A Scalable Virtual Channel with Privacy Protection and Parallel Processing for CPSS Payments

Songyou Xie, Wei Liang, Kun Xie, Xiong Li, Kuanching Li, Weizhi Meng

***Abstract*—Blockchain has emerged as one of the ideal payment solutions in Cyber-Physical-Social Systems (CPSS) due to its distributed network architecture, data validation, and recording mechanism. Nevertheless, the block size and consensus mechanism of blockchain significantly limit its transaction throughput, resulting in a slow transaction confirmation process. Although traditional Payment Channel Network (PCN) schemes can enhance the confirmation speed of blockchain transactions, these schemes require intermediaries to validate and route each payment, resulting in high fees and a risk of privacy leakage. Due to the abovementioned issues, we propose a lightweight Scalable Virtual Channel (SVC) construction that eliminates the reliance on intermediaries during the payment process by utilizing contracts. The constructed SVC splits the primary payment channel into two sub-channels that can operate concurrently, thereby enhancing transaction efficiency and privacy; also, it sets the lifetime of the virtual channel using adjustable time locks, eliminating the need for continuous online monitoring by users. Formal security proof demonstrates that the proposed SVC ensures balance security, atomicity, and transaction privacy. Performance analysis reveals that SVC reduces communication overhead by approximately 50% during the open phase and by approximately 15% during the close phase, compared to other lightning channel-based virtual channel constructions.**

***Index Terms*—CPSS, Blockchain, Virtual Channel, Smart Payments, Distributed Network Architecture.**

I. Introduction

Cyber-Physical-Social Systems (CPSS) integrate network systems, physical systems, and social systems, representing a promising interdisciplinary research field. Through sensor networks and actuators, CPSS enables the monitoring, analysis, and optimization of complex systems [1]. However, payments in CPSS still need to address the potential trust risks and limitations associated with centralized payment systems [2].

Songyou Xie and Kun Xie are with the College of Computer Science and Electronic Engineering, Hunan University, China. Email:{syxie, cskxie}@hnu.edu.cn

Wei Liang is with the School of Computer/School of Software, University of South China, China, the School of Computer Science and Engineering, Hunan University of Science and Technology, China, and the Hunan Engineering Research Center of Software Design and Services of Intelligent Connected Vehicles, China. Email: 2025001208@usc.edu.cn

Xiong Li is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, China, and Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, China. E-mail: lixiongzhq@163.com

Kuanching Li is with the School of Computer Science and Engineering, Hunan University of Science and Technology, China, and the Hunan Engineering Research Center of Software Design and Services of Intelligent Connected Vehicles, China. Email: aliric@hnust.edu.cn

Weizhi Meng is with the School of Computing and Communications, Lancaster University, United Kingdom. Email: w.meng3@lancaster.ac.uk

(Corresponding authors: Kuanching Li, Wei Liang.)

Traditional CPSS payment systems rely on intermediaries to provide payment services. These intermediaries typically charge high transaction and withdrawal fees, thereby increasing users' costs. Blockchain's distributed architecture and decentralized payment platforms eliminate intermediaries while ensuring openness, transparency, and immutability [3]. Moreover, smart contracts in blockchain systems can automatically execute operations according to predefined terms, thereby enhancing the intelligence and automation level of CPSS . Consequently, blockchain is regarded as one of the ideal solutions for CPSS payments [4], [5].

However, blockchain systems require the entire network to reach consensus to validate each transaction, which severely limits transaction throughput and increases confirmation latency [6]. A payment channel enables two users to conduct instant off-chain payments by locking on-chain collateral, addressing the aforementioned issues. In recent years, Payment Channel Networks (PCNs) [7], [8] have been proposed to allow users to make off-chain payments even without a direct payment channel, thereby reducing the overhead of opening channels. The Lightning Network (LN) is a common Bitcoin PCN construction that allows users to make payments via intermediaries without directly opening payment channels [9]. As illustrated in Fig. 1, suppose Alice plans to make an off-chain payment to Bob, but there is no direct payment channel between them. However, there are payment channels between Alice and Hub, and between Hub and Bob. By constructing a Hash Time-Locked Contract (HTLC), Alice can pay Bob via Hub. To incentivize Hub to participate, Alice pays a fee. However, PCNs rely on intermediaries to validate and route each payment, potentially exposing transaction details and frequency to intermediaries [10]–[12].

