PCMCPS: A Secure Privacy-Preserving Cross Multi-Party Channel Payment Scheme In Payment Channel Networks

***Abstract*—As digital currencies continue to evolve, blockchain scalability has gradually emerged as a major constraint. Payment channel networks partially address this issue by connecting multiple channels. However, when payment channels expand from bilateral to multilateral arrangements, the resulting challenges—particularly privacy concerns—cannot be overlooked. To address the above challenges, we propose a Privacy-Preserving Cross Multi-party Channel Payment Scheme (PCMCPS) based on Paillier cryptography. This scheme employs a Watchtower as a third-party supervisor to oversee cross-channel payments, utilizes redundant lookup tables as Watchtower's storage structure, and leverages the homomorphic properties of Paillier cryptosystems to implement balance updates. We have proven the protocol's security in Universal Composable framework. Compared to privacy-preserving payment schemes based on zk-SNARKs, PCMCPS exhibits significantly lower overhead (approximately 70% of zk-PerHop at a key size of 3072 bits, and 22% with 2048 bits at 10 hops), further demonstrating its feasibility.**

***Index Terms*—blockchain, payment channel, privacy-preserving, Paillier cryptography, smart contract**

I. Introduction

Blockchain-based payment systems have seen rapid adoption in recent years, driven by the need for fast, low-fee, and scalable digital transactions. According to Chainalysis [1], on-chain crypto value received in the Asia-Pacific region grew from approximately US$1 billion per month in mid-2022 to a peak of US$244 billion per month in December 2024, underscoring the massive scale and rapid growth of blockchain-based payments. While on-chain payments provide strong security guarantees, their throughput and latency limitations prevent them from supporting high-volume, real-time payment workloads. To address this issue, off-chain Payment Channel Networks (PCNs) — most notably the Lightning Network (LN) [2] on Bitcoin and Raiden Network (RN) [3] on Ethereum — have emerged as the leading Layer-2 solution.

Recent empirical measurements of the LN indicate that redundancy in channel topology incurs substantial hidden costs. Studies show that more than half of all public channels appear to be inactive [4] — opened but never used to forward a single payment — while only around 8–9% of channels exhibit frequent fee or state updates [5], a strong proxy for active routing participation. Such skewed utilization implies that a large fraction of the network's locked liquidity, on-chain channel-opening fees, and ongoing maintenance costs are effectively spent on redundant connectivity that contributes little to overall throughput. This structural inefficiency stems from the fact that current PCN model relying on channels restrict to participation by both parties, forcing users to open and maintain separate bilateral links even when their traffic patterns are sparse or unpredictable. As the network scales, the cost of maintaining many underutilized channels grows disproportionately.

To overcome these limitations, recent researches has explored Multi-Party Payment Channels (MPPCs) [6], which allow multiple users to establish a "large channel" capable of accommodating payments from multiple parties. Inheriting the advantages of two-way channels, like LN and RN, in enhancing scalability: multiple payments from channel participants are aggregated into a single transaction on-chain [6]–[8]; simultaneously, it enables more complex payment and coordination patterns (such as multi-party-to-multi-party payments) [9], [10]. And the reduction in the number of channels further lowers the overhead of channel establishment and closure [11]. The introduction of MPPC also brought issues such as centralized oversight authority and excessive redundant user overhead within channels. To the best of our knowledge, the SMPC scheme proposed in [12] is the only work that attempts to address these problems. By designing a dual-supervisory user mechanism and a reverse supervision mechanism, [12] enhanced payment security within channels while strengthening distributed characteristics; the introduction of a channel threshold enables the periodic removal of inactive users.

Nevertheless, there are several limitations in [12]. Firstly, it does not consider scenarios involving multiple MPPCs — users may have payment needs with users of other channels. Integrating multiple MPPCs — each accommodating multiple users — into PCNs presents significant challenges. Existing PCNs are connected via multiple two-way channels, enabling cross-channel payments without complex operations. In MPPCs, however, increased user numbers lead to a geometric increase in complex operations. Moreover, resisting malicious double-spend attacks becomes significantly harder when multiple payments occur concurrently. More critically, channel overhead inevitably rises with the number of channels, making the implementation of an efficient payment scheme highly challenging.

Secondly, it lacks thorough consideration of privacy protection. The payment reliability in [12] relies entirely on its designed supervisory mechanisms, yet completely overlooks the privacy security of users and payments themselves. In a PCN scenario, the increase in payment participants means

more privacy to protect and more potential points of privacy leakage. Furthermore, cross-channel payments cannot simply rely on a single supervisory user within a channel; they also require a third-party entity capable of maintaining and overseeing the cross-channel payment process. Given that third-party entities are not fully trustworthy, establishing mechanisms to ensure user privacy is not disclosed to them and even resisting their statistical analysis of payment processes is also extremely challenging. Addressing this challenge entails simultaneously safeguarding user privacy—specifically identity privacy and balance privacy—while also protecting payment privacy (such as payment amount, payment time, or other details pertaining to a specific payment). Therefore, for a PCN composed of multiple MPPCs, maintaining existing supervisory capabilities without compromising privacy represents a significant challenge.

Motivated by these challenges, we propose a secure Privacy-preserving Cross Multi-party Channel Payment Scheme (PCMCPS), the first cross-channel payments scheme achieving balance between regulations and privacy protection based on Paillier cryptosystem [13]. The homomorphic property of the Paillier cryptosystem enables untrusted third parties to update balances across channels without decryption, while simultaneously safeguarding privacy against disclosure. For the aforementioned first challenge, following that in [12], we retain the supervisory user within each channel, who acts as an intermediate user agent within their respective channel. But we separate authorization for request processing from authorization for balance updates, enabling unified execution of operations such as fund retention and balance updates. Since supervisory users across channels can process payment requests in parallel, this significantly reduces time overhead, and this overhead does not increase with the count of hops. For the second challenge, we introduce the Watchtower as an untrusted third-party regulator and designed a redundant lookup table as its storage structure. User identity information and balances are encrypted and redundantly stored in the lookup table, ensuring the Watchtower cannot directly access user identity or balance privacy, while realizing the amount updates through homomorphic properties of Paillier cryptosystems. By configuring a guide set to obfuscate the Watchtower's processing targets, we protect payment privacy and resist statistical analysis attacks.

The main contributions are as follows:

- Based on the work in [12], we extend the off-chain payment process from a single MPPC to PCNs and optimize the cross-channel payment flow, thereby achieving efficient cross-channel payment functionality.
- Maintaining the previously established supervision of payments within channels, we introduce an untrusted third-party Watchtower as the supervisory entity within the PCN. Concurrently, we design a redundant lookup table as its storage structure and develop a user lookup and balance homomorphic update scheme based on the Paillier cryptosystem. This achieves supervisory capability for cross-channel payments while preserving privacy.
- We implemented PCMCPS within the Universal Composability (UC) framework and demonstrated its security.[1] Experiments comparing the overhead with mainstream zk-SNARK privacy-preserving schemes like [14] revealed significant efficiency advantages for PCMCPS.

The remainder of this paper is structured as follows: Section II reviews related work on MPPCs and privacy issues; Section III introduces preliminary knowledge; Section IV presents the models and security definitions; Section V proposes the scheme along with its design and protocol; Section VI conducts a security analysis; Section VII evaluates efficiency of PCMCPS and compares it with other scheme; Section VIII and IX give discussion and conclusion.

II. Related Works

A. Multi-party Payment Channel

Traditional off-chain payment channels facilitate transfers by establishing channels between two parties. Multi-party payment channels extend participation from two to multiple parties. There are two design approaches. Channel factories [11], [15] allow a set of users to create a shared on-chain contract that manages multiple off-chain sub-channels based on current bidirectional channels. Sahoo et al. [10] proposed a concurrently executed n-party payment model based on virtual channels, which solves payment disputes by a global smart contract, GSCC, to ensure secure payments among n parties. Although factories significantly improve scalability, it also introduces several privacy challenges [16], [17]: Participants from inactive channels may observe all state transitions; sub-channel relationships are revealed to all members; and joining or exiting procedures leak structural information about past payments.

Another solution is for a single multi-party channel to accommodate multiple users or nodes. Huang et al. [18] proposed a multi-node payment channel scheme, MPC+, which can accommodate an almost infinite number of nodes, thereby avoiding the overhead associated with repeatedly establishing and terminating channels. However, this channel design relies on the payee node as the sole central node, hindering multi-party payments. Concurrently, the maintenance challenges posed by an infinite number of nodes warrant further consideration. Other multi-party channel solutions [7], [19] have significantly enhanced efficiency and scalability, but privacy protection within such architectures remains relatively understudied [6], [12]. Most existing designs assume cooperative or semi-honest user behavior and offer limited defenses against internal attackers with strong observation capabilities.

