is hashed and signed by both parties. The state is then assigned a sequence number to indicate its order. This sequence number, along with the hashed state, is signed and broadcast to the wardens. Once at least $2f+1$ wardens have signed the sequence number, the state is considered valid, and the parties can execute it.

Brick close. The Brick channel closing protocol has two scenarios: *optimistic* and *pessimistic*. In the optimistic case, both parties are online and responsive, allowing them to collaboratively generate and sign a closing request, which is then published on-chain. In the pessimistic case, one party, say Alice, may be offline for an extended period and unresponsive to Bob's close request. To prevent the channel from being indefinitely locked, the Brick protocol allows Bob to close the channel unilaterally. However, Bob alone does not have a valid closing request, as during the channel update, the parties only exchange signatures on the hashed state, which is insufficient to close the channel. To close unilaterally, Bob initiates a closing request on-chain, prompting the wardens to publish the latest stored sequence number. The valid closing state is defined by the highest sequence number signed by Alice, Bob, and at least one warden. To incentivize honest behavior from the wardens, Brick employs a punishment mechanism based on proofs-of-fraud. A *proof-of-fraud* consists of a warden's signature on an update with a higher sequence number than the one they submitted for the closing request. Bob collects these signatures during the channel update process. If a valid *proof-of-fraud* is provided, Bob can claim the corresponding warden's collateral on-chain. According to Brick's security analysis, if the aggregate collateral of wardens submitting outdated states exceeds the channel balance, Bob has no incentive to close the channel incorrectly. Instead, he would claim the wardens' collateral and award the entire channel balance to his counterparty. Specifically, for a Brick channel with v coins as the balance, each of the $3f+1$ wardens must deposit at least $\frac{v}{f}$ coins on-chain to ensure security.

2.3 Virtual channels

Virtual channels [4, 18] offer a solution for executing transactions between parties who do not share a direct channel while minimizing the involvement of an intermediary (Ingrid). Specifically, Alice and Bob can establish a virtual channel on top of two existing payment channels, as shown in Fig 2. Similar to payment channel protocols, virtual channels typically involve three operations: *open*, *update*, and *close*. However, Ingrid only needs to participate in the *open* and *close* operations. For example, if Alice and Bob want to create a virtual channel with a balance of $x_A + x_B$ coins, not only do Alice and Bob contribute coins, but Ingrid also deposits x_B and x_A coins, respectively, to open the virtual channel, as shown in Fig 2. Importantly, no additional coins are deposited on the blockchain for the virtual channel; instead, the virtual channel's funding is achieved by updating the payment channel state and utilizing a portion of the payment channel's funds. Unlike payment channels, opening a virtual channel generally occurs off-chain. After the virtual channel is opened, Alice and Bob can transact directly with each other during the *update* operation without needing Ingrid's involvement. When Alice and Bob decide to close the virtual channel, Ingrid participates in updating or closing the two underlying payment channels. The close operation only affects the underlying payment channels without directly impacting the blockchain. Accessing the blockchain is only required in pessimistic situations, such as when one party becomes unresponsive. Therefore, virtual channels can be seen as a "Layer 3" solution in blockchain architecture.

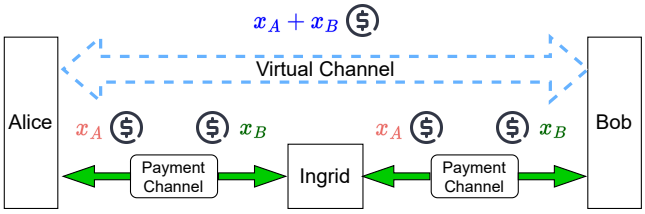

Figure 2: Virtual channel structure

3 Model

In this section, we first present our system model and underlying assumptions. We then formally define two distinct threat models: one assumes Byzantine participants, while the other assumes participants are rational agents aiming to maximize their profit. Finally, we outline the desired security properties of Thunderdome under both models.

3.1 System model and assumptions

There are two types of participants in Thunderdome: (1) the *main parties*, which include Alice (A), Bob (B), and Ingrid (I) in the two-hop scenario; and (2) the *wardens*, organized into warden committees C_A and C_B. The three main parties maintain two underlying timelock-free payment channels (TPC), such as Brick. The Alice-Ingrid and Ingrid-Bob channels are each monitored by the wardens in C_A and C_B, respectively. The general structure of Thunderdome is illustrated in Fig. 3.

We assume that all participants are computationally bounded and that cryptographically secure communication channels, signatures, hash functions, and encryption mechanisms are in place. The communication channels between Thunderdome participants are considered asynchronous,

meaning messages from honest parties may be delayed indefinitely but will eventually be delivered. This includes messages sent to the blockchain (miners), which can also experience arbitrary delays. We further assume that the underlying blockchain is safe and guarantees persistence [21].

3.2 Threat models

We introduce two threat models: the conventional security model, which assumes the presence of Byzantine participants, and a game-theoretic model, which considers rational participants. A game-theoretic analysis is particularly important for off-chain protocols, as the standard Byzantine security model may fail to capture certain threats that arise from party collusion, such as the WormHole attack [25]. Given the significant impact of Byzantine behavior on participant utilities and the lack-to the best of our knowledge—of a unified framework that addresses both models, it is crucial to assess the security of our protocol under both perspectives.

Byzantine security model. In this model, we assume each committee consists of $3f+1$ wardens, of which f may be Byzantine, while the rest are honest and follow the protocol as specified. We note that the protocol's security can be extended to scenarios where the number of wardens differs as long as the Byzantine ratio remains no larger than $\frac{f}{3f+1}$ (i.e., the total number of wardens may vary depending on the setup, but the Byzantine proportion must be maintained). In addition to wardens, the security model also accounts for cases where the main Thunderdome parties may be Byzantine, ensuring security for any honest participant. This is the weakest meaningful assumption, as collusion between all parties against a single honest party could occur; however, considering all parties deviating from the protocol holds no value since security properties apply only to honest participants. Byzantine nodes can arbitrarily deviate by delaying, crashing, or signing/publishing incorrect states or messages. However, they cannot drop honest messages or forge signatures, as these actions violate network and cryptographic assumptions.

Game-theoretic model. In this model, all participants are treated as rational, mutually distrusting agents. Rational agents will deviate from the protocol – such as double signing, publishing outdated states, or crashing – when doing so allows them to maximize their utility, i.e., gain more profit. Wardens are required to lock collateral for their participation in each channel. Overlapping wardens lock collateral in separate smart contracts for each channel, ensuring that punishments remain independent. Additionally, the participants' budgets are limited to their on-chain collateral, and external budgets are not considered in our analysis. This approach is consistent with most blockchain protocols, as proving economic security in the presence of infinite shorting markets (e.g., Bitcoin or Ethereum) is infeasible.

3.3 Protocol Goals

The two primary goals of our protocol are *balance security* and *liveness*. Balance security ensures the basic safety of channels, meaning that no honest participant loses coins during the execution of the virtual channel protocol. Honest participants are defined as those who adhere to the protocol specifications.

Definition 1 (Balance security)**.** Any honest participant of Thunderdome does not lose coins.

In addition to balance security, Thunderdome must ensure that the protocol progresses meaningfully. For example, consider a channel protocol that only contains dummy functions that make no changes to the channel state. Although no participant would lose coins, such a protocol is useless, as no valid channel state could ultimately be committed on-chain. More broadly, hostage situations can still cause losses for participants, even if the channel funds remain intact. The liveness property addresses these issues, complementing balance security by ensuring meaningful progress.

Definition 2 (Liveness)**.** Any valid operation (update or close) on the state of the virtual channel, involving at least one honest participant (Alice, Bob, or Ingrid), will eventually either be committed on-chain or invalidated.

We note that the *validity* of operations is determined by the protocol specification. For example, in our protocol, an operation is considered valid if it is agreed upon by the two main parties, involving at least one honest participant (Alice, Bob, or Ingrid). In contrast, in the Lightning Network, a valid update corresponds to the so-called commitment transaction [33]. A valid update is either the latest one, capable of being committed on-chain, or it is replaced by a newer valid update and thereby invalidated.