Subsequently, researchers subsequently proposed the construction of virtual channels by locking the state on the PCN [13]. Once the virtual channel is built, the sender can make off-chain payments to the receiver without interacting with intermediaries, thereby addressing the aforementioned issues. Virtual channels can be easily implemented on the blockchain using Turing-complete scripts, such as those found in Ethereum [14]. Because its local state tree can store the off-chain payment states corresponding to different version numbers, the latest state can be selected as the valid state in the event of a conflict [13]. However, such virtual channels present challenges for certain blockchains, such as Bitcoin, which is based on the Unspent Transaction Output (UTXO) model. The UTXO model only supports a limited set of transaction logic

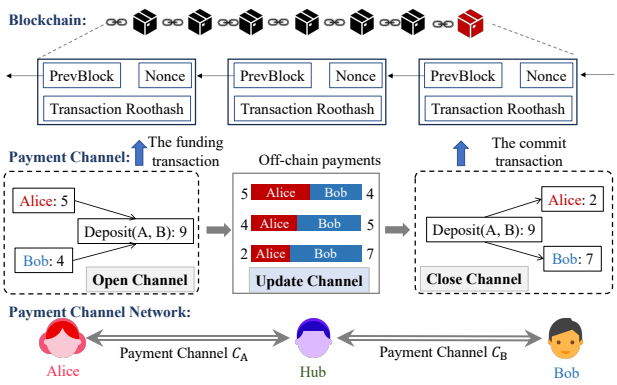


Fig. 1: Traditional payment channel network construction.

and cannot express complex on-chain conditional branches, loops, or state updates. Thus, the restricted scripting capabilities make it difficult to implement such virtual channels on Bitcoin.

A. Contributions

To address the aforementioned challenges, we propose a Scalable Virtual Channel (SVC) construction for CPSS payments, specifically designed for UTXO-based blockchains with restricted scripting capabilities. The main contributions of this work are as follows:

- To propose a virtual channel construction for CPSS aimed at protecting transaction privacy and reducing fees.
- To propose a channel split mechanism that divides the primary payment channel into two sub-channels to support parallel operation, allowing multiple payments to be executed independently and concurrently.
- In the design of the virtual channel, absolute and relative timelocks are combined to encode transaction order and timeout logic, ensuring that honest users can safely settle even when offline or facing malicious users.
- Experimental analyses on multiple Lightning Network instances and the Bitcoin testnet indicate that SVC exhibits advantages in both security and overhead.

The remainder of this paper is organized as follows. Sec. II reviews related work, Sec. III presents preliminaries, Sec. IV proposes the scalable virtual channel, Sec. V defines the security model, Sec. VI analyzes SVC security, Sec. VII evaluates performance, and Sec. VIII concludes.

II. Related Work

Intelligent devices in CPSS [15] can leverage blockchain to facilitate machine-to-machine (M2M) and machine-to-person (M2P) payments on behalf of users [16], [17]. However, with the rapid increase in payment volume and frequency within CPSS, traditional blockchain payment protocols struggle to meet the requirements for real-time and low-cost payment [18], [19]. Consequently, researchers have proposed off-chain payment protocols, including PCNs and virtual channels.

However, a PCN [5], [7] requires intermediaries to be online and participate in each payment process [20]. In Ethereum, Dziembowski *et al.* introduced a virtual channel protocol for Turing-complete smart contracts [13]. Nevertheless, this protocol is not suitable for script-constrained blockchains. Jia *et al.* illustrated a cross-chain virtual channel scheme to realize cross-chain payments and enhance the privacy of payments [21]. However, it also requires blockchain support in a Turing-complete scripting language. Jourenko *et al.* first presented a slim virtual channel protocol for script-constrained blockchains [22]. However, this protocol utilizes staggered time-locks for closing virtual channels, which may expose it to malicious locking attacks. Aumayr *et al.* proposed a virtual channel construction built over generalized channels called BCVC-V [10]. The BCVC-V construction uses hash time locks to secure collateral for virtual channel payments. However, this construction is built over generalized channels and requires a specific signing mechanism, which imposes higher demands on the deployment environment. Ref. [10] also introduced a virtual channel construction based on lightning channels, called LNVC-V. Nevertheless, LNVC-V requires the creation of funding transactions for the virtual channel, which involves various transaction combinations within each underlying channel, resulting in increased communication overhead for the system. Subsequently, Aumayr *et al.* [11] introduced a multi-hop virtual channel structure termed Donner, which is a virtual channel jointly funded by the sender. However, in its single-instance construction, Donner does not support bidirectional payments, which limits its scalability.

In summary, existing off-chain payment schemes have shortcomings in addressing issues such as privacy leakage and limited scalability. Therefore, this work introduces a scalable virtual channel construction to address these issues.