B. Privacy of Payment

In terms of privacy protection in payment, current research primarily focuses on safeguarding user identity privacy and payment privacy.

[1]The full UC security proof is available in an anonymous GitHub repository for review purposes: https://github.com/anonymous-submission-123/submission-supplementary-material

1) Identity Privacy: To address malicious abort attacks by intermediaries in existing privacy-preserving payment channel hubs supporting variable amounts, Accio [20] proposes a novel structure combining zero-knowledge proofs with adaptor signatures, thereby enhancing relational anonymity and optimising efficiency. Li et al. [21] propose a novel anonymous PCH solution. Through a designed Linkable Randomizable Puzzle (LRP), it achieves unlinkability between hubs and external observers, whilst restoring limited relational information via an audit mechanism in dispute or regulatory scenarios. The SAMCU scheme [19] employs Update Graph Splitting (UGS) technology to achieve secure multi-channel updates while concealing which specific channels or users participated in the channel update, thereby realizing internal anonymity. EM-CPF [8] employs a hybrid encryption mechanism, providing layered protection for user privacy and ensuring anonymous communication between users.

2) Payment Privacy: Thora [22] addresses the issue where path length in payment channel networks significantly impacts locking times by designing an atomic update mechanism. This enables cross-channel payments via non-specific script functions without forming paths (thus avoiding disclosure of individual payment amounts), thereby safeguarding value privacy. Chen et al. [7]'s MBPChannel protocol for multi-party payment channels achieves balance security and on-chain value privacy through Non-Interactive Zero-Knowledge proofs (NIZK) and blind signatures. Another scheme, CAPE [23], also uses the technology of zero-knowledge proof. It converted funds in payment channels into zero-knowledge proof currencies and designed a commitment-based cross-bidirectional payment channel scheme. By leveraging the security of zero-knowledge proof technology, this approach safeguards payment privacy, achieving the privacy of amounts of transactions. Wang et al. [24] designed an RLWE-based cryptographic scheme that implements an on-demand privacy mechanism through puzzle overlap, ensuring user value privacy throughout the payment process.

Although the above approach provides a degree of privacy protection for the payment process, malicious attackers may still obtain information such as the user's transaction frequency and timing through statistical analysis.

III. Preliminaries

A. Paillier Cryptosystem

The Paillier cryptosystem [13] is a public-key cryptographic algorithm possessing additive homomorphic properties.

1) Algorithms: The Paillier cryptosystem typically comprises three polynomial-time algorithms (Gen, Enc and Dec).

$Gen(1^n)$: First, two independent large prime numbers p and q are randomly selected. Then $N = p \cdot q$ and $\lambda = lcm(p-1, q-1)$, where $lcm(\cdot)$ denotes the least common multiple function. Finally, the public key $pk = N$ and private key $sk = (\lambda, \psi(N))$ are obtained, where $\psi(N) = \lambda^{-1} \pmod N$.

$Enc(pk, m)$: Assume m is a plaintext to be encrypted. Firstly, a random number $r \in \mathbb{Z}_N^*$ is selected. Then the encrypted result c can be computed by equation (1):

$$c = [(1+N)^m \cdot r^N \bmod N^2] \tag{1}$$

$Dec(sk, c)$: To get the plaintext m, the encrypted result c can be recovered with the private key sk by equation (2):

$$m = \frac{(c^\lambda \bmod N^2) - 1}{N} \cdot \psi(N) \bmod N \tag{2}$$

2) Homomorphism: Given two plaintext messages m_1 and m_2, the corresponding ciphertexts obtained from the encryption algorithm are $c_1 = [(1+N)^{m_1} \cdot r^N \bmod N^2]$ and $c_2 = [(1+N)^{m_2} \cdot r^N \bmod N^2]$. Given that the Paillier cryptosystem possesses homomorphic properties, to compute the sum m_{sum} of plaintexts m_1 and m_2, one need only perform homomorphic addition on c_1 and c_2:

$$\begin{aligned} c_{sum} &= c_1 \cdot c_2 \bmod N^2 \\ &= [(1+N)^{m_1} \cdot (1+N)^{m_2} \cdot r^N \cdot r^N \bmod N^2] \\ &= [(1+N)^{m_1+M_2} \cdot r^N \bmod N^2] \\ &= [(1+N)^{m_{sum}} \cdot r^N \bmod N^2] \end{aligned} \tag{3}$$

where c_{sum} is the ciphertext of m_{sum}.

IV. Model Construction and Security Definition

A. System Model

The cross multi-party channel payment (MPPC) framework is established based on MPPC in [12], as illustrated in Fig. 1.

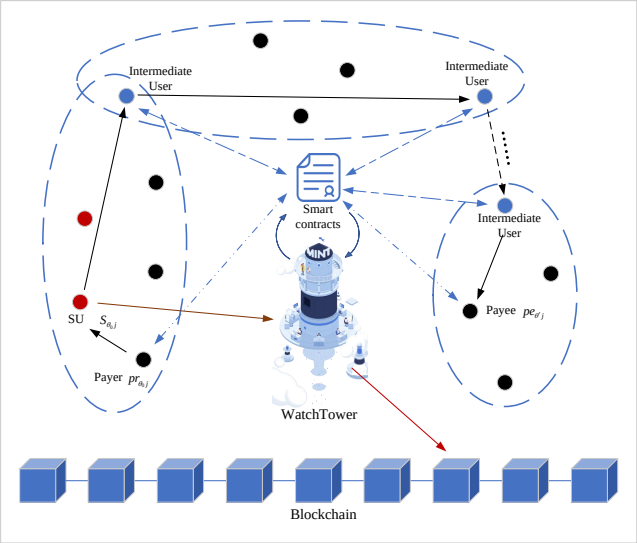


Fig. 1: System Model

It involves five entities: Trading User (U), Supervisory User (SU), Intermediate User (IU), WatchtTower (WT) and Blockchain ($Ledger$). Multiple users with payment requirements establish an MPPC by transferring the intended amount to their account balance within the channel. Trading users

participating in multiple MPPCs, acting as intermediate users, establish the PCN. Should a trading user A within a given payment channel PC^{θ_0} wish to pay an amount m to another trading user B not within the channel PC^{θ_0}, to avoid the unnecessary overhead of directly establishing and closing a channel with B, a cross-channel payment may be initiated by supervisory user SU to facilitate the transaction between parties not within the same payment channel. Intermediate users IUs participating in multiple channels simultaneously act as payment intermediaries, using their account balances across the two payment channels as the payment medium. A third-party Watchtower WT, always acting as a monitor to prevent participants from being required to remain perpetually online [25], serves as the supervisor for cross-channel payments, enabling efficient cross-channel payments.

B. Threat Model

We consider a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ in the Universal Composability (UC) framework [26].

1) Adversarial Model: We assume static corruption. At the beginning of the execution, the adversary $\mathcal{A}$ may corrupt any subset of the following parties: ordinary users participating in payment channels; channel supervisors SU; the watchtower WT. A corrupted party is fully controlled by $\mathcal{A}$ and may arbitrarily deviate from the protocol. The ledger functionality $\mathcal{L}_{CMC}$ is assumed to be honest and incorruptible. It correctly execute deposit locking, evidence verification, and slashing according to protocol.

2) Communication and Execution Model: All communication channels are assumed to be authenticated. The adversary controls message scheduling and delivery but cannot forge messages from honest parties. The Paillier cryptosystem used in the protocol is assumed to be semantically secure and correctly homomorphic.

We consider the standard UC execution model consisting of: a real-world protocol Π_{CMC}, an ideal-world functionality $\mathcal{F}_{CMC}$, a watchtower functionality $\mathcal{F}_{WT}$, a simulator $\mathcal{S}_{CMC}$, an environment $\mathcal{Z}$ that provides inputs to and receives outputs from protocol participants. The environment $\mathcal{Z}$ may arbitrarily interleave multiple protocol executions, invoke concurrent sessions, and adaptively schedule messages, subject to the static corruption assumption.

3) Security Definition: We now formally define the security of the proposed payment channel protocols.

Definition 1 (UC Security)**.** *A protocol Π_{CMC} UC-realizes the ideal functionality $\mathcal{F}_{CMC}$ in the $(\mathcal{F}_{WT}, \mathcal{L}_{CMC})$-hybrid model if for every PPT adversary $\mathcal{A}$ interacting with Π_{CMC} in the real world, there exists a PPT simulator $\mathcal{S}_{CMC}$ such that for every PPT environment $\mathcal{Z}$, the following ensembles are computationally indistinguishable:*

$$\mathsf{EXEC}_{\Pi_{CMC},\mathcal{A},\mathcal{Z}}(1^\lambda) \approx_c \mathsf{EXEC}_{\mathcal{F}_{CMC},\mathcal{F}_{WT},\mathcal{S}_{CMC},\mathcal{Z}}(1^\lambda),$$

where λ is the security parameter, "$\approx_c$" denotes computational indistinguishability..