In our analysis, we demonstrate that Thunderdome satisfies these properties in both the Byzantine and game-theoretic models. However, to prove these properties in the game-theoretic model, we employ different tools: we model the protocol as an Extensive Form Game (EFG) with *perfect information* and show that the correct strategy profile forms a Subgame Perfect Nash Equilibrium (SPNE).

Definition 3 (Extensive Form Game-EFG)**.** And Extensive Form Game (EFG) is a tuple $\mathcal{G} = (N, H, P, u)$, where set N represents the game player, the set H captures EFG game history, $T \subseteq H$ is the set of terminal histories, P denotes the next player function, and u is the utility function. The following properties are satisfied.

(A) The set H of histories is a set of sequence actions with

1. $\emptyset \in H$;
2. if the action sequence $(a_k)_{k=1}^{K} \in H$ and $L < K$, then also $(a_k)_{k=1}^{L} \in H$;
3. an action sequence is terminal $(a_k)_{k=1}^{K} \in T$, if there is no further action a_{K+1} that $(a_k)_{k=1}^{K+1} \in H$.

(B) The next player function P

1. assigns the next player $p \in N$ to every non-terminal history $(a_k)_{k=1}^{K} \in H \setminus T$;
2. after a non-terminal history h, it is player $P(h)$'s turn to choose an action from the set $A(h) = \{a : (h,a) \in H\}$.

A player p's strategy is a function σ_p mapping every history $h \in H$ with $P(h) = p$ to an action from $A(h)$. Formally, $\sigma_p : \{h \in H : P(h) = p\} \to \{a : (h,a) \in H, \forall h \in H\}$, such that $\sigma_p(h) \in A(h)$.

The subgame of an EFG is a subtree determined by a certain history (i.e., whose root note is the last history node) and is formalized by the following definition [34]:

Definition 4 (EFG subgame)**.** The subgame of an EFG $\varphi = (N, H, P, u)$ associated to history $h \in H$ is the EFG $\varphi(h) = (N, H_{|h}, P_{|h}, u_{|h})$ defined as follows: $H_{|h} := h'|(h,h') \in H$, $P_{|h}(h') := P(h,h')$, and $u_{|h}(h') := u(h,h')$.

In an EFG, a strategy profile that prescribes utility-maximizing choices at every decision point in every subgame is named Subgame Perfect Nash Equilibrium (SPNE) [32].

Definition 5 (Subgame Perfect Nash Equilibrium (SPNE))**.** A subgame perfect equilibrium strategy is a joint strategy $\sigma = (\sigma_1, ..., \sigma_n) \in S$, s.t. $\sigma_{|h} = (\sigma_{1|h}, ..., \sigma_{n|h})$ is a Nash Equilibrium of the subgame $\varphi(h)$, for every $h \in H$. The strategies $\sigma_{i|h}$ are functions that map every $h' \in H_{|h}$ with $P_{|h}(h') = i$ to an action from $A_{|h}(h')$.

Using the definitions above, we establish below a security property for Thunderdome in the game-theoretic model that encompasses that rational participants will consistently follow the protocol. We primarily focus on the closing procedure of Thunderdome, as the security of opening Thunderdome is straightforward. In the following, α represents the regular profit parties receive from closing Thunderdome, ensuring that they are incentivized to close the virtual channel, and ε denotes the small value cost associated with unilateral closing, required when some parties are offline.

Definition 6 (Game-theoretic security)**.** A closing protocol is *game-theoretic secure* if each main party $n \in \{A, I, B\}$'s utility of the closing game $\mathcal{G}$'s history h^*, which is composed by the SPNE joint strategy σ^*, is no less than $\alpha - \varepsilon$, $u_n(h^*) \geq \alpha - \varepsilon$.

4 Protocol Overview

The core structure of Thunderdome is illustrated in Fig. 3. Despite its apparent simplicity, several challenges arise in securely designing the protocol: a) unlike the two-party TPC, Thunderdome involves three main parties, introducing the possibility of collusion between two parties (and the wardens) to defraud the third party; b) the wardens in Thunderdome can only be penalized in the TPC if they have deposited collateral, even though they may verify transactions that affect both underlying channels; and c) Thunderdome transactions are only known to Alice and Bob, leaving Ingrid unaware of the most recent state of the virtual channel. In the following, we present Thunderdome using a strawman approach.

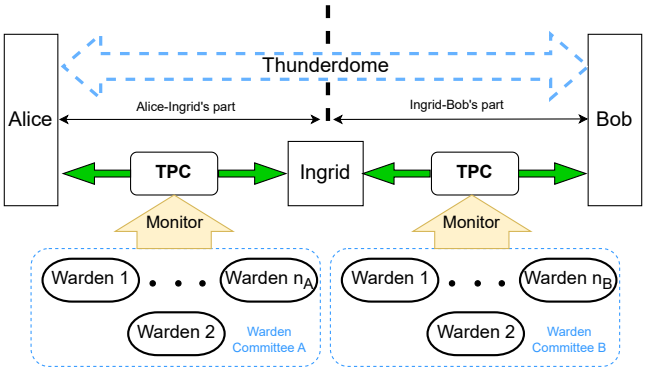

Figure 3: Thunderdome structure

Naive design. In a naive design of Thunderdome, we follow the same rationale as regular TPC protocols like Brick. When opening the virtual channel, the wardens of each TPC lock a portion of their collateral into Thunderdome. For example, if 2 out of 10 coins in a TPC are locked in the virtual channel, each warden locks 20% of their collateral during the funding transaction. The update phase of Thunderdome operates similarly to Brick. However, the closing phase differs during collaborative closing: Thunderdome cannot be closed solely by Alice and Bob; Ingrid's agreement is also required. For unilateral closing, the smart contract only requires a quorum certificate ($2f+1$ signatures) from the closing party's warden committee (e.g. C_A or C_B).

However, the security of this protocol can be easily compromised if two parties collude. Suppose Alice and Bob collude to cheat Ingrid. Since unilateral closing only requires the quorum certificate from each warden committee independently, Alice and Bob can create and broadcast two different update states, one to each warden committee, C_A and C_B. For instance, in one state, Alice holds all the coins, while in the other, Bob holds all the coins of the virtual channel. During the Thunderdome unilateral closing procedure, neither Alice nor Bob will publish any proof-of-fraud, even if the wardens from different committees submit different states on-chain, as each warden only signs one state and publishes it honestly. Consequently, Thunderdome will close with two different states in the underlying TPCs, causing Ingrid to lose coins. A straightforward solution would be to require the wardens to send their signatures to Ingrid as well. However, under asynchronous conditions, it becomes impossible to differentiate between a Byzantine warden who deliberately withholds the signature from Ingrid (but sends it to Alice) and an honest warden whose signature simply doesn't arrive in time.

Smart contract cross-checking. To prevent attacks and protect Ingrid in Thunderdome, we introduce a *smart contract cross-checking* scheme during the unilateral closing procedure. Specifically, after a smart contract receives enough states from wardens that have not been proven fraudulent, it notifies the other contract of the latest state it has learned and queries the other contract's latest state by sending a transaction. Both contracts then decide on the state with the higher sequence number, which will be used as the final closing state.

However, this approach remains vulnerable to certain safety attacks. For example, if Alice and Bob collude to generate two different states with the same sequence number, they can broadcast these states to their respective warden committees and initiate unilateral closing simultaneously. In the worst-case scenario, both smart contracts may send transactions to notify each other of the states they have received, and these transactions could be included in the same block, arriving simultaneously. If a naive scheme such as "only storing the state from the counterparty contract" is applied in this situation, the two parts of Thunderdome could still be closed with different states, resulting in potential losses for Ingrid.

Use leader contract to solve collision. To address the potential collision when transactions from two contracts arrive simultaneously, we designate one contract as the *leader contract*. In the event that two different states have the same sequence number, both contracts will store only the state from the *leader contract*. The leader contract must be agreed upon by all parties before the virtual channel is opened and should notify the corresponding smart contract as required.