TABLE I: Notations used.

Notations	Description
$\mathcal{PCH} := (\mathbb{N}, \mathbb{PC})$	Bidirected graph with the user set $\mathbb{N}$ and channel set $\mathbb{PC}$
$\mathbb{VC}$	Set of constructed virtual channels
C_{ij}	Payment channel between users i and j
SC_{ij}	Sub-payment channel between users i and j
$\overline{\mathsf{tx}}$	Old transaction.
γ	Virtual channel controlled by A and B
C_{ij}.users	Users of payment channel C_{ij} (i.e., i and j)
γ.users	Users of virtual channel γ (i.e., A, H, and B)
γ.endUsers	End users of virtual channel γ (i.e., A and B)
τ	Maximum rounds required for tx to be recorded on-chain
Γ	Lifetime of the virtual channel γ
Δ_{x}	Maximum number of rounds required for phase x
$\wedge$	Logical symbol AND
pk_i	Spending conditions corresponding to user i's signature
$\sigma_{ij}(\mathsf{tx})$	Multi-signature of users i and j for transaction tx
$\sigma_i(\mathsf{tx})$	Single signature of user i for transaction tx
$\mathcal{L}$	Global ledger
$\mathcal{F}_{svc}$	Ideal functionality for SVC
$\mathcal{F}_{smt}$	Ideal functionality for secure message transmission
$\mathcal{Z}, \mathcal{A}, \mathcal{S}im$	PPT environment, adversary, and simulator

III. Preliminaries

In this section, we describe the preliminaries necessary to understand SVC, with Table I displaying the notation used.

A. Payment Channel Hub

This work considers payment channel networks based on a Payment Channel Hub (PCH), in which only a single intermediary acts as the payment hub [23], [24].

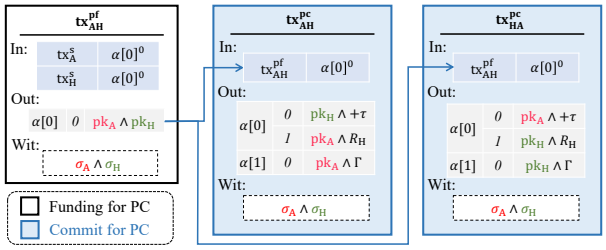


Fig. 2: Lightning style payment channel. We use different colors to represent information related to different users: red for user Alice and olive for user Hub.

Definition 1. *A $\mathcal{PCH}$ can be formally defined as a directed graph $\mathcal{PCH} := (\mathbb{N}, \mathbb{PC})$, where $\mathbb{N}$ denotes the set of participating users with $|\mathbb{N}| = 3$, and $\mathbb{PC}$ denotes the set of participating payment channels with $|\mathbb{PC}| = 2$. Each payment channel in $\mathbb{PC}$ is defined as a tuple $\mathsf{C}_{ij} := (\mathsf{id}, \mathsf{users}, \mathsf{st}, \mathsf{t}, \mathsf{FTx}, \mathcal{CT}, \mathcal{CT}^{\mathsf{old}})$, where id represents the channel's identifier, $\mathsf{users} := (i, j)$ represents the users of C_{ij}, $\mathsf{st} := \{(\alpha_i, \phi_i), (\alpha_j, \phi_j)\}$ represents the channel state of C_{ij}, t represents the lifetime, FTx is the funding transaction of C_{ij}. We denote by $\mathcal{CT} := \{\mathsf{tx}^{\mathsf{pc}}_{ij}, \mathsf{tx}^{\mathsf{pc}}_{ji}\}$ the set of the latest commitment transactions, and by $\mathcal{CT}^{\mathsf{old}} := \{((\overline{\mathsf{tx}^{\mathsf{pc}}_{ij}}, r_i), (\overline{\mathsf{tx}^{\mathsf{pc}}_{ji}}, r_j))\}$ the set of all old commitment transactions.*

B. UTXO-based Blockchains

In this model, each transaction output can specify multiple spending conditions, but can be spent only once by subsequent transactions. For clarity, we express a transaction as the attribute tuple $\mathsf{tx} := (\mathsf{id}, \mathsf{In}, \mathsf{Out}, \mathsf{Wit})$. The id represents the unique identifier of tx.