C. Composability Guarantee

By the UC composition theorem, the security of the proposed system is preserved under arbitrary concurrent executions of: Multi-party payment channels; Cross Multi-party payment channels; interactions with other protocols that access the same ledger functionalities. Thus, the protocol remains secure even when composed with other UC-secure blockchain applications.

V. Proposed Scheme and Protocol

A. Cross-channel Payment Process

When a user $pr_j^{\theta_0}$ within a given channel PC^{θ_0} wishes to make a payment to another user $pe_j^{\theta'}$ within a different channel $PC^{\theta'}$, in order to align with the transaction process within the channel and simplify user operations, this scheme maintains the approach whereby the trading user initiates the transaction request $(pay_j^{\theta_0}, SU_k^{\theta_0}, \tau_{pr_j^{\theta_0}})$ to the supervisory user $SU_k^{\theta_0}$ within their own channel, where $pay_j^{\theta_0} = (pr_j^{\theta_0}, pe_j^{\theta'}, m'', t_j)$, $k \in \{a, b\}$. $m'' = m + \varphi m$ is the amount actually paid by the payer $pr_j^{\theta_0}$, including the payment amount m and the handling fees φm paid to the supervisory users, intermediate users and Watchtower. And t_j is the processed payment request timestamp. After receiving the payment request, the supervisory user determines whether the payment request is an in-channel or cross-channel payment (i.e., determines whether the payee $pe_j^{\theta'}$ belongs to the channel PC^{θ_0}). If it is an in-channel payment, the subsequent payment process is the same as the process in [12]. If it is a cross-channel payment, the payment request is processed by the supervisory user and sent to Watchtower as a cross-channel payment request, and the processed payment request is as follows:

$$\begin{aligned}&\left(pay_j^{\theta_0}, SU_k^{\theta_0}, WT, num^{\theta_0}, \tau_{pr_j^{\theta_0}}, \tau_{SU_k^{\theta_0}}, t_j\right)\\ =&\left((pr_j^{\theta_0}, pe_j^{\theta'}, m''), SU_k^{\theta_0}, WT, num^{\theta_0}, \tau_{pr_j^{\theta_0}}, \tau_{SU_k^{\theta_0}}, t_j\right),\end{aligned} \tag{4}$$

where WT stands for Watchtower, num^{θ_0} is the identifier of the channel where the payment request is initially sent, $\tau_{pr_j^{\theta_0}}$ and $\tau_{SU_k^{\theta_0}}$ are the signatures of the payer and the supervisory user for the payment request.

After receiving the payment request, Watchtower sends a cross-channel participation request to the supervisory users of the channels involved in the payment according to the determined channel routes (Dijkstra's algorithm, etc., can be used to find the optimal route, and the specific routing algorithm is beyond the scope of this paper):

$$\begin{aligned}&(agent_j, \tau_{WT})\\ =&\left((num^{\theta}, pr_j^{\theta}, pe_j^{\theta}, m), \tau_{WT}(agent_j)\right)\end{aligned} \tag{5}$$

Ultimately, the transfers are completed by the supervisory users of each channel, with Watchtower acting as the monitor, consolidating the completed payments at periodic intervals and storing them on the chain.

Problem statement: Under ideal conditions, the aforementioned scheme can indeed facilitate cross-channel payments within the payment channel network without error. However, in practice, Watchtower—as the third party involved in the cross-channel payment process—will, on the one hand, faithfully execute the designed scheme to earn supervisory commissions; on the other hand, it may also harbour curiosity about transaction information: it could analyse the transaction process to obtain illicit information. The primary privacy concerns include:

(1) User identity and relevant information. Trading user identity details encompass the user's blockchain address, which channels they have joined, and their account balances across each channel. During cross-channel payment oversight facilitated by Watchtower, user identity information may be disclosed to attackers through Watchtower, leading to the compromise of personal privacy. In certain circumstances, users may wish to maintain payment anonymity. However, the participation of honest but curious third-party entities inherently diminishes the possibility of conducting anonymous transactions.

(2) Payment details. The specifics of a payment transaction encompass the time, amount, and other particulars. Within the aforementioned framework, Watchtower can directly access this information, thereby threatening user privacy. Even if payment details themselves are encrypted, analysis of factors such as payment frequency and patterns may still enable the inference of private user information. For instance, data revealing frequent small-value transactions with a known service provider could expose the user's consumption habits.

B. Privacy-Preserving Cross Multi-Party Channel Payment Scheme

To address the aforementioned issues, we propose a secure privacy-preserving cross-multi-party channel payment scheme, PCMCPS. For guaranteeing the user's identity and relevant information not to be obtained by Watchtower, PCMCPS uses the Paillier cipher to encrypt the user's information and changes the subject of the routing search from Watchtower to the supervisory user in the payer's channel, so that Watchtower cannot get the users' valid identity from the process of supervising the payment directly , so as to protect the identity privacy and balance privacy. To ensure the surveillance effectiveness of Watchtower, we have designed a redundant lookup table as its storage structure. By employing a guide set approach to randomise the indexing process, we effectively prevent statistical analysis attacks on Watchtower that could lead to unauthorized acquisition of user information.

1) Design of Lookup Table: The storage structure of Watchtower in PCMCPS is designed as a multi-redundant lookup table. The dual redundant lookup table (i.e. redundancy degree $n = 2$) is shown in Table I.

The first column **Channel Identifier** contains the channel identifier for each channel, used to categorize and store users within that channel. The second column $PRF_k(ID)$ holds the obfuscated encrypted user identifier, enabling Watchtower to query users participating in payment. The third column D_{NODE} displays the encrypted account balance within the channel corresponding to the user in the second column. Watchtower can process cross-channel payment by executing balance transfers based on payment requests.

TABLE I: Lookup tables for users and account balances stored in different channels

Channel Identifier	$PRF_k(ID)$	D_{NODE}
num^{θ_0}	$prf_k(id_j), prf_k(id_{j+\zeta})$	$C(d^{\theta_0}_{node_j}), C(d^{\theta_0}_{node_{j+\zeta}})$
	$prf_k(id_{j+1}), prf_k(id_{j+1+\zeta})$	$C(d^{\theta_0}_{node_{j+1}}), C(d^{\theta_0}_{node_{j+1+\zeta}})$
	.....	
num^{θ_1}	$prf_k(id_j), prf_k(id_{j+\zeta})$	$C(d^{\theta_1}_{node_j}), C(d^{\theta_1}_{node_{j+\zeta}})$
	$prf_k(id_{j+1}), prf_k(id_{j+1+\zeta})$	$C(d^{\theta_1}_{node_{j+1}}), C(d^{\theta_1}_{node_{j+1+\zeta}})$
	.....	
...	.....	

2) Design of Guide Set: Based on the structure of Watchtower lookup table, a guide set like $F = \{f_q | f_q \in \{0, 1\}, 1 \leq q \leq n\}$ is set up in this scheme, where n is the number of redundancy storage of data (in Table 1, the redundancy degree $n = 2$). Specifically, the number of elements in the guide set represents the number of the same data items in Watchtower lookup table after redundant storage, where the binary element "1" represents the location of the real data in the guide set, and the rest of the binary elements "0" represent redundant data. Note that regardless of the redundancy degree, the guide set contains only one binary "1".

3) The PCMCPS Protocol: The detailed design of the cross-channel payment is shown in Fig .2.

Payment Request Initiation: As shown in ① in Fig. 2, trading user $pr_j^{\theta_0}$ within the channel PC^{θ_0} initiates a payment request $(pay_j^{\theta_0}, SU_a^{\theta_0}, \tau_{pr_j^{\theta_0}})$ to the supervisory user $SU_a^{\theta_0}$ (Note that in this example, for the sake of convenience in later discussion, the request is sent to the supervisory user $SU_a^{\theta_0}$, and the other supervisory user is $SU_b^{\theta_0}$. In practice, the request will be received by one of the two users randomly).