Once we establish how smart contracts should handle states published by wardens, the next question is how many on-chain publications are necessary, given that asynchrony only guarantees eventual message delivery. If an updated state is confirmed with only $2f+1$ signatures from each warden committee, there could still be up to f (honest) wardens who have not received the latest state, even after the protocol was closed (incorrectly). To mitigate this, we require the smart contract to wait for at least $f+1$ publishments from the corresponding committee before closing the channel. This ensures that at least one warden with locked collateral knows the latest state.

However, if we consider rational, profit-maximizing players, Bob could simply bribe that specific warden and close the channel using an outdated state. Since the collateral held by each warden in TPC is much smaller than the total channel value (which could be equal to the value locked in Thunderdome), Bob could benefit by bribing the warden and closing the channel in a more favorable state. Thus, this protocol would not be secure in the rational model.

Ensuring enough collateral. A straightforward solution to this bribing problem is to require each warden to lock collateral equal to the full channel value. However, this approach is impractical due to over-collateralization. Additionally, either the funding transactions of the underlying TPCs would need to be topped up to support future virtual channels, or additional on-chain transactions would be required when a virtual channel opens. These implications are undesirable, so we aim for a solution that does not require additional collateral.

The first step is determining the necessary amount of collateral. Let the balance of Thunderdome be v, and assume the parties have locked v_i, $i \in \{A, I_A, I_B, B\}$ coins in the Alice-Ingrid and Ingrid-Bob payment channels, respectively. Since Ingrid participates in both channels, she locks coins in both, and the locked balances must satisfy the relationship: $v_A + v_B = v_{I_A} + v_{I_B} = v_A + v_{I_A} = v_{I_B} + v_B = v$. Any collusion set could potentially gain up to v coins. On the one hand, a single Byzantine party could cheat their counterparty and gain at most the number of coins locked by the counterparty, which is no more than the Thunderdome balance v. On the other hand, collusion between Alice and Bob could cheat Ingrid, who has locked $v_{I_A} + v_{I_B} = v$ coins, or Alice (or Bob) could collude with Ingrid to cheat Bob (or Alice), who has locked v_A (or v_B) coins, which are also less than or equal to v. Therefore, to guarantee security, we must ensure that the total collateral of wardens that can be punished in the event of fraud is no less than Thunderdome's balance v.

Thus, we require at least $2f+1$ wardens from each committee, C_A and C_B, to publish their closing state in Thunderdome's unilateral closing protocol. Combined with the quorum certificates required to update the Thunderdome state, we ensure that at most f wardens per committee can be slow. By waiting for $2f+1$ publications from accountable wardens, at least $f+1$ publications can be punished if they submit outdated information. Given that each warden locks v/f coins in Thunderdome, the total collateral that can be claimed per channel in case of fraud is $(f+1) \times \frac{v}{f} > v$.

5 Thunderdome Design

In this section, we present the detailed Thunderdome protocol, which consists of three operations: *Open*, *Update*, and *Close*. We assume two consecutive underlying TPCs: Alice-Ingrid's and Ingrid-Bob's. Each channel is managed by an on-chain smart contract, which handles the locked coins of both channel parties as well as the collateral of the wardens in the respective committee. The smart contract only has information about the specific payment channel it governs and has no knowledge of other payment channels or virtual channels.

5.1 Thunderdome Open

To initiate the opening of Thunderdome, agreement among all three main parties is required. As evidence for the channel opening, our protocol mandates that all three parties collaboratively generate a unique transaction, referred to as the "register transaction," TX_r. This transaction includes the signatures of all three parties, serving as proof of their unanimous consent to open the virtual channel. Additionally, TX_r contains essential information related to the virtual channel, such as its

initial state, total channel balance, and *contract information*, including the contract address and leader contract identifier. Under optimistic conditions, TX_r remains off-chain and is not published on the blockchain. However, in pessimistic scenarios, such as when some parties go offline, the remaining online parties may involve the payment channel smart contract to help close Thunderdome. In this case, TX_r will be submitted to the smart contract and used to facilitate decisions during the Thunderdome unilateral closing process when some parties are offline. Protocol 1 outlines the process of opening a Thunderdome channel.

Protocol 1 Thunderdome Open

Input: Parties A, I and B, TPC committees C_A and C_B, initial virtual channel state s_1, total channel balance v and contract information ci.

Result: Virtual channel register transaction TX_r and open a virtual channel

/*Combine wardens to form a virtual channel committee*/

1: A, I and B combine wardens from C_A and C_B to form virtual channel committee $C_V = C_A \cup C_B$.

/*Parties reach agreement on the open state*/

2: A and B generate two pre-register transaction with their signatures $TX'_r = \{pk_A, pk_B, pk_I, \{pk_{C_V}\}, s_1, v, ci, \sigma_A/\sigma_B\}$ and send to I.

3: After receiving two transactions, I responds with $TX''_r = \{pk_A, pk_B, pk_I, \{pk_{C_V}\}, s_1, v, ci, \sigma_A/\sigma_B, \sigma_I\}$.

4: A and B exchange responses of I and generate final register transaction with all partis' signatures $TX_r = \{pk_A, pk_B, pk_I, \{pk_{C_V}\}, s_1, v, ci, \sigma_A, \sigma_B, \sigma_I\}$.

5: All parties broadcast TX_r to wardens in virtual channel committee C_V, and receive more than t responses with wardens' signatures, then the virtual channel is considered to be opened.

6: A and I (I and B) update payment channels according to initial virtual channel state s_1.

First, all three parties—Alice, Ingrid, and Bob—must select a new committee for Thunderdome based on the two existing payment channel committees, C_A and C_B. A straightforward approach is to combine the two committees into one. While selecting a subset of wardens from each committee could reduce message complexity, it introduces additional security concerns, such as ensuring the safety of the selected virtual channel committee, which is beyond the scope of this paper. Therefore, to simplify the analysis, we assume the virtual channel committee C_V is the combined set of the two payment channel committees, C_A and C_B.

Once C_V is finalized, the three main parties collaboratively generate the register transaction TX_r. To ensure that all parties eventually have the same TX_r, or none at all, Alice and Bob first generate and send two pre-register transactions with their signatures to Ingrid. Upon receipt, Ingrid verifies the transactions match, and sends the new pre-register transaction back to Alice and Bob. Alice and Bob then exchange response messages to collect each other's signatures. Finally, all three parties obtain the complete TX_r, which includes the public keys of the wardens in C_V, the public keys of Alice, Bob, and Ingrid (pk_A, pk_B, and pk_I), the initial state s_1, and the signatures of the three parties (σ_A, σ_B, and σ_I). After generating the register transaction TX_r, the main parties broadcast it to all wardens in C_V for verification.

The final step in opening Thunderdome, after all parties reach an agreement, is to "virtually lock" the coins for the virtual channel. Unlike payment channels, which require publishing funding transactions on-chain, the parties of both underlying payment channels—Alice and Ingrid, and Ingrid and Bob—update their respective payment channels to lock coins for use in the virtual channel. For example, if the payment channel state between Alice and Ingrid is (a,b), and Alice wants to open a virtual channel with Bob with an initial state of (c,d), then Alice and Ingrid must update their payment channel state to $(a-c, b-d)$. This means Alice deposits c coins and Ingrid deposits d coins for the virtual channel. This update locks the coins $(c+d)$ for exclusive use in the virtual channel, while the wardens' collateral is allocated based on the balance of both the payment channel and the virtual channel. The virtual channel is considered open once the payment channel has been updated accordingly. It's important to note that opening and closing are the only operations that require changes to the underlying payment channels. Updates to Thunderdome and the payment channels can occur in parallel, as long as the coins locked for the virtual channel are not used during the payment channel update.

5.2 Thunderdome Update

The *update* procedure of Thunderdome is outlined in Protocol 2. After Alice and Bob agree on the new Thunderdome state s_i and sequence number i, they generate an announcement containing their signatures, $M = \{s_i, i, \sigma_A(s_i,i), \sigma_B(s_i,i)\}$, and broadcast it to all wardens in the Thunderdome committee C_V. To limit the influence of slow wardens in each committee, both parties must wait for the quorum certificate from each warden committee (C_A and C_B) before proceeding with the next state update. Although Byzantine or rational wardens could deviate from this protocol by broadcasting different states to wardens or by not waiting for sufficient responses, our closing protocol, which will be introduced in the next section, ensures that these deviations will not compromise the protocol's security.