A valid transaction can be stored in a public ledger $\mathcal{L}$, namely blockchain. We define the inputs of a transaction as $\mathsf{tx.In} := \{\vartheta_0, \vartheta_1, ..., \vartheta_n\}$. Each transaction input is defined as $\vartheta_i := (\mathsf{ptx}.\alpha)$, where ptx is the previous transaction and α indexes its output. The outputs of a transaction are defined as $\mathsf{tx.Out} := \{\theta_0, \theta_1, ..., \theta_n\}$. Each output is defined as $\theta_i := (\mathsf{vout}, \phi)$, where vout is the output value and ϕ specifies the spending condition. These conditions may include any set of blockchain-supported scripts. For simplicity, we denote the j-th spending condition of the i-th output in tx by $\mathsf{tx}.\alpha[i]^j$, where $i, j \in \mathbb{N}_0$ are zero-based indices and α represents the associated funds. In UTXO-based blockchains, the output scripts of a transaction can include an absolute timelock $\mathsf{ATL}(\Gamma)$ (or simply Γ), meaning that the condition becomes valid only after Γ rounds have passed since the blockchain was created. Additionally, a relative timelock $\mathsf{RTL}(\tau)$ (or simply $+\tau$) can be specified, indicating that the condition becomes valid only after at least τ times have elapsed since the transaction was stored on $\mathcal{L}$. The attribute tx.Wit is a list of witnesses associated with tx.

C. Lightning-style Payment Channel

Fig. 2 shows a Lightning-style payment channel. In the input of the funding transaction $\mathsf{tx}^{\mathsf{pf}}_{\mathsf{AH}}$, $\mathsf{tx}^{\mathsf{s}}_{\mathsf{A}}.\alpha[0]^0$ and $\mathsf{tx}^{\mathsf{s}}_{\mathsf{H}}.\alpha[0]^0$ means that $\mathsf{tx}^{\mathsf{pf}}_{\mathsf{AH}}$ spends the outputs $\alpha[0]^0$ of transactions $\mathsf{tx}^{\mathsf{s}}_{\mathsf{A}}$ and $\mathsf{tx}^{\mathsf{s}}_{\mathsf{H}}$. $\mathsf{tx}^{\mathsf{pf}}_{\mathsf{AH}}$ also includes the witnesses corresponding to the spending condition, *i.e.*, providing the signatures of Alice and Hub. $\mathsf{tx}^{\mathsf{pf}}_{\mathsf{AH}}$ contains an output $\mathsf{tx}^{\mathsf{pf}}_{\mathsf{AH}}.\alpha[0]^0$, and its unique spending condition is $\langle \mathsf{pk}_{\mathsf{A}} \wedge \mathsf{pk}_{\mathsf{H}} \rangle$. $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ can spend $\mathsf{tx}^{\mathsf{pf}}_{\mathsf{AH}}.\alpha[0]^0$ after providing multi-signature for Alice and Hub. $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ contains two outputs, with $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}.\alpha[0]$ having two spending conditions: (i) $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}.\alpha[0]^0 := \langle \mathsf{pk}_{\mathsf{H}} \wedge +\tau \rangle$, which can be spent by Hub after the relative time-lock $+\tau$; (ii) $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}.\alpha[0]^1 := \langle \mathsf{pk}_{\mathsf{A}} \wedge R_{\mathsf{H}} \rangle$, which can be spent immediately by Alice if r_{H} is revealed, where $(R_{\mathsf{H}}, r_{\mathsf{H}}) \in R$. $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}.\alpha[1]^0$ can be spent by Alice after the absolute time-lock Γ. Similar to $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$, $\mathsf{tx}^{\mathsf{pf}}_{\mathsf{AH}}$ can also be spent by $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{HA}}$, depending on which transaction is stored on the ledger $\mathcal{L}$ first. Note that $\mathsf{tx}^{\mathsf{pf}}_{\mathsf{AH}}.\alpha[0]^0$ can only be spent once, *i.e.*, only one of $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ and $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{HA}}$ can be stored on $\mathcal{L}$.

D. Digital Signatures

A digital signature scheme Σ consists of the following algorithms $\Sigma := (\mathrm{KeyGen}, \mathrm{Sign}, \mathrm{Vrfy})$. Among them, $\mathrm{KeyGen}(\lambda)$ is the key generation algorithm, which takes the security parameter λ as input and outputs the public/private key pair $(\mathsf{pk}, \mathsf{sk})$. $\mathrm{Sign}(\mathsf{sk}, m)$ is the signature algorithm, input the private key sk and message $m \in \{0, 1\}^*$, and output the signature σ. In the verification algorithm $\mathrm{Vrfy}(\mathsf{pk}, m, \sigma)$, when σ is a valid signature in m with public key pk, the output is 1; otherwise, the output is 0. Our signature scheme security requirement is unforgeable under chosen message attacks [25].