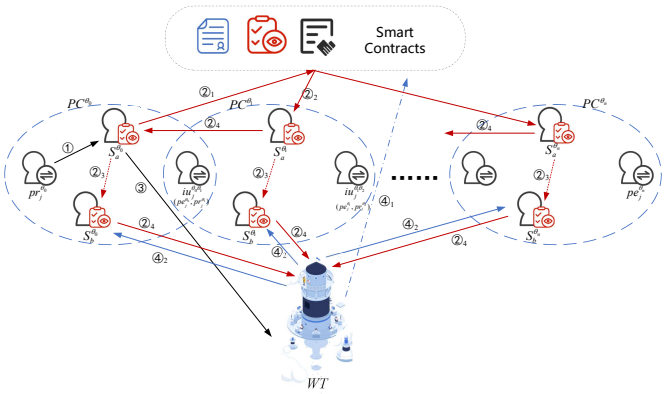


Fig. 2: Cross-channel payment

Upon receiving a payment request, $SU_a^{\theta_0}$ first verifies the authenticity of the signature $\tau_{pr_j^{\theta_0}}(pay_j^{\theta_0})$ in the request and the reasonableness of the payment. Then $SU_a^{\theta_0}$ determine whether the request is a cross-channel payment, i.e. $pe_j^{\theta''} \notin PC^{\theta_0}$. Next, for the request that meets the condition of cross-channel payment, $SU_a^{\theta_0}$ triggers the route finding smart contract to start the route finding, and determines whether there is a payment path that meets the conditions in PCNs, i.e., there exists a cross-channel route from the payer $pr_j^{\theta_0}$ to the payee $pe_j^{\theta''}$, and the balances of the payment channels of the cross-channel intermediate users meet the amount of the payment. If there is no satisfying route, the payment fails.

Fund holding and pre-deduction: For the optimal payment route that satisfies the conditions, $SU_a^{\theta_0}$ triggers the retention smart contract to send a retention request $\left(Reqiu_j^{\theta_i}, \tau\right) = \left((transid, num^{\theta_i}, pr_j^{\theta_i}, pe_j^{\theta_i}, m), \tau(Reqiu_j^{\theta_i})\right)$ to each participation channel, where $i \in [0, n]$.The retention request is to inform intermediates that a cross-channel payment will be generated, requiring intermediate users to retain the balance in advance to ensure that the payment goes well. Upon receiving a request, one of the supervisory users $SU_a^{\theta_i}$ within each channel verifies the request's validity. Following this, verification is conducted to determine whether the intermediary user's balance within that channel satisfies the payment requirements. Should the requirements be met, the request is synchronised to the other supervisory user $SU_b^{\theta_i}$ to initiate a pre-payment deduction from the intermediary user.

$$\widetilde{d}_{pr_j^{\theta_i}} := \widetilde{d}_{pr_j^{\theta_i}} - m. \tag{6}$$

Note that this step also proceeds within the payer's channel, with $SU_b^{\theta_0}$ making a pre-payment to $pr_j^{\theta_0}$.

Subsequently, $SU_b^{\theta_i}$ processes the user identifiers of the users involved in this payment in its channel using the pseudo-random function $prf(\cdot)$ with the shared key k to obtain $\{prf_k(id_j)\}$. Then, $SU_b^{\theta_i}$ sets the Guide Set F of $pr_j^{\theta_i}$ randomly and performs Paillier encryption of the element f_q using the public key $pk = N$ and the random number $r_q \in \mathbb{Z}_N^*$ to obtain c_q:

$$c_q = [(1+N)^{f_q} \cdot r_q^N \bmod N^2] \quad . \tag{7}$$

To enable Watchtower to locate the required user identity and its balance within the query results, we have also designed e_j. The process is shown below:

$$e_j = [(1+N)^{prf_k(id_j)} \cdot (\prod_{q=1}^{n} r_q^{prf_k(id_q)})^N \bmod N^2] \quad . \tag{8}$$

The account balances of these users are encrypted using Paillier algorithm $C(\cdot)$ to obtain $C(d_{node_j})$. Next, in order to store redundantly in lookup tables, thereby resisting statistical analysis attacks from Watchtower, $SU_b^{\theta_i}$ performs the same operation on users with identifiers $id_{j+\zeta \bmod |N^{\theta_i}|}$ and $id_{j-\zeta \bmod |N^{\theta_i}|}$, which are subjected to identical processing, yielding $prf_k(id_{j\pm\zeta \bmod |N^{\theta_i}|})$ and $C(d_{node_{j\pm\zeta \bmod |N^{\theta_i}|}})$ respectively. At this point, the processing of one intermediate user's information involved in this payment within the channel has been finalised. To balance the workload among supervisory users, the information pertaining to the other intermediate user $iu_j^{\theta_i\theta_{i+1}}$ (also described as $pe_j^{\theta_i}$ or $pr_j^{\theta_{i+1}}$) shall be handled by $SU_a^{\theta_i}$. The procedure will follow the same pattern as described above and shall therefore not be repeated here.

Then $SU_a^{\theta_i}$ integrates the intermediate users' information and transmits it back as $((prf_k(id_j), \{c_q\}, e_j), \tau)$ to $SU_a^{\theta_0}$ to form a Watchtower assistance request. $SU_b^{\theta_i}$ update the corresponding data according to the structure of the lookup table of Watchtower. That is, the updated data is encrypted and redundantly stored in Watchtower.

Request for Watchtower assistance: After data updates have been completed across all channels, $SU_a^{\theta_0}$ initiates a request for assistance to Watchtower WT:

$$\begin{aligned}
&\left(Reqwt, WT, \tau_{SU_k^{\theta_0}}\right) \\
=&\left((NUM, prf_k(ID), CQ, E, C(m)), WT, \tau_{SU_k^{\theta_0}}\right) \\
=&\Big((num^{\theta_0}, num^{\theta_1}, \cdots, prf_k(id_{j1}), prf_k(id_{j2}), \cdots, \\
&(Cq)_{j1}, (Cq)_{j2}, \cdots, e_{j1}, e_{j2}, \cdots, C(m)), WT, \\
&\tau_{SU_k^{\theta_0}}(Reqwt)\Big),
\end{aligned} \tag{9}$$

where $CQ = \{(Cq)_{j1}, (Cq)_{j2}, \cdots\}$ and $E = \{e_{j1}, e_{j2}, \cdots\}$ correspond to each $prf_k(id_j)$. Specifically, Cq is a cipher set consisting of a series of encrypted data of the form $\{c_q | 1 \leq q \leq n\}$.

Supervision of the Watchtower: Watchtower, after receiving the assistance request $(reqwt, WT, \tau_{S_k^{\theta_0}})$, uses $prf_k(ID)$ as a lookup index to find $prf_k(id_j)$ within corresponding channel from the column $PRF_k(ID)$ of its own storage table. Because Watchtower storage structure designed by the scheme is a redundant lookup table, i.e., the $prf_k(id_j)$ within a channel is also redundant and has the same amount of redundancy. Take the double redundancy shown in Table I as an example, Watchtower lookup to the content of $(prf_k(id_j), prf_k(id_x))$ and $(prf_k(id_y), prf_k(id_j))$. Both items contain $prf_k(id_j)$. Next, Watchtower will perform the following calculations on the two items found:

$$\begin{aligned}
P(PRF_k(ID)) &= [\prod_{q=1}^{n} c_q^{PRF_k(ID)} \bmod N^2] \\
&= [\prod_{q=1}^{n} c_q^{prf_k(id_q)} \bmod N^2] \\
&= [(1+N)^{\sum_{q=1}^{n} f_q \cdot prf_k(id_q)} \cdot (\prod_{q=1}^{n} r_q^{prf_k(id_q)})^N \bmod N^2] \\
&= [(1+N)^{1 \cdot prf_k(id_j)} \cdot (\prod_{q=1}^{n} r_q^{prf_k(id_q)})^N \bmod N^2] \\
&= [(1+N)^{prf_k(id_j)} \cdot (\prod_{q=1}^{n} r_q^{prf_k(id_q)})^N \bmod N^2] \quad ,
\end{aligned} \tag{10}$$

Watchtower then compares to find the data item that is equal to e_j (i.e., $P(PRF_k(ID)) == e_j$), which is the target data item. After finding the target data item, Watchtower continues to calculate $P(D_{NODE})$:

$$\begin{aligned} P(D_{NODE}) &= [\prod_{q=1}^{n} c_q^{D_{NODE}} \bmod N^2] \\ &= [(1+N)^{C(d^{\theta}_{node_{ej}})} \cdot r^N \bmod N^2] \end{aligned} \tag{11}$$

For each payment, Watchtower can obtain $P(C(d^{\theta_0}_{node_{j1}})), P(C(d^{\theta_0}_{node_{j2}})), P(C(d^{\theta_1}_{node_{j2}})), \cdots$ after going through the above series of calculations. The complete Watchtower assistance algorithm is shown in Algorithm 1.