5.3 Thunderdome Close

The final operation of Thunderdome is the *close*, which is initiated when at least one participant wishes to close the virtual

Protocol 2 Thunderdome Update

Main Parties

Input: Parties A and B, warden committee C_V comprising C_A and C_B, current state $E_{P_I}(s_i)$.

Result: Update Thunderdome to a new valid state.

1: Both parties A and B sign and exchange the new virtual channel state: $M = \{s_i, i, \sigma_A(s_i, i), \sigma_B(s_i, i)\}$.
2: A and B broadcast M to committee C_V. /*Along with a fee r*/
3: A (B) waits for a quorum certificate ($2f+1$ signatures) from each warden committee, then A (B) execute the virtual channel state s_i.

Wardens

Input: Parties A and B, warden committee C_V comprising C_A and C_B, sequence number i.

Result: C_V updates to a new valid state.

1: Each warden W_j, upon receiving M, verifies that both parties' signatures are present and the sequence number is exactly one higher than the previously stored sequence number. If the warden has published a state or has signed an M or M' with the same sequence number, it ignores the state update. Otherwise, W_j stores M (as a possible update, not yet replacing the $i-1$-th committed state), and sends its signature $\sigma_{W_j}(M)$ to both parties. /*Only to parties that paid the fee*/

channel. This can be optimistically executed by unlocking the previously virtually locked coins in the underlying TPCs through a simple channel update, provided that all parties are online and responsive. However, to ensure that coins are not locked indefinitely if some participants are unresponsive, Thunderdome includes a mechanism that allows any party to unilaterally initiate the closing of the virtual channel on-chain, using the wardens. In the following, we outline the different subroutines for the Thunderdome close operation and explain the rationale behind the design, starting from the most optimistic scenario.

5.3.1 Optimistic situation

In the optimistic scenario, all three parties—Alice, Bob, and Ingrid—are online and acting honestly. In this case, Thunderdome can be closed by simply updating the underlying payment channels. Alice and Bob first send closing requests to Ingrid. Once Ingrid receives matching announcements from both parties, she verifies that the requests align. The parties then collaborate to update their payment channels according to the final virtual channel state.

However, the optimistic closing procedure is insufficient on its own for several reasons. First, the closing requests from Alice and Bob may differ, leading to inconsistencies. Additionally, any of the three parties could be offline, preventing them from sending or responding to closing requests and resulting in a deadlock. To address these issues, we ensure the correct closing of Thunderdome by allowing any online party to unilaterally close the channel. This protocol handles cases where counterparty agreement cannot be obtained due to an asynchronous network, offline status, or Byzantine behavior. To facilitate unilateral closure in Thunderdome, we leverage the assistance of wardens, as described below. Crucially, the initiating party does not need to determine whether the counterparty is offline or if their message has not arrived; the mere absence of a response triggers the unilateral closing process.

Protocol 3 Thunderdome Closing protocol

Input: Closing party $P \in \{A, I, B\}$ and payment channel counter party $Q \in \{A, I, B\}$, closing request sent by party $m \in \{P, Q\}$: $VS_m = \{s_{i_m}, i, \sigma_m(s_{i_m}, i), \sigma_m(s_{i_m}, i)\}$.

Result: Close the virtual channel when all parties are online and responsive.

/*Parties send closing request*/

1: Closing party P sends closing requests to others.

/*Counterparty is offline or sends incorrect request*/

2: **if** Q is offline or $VS_Q \neq VS_P$ **then**
3: P runs Protocol 4 separately to close the channel with Q
4: **else**
5: P update the payment channel with Q
6: **end if**
7: If both parts of Thunderdome are closed, the virtual channel is successfully closed.

5.3.2 Pessimistic situation

Ingrid is offline or misbehaves. The first situation occurs when Ingrid is offline while Alice and Bob wish to close the virtual channel, or when Ingrid responds inconsistently to Alice's and Bob's honest closing requests. In this case, Thunderdome allows Alice and Bob to close the channel unilaterally, requiring them to interact with the smart contract, effectively offloading the virtual channel closure to the underlying payment channel. Protocol 4 outlines the steps from Alice's perspective.

The core idea of Protocol 4 is for the warden committee C_V to publish the latest channel state, enabling the closing party to punish malicious wardens by slashing their collateral in the smart contract. At the start of Protocol 4, the closing party submits the register transaction TX_r to the smart contract, allowing the payment channel contract to verify published states and proofs-of-fraud using the public keys of the Thunderdome parties and wardens from the respective committees included in TX_r. With $3f+1$ wardens in each committee, the closing party publishes proofs-of-fraud only after at least $2f+1$ wardens from their committee have published closing states, ensuring sufficient collateral for penalties.

As discussed in Section 5.2, it is possible for parties to

deviate from the update protocol. As a result, when Alice and Bob execute Protocol 4 simultaneously, the closing states published by the wardens could differ. To address this potential security issue, we require smart contracts to cross-check the states published by the wardens, as shown in Protocol 5. In this process, the two contracts exchange the latest state they have received and only retain the one with the higher sequence number. If two states have the same sequence number but different values, only the state from the *leader contract*, which is pre-determined in the register transaction TX_r during the channel opening, will be stored. Ultimately, Thunderdome will be closed using the stored state after cross-checking, if there are not enough fraud proofs to invalidate it.

In Thunderdome, parties can close the virtual channel unilaterally only by simultaneously closing the payment channel, as the latter becomes redundant if Ingrid does not respond to the Thunderdome closing request. Furthermore, Thunderdome enforces the virtual channel to be closed before the payment channel. To close both channels, parties first interact with the TPC's smart contract to verify the virtual channel's existence. The virtual channel is then closed through the combined warden committees, followed by the payment channel's closure. If a party attempts to close the payment channel without settling the virtual channel first, the Thunderdome balance is forfeited to the payment channel counterparty as a penalty.

Bob (Alice) is offline or misbehaves. The second scenario occurs when one party, such as Bob, is offline. In this case, Ingrid only receives the closing request from Alice. However, Ingrid cannot immediately agree to Alice's request, as she has no knowledge of the virtual channel state and could potentially be deceived if Alice and Bob are colluding. To address this, Ingrid is allowed to unilaterally close the part of Thunderdome involving the offline party first, thereby gaining access to the virtual channel state. Once Ingrid has this information, she can verify Alice's closing requests and act accordingly. In our game-theoretic analysis, we demonstrate that wardens will continue to publish honestly, even if Ingrid is initially unaware of the Thunderdome state.

Alice and Bob are both offline or collude. The final pessimistic situation covers two scenarios: (1) Ingrid wants to close Thunderdome, but both Alice and Bob are offline; (2) Alice and Bob send valid closing requests but with different closing states. The second scenario suggests that Alice and Bob are colluding during the Thunderdome update procedure, generating two different final states to deceive Ingrid. In both cases, Ingrid cannot rely on Alice and Bob to close Thunderdome properly. Therefore, Ingrid must execute Protocol 4 sequentially to close both parts of Thunderdome unilaterally. We stress that if Alice and Bob send identical but outdated closing requests, Ingrid incurs no loss by agreeing to them; so this scenario is excluded from our security analysis.

Ingrid and Bob (Alice) are offline. In the final possible scenario, which is not covered in Protocol 3, both Ingrid and Bob (or Alice) are offline during the closing process. In this case, the online party can execute Protocol 4 to close their part of Thunderdome unilaterally. Once the other two parties come back online, they can close their portion of Thunderdome either unilaterally or collaboratively.

Protocol 4 Unilateral closing

Input: Party A, wardens $W_1, \cdots, W_{n_A}$ from payment channel committee C_A, virtual channel state VS, virtual channel register transaction TX_r.

Result: Close the payment channel and one part of the virtual channel.

/*Alice registers virtual channel*/

1: A publishes the register transaction TX_r on-chain and prove the validity of virtual channel.