IV. Proposed Construction

A. Security and Privacy Goals

Considering the security and privacy concerns that may arise in the virtual channel, we define the following goals for SVC.

(S1) Balance security. It ensures that the balances of honest users within channels are correctly distributed, while malicious users are punished and lose their funds. This mechanism allows off-chain transactions constructed by users to be enforced on-chain, ensuring that funds are allocated to honest users.

(S2) Atomicity. It guarantees that all payments are executed as an indivisible operation—either all succeed together or all fail together.

(P1) Transaction privacy. It ensures that once the virtual channel is constructed, the intermediary cannot directly access the subsequent transaction information of end users, including transaction numbers and amounts.

B. Formal Description of SVC

In this subsection, we introduce a virtual channel construction called SVC. It can be constructed over Lightning style payment channels (as illustrated in Fig. 2) and adapted to the LN and other payment networks with limited scripting capabilities. We formalize the proposed SVC in the $(\mathcal{F}_{smt}, \mathcal{F}_{syn}, \mathcal{G}_{\mathcal{L}})$-hybrid model (as defined in Section V-A).

Suppose that Alice and Bob don't have a direct payment channel built on-chain, but they both construct a direct payment channel with the intermediary Hub. Let C_{A} and C_{B} denote the payment channels established between Alice and Hub, and between Hub and Bob, respectively. A virtual channel γ can be defined as a tuple

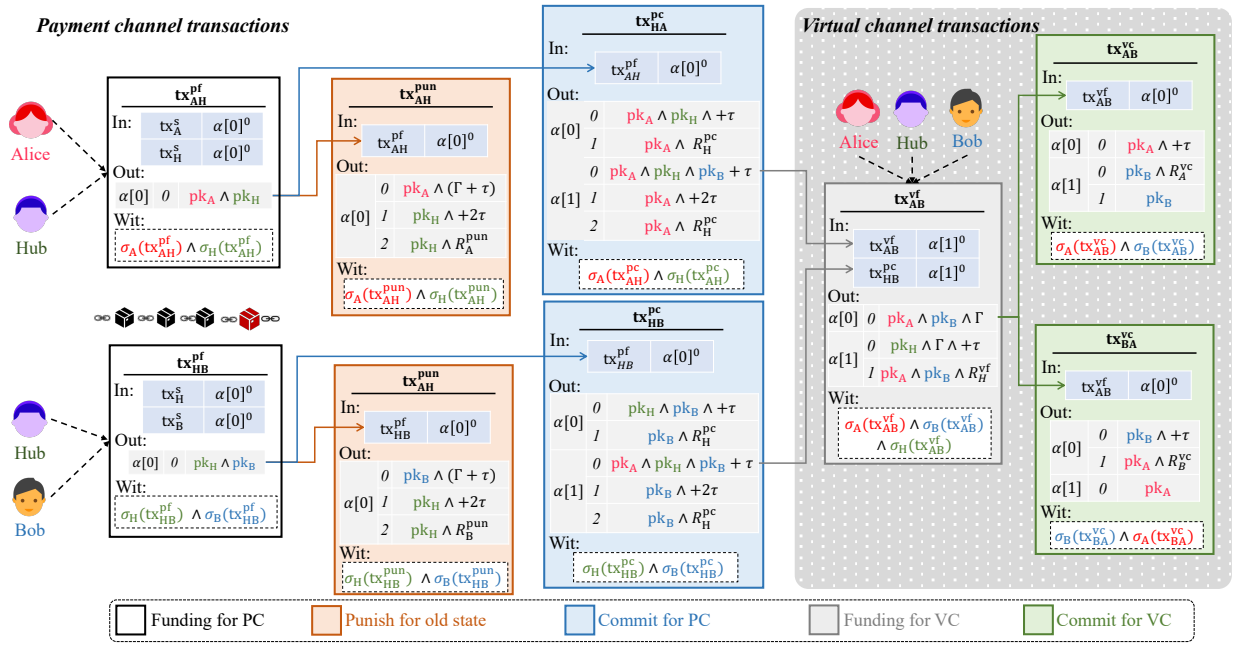


Fig. 3: Transaction contracts for setting up the virtual channel.