Algorithm 1 Watchtower

Input: Lookup Table LT, Channel set NUM, encrypted set of user identifiers $prf_k(ID)$, CQ, $E = \{e_{j1}, e_{j2}, \cdots\}$
Output: $P(D_{NODE})$

$P(D_{NODE}) \leftarrow \emptyset$
for each $num^{\theta_i} \in NUM$ **do**
  $entry \leftarrow LT[num^{\theta_i}]$
  $IDset \leftarrow entry[PRF_k(ID)]$
  $Dset \leftarrow entry[D_{NODE}]$
  for $t = 0$ to $|IDset| - 1$ **do**
    $ID \leftarrow IDset[t]$
    $Dnode \leftarrow Dset[t]$
    if $prf_k(id_{ji}) \in ID$ **then**
      $y1 \leftarrow P(ID)$
      if $y1 = e_{ji}$ **then**
        $P(D_{NODE}) \leftarrow P(D_{NODE}) \cup \{P(Dnode)\}$
      end if
    end if
    if $prf_k(id_{j(i+1)}) \in ID$ **then**
      $y2 \leftarrow P(ID)$
      if $y2 = e_{j(i+1)}$ **then**
        $P(D_{NODE}) \leftarrow P(D_{NODE}) \cup \{P(Dnode)\}$
      end if
    end if
  end for
end for
return $P(D_{NODE})$;

After calculating $P(C(m))$, Watchtower proceeds to process $P(D_{NODE})$ sequentially as follows:

$$\begin{cases} P(C(d^{\theta_0}_{node_{j1}})) = & P(C(d^{\theta_0}_{node_{j1}}))^{\frac{1}{P(C(m))}} \\ P(C(d^{\theta_0}_{node_{j2}})) = & P(C(d^{\theta_0}_{node_{j2}}))^{P(C(m))} \\ P(C(d^{\theta_1}_{node_{j2}})) = & P(C(d^{\theta_1}_{node_{j2}}))^{\frac{1}{P(C(m))}} \\ P(C(d^{\theta_1}_{node_{j3}})) = & P(C(d^{\theta_1}_{node_{j3}}))^{P(C(m))} \\ \vdots \end{cases} \tag{12}$$

The homomorphism of Paillier's cryptographic regime follows:

$$\begin{cases} d^{\theta_0}_{node_{j1}} = & d^{\theta_0}_{node_{j1}} - m \\ d^{\theta_0}_{node_{j2}} = & d^{\theta_0}_{node_{j2}} + m \\ d^{\theta_1}_{node_{j2}} = & d^{\theta_1}_{node_{j2}} - m \\ d^{\theta_1}_{node_{j3}} = & d^{\theta_1}_{node_{j3}} + m \\ \vdots \end{cases} \tag{13}$$

Finally, Watchtower triggers the relevant smart contract using the proof of outcome, demonstrating its participation in the payment and claiming the corresponding fee. It then returns confirmation of successful payment to the supervisory users within each channel, instructing them to update their local account records. That is, the cross-channel payment process is completed.

C. Ideal Functionality of PCMCPS

Ideal Functionality of PCMCPS. The ideal functionality $\mathcal{F}_{CMC}$ interacts with trading users, Watchetower $\mathcal{F}_{WT}$, the simulator $\mathcal{S}_{CMC}$, and the ledger $\mathcal{L}_{CMC}$. The ideal functionality $\mathcal{F}_{CMC}$ is shown in Fig. 3.

Payment request. Upon receiving $(\mathsf{pay}, pid, pr, pe, m, t)$ from $SU_a^{\theta_0}$ on behalf of pr^{θ_0}:
1) Determine whether the payment is cross-channel.
2) If not, ignore; otherwise derive (NUM, ID) and set $\mathsf{status} := \mathsf{init}$.

Fund holding. For each channel $num^{\theta_r} \in NUM$:
1) Send $(\mathsf{hold\_req}, pid, num_{\theta_r}, ID_r, m)$ to $SU_a^{\theta_r}$.
2) If any channel rejects, set $\mathsf{status} := \mathsf{abort}$.
3) Send $(\mathsf{wt\_update}, pid, num^{\theta_r}, \mathsf{UpLT}^{\theta_r})$ to $\mathcal{F}_{WT}$ on behalf of $SU_b^{\theta_r}$.

Upon receiving $(\mathsf{hold\_ok}, pid, num^{\theta_r})$ from all channels accept, set $\mathsf{status} := \mathsf{ready}$.

Watchtower assistance. If $\mathsf{status} = \mathsf{ready}$, send $(\mathsf{reqwt}, pid, NUM, C(m), prf_k(ID), CQ, E)$ to $\mathcal{F}_{WT}$ on behalf of $SU_a^{\theta_0}$.

Finalization.
1) Upon receiving $(\mathsf{wt\_confirm}, pid)$, finalize local balances and set $\mathsf{status} := \mathsf{settled}$.
2) Upon receiving $(\mathsf{wt\_rej}, pid)$, set $\mathsf{status} := \mathsf{abort}$.

Fig. 3: Ideal Functionality $\mathcal{F}_{CMC}$

The ideal functionality $\mathcal{F}_{WT}$ interacts with supervisory users, $\mathcal{F}_{CMC}$, the simulator $\mathcal{S}_{CMC}$, and the ledger $\mathcal{L}_{CMC}$. The ideal functionality $\mathcal{F}_{WT}$ is shown in Fig. 4.

Updates Lookup Table
Upon receiving $(\mathsf{wt\_update}, pid, num^{\theta_r}, \mathsf{UpLT}^{\theta_r})$, update UpLT^{θ_r}.

Execution request. Upon receiving $(\mathsf{reqwt}, pid, NUM, C(m), prf_k(ID), CQ, E)$:
- Search $prf_k(ID)$ from LT, and perform Paillier-homomorphic computations to finish payment. Output

$(\mathsf{wt\_confirm}, pid)$.
- Otherwise output $(\mathsf{wt\_rej}, pid)$.

Abort and evidence generation.
If malformed inputs, inconsistent updates, or timeout occur:
- Set $\mathsf{status} := \mathsf{abort}$.
- Output $(\mathsf{wt\_rej}, pid, \mathsf{reason})$ to $\mathcal{F}_{CMC}$.
- Optionally generate a proof object $\pi_{WT}(pid)$ that can be submitted to $\mathcal{L}_{CMC}$ for dispute resolution.

Fig. 4: Ideal Functionality $\mathcal{F}_{WT}$

VI. Security Analysis

In this section, we provide a comprehensive security analysis of the proposed scheme.

Theorem 1. *Let Π_{CMC} denote the proposed PCMCPS protocol. Assume static corruption and a polynomiamtime adversary $\mathcal{A}$.*

In the $(\mathcal{F}_{WT}, \mathcal{L}_{CMC})$-hybrid model, Π_{CMC} UC-realizes the ideal functionality $\mathcal{F}_{CMC}$.

Consequently, for any probabilistic polynomial-time environment $\mathcal{Z}$ and any adversary $\mathcal{A}$, there exists a probabilistic polynomial-time simulator $\mathcal{S}_{CMC}$ such that the real-world execution of Π_{CMC} is computationally indistinguishable from the ideal-world execution with $\mathcal{F}_{CMC}$.

Proof Sketch. We sketch why the proposed cross-channel protocol UC-realizes $\mathcal{F}_{CMC}$.

The simulator $\mathcal{S}_{CMC}$ forwards all payment requests to the ideal functionality and simulates supervisor and Watchtower transcripts on behalf of honest parties. Tentative fund holding prevents double spending even if the cross-channel procedure spans multiple rounds.

Watchtower only proceeds after receiving all required encrypted updates. If a corrupted Watchtower confirms prematurely, outputs inconsistent results, or stalls, the simulator reduces such behavior to a valid evidence submission and slashing event in $\mathcal{L}_{CMC}$. Thus, any deviation has an equivalent ideal-world outcome.

Since Watchtower observes only encrypted identifiers and balances, its view can be simulated using ideal-world leakage. Therefore, no environment can distinguish the real and ideal executions, which establishes UC security. □

VII. Implementation and Evaluation

In this section, we designed two experiments to test and evaluate the effectiveness of PCMCPS. The first experiment evaluates the smart contracts within PCMCPS. The second experiment compares its time overhead with mainstream ZK-SNARKs.

A. Smart-Contract–Based Cross-Channel Evaluation

We implement the proposed PCMCPS on an EVM [27]-compatible blockchain and evaluate its feasibility and on-chain overhead. The implementation supports: Multi-party payment channels with N participants, where off-chain balances are updated iteratively and committed via on-chain settlement rounds. Cross-channel payments, coordinated by an external Watchtower, enabling atomic balance updates across multiple independent channels.

1) Experimental Setup: All experiments are conducted using the Foundry framework [28], which provides a deterministic local EVM execution environment and a programmable testing infrastructure for smart contracts. We deploy the proposed payment channel and cross-channel contracts on a local Anvil instance and execute protocol logic through Foundry's testing harness. The framework enables precise control over block timestamps, caller identities, and signature generation, which is essential for simulating multi-round channel execution, supervisory user behavior, and cross-channel coordination. The experiments are performed on a machine equipped with an Apple M4 Pro processor and 24 GB of memory, running macOS Sequoia 15.5. Smart contracts are compiled using Solidity 0.8.24, and all tests are executed with Foundry (forge 1.4.4-stable). In the experiments, each payment channel supports up to $|N| = 15$ participants, which is a moderate size for a MPPC. And the threshold parameter is set to $T = 10$ that governs participant lifetime across payment rounds. These settings address the practical reality that cross-channel payments are slower than intra-channel payments, while also preventing excessive channel overhead caused by accommodating too many users without demand. Cross-channel payments involve two more independent channels and are executed through atomic intents coordinated by Watchtower. On-chain execution cost is measured using Foundry's built-in gas reporting mechanism, while all experiments are repeated under identical conditions to ensure reproducibility.