/*Wardens publish virtual channel states */

2: Each warden W_j publishes on-chain signature lists on the stored announcements for payment channel and virtual channel $\{VS, \sigma_{W_j}(VS)\}$.

/*Closing party submits proof-of-fraud*/

3: After verifying $2f+1$ on-chain signed announcements from committee C_A, A publishes proofs-of-fraud.

/*Closing with punishment*/

4: After the state is included in a block, the smart contract verifies the signature of VS based on the information in the register transaction TX_r.

5: Then the smart contract verifies the proofs-of-fraud:

(a) If the valid proofs about a channel are $x \leq f$, the smart contract follows Protocol 5 to decide the *wardens' published state*, WS_A. Then the smart contract virtually distributes coins by updating the parties' current balances according to WS_A.

(b) If the valid proofs about a channel are $x \geq f+1$, smart contracts award all channel balances to the counterparty.

6: Smart contracts award all the cheating wardens' collateral to party A.

7: First $2f+1$ wardens that have no cheating behaviors get an equal fraction of the closing fee.

8: If the virtual channel is not closed before closing the payment channel, the whole channel balance will be given to the payment channel counterparty.

5.4 Multi-hop Thunderdome

Thunderdome can also be extended to multi-hop scenarios. We illustrate this with example involving four parties: Alice, Bob, Charlie, and Dave, connected through TPCs (see Fig. 4).

The main difference in a multi-hop scenario is the involvement of additional intermediary parties who are unaware of the Thunderdome states. As with the two-hop Thunderdome, it is crucial to ensure that each committee has enough wardens

Protocol 5 Smart contract cross-checking

Sending transaction

Input: Alice-Ingrid smart contract SC_A, Ingrid-Bob smart contract SC_B, wardens' published state WS_A.

Result: Send notification and query transaction.

/*Decide state*/

1: SC_A stores the state with the highest sequence number as WS_A. If there is more than one, SC_A selects the first one it receives.

/*Exchange cross-checking transaction*/

2: SC_A sends the notification and query transaction TX_{qA} containning WS_A to SC_B.

3: SC_A waits for the transaction TX_{qB} from SC_B.

Receiving transaction

Input: Alice-Ingrid smart contract SC_A, Ingrid-Bob smart contract SC_B, wardens' published state WS_A.

Result: SC_A decides on the closing state.

/*Receive transaction from the other contract*/

1: SC_A receives transaction TX_{qB} containing WS_B from SC_B.

/*React when state is not decided yet*/

2: If SC_A hasn't deceided WS_A yet, then stores WS_B as WS_A and reply with $NULL$.

/*React when state is decided decided*/

3: If WS_A has already been decided, then SC_A reacts as follow:

(a) If WS_A and WS_B have different sequence number. Then SC_A updates WS_A to the one with the higher sequence number.

(b) If WS_A and WS_B have the same sequence number but different value. Then SC_A updates WS_A to the state from the *leader contract*.

who know the latest Thunderdome state. The key steps for a multi-hop Thunderdome are as follows: (1) During the opening procedure, all four parties, along with the wardens, must collaborate to generate the *register transaction*. Each party considers the channel open after receiving a quorum certificate from each committee. (2) To successfully update Thunderdome, Alice and Dave must broadcast the updated state to all wardens and wait for a quorum certificate from each committee. (3) Thunderdome can be optimistically closed once all four parties reach an agreement. In a pessimistic closing scenario, each smart contract should wait for at least $2f+1$ wardens to publish their states from the corresponding payment channel committee and cross-check with other contracts to determine the final closing state.

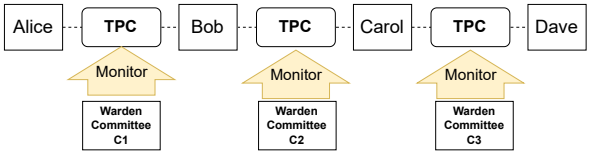

Figure 4: Multi-hop Thunderdome

6 Byzantine Security Analysis

We prove that Thunderdome is secure under the Byzantine model, provided at least $t = 2f+1$ wardens remain honest in each committee, C_A and C_B. We then extend our analysis to multi-hop Thunderdome. Due to space constraints, we present only sketch proofs below, with full proofs available in the extended version [11].

Balance Security. According to Protocol 5, the contracts governing both parts of Thunderdome always close with the same state, ensuring balance security for Ingrid. We thereby analyze two possible closing scenarios: (i) In collaborative closing, all three main parties (Alice, Ingrid, and Bob) must agree. Hence, collaborative closing cannot succeed with an outdated state, as long as at least one party is honest. (ii) In unilateral closing, any main party can close the channel, but this requires $2f+1$ signatures from the respective warden committee. If either Alice or Bob is honest, at least $2f+1$ wardens in each committee will be notified by Protocol 2, ensuring that at least one honest warden will publish the latest state. Even in the worst-case scenario where both Alice and Bob are Byzantine, the cross-checking mechanism guarantees Ingrid's security by preventing state inconsistencies between the two parts of Thunderdome.

Theorem 1. *Thunderdome achieves **balance security** for honest parties under asynchrony, assuming at most f Byzantine wardens in each committee.*

Liveness. Liveness can fail if an insufficient number of wardens are available to process a valid request. However, with at most f Byzantine wardens in each committee, every valid request will always be processed by at least $2f+1$ honest wardens, ensuring execution. Thus, valid operations always proceed, and Byzantine parties cannot indefinitely stall progress.

Theorem 2. *Thunderdome achieves **liveness** for honest parties under asynchrony, assuming at most f Byzantine wardens in each committee.*

Multi-hop Thunderdome. The same guarantees hold for multi-hop Thunderdome. As long as at least one main party is honest and each committee contains at least $2f+1$ honest wardens, invalid operations cannot succeed, and valid operations will always execute. Additionally, the cross-checking mechanism ensures state consistency across all parts of the multi-hop Thunderdome.

Theorem 3. *Multi-hop Thunderdome achieves* ***balance security*** *for honest parties under asynchrony, assuming at most f Byzantine wardens in each committee.*

Theorem 4. *Multi-hop Thunderdome achieves* ***liveness*** *for honest parties under asynchrony, assuming at most f Byzantine wardens in each committee.*

7 Game-theoretic Security Analysis

We prove Thunderdome is game-theoretically secure (Def. 6) by showing that no party can increase their utility by deviating from the protocol specification during the open, update, or close phases. While Thunderdome comprises three phases, the update phase is effectively reflected in the closing process. Therefore, we model the opening and closing procedures as an EFG to prove the scheme's security.

The correctness of the Thunderdome open procedure is ensured by the fact that if the two parts of Thunderdome are initialized inconsistently, certain state transitions become impossible due to channel balance constraints. This imbalance disrupts state consistency, ultimately causing a financial loss for Ingrid. As a rational participant, Ingrid will thus reject any inconsistent initialization of Thunderdome.

We assume the whole Thunderdome closing process is initiated by Alice and Bob. According to Protocol 3, the closing of Alice-Ingrid's part and Ingrid-Bob's part are symmetric; for simplicity, we only show the game involving Bob and Ingrid. The possible strategies for these two parties are defined in Table 3 and 4. Depending on whether Ingrid knows the Thunderdome state, there could be two different game trees depicted in Fig. 6 and 7. The blue dotted circle in Fig. 6 represents that the parties inside the circle share the same information set [32]. Specifically, if Bob sends a closing request while Alice is offline, Ingrid cannot determine whether the request is correct, which makes the game an EFG with imperfect information. Conversely, if Alice also sends a closing request or Ingrid gains knowledge of the virtual channel through unilateral closing, the closing game changes to an EFG with perfect information, as shown in Fig. 7.

Subgame 1 models the unilateral closing game between the closing party and the wardens, as illustrated in Fig.5. We consider a setting with 10 wardens in each committee, with their strategies detailed in Tables 1 and 2. The wardens choose their strategies simultaneously. By comparing utilities, we establish that the SPNE strategy for wardens is to publish honestly, avoiding penalties on their collateral. This outcome extends to Ingrid, as wardens cannot ascertain whether she has obtained information from other parties and can impose penalties. Notably, this game also applies to the update protocol, where wardens decide whether to sign honestly. Additionally, the cross-checking scheme ensures that both parts of Thunderdome can only be unilaterally closed with the same state, guaranteeing Ingrid's security and incentivizing the other two rational parties to honestly follow the update protocol. The reward k here is used as an incentive to guarantee *liveness* from the perspective of wardens.