$\gamma := \{\mathsf{id}, \mathsf{users}, \mathsf{endUsers}, \mathsf{st}, \Gamma, \mathsf{subchan}, \mathsf{fee}\}$. The attribute $\gamma.\mathsf{users} := \{\mathsf{Alice}, \mathsf{Hub}, \mathsf{Bob}\}$ denotes the user set of γ, $\gamma.\mathsf{endUsers} := \{\mathsf{Alice}, \mathsf{Bob}\}$ denotes the end user set of γ. The attribute $\gamma.\mathsf{st}$ consists of a series of outputs and represents the state of γ , $\gamma.\Gamma$ is an absolute time-lock used to represent the lifetime of γ. The attributes $\gamma.\mathsf{subchan} := \{\mathsf{C_A}, \mathsf{C_B}\}$ and $\gamma.\mathsf{fee}$ respectively represent the set of underlying payment channels and the fee paid to Hub. We use $\mathsf{ATL}(\Gamma)$ (or simply Γ) and $\mathsf{RTL}(\tau)$ (or simply $+\tau$) to denote absolute and relative timelocks, respectively, and $\mathsf{Sig}(\mathsf{P})$ (or simply pk_P) and $\mathsf{MSig}(\mathsf{P}, \mathsf{Q})$ (or simply $\mathsf{pk_P} \wedge \mathsf{pk_Q}$) to denote single-signature and multi-signature conditions, respectively.

For clarity, Fig. 3 illustrates the transaction contracts required to construct the virtual channel γ, while Fig. 4 presents the corresponding transaction construction procedure. We provide a detailed description of each transaction within the contract as we discuss the different phases of SVC. SVC consists of five phases: **Open**, **Update**, **Release**, **Offload**, and **Close**. In these phases, Alice in $\mathsf{C_A}$ and Bob in $\mathsf{C_B}$ perform similar operations, including transaction construction, transaction signing, and message transmission. We define Δ_{op} as the maximum allowable time interval for identical instruction messages to reach Hub. To ensure balanced security, identical instruction messages from Alice and Bob must be delivered to Hub within Δ_{op}; otherwise, Hub will abort the phase.

*1) **Open**:* In the open phase, seven transactions need to be constructed using the transaction construction procedure shown in Fig. 4, where $\mathsf{tx}^{\mathsf{pf}}_{\mathsf{AH}}$ and $\mathsf{tx}^{\mathsf{pf}}_{\mathsf{HB}}$ respectively serve as the funding transactions for channels $\mathsf{C_A}$ and $\mathsf{C_B}$ (having the same structure as $\mathsf{tx}^{\mathsf{pf}}_{\mathsf{AH}}$ in Fig. 2). Fig. 5 illustrates the creation and signing process of transactions in $\mathsf{C_A}$ during the open phase.

In Step 1 of Fig. 5, Alice sends the revocation public value $R^{\mathsf{pun}}_{\mathsf{A}}$, the virtual channel γ, and the transaction $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}$ to Hub. Specifically, Alice generates

$$(R_{\mathsf{A}}, r_{\mathsf{A}}) \leftarrow \mathsf{GenR},$$
$$\mathsf{st}^{\mathsf{pun}} := \{(\alpha[0], \langle \mathsf{Sig}(\mathsf{A}) \wedge \mathsf{ATL}(\Gamma + \tau)\rangle^0, \langle \mathsf{Sig}(\mathsf{H}) \wedge \mathsf{RTL}(2\tau)\rangle^1, \langle \mathsf{Sig}(\mathsf{H}) \wedge \mathsf{Wit}(R^{\mathsf{pun}}_{\mathsf{A}})\rangle^2)\}, \quad (1)$$

where GenR is a Probabilistic Polynomial-Time (PPT) sampling algorithm that, on input 1^n, outputs a public/secret pair $(R_A, r_A) \in \mathcal{R}$, where $\mathcal{R}$ denotes a hard relation [26]. Then Alice invokes $\mathtt{GenTxPun}$ with $(\mathsf{C_A}, \mathsf{A}, \mathsf{tx}^{\mathsf{pf}}_{\mathsf{AH}}, R^{\mathsf{pun}}_{\mathsf{A}}, \tau, \Gamma, \mathsf{st}^{\mathsf{pun}})$ to generate the punishment transaction $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}$. The $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}$ sets the spending conditions of its outputs with different time-locks, thereby preventing a malicious Alice from prematurely broadcasting $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}$ or preventing Hub from withholding $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$, which could invalidate the virtual channel.