2) Evaluation Results: Based on [12], we have modified and introduced several new smart contracts related to cross-channel payments, measuring their respective Gas consumption. The deployment cost for smart contracts related to intra-channel payments is approximately 1,466,472 gas, with a deployment size of approximately 7,227 bytes. The deployment cost for smart contracts related to cross-channel payments is approximately 1,168,923 gas, with a deployment size of approximately 5,413 bytes. The results of several main functions are shown in Fig. 5.

Within the smart contracts pertaining to payments within the channel, establishing a channel incurs relatively high costs. Consequently, the gas expenditure for `InitializeChannel` in the figure is significantly higher than other components, accounting for approximately 25.5% of the total expenditure. This is one of the reasons for selecting multi-party payment channel and payment channel networks. The gas costs for the remaining primary functions all remain below 150k gas. Regarding cross-channel payment smart contracts, the PCMCPS design offloads cryptographic operations such as homomorphic computation to Watchtower for off-chain processing. Consequently, expenses for these contracts are generally low. Overall, the smart contract costs within the PCMCPS design are within reasonable parameters, with deployment sizes well below the EIP-170 [29] contract

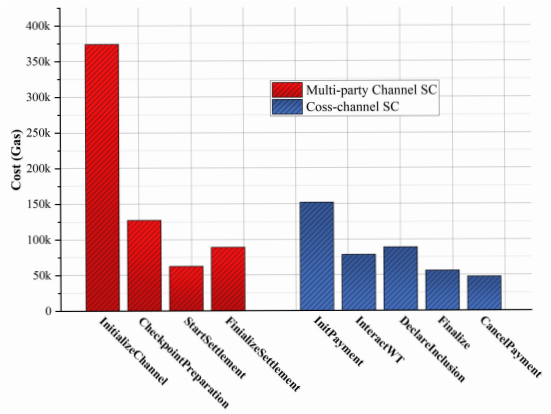


Fig. 5: Gas consumption of main function in smart contracts

size limit.

The experimental results demonstrate that the proposed design is practical on EVM-compatible blockchains. By separating cross-channel coordination from individual channel logic and offloading privacy-preserving computation to off-chain components, the system achieves atomic cross-channel payments with low on-chain overhead and strong privacy guarantees.

B. Time Overhead of PCMCPS

This subsection evaluates the efficiency of the proposed cross-channel payment scheme, with a particular focus on the cryptographic overhead introduced by privacy-preserving balance verification. We first analyze the performance of the Paillier-based design adopted in our scheme and then compare it with zk-SNARK–based approaches [14], which are commonly used in existing privacy-preserving payment systems.

1) Motivation for zk-SNARK–Based Comparison: In the proposed scheme, privacy protection is achieved through additively homomorphic encryption based on the Paillier cryptosystem. Sensitive operations such as user identification, balance lookup, and balance updates are performed on encrypted data by Watchtower. The on-chain component only verifies cryptographic commitments and signatures and does not participate in heavy cryptographic computation.

zk-SNARKs have been widely adopted in prior works to provide strong privacy guarantees in blockchain systems, including confidential transactions and privacy-preserving payment channels. Their succinct verification property enables constant-size proofs and efficient on-chain verification. However, zk-SNARK–based solutions typically incur substantial proof generation overhead, which grows with the complexity of the circuit or the number of payment hops.

To highlight the design trade-offs, we compare the efficiency of PCMCPS against zk-SNARK–based implementations under comparable experimental conditions.

2) Experimental Setup: The Paillier-based experiments are implemented in Java using the Java Microbenchmark Harness (JMH) framework. All benchmarks are executed in a single-threaded configuration on an Apple M4 Pro processor with 24 GB memory, running macOS Sequoia 15.5 (Apple Silicon). The implementation is based on a self-developed Paillier cryptosystem using Java `BigInteger`, without CRT optimization and without GPU acceleration. First, we illustrate the impact of Paillier key size on the cost of key generation, encryption, and decryption.

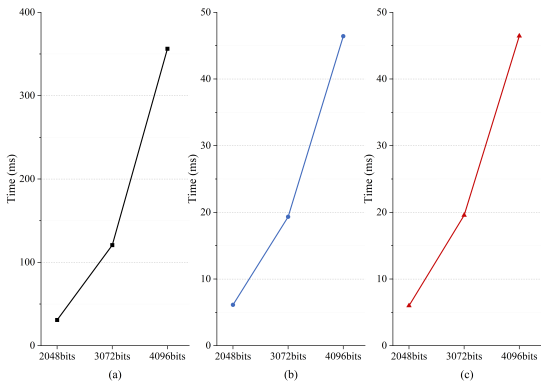
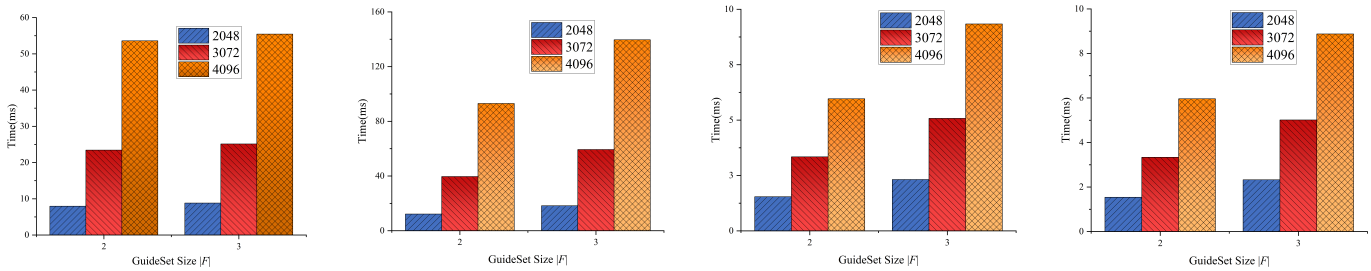


Fig. 6: Impact of Paillier key size on cryptographic operations. (a) Key generation time, which is incurred once during initialization; (b) Encryption time for computing $C(m)$, incurred per payment; (c) Decryption time, which dominates the recurring cryptographic cost. All measurements are conducted in a single-threaded CPU-only environment.

As shown in Fig. 6(a), key generation incurs a one-time initialization cost that increases rapidly with the security parameter, which is expected given the underlying large integer arithmetic. However, this cost is amortized over the lifetime of the channel and does not affect per-payment performance. Fig. 6(b) and 6(c) show the cost of encryption and decryption. The actual time of decryption operations is even lower than the expected number of payments, as Watchtower can adjust balances without decryption due to the homomorphic addition properties of the Paillier scheme. This means decryption may only be required when consolidating multiple payments onto the blockchain after they have been processed. The absolute cost of both operations remains within a practical range under commonly used security parameters, indicating that Paillier-based homomorphic operations are feasible for cross-channel payment execution.

And we evaluate the computation cost of the cryptographic operations involved in encrypted lookup and balance update during the process. These operations represent the dominant overhead introduced by privacy preservation in PCMCPS.

As shown in Fig. 7(a)–(d), We conducted tests on the computational cost of the correlation function for GuideSet sizes $|F| = 2$ and $|F| = 3$. It can be observed that the computation of ej is independent of the GuideSet size and solely dependent on the key size. Increasing the GuideSet size from $|F| = 2$ to $|F| = 3$ results in a moderate increase in computational cost for other algorithms, while the overall performance trend remains stable across different Paillier key sizes. The absolute



(a) Overhead of e_j (b) Overhead of C_q (c) Overhead of $P(PRF_k(ID))$ (d) Overhead of $P(D_{NODE})$

Fig. 7: Impact of GuideSet size on cryptographic operations related to encrypted lookup and balance update. The computation cost increases mildly with the GuideSet size and remains predictable across different Paillier key sizes, indicating that the proposed lookup design introduces limited overhead.

computation cost remains within a practical range across all tested configurations, indicating that the proposed design can support privacy-preserving cross-channel payments without incurring prohibitive cryptographic overhead.