Table 1: Strategy for unilateral closing party (P)

PF	Number of valid proofs-of-fraud she publishes on-chain

Table 2: Strategy for warden i (W_i)

P_l	Publish the honest state \ Sign honestly
P_o	Publish the dishonest state \ Sign dishonestly

Table 3: Strategy for Bob (B)

Old	Send a dishonest collaborative closing request with the old virtual channel state
New	Send an honest collaborative closing request with the latest virtual channel state
Uni	Unilaterally close the part of Thunderdome with Ingrid

Table 4: Strategy for Ingrid (I)

$Agree$	Agree and sign the closing request
$Disagree$	Disagree and go for unilateral closing
$Ignore$	Ignore the closing request or go offline

By incorporating the SPNE utility from Subgame 1 into the closing game illustrated in Fig.6 and Fig.7, we analyze the SPNE strategies in both cases. The SPNE of an extensive-form game (EFG) with imperfect information is equivalent to the Nash Equilibrium (NE) strategy of the normal-form game (NFG) derived from it [32]. Thus, to determine the SPNE, we first transform the closing game with imperfect information into its corresponding NFG, represented in Table 5, where the NE strategy is highlighted in blue.

Following this transformation, Ingrid's optimal SPNE strategy is to initially ignore Bob's closing request when lacking sufficient knowledge. Two possible outcomes then emerge: (1) If Bob aims to expedite the closure of his part of Thunderdome, he may unilaterally initiate Protocol 4. From the analysis of Subgame 1, Ingrid will not incur any loss in this scenario. (2) Alternatively, Ingrid may acquire knowledge of Thunderdome through either Alice's closing request or by closing the Alice-Ingrid part of Thunderdome. If, at this point, the Ingrid-Bob part remains open, the game transitions into an EFG with perfect information, as shown in Fig. 7. By evaluating the utilities in this game, we conclude that the unique SPNE outcome is for all parties to close their respective parts of Thunderdome with the latest state.

In case both Alice and Bob are offline, Ingrid must unilaterally close the Alice-Ingrid and Ingrid-Bob parts by executing Protocol 4. As previously established, Ingrid remains protected from loss during unilateral closing. Furthermore, Ingrid

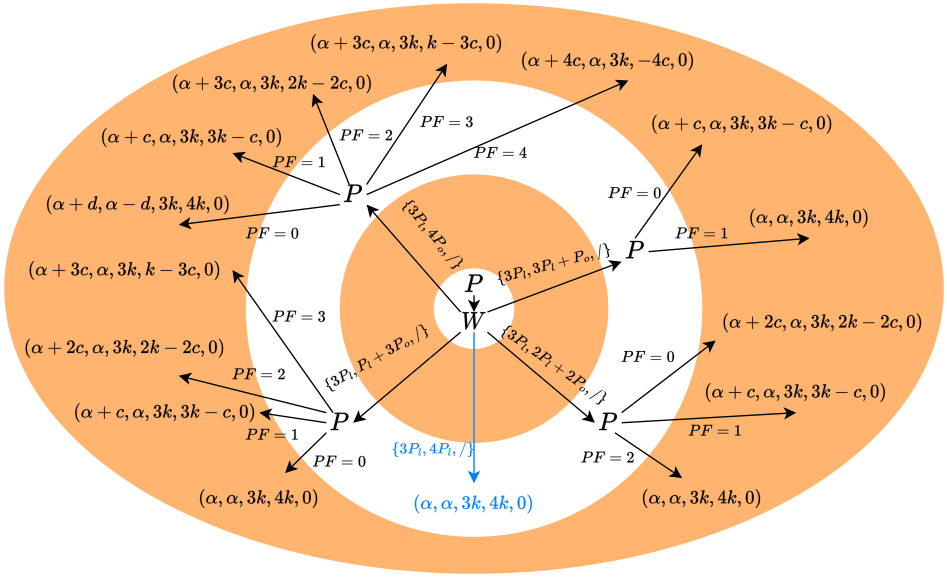

Figure 5: Subgame 1 – Unilateral closing game started by party P, $P \in \{A,B,I\}$. The utility is shown in order of $\{P, P$'s counterparty, warden group 1 (3 wardens), warden group 2 (4 wardens), warden group 3 (3 wardens)$\}$. α represents the default value for closing, c for warden collateral, d for profit of incorrect closing and k for reward of correct warden publishment. The SPNE of the game is shown with the blue arrow.

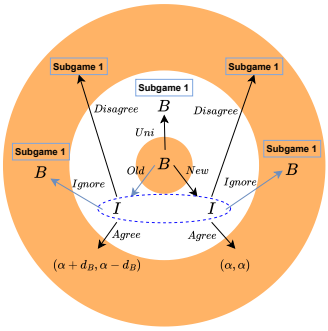

Figure 6: Closing game started by Bob. Utilities are in the form of (Bob, Ingrid), where α is the incentive for closing Thunderdome, and d_B is Bob's profit for incorrect closure. The SPNE utility of subgame 1 is $(\alpha-\varepsilon,\alpha)$, with ε representing a small preference cost. The SPNE strategy is indicated by the blue arrow, and the blue-dotted circle denotes a shared information set.

	Ignore	*Agree*	*Disagree*
Uni	$(\alpha-\varepsilon,\alpha)$	$(\alpha-\varepsilon,\alpha)$	$(\alpha-\varepsilon,\alpha)$
Old	$(\alpha-\varepsilon,\alpha)$	$(\alpha+d_B,\alpha-d_B)$	$(\alpha,\alpha-\varepsilon)$
New	$(\alpha-\varepsilon,\alpha)$	(α,α)	$(\alpha,\alpha-\varepsilon)$

Table 5: Transformed NFG of Bob's closing game with imperfect information.

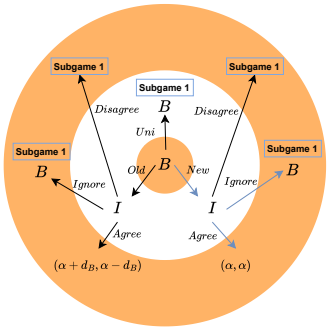

Figure 7: Closing game started by Bob; Ingrid gains knowledge about Thunderdome, distinguishing Bob's requests. α represents the incentive for closing Thunderdome, and d_B represents Bob's profit for close Thunderdome incorrectly. The SPNE utility of subgame 1 is $(\alpha-\varepsilon,\alpha)$, where ε is a small value of cost to denote preference. The SPNE strategy is shown by the blue arrow.

cannot exploit Alice or Bob, as the cross-checking mechanism ensures that both parts of Thunderdome close with identical states. We formally establish the game-theoretic security of Thunderdome in Theorem 5.

Theorem 5. *Thunderdome closing protocol is game-theoretic secure.*

The multi-hop Thunderdome remains secure in the game-

theoretic model, as any closing game in multi-hop Thunderdome can be reduced to the two-hop case.

Channel open: Rational intermediary main parties will prevent potential losses from incorrect state initialization, similar to the two-hop Thunderdome.

Channel update: The unilateral closing subgame remains unchanged for multi-hop Thunderdome. Therefore, rational wardens will follow the update protocol honestly to avoid penalties, while the cross-checking scheme prevents rational main parties from misbehaving.

Channe closing: A key observation is that when a party, such as Dave, initiates the closure of a multi-hop Thunderdome, the closing game effectively involves only Dave and Carol, despite multiple intermediaries. This is because Dave's objective is to modify the underlying payment channel with Carol, which requires either Carol's consent or the wardens' intervention. Additionally, the cross-checking scheme ensures that all parts of Thunderdome close with the same state, preventing rational parties from colluding to close Thunderdome inconsistently. Since the presence of intermediaries does not alter the game tree or utility outcomes, the analysis for two-hop Thunderdome extends naturally to the multi-hop setting.