The output $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}.\alpha[0]$ contains one output with three spending conditions: (i) Alice can spend $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}.\alpha[0]^0$ after absolute time-lock $\Gamma + \tau$ has expired; (ii) Hub can spend $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}.\alpha[0]^1$ once the relative time-lock $+2\tau$ has expired; and (iii) if Hub knows the revocation secret value $r^{\mathsf{pun}}_{\mathsf{A}}$, it can spend $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}.\alpha[0]^2$ immediately after the transaction is recorded on-chain.

In Step 2 of Fig. 5, Alice receives the revocation public value $R^{\mathsf{pc}}_{\mathsf{H}}$ from Hub and sends the transaction $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ to Hub. Specifically, Alice generates

$$\mathsf{st}^{\mathsf{pc}} := \{(\alpha[0], \langle \mathsf{MSig}(\mathsf{A}, \mathsf{H}) \wedge \mathsf{RTL}(\tau)\rangle^0, \langle \mathsf{Sig}(\mathsf{A}) \wedge \mathsf{Wit}(R^{\mathsf{pc}}_{\mathsf{H}})\rangle^1), (\alpha[1], \langle \mathsf{MSig}(\mathsf{A}, \mathsf{H}, \mathsf{B}) \wedge \mathsf{RT}(\tau)\rangle^0, \langle \mathsf{Sig}(\mathsf{A}) \wedge 2\tau\rangle^1, \langle \mathsf{Sig}(\mathsf{A}) \wedge \mathsf{Wit}(R^{\mathsf{pc}}_{\mathsf{H}})^2\rangle)\} \quad (2)$$

and calls the procedure $\mathtt{GenTxPc}(\mathsf{C_A}, \mathsf{A}, \mathsf{H}, \mathsf{tx}^{\mathsf{pf}}_{\mathsf{AH}}, R^{\mathsf{pc}}_{\mathsf{H}}, \tau, \mathsf{st}^{\mathsf{pc}})$ to create the transaction $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ (similarly, on the $\mathsf{C_B}$ side, Bob constructs $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}$). The transaction $\mathsf{tx}^{\mathsf{pf}}_{\mathsf{AH}}$ is an on-chain funding transaction, and it serves as the input transaction for both $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ and $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}$. The output $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}.\alpha[0]$ contains two spending conditions: (i) $\langle \mathsf{pk_A} \wedge \mathsf{pk_H} \wedge +\tau\rangle^0$ is used to construct the sub-payment channel $\mathsf{SC_{AH}}$ between Alice and Hub; and (ii) $\langle \mathsf{pk_A} \wedge R^{\mathsf{pc}}_{\mathsf{H}}\rangle^1$ incorporates the revocation public value $R^{\mathsf{pc}}_{\mathsf{H}}$

$\texttt{GenTxPc}(\mathsf{ch}, \mathsf{P}, \mathsf{Q}, \mathsf{tx}^{\mathsf{pf}}_{\mathsf{PH}}, R^{\mathsf{pc}}_{\mathsf{H}}, \tau, \mathsf{st}^{\mathsf{pc}})$:

1. For $\mathsf{P}, \mathsf{Q} \in \{\mathsf{A}, \mathsf{B}\}$ with $\mathsf{P} \neq \mathsf{Q}$ and $\mathsf{ch} \in \{\mathsf{C}_\mathsf{P}, \mathsf{C}_\mathsf{Q}\}$, parse $\mathsf{st}^{\mathsf{pc}}$ as $\{\theta_0, \theta_1\}$, where
   - $\theta_0 := (\alpha[0], \langle \mathsf{MSig}(\mathsf{P}, \mathsf{H}) \wedge \mathsf{RTL}(\tau)\rangle^0, \langle \mathsf{Sig}(\mathsf{P}) \wedge \mathsf{Wit}(R^{\mathsf{pc}}_{\mathsf{H}})\rangle^1)$.
   - $\theta_1 := (\alpha[1], \langle \mathsf{MSig}(\mathsf{P}, \mathsf{H}, \mathsf{Q}) \wedge \mathsf{RTL}(\tau)\rangle^0, \langle \mathsf{Sig}(\mathsf{P}) \wedge \mathsf{RTL}(2\tau)\rangle^1, \langle \mathsf{Sig}(\mathsf{P}) \wedge R^{\mathsf{pc}}_{\mathsf{H}}\rangle^2)$.
2. Let $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{PH}}$ denote the commitment transaction of ch, where $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{PH}}.\mathsf{Out} := \mathsf{st}^{\mathsf{pc}}$, and set $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{PH}}.\mathsf{In} := \{\mathsf{tx}^{\mathsf{pf}}_{\mathsf{PH}}.\alpha[0]^0\}$.
3. Return $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{PH}}$.