For comparison, we implement zk-SNARK–based baselines using the Groth16 [30] proving system, with circuits written in circom and executed using snarkjs. All zk-SNARK experiments are conducted on the same machine and in a CPU-only environment, without GPU acceleration. We consider two zk-SNARK execution models: 1) Per-hop independent proofs, where each hop in a cross-channel payment path generates and verifies an independent proof; 2) Single aggregated circuit, where all hops are encoded into one circuit and verified once. Given that the defining characteristic of zero-knowledge proof technology is its extremely fast verification speed (a single verification takes only a few milliseconds), we consider only the proof time for both models.

The hop count is varied across $h \in \{1, 2, 4, 5, 10\}$. The trusted setup phase of Groth16 is executed once offline and is excluded from all measurements, and only proving costs are reported.

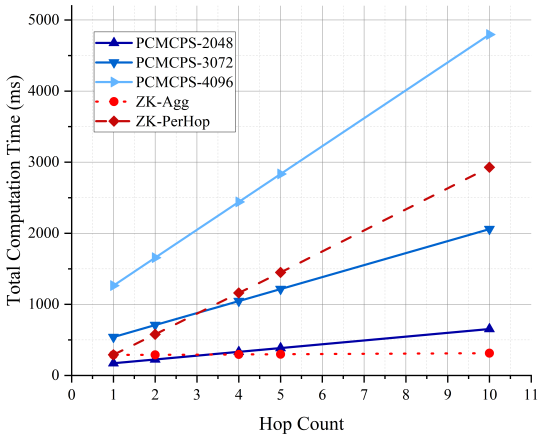


Fig. 8: End-to-end computation cost of cross-channel payments under different hop counts.

Fig. 8 shows the primary payment time costs of PCMCPS versus zk-SNARK-based methods across different hop counts and Paillier key sizes. Results indicate that for small hop counts, specifically around $h < 3$ ($|PC| < 4$), PCMCPS-2048 exhibits lower time costs than zk-SNARK proof generation—even when zk-SNARK employs aggregated single circuits. In practice, aggregated single circuits inherently suffer from privacy leaks and proof generation difficulties. For stronger privacy protection, having each intermediate user generate their own proof is the appropriate choice.

For zk-SNARKs with per hop circuit, PCMCPS-3072 exhibits lower overhead starting from 4 hops and maintains this trend (approximately 72% of zk-PerHop, 10 hops). Furthermore, PCMCPS-2048 consistently demonstrates lower overhead than zk-PerHop throughout all hop counts. This indicates that while both protocols exhibit linear growth with hop count, PCMCPS possesses a significant advantage. We also measured PCMCPS's overhead with a key size of 4096bits. Result indicates that even with such an absolutely secure key size, PCMCPS's total time overhead at 10 hops is approximately 4796ms—still within reasonable bounds (considering 11 payment channels and 12 users involved).

These results demonstrate that PCMCPS's Paillier-based privacy-preserving solution offers a more efficient alternative than zk-SNARK-based approaches.

VIII. Discussion

While PCMCPS demonstrates practical efficiency and favorable performance compared to zk-SNARK–based approaches for small hop counts, it also exhibits several inherent limitations that merit discussion.

First, the efficiency advantage of the proposed design relies on the use of homomorphic encryption, whose computation cost grows linearly with the hop count and depends on the selected security parameter. Although our evaluation shows that this cost remains practical under commonly used Paillier key sizes and realistic payment path lengths, the scheme may become less competitive for extremely large key size.

Second, the proposed design assumes an honest-but-curious Watchtower that correctly executes the encrypted lookup and balance update protocols while being prevented from

learning sensitive payment information. Although misbehavior by Watchtower can be detected and penalized through on-chain mechanisms, exploring stronger adversarial models and decentralized Watchtower deployments remains an interesting direction for future research.

IX. Conclusion

In this paper, we propose a secure privacy-preserving cross multi-party channel payment scheme (PCMCPS) in payment channel networks. We introduced Watchtower as a third-party regulator and designed a privacy-preserving algorithm based on the Paillier cryptosystem, enabling efficient and secure cross-MPPC payments within PCNs. The security of PCMCPS is formally defined and proved in UC framework. Furthermore, a comparison with the mainstream privacy protection technology zk-SNARK demonstrates the advantages of our proposed PCMCPS.

Acknowledgment

Appendix

Stage I: Transaction type identification.

- *Case I1 (honest payer, honest payer-channel supervisors).* Upon receiving $(\mathsf{pay}, pid, pr, pe, m, t)$, the simulator forwards it to $\mathcal{F}_{CMC}$. If the payment is identified as cross-channel, the simulator initializes the internal record for pid and outputs an acceptance transcript on behalf of the payer-channel supervisor.
- *Case I2 (corrupt payer, honest payer-channel supervisors).* The simulator forwards the request to $\mathcal{F}_{CMC}$ and outputs either acceptance or rejection transcripts according to the ideal decision.
- *Case I3 (honest payer, corrupt payer-channel supervisor).* The simulator forwards the request to $\mathcal{F}_{CMC}$. Any externally inconsistent acceptance or rejection transcript generated by the adversary is recorded as a deviation for potential reverse supervision.
- *Case I4 (corrupt payer and corrupt payer-channel supervisor).* The simulator lets the adversary determine external transcripts while maintaining internal consistency with the ideal functionality.

Stage II: Fund holding and pre-deduction.

- *Case H1 (honest supervisors in all involved channels).* For each channel in NUM, the simulator delivers a holding request on behalf of the payer-channel supervisor. The other supervisor in the channel performs tentative pre-deduction. Upon success, the simulator outputs a holding confirmation and records the channel as successfully held.
- *Case H2 (some channel supervisors corrupt, honest payer).* For honest channels, the simulator behaves as in Case H1. For corrupted channels, the adversary may refuse holding, apply incorrect pre-deduction, or output inconsistent confirmations. All such behavior is recorded in the deviation log. If any honest channel rejects holding, the simulator aborts the payment.
- *Case H3 (corrupt payer, honest supervisors).* If the ideal functionality reaches the holding stage, the simulator proceeds as in Case H1. Concurrent reuse of funds by the payer is prevented by the tentative balance updates in the ideal world, and subsequent requests are simulated as rejected.
- *Case H4 (corrupt payer and corrupt supervisors).* The simulator combines the behaviors of Case H2 and Case H3 and records all detected inconsistencies.

Stage III: Encrypted lookup-table updates.

- *Case U1 (honest supervisors, arbitrary watchtower).* For each successfully held channel, the simulator constructs a well-formed encrypted update consistent with the ideal state and delivers it to $\mathcal{F}_{WT}$ on behalf of the channel supervisors.
- *Case U2 (corrupt supervisors in some channels, honest watchtower).* Encrypted updates from honest channels are delivered as in Case U1. Malformed or missing updates from corrupted channels prevent the watchtower from satisfying its execution condition.
- *Case U3 (corrupt watchtower).* The adversary may drop, reorder, or falsely acknowledge updates. Any discrepancy between submitted updates and watchtower behavior is recorded as a deviation.

Stage IV: Watchtower assistance.

- *Case W1 (honest watchtower).* If encrypted updates from all channels are available, the simulator outputs a watchtower confirmation transcript and marks the payment as completed by the watchtower.
- *Case W2 (corrupt watchtower, stalling or rejecting).* If the watchtower rejects or stalls despite all conditions being satisfied, the simulator aborts the payment and records the deviation.
- *Case W3 (corrupt watchtower, premature confirmation).* If the watchtower confirms execution without receiving all required updates, the simulator records a critical inconsistency for later slashing.

Stage V: Local ledger finalization.

- *Case F1 (honest supervisors in all channels).* After watchtower confirmation, the simulator instructs each involved channel to finalize its local balances and outputs a successful cross-channel payment completion transcript.
- *Case F2 (corrupt supervisors in some channels).* If corrupted supervisors refuse to finalize or output inconsistent local states, the simulator records the deviation and reduces the execution to dispute resolution.

Stage VI: Dispute resolution and slashing.

- *Reverse supervision.* When an inconsistency is detectable by honest users, the simulator constructs abstract evidence and submits it to $\mathcal{L}_{CMC}$ on behalf of the user.
- *Mutual supervision.* When contradictory transcripts from the watchtower or supervisors exist, the simulator

submits corresponding evidence on behalf of an honest supervisor.

Fig. 9: Simulation $\mathcal{S}_{CMC}$

The ledger functionality $\mathcal{L}_{CMC}$ interacts with $\mathcal{F}_{CMC}$, $\mathcal{F}_{WT}$

Deposit locking. Upon receiving $(\mathsf{LockWT}, WT, Dep_{WT})$, lock Dep_{WT} as the collateral of WT.

Deposit release. Upon receiving $(\mathsf{ReleaseWT}, pid, WT)$, if pid is marked as successfully settled and no valid evidence against WT has been accepted, release Dep_{WT} to WT.