Theorem 6. *The multi-hop Thunderdome is game-theoretic secure.*

8 Implementation and Evaluation

In this section, we present a proof-of-concept implementation of Thunderdome. The smart contract for the channel is written in Solidity V0.8.21 and deployed using Truffle V5.11.4 on a private Ethereum testing blockchain set up with Ganache V7.9.1. The source code is publicly available at https://github.com/BartWaaang/Thunderdome. We evaluate the gas fee cost of smart contracts on Ethereum as our primary metric. Gas fees are paid by blockchain users to miners as transaction fees, which depend on both the amount of data and the complexity of operations in the transaction. In our implementation, we set the per unit gas price to 2×10^{-9} ETH, a typical setting. Additionally, we set the exchange rate between ETH and USD at 1668 : 1, based on the latest rate at the time of writing. Under these conditions, the total cost to deploy the payment channel contract supporting our Thunderdome protocol is approximately 0.006 ETH (3,174,554 gas, around 10.59 USD). We evaluate the cost for each procedure using an evaluation script written in JavaScript, which generates testnet accounts for the main parties and wardens, simulating their actions. The gas fee costs for various procedures in our protocol are summarized in Table 6.

We compare our gas fee costs with Perun, as shown in Table 6. To deploy and open an underlying payment channel (PC), Alice (Bob) and Ingrid, along with 10 wardens, send funding transactions on-chain. The total cost for Thunderdome amounts to 0.0089 ETH, which is 1.5 times the cost of

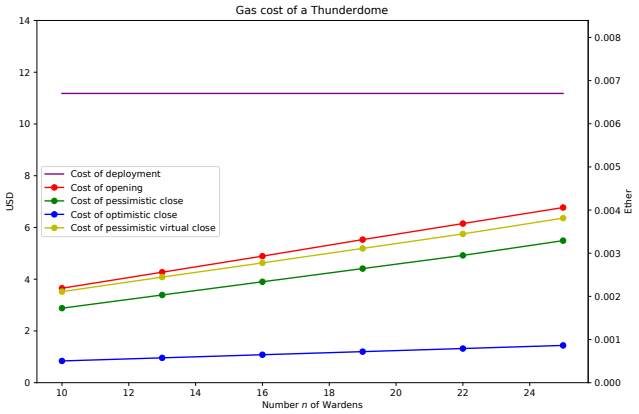

Figure 8: Gas fee changes in Thunderdome's procedures as the number of wardens per committee grows from 10 to 25.

Perun, primarily due to the additional funding transactions required from the wardens. Given the need for extra funding transactions and the removal of timelocks, this increase is reasonable. Regarding off-chain communication, Thunderdome requires quorum certificates from committees to open the channel. Additionally, updating the payment channel for Thunderdome's "virtual lock" involves two rounds of broadcasting, leading to higher message complexity compared to updating a standard payment channel. To close the payment channel optimistically, Alice and Ingrid must publish the closing state agreed upon by both parties on-chain, costing 0.0005 ETH. The pessimistic closing, where at least 7 wardens publish the state on-chain after one party (e.g., Alice) submits a closing request, costs approximately 0.0017 ETH. The message cost for the optimistic closing of the virtual channel is the same as for a payment channel update. In pessimistic situations, closing the virtual channel requires the closing party to submit the register transaction to the smart contract and wardens to publish Thunderdome states on-chain, costing 0.0024 ETH — 3 times the cost of Perun. This additional closing cost arises because all wardens must publish on-chain transactions for the closing state, similar to the Thunderdome opening process. However, given that pessimistic scenarios like censorship are infrequent and the difference in optimistic payment channel closing is only around 0.4 USD, Thunderdome remains comparable to Perun in most cases. We also evaluate how gas fees change as the size of each warden committee increases from 10 to 25. Naturally, costs for all procedures rise with more wardens, as more participants need to interact with the smart contract during Thunderdome execution. The results are illustrated in Fig. 8.

9 Extensions

Privacy-preserving protocol. In Thunderdome, the privacy of virtual channel states is not protected from the wardens.

	On-chain transaction	off/on-chain message			Thunderdome Cost			Perun Cost		
		Alice (Bob)	Ingrid	Warden (10)	Gas	ETH	USD	Gas	ETH	USD
Deploy & Open PC	2+10	1+10 / 1	1+10 / 1	$14 \leq m \leq 20$ / 10	4444861	0.0089	14.83	2819448	0.0057	9.54
Update PC	0	1+10 / 0	1+10 / 0	$14 \leq m \leq 20$ / 0	0	0	0	0	0	0
Open VC	0	3+30 / 0	2+40 / 0	$35 \leq m \leq 50$ / 0	0	0	0	0	0	0
Update VC	0	1+20 / 0	0 / 0	$14 \leq m \leq 20$ / 0	0	0	0	0	0	0
Optimistic VC close	0	1+10 / 0	1+10 / 0	$14 \leq m \leq 20$ / 0	0	0	0	0	0	0
Optimistic PC close	2	1 / 1	1 / 1	0 / 0	252760	0.0005	0.84	147788	0.0003	0.49
Pessimistic VC close by Alice	$2+7 \leq m \leq 2+10$	0 / 2	0 / 0	$0 / 7 \leq m \leq 10$	1217307	0.0024	4.06	418318	0.0008	1.40
Pessimistic PC close by Alice	$1+7 \leq m \leq 1+10$	0 / 1	0 / 0	$0 / 7 \leq m \leq 10$	862459	0.0017	2.88	275049	0.0006	0.92

Table 6: Gas and communication cost for different procedures in Thunderdome and Perun [18]. Warden refers to a single payment channel warden committee. PC denotes a payment channel and VC denotes a virtual channel. The column "on-chain" transaction counts transactions sent by the main parties (before "+") and by wardens (after "+"). The column "off/on-chain message" counts the off-chain messages among the main parties (before "+") and to wardens (after "+"), as well as the on-chain messages.

It is not straightforward to design a privacy-preserving protocol that ensures both: a) Ingrid can recover the state, and b) the wardens cannot retrieve the state, even in collusion with Ingrid, while still being able to verify the validity of the private state. A compromise is to enable state recovery and privacy against wardens, assuming Ingrid does not collude with them. However, even in this case, the repetitive nature of updates poses challenges. Trapdoor one-way functions, though promising candidates, lack IND-CPA security. On the other hand, asymmetric encryption schemes offering such protection require a new random number for each encrypted value [12]. In our protocol, this can be implemented by Alice, Bob, and Ingrid sharing a vector of random values during the creation of the register transaction in Thunderdome open. However, this solution has two main drawbacks: a) the number of states that can be updated in the virtual channel is limited by the length of this vector. Optimistically, Ingrid may occasionally renew it, but this depends on Ingrid being online and cooperative; b) the virtual channel parties must provide the wardens with corresponding evidence (e.g., ZK proof) at each update to certify that the correct random number was used, adding communication and computational overhead to Thunderdome's update process. Other cryptographic tools, such as Verifiable Secret Sharing [37] or threshold signatures [35], could be leveraged to enhance privacy in Thunderdome transactions. For example, one could explore the possibility of allowing Ingrid and $f+1$ wardens from her committee to recover the states. This scheme may be secure under the assumption of $2f+1$ honest wardens, but it remains an open question whether suitable incentive mechanisms can ensure security in the rational model.

Scaling the multi-hop protocol. An interesting question is whether it is feasible to scale the multi-hop Thunderdome by involving fewer wardens. We argue that this is possible by defining the employed wardens and the closing rules during the opening phase. Assuming that only $n' < n$ wardens are selected per committee, two necessary compromises arise: (1) *Selection algorithm*: Among $n' = 3f' + 1$ wardens, there should still be at most f' Byzantine wardens; (2) *Extra deposit*: If the multi-hop Thunderdome channel balance is v, then each warden must lock collateral of at least $\frac{v}{f'} > \frac{v}{f}$. However, the validity rule (e.g., requiring a quorum certificate per committee during the update procedure) remains unchanged for two key reasons. First, the security model assumes that the two end parties of the multi-hop Thunderdome could collude, necessitating warden committees to ensure there are no two conflicting valid states. Second, intermediary parties without knowledge of the virtual channel state rely on the wardens in the local committee to publish the correct state.