$\texttt{GenTxPun}(\mathsf{ch}, \mathsf{P}, \mathsf{tx}^{\mathsf{pf}}_{\mathsf{PH}}, R_\mathsf{P}, \tau, \Gamma, \mathsf{st}^{\mathsf{pun}})$:

1. For $\mathsf{P}, \mathsf{Q} \in \{\mathsf{A}, \mathsf{B}\}$ with $\mathsf{P} \neq \mathsf{Q}$ and $\mathsf{ch} \in \{\mathsf{C}_\mathsf{P}, \mathsf{C}_\mathsf{Q}\}$, parse $\mathsf{st}^{\mathsf{pun}}$ as $\{\theta_0\}$, where
   - $\theta_0 := (\alpha[0], \langle \mathsf{Sig}(\mathsf{P}) \wedge \mathsf{ATL}(\Gamma + \tau)\rangle^0, \langle \mathsf{Sig}(\mathsf{H}) \wedge \mathsf{RTL}(2\tau)\rangle^1, \langle \mathsf{Sig}(\mathsf{H}) \wedge \mathsf{Wit}(R^{\mathsf{pun}}_{\mathsf{P}})\rangle^2)$.
2. Let $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{PH}}$ denote the punishment transaction of channel ch, where $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{PH}}.\mathsf{Out} := \mathsf{st}^{\mathsf{pun}}$, and set $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{PH}}.\mathsf{In} := \{\mathsf{tx}^{\mathsf{pf}}_{\mathsf{PH}}.\alpha[0]^0\}$.
3. Return $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{PH}}$.

$\texttt{GenTxVf}(\gamma, \mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}, R^{\mathsf{vf}}_{\mathsf{H}}, \mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}, \mathsf{st}^{\mathsf{vf}})$:

1. Parst $\mathsf{st}^{\mathsf{vf}}$ as $\{\theta_0, \theta_1\}$, where
   - $\theta_0 := (\alpha[0], \langle \mathsf{MSig}(\mathsf{A}, \mathsf{B}) \wedge \mathsf{ATL}(\Gamma)\rangle^0)$
   - $\theta_1 := (\alpha[1], \langle \mathsf{Sig}(\mathsf{H}) \wedge \mathsf{ATL}(\Gamma) \wedge \mathsf{RT}(\tau)\rangle^0, \langle \mathsf{MSig}(\mathsf{A}, \mathsf{B}) \wedge \mathsf{Wit}(R^{\mathsf{vf}}_{\mathsf{H}})\rangle^1)$
2. Let $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$ denote the funding transaction of γ, where $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}.\mathsf{Out} := \mathsf{st}^{\mathsf{vf}}$, set $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}.\mathsf{In} := \{(\mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}.\alpha[1]^0), (\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}.\alpha[1]^0)\}$.
3. Return $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$.

$\texttt{GenTxc}(\mathsf{ch}, \mathsf{P}, \mathsf{Q}, R_\mathsf{P}, R_\mathsf{Q}, \mathsf{st}^c, \tau)$:

1. For $\mathsf{P}, \mathsf{Q} \in \{\mathsf{A}, \mathsf{B}\}$ with $\mathsf{P} \neq \mathsf{Q}$ and $\mathsf{ch} \in \{\alpha, \beta, \gamma\}$, parse st^c as$\{\theta_0, \theta_1\}$ and $\{\theta_2, \theta_3\}$, where
   - $\theta_0 := (\alpha[0], \langle \mathsf{Sig}(\mathsf{P}) \wedge \mathsf{RTL}(\tau)\rangle^0, \langle \mathsf{Sig}(\mathsf{Q}) \wedge \mathsf{Wit}(R_\mathsf{P})\rangle^1)$.
   - $\theta_1 := (\alpha[1], \langle \mathsf{Sig}(\mathsf{Q})\rangle^0)$.
   - $\theta_2 := (\alpha[0], \langle \mathsf{Sig}(\mathsf{Q}) \wedge \mathsf{RTL}(\tau)\rangle^0, \langle \mathsf{Sig}(\mathsf{P}) \wedge \mathsf{Wit}(R_\mathsf{Q})\rangle^1)$.
   - $\theta_3 := (\alpha[1], \langle \mathsf{Sig}(\mathsf{P})\rangle^0)$.
2. Let $\mathsf{ftx} := \mathsf{ch}.\mathsf{FTx}$ is the funding transaction of ch. Let $\mathsf{tx}_{\mathsf{PQ}}$ denote one of the commitment transactions of ch, where $\mathsf{