Evidence submission. Upon receiving $(\mathsf{SubmitEvidence}, pid, WT, \mathsf{etype}, \pi)$ from any party, where $\mathsf{etype} \in \{\mathsf{reverse}, \mathsf{mutual}\}$:

1) If pid is already resolved, ignore the submission.
2) Otherwise, evaluate the abstract predicate $\mathsf{Verify}_{\mathsf{etype}}(pid, WT, \pi)$.

Slashing. If $\mathsf{Verify}_{\mathsf{etype}}(pid, WT, \pi) = 1$, slash Dep_{WT}, transfer it to the reporter, and mark pid as resolved.

Rejection. If $\mathsf{Verify}_{\mathsf{etype}}(pid, WT, \pi) = 0$, ignore the evidence and keep Dep_{WT} locked.

Finalization marker. Upon receiving $(\mathsf{Finalize}, pid)$ from $\mathcal{F}_{CMC}$, mark pid as successfully settled unless it has already been resolved by slashing.

Fig. 10: Ledger functionality $\mathcal{L}_{CMC}$

REFERENCES

[1] Chainalysis Team, "APAC Crypto Adoption Accelerates with Distinct National Pathways," Chainalysis Blog, Sep. 24, 2025. [Online]. Available: https://www.chainalysis.com/blog/asia-pacific-crypto-adoption-2025/. Accessed: Jan. 5, 2026.

[2] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," White paper, Draft Version 0.5.9.2, Jan. 2016.

[3] Raiden Network, "Raiden Network 101: What is the Raiden Network?," Raiden.network. [Online]. Available: https://raiden.network/101.html (accessed Jan 5, 2026).

[4] P. Zabka, L. Luu, T. Nasr, and A. Gervais, "Empirical evaluation of nodes and channels of the Lightning Network," *Pervasive and Mobile Computing*, vol. 83, art. no. 101584, 2022.

[5] F. Grötschla, L. Heimbach, S. Richner, and R. Wattenhofer, "On the lifecycle of a lightning network payment channel," in *Proc. Int. Conf. Financial Cryptography and Data Security*, pp. 1–16, 2025.

[6] Z. Ge, Y. Zhang, Y. Long, D. Gu, Y. Liu, and J. Wang, "Magma: Robust and flexible multi-party payment channel," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 6, pp. 5024–5042, 2023, doi:10.1109/TDSC.2023.3238332.

[7] R. Chen, Y. Zhang, D. Li, Y. Liu, J. Liu, Q. Wu, J. Zhou, and W. Susilo, "Bitcoin-compatible privacy-preserving multi-party payment channels supporting variable amounts," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 2984–2998, 2025, doi:10.1109/TIFS.2025.3528827.

[8] H. F. Hasan, M. N. Yusoff, J. S. Teh, and S. M. Abd Ali, "EMCPF: Efficient multi-party conditional payment flow payment protocol on Lightning payment channel network," *Blockchain: Research and Applications*, vol. 6, p. 100359, Aug. 2025, doi:10.1016/j.bcra.2025.100359.

[9] Y. Ye, Z. Ren, X. Luo, J. Zhang, and W. Wu, "Garou: An Efficient and Secure Off-Blockchain Multi-Party Payment Hub," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4450–4461, Dec. 2021.

[10] S. S. Sahoo, A. R. Menon, and V. K. Chaurasiya, "Blockchain based n-party virtual payment model with concurrent execution," *Arabian Journal for Science and Engineering*, vol. 49, no. 3, pp. 3285–3312, 2024, doi:10.1007/s13369-023-08241-7.

[11] A. Tian, P. Ni, Y. Liu, and L. Huang, "Blockchain-based payment channel network: challenges and recent advances," in *Proc. 2023 Int. Conf. Blockchain Technology and Information Security (ICBCTIS)*, pp. 187–194, 2023.

[12] K. Xiao, J. Li, Y. He, X. Wang, and C. Wang, "A secure multi-party payment channel on-chain and off-chain supervisable scheme," *Future Generation Computer Systems*, vol. 154, pp. 330–343, May 2024, doi:10.1016/j.future.2024.01.012.

[13] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techniques (EUROCRYPT '99)*, Prague, Czech Republic, May 1999, pp. 223–238.

[14] W. Yu, M. Xu, D. Yu, X. Cheng, Q. Hu, and Z. Xiong, "zk-PCN: A Privacy-Preserving Payment Channel Network Using zk-SNARKs," in *Proc. 41st International Performance, Computing and Communications Conference (IPCCC)*, Austin, TX, USA, Nov. 2022, pp. 57–64, doi:10.1109/IPCCC55821.2022.00018.

[15] M. Qi, Q. Wang, Z. Wang, M. Schneider, T. Zhu, S. Chen, W. Knottenbelt, and T. Hardjono, "SoK: Bitcoin layer two (L2)," *ACM Computing Surveys*, vol. 58, no. 3, pp. 1–37, 2025.

[16] X. Chen, J. Lai, C. Lin, X. Huang, and D. He, "Cryptographic primitives in script-based and scriptless payment channel networks: A survey," *ACM Computing Surveys*, vol. 57, no. 10, pp. 1–34, 2025, doi:10.1145/3725846.

[17] M. H. Ziegler, M. Nowostawski, and B. Katt, "A systematic literature review of information privacy in blockchain systems," *Journal of Cybersecurity and Privacy*, vol. 5, no. 3, p. 65, 2025, doi:10.3390/jcp5030065.

[18] L. Huang, H. Lei, and L. Wang, "MPC+: Secure, compatible and efficient off-blockchain multi-node payment channel," *IEEE Transactions on Network and Service Management*, vol. 21, no. 3, pp. 3152–3166, 2024, doi:10.1109/TNSM.2024.3379475.

[19] J. Wang, S. Gao, G. Li, K. Gai, and B. Xiao, "SAMCU: Secure and anonymous multi-channel updates in payment-channel networks," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 9115–9128, 2024, doi:10.1109/TIFS.2024.3451366.

[20] Z. Ge, J. Gu, C. Wang, Y. Long, X. Xu, and D. Gu, "Accio: Variable-amount, optimized-unlinkable and NIZK-free off-chain payments via hubs," in *Proc. 2023 ACM SIGSAC Conf. on Computer and Communications Security (CCS '23)*, Copenhagen, Denmark, Nov. 2023, pp. 1541–1555, doi:10.1145/3576915.3616577.

[21] Y. Li, J. Weng, J. Lai, Y. Li, J. Wu, M. Li, J. Sun, P. Wu, and R. H. Deng, "AuditPCH: Auditable payment channel hub with privacy protection," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 1251–1261, 2025, doi:10.1109/TIFS.2024.3515820.

[22] L. Aumayr, K. Abbaszadeh, and M. Maffei, "Thora: Atomic and privacy-preserving multi-channel updates," in *Proc. 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*, Los Angeles, CA, USA, Nov. 2022, pp. 1679–1693, doi:10.1145/3548606.3559260.

[23] K. Gai, Y. Guo, J. Yu, W. Chan, L. Zhu, Y. Zhang, and W. Meng, "CAPE: Commitment-based privacy-preserving payment channel scheme in blockchain," *IEEE Transactions on Dependable and Secure Computing*, vol. 2025, pp. 3977–3992, 2025, doi:10.1109/TDSC.2025.3542906.

[24] C. Wang, Y. Long, X. Xu, S.-F. Sun, Y. Liu, and D. Gu, "UniCross: A universal cross-chain payment protocol with on-demand privacy and high scalability," IACR Cryptology ePrint Archive, Report 2025/1554, 2025. [Online]. Available: https://eprint.iacr.org/2025/1554. Accessed: Jan. 5, 2026.

[25] G. Avarikioti, C. Badertscher, M. Gabel, and R. Wattenhofer, "Watchtowers: Outsourcing the Monitoring of Payment Channels," arXiv:1811.12740, 2018.

[26] R. Canetti, "Universally Composable Security: A New Paradigm for Cryptographic Protocols," in *Proc. 42nd IEEE Symp. Foundations of Computer Science (FOCS)*, Las Vegas, NV, USA, 2001, pp. 136–145.

[27] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Yellow Paper, 2014. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[28] Foundry Contributors, "Foundry: A blazing fast, portable and modular toolkit for Ethereum application development," *GitHub repository*. [Online]. Available: https://github.com/foundry-rs/foundry. Accessed: 2025.

[29] V. Buterin and A. Beregszaszi, “EIP-170: Contract code size limit,” Ethereum Improvement Proposals, Nov. 2016. [Online]. Available: https://eips.ethereum.org/EIPS/eip-170

[30] J. Groth, “On the Size of Pairing-Based Non-interactive Arguments,” in *Proc. Advances in Cryptology – EUROCRYPT 2016 (Part II)*, ser. LNCS, vol. 9666. Berlin, Germany: Springer, 2016, pp. 305–326, doi: 10.1007/978-3-662-49896-5_11.