Extend to state channel. Thunderdome is introduced as a virtual payment channel but can also be extended to function as a virtual state channel. Although numerous state channel primitives exist in the literature [19, 27–29] that generalize payment channels to more complex applications, these constructions cannot be directly applied to Thunderdome due to their synchronous nature, which relies on timelocks. However, Thunderdome can utilize two Brick state channels [5] as the underlying layer, provided that each state can be mapped to a monetary valuation to ensure security in the game-theoretic model. This assumption, as proposed in [5], is essential for converting the payment channel primitive into a rationally secure state channel, which Thunderdome also adopts.

References

[1] Txwithhold smart contracts by gleb naumenko, 2022.

[2] Lukas Aumayr, Kasra Abbaszadeh, and Matteo Maffei. Thora: Atomic and privacy-preserving multi-channel updates. In *Proceedings of the 2022 ACM SIGSAC*

Conference on Computer and Communications Security, pages 165–178, 2022.

[3] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. In *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security,*, volume 13091 of *Lecture Notes in Computer Science*, pages 635–664. Springer, 2021.

[4] Lukas Aumayr, Matteo Maffei, Oğuzhan Ersoy, Andreas Erwig, Sebastian Faust, Siavash Riahi, Kristina Hostáková, and Pedro Moreno-Sanchez. Bitcoin-compatible virtual channels. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 901–918. IEEE, 2021.

[5] Georgia Avarikioti, Eleftherios Kokoris-Kogias, and Roger Wattenhofer. Brick: Asynchronous state channels. corr abs/1905.11360 (2019). *arXiv preprint arXiv:1905.11360*, 2019.

[6] Georgia Avarikioti, Felix Laufenberg, Jakub Sliwinski, Yuyi Wang, and Roger Wattenhofer. Towards secure and efficient payment channels. *arXiv preprint*, 2018.

[7] Zeta Avarikioti, Pawel Kedzior, Tomasz Lizurej, and Tomasz Michalak. Bribe & Fork: Cheap bribing attacks via forking threat. *Advances in Financial Technologies (AFT)*, 2024.

[8] Zeta Avarikioti, Eleftherios Kokoris-Kogias, Roger Wattenhofer, and Dionysis Zindros. Brick: Asynchronous incentive-compatible payment channels. In *Financial Cryptography and Data Security (FC)*, pages 209–230. Springer, 2021.

[9] Zeta Avarikioti and Orfeas Stefanos Thyfronitis Litos. Suborn channels: Incentives against timelock bribes. In *International Conference on Financial Cryptography and Data Security*, pages 488–511. Springer, 2022.

[10] Zeta Avarikioti, Orfeas Stefanos Thyfronitis Litos, and Roger Wattenhofer. Cerberus channels: Incentivizing watchtowers for bitcoin. In *Financial Cryptography and Data Security (FC)*, pages 346–366. Springer, 2020.

[11] Zeta Avarikioti, Yuheng Wang, and Yuyi Wang. Thunderdome: Timelock-free rationally-secure virtual channels. *arXiv: 2501.14418*, 2025.

[12] Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In *Advances in Cryptology–ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security*, pages 232–249. Springer, 2009.

[13] Same Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, et al. Sui lutris: A blockchain combining broadcast and consensus. *arXiv preprint arXiv:2310.18042*, 2023.

[14] Hao Chung, Elisaweta Masserova, Elaine Shi, and Sri Aravinda Krishnan Thyagarajan. Rapidash: Improved constructions for side-contract-resilient fair exchange. *IACR Cryptol. ePrint Arch.*, 2022:1063, 2022.

[15] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains: (a position paper). In *Financial Cryptography and Data Security (FC)*, pages 106–125. Springer, 2016.

[16] George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and tusk: a dag-based mempool and efficient bft consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 34–50, 2022.

[17] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Stabilization, Safety, and Security of Distributed Systems - International Symposium, SSS*, volume 9212 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2015.

[18] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual payment hubs over cryptocurrencies. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 106–123. IEEE, 2019.

[19] Stefan Dziembowski, Sebastian Faust, and Kristina Hostáková. General state channel networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 949–966, 2018.

[20] Oguzhan Ersoy, Jérémie Decouchant, Satwik Prabhu Kumble, and Stefanie Roos. Syncpcn/psyncpcn: Payment channel networks without blockchain synchrony. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pages 16–29, 2022.

[21] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. *J. ACM*, apr 2024. Just Accepted.

[22] Karan Singh Garewal and Karan Singh Garewal. The helium cryptocurrency project. *Practical Blockchains and Cryptocurrencies: Speed Up Your Application Development Process and Develop Distributed Applications with Confidence*, pages 69–78, 2020.

[23] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In *Financial Cryptography and Data Security (FC)*, pages 201–226. Springer, 2020.

[24] Joshua Lind, Oded Naor, Ittay Eyal, Florian Kelbert, Emin Gün Sirer, and Peter Pietzuch. Teechain: a secure payment network with asynchronous blockchain access. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 63–79, 2019.

[25] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *Network and Distributed System Security (NDSS) Symposium*, 2019.

[26] Subhra Mazumdar, Prabal Banerjee, Abhinandan Sinha, Sushmita Ruj, and Bimal Kumar Roy. Strategic analysis of griefing attack in lightning network. *IEEE Trans. Netw. Serv. Manag.*, 20(2):1790–1803, 2023.

[27] Patrick McCorry, Surya Bakshi, Iddo Bentov, Sarah Meiklejohn, and Andrew Miller. Pisa: Arbitration outsourcing for state channels. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 16–30, 2019.

[28] Patrick McCorry, Chris Buckland, Surya Bakshi, Karl Wüst, and Andrew Miller. You sank my battleship! a case study to evaluate state channels as a scaling solution for cryptocurrencies. In *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC*, pages 35–49. Springer, 2020.

[29] Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. Sprites and state channels: Payment networks that go faster than lightning. In *Financial Cryptography and Data Security (FC)*, volume 11598 of *Lecture Notes in Computer Science*, pages 508–526. Springer, 2019.

[30] Tejaswi Nadahalli, Majid Khabbazian, and Roger Wattenhofer. Timelocked bribing. In *Financial Cryptography and Data Security*, pages 53–72. Springer, 2021.

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

[32] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[33] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. 2016.

[34] Sophie Rain, Georgia Avarikioti, Laura Kovács, and Matteo Maffei. Towards a game-theoretic security analysis of off-chain protocols. In *2023 2023 IEEE 36th Computer Security Foundations Symposium (CSF)*, pages 31–46. IEEE Computer Society, 2022.

[35] Victor Shoup. Practical threshold signatures. In *Advances in Cryptology – EUROCRYPT: International Conference on the Theory and Application of Cryptographic Techniques*, pages 207–220. Springer, 2000.

[36] Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. Bullshark: Dag bft protocols made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2705–2718, 2022.

[37] Markus Stadler. Publicly verifiable secret sharing. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 190–199. Springer, 1996.

[38] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. Post-quantum adaptor signature for privacy-preserving off-chain payments. In *Financial Cryptography and Data Security (FC)*, volume 12675 of *Lecture Notes in Computer Science*, pages 131–150. Springer, 2021.

[39] Sri Aravinda Krishnan Thyagarajan, Adithya Bhat, Giulio Malavolta, Nico Döttling, Aniket Kate, and Dominique Schröder. Verifiable timed signatures made practical. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1733–1750, 2020.

[40] Sri Aravinda Krishnan Thyagarajan, Giulio Malavolta, Fritz Schmid, and Dominique Schröder. Verifiable timed linkable ring signatures for scalable payments for monero. In *Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security*, volume 13555 of *Lecture Notes in Computer Science*, pages 467–486. Springer, 2022.

[41] Itay Tsabary, Matan Yechieli, Alex Manuskin, and Ittay Eyal. Mad-htlc: because htlc is crazy-cheap to attack. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1230–1248. IEEE, 2021.

[42] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International journal of web and grid services*, 14(4):352–375, 2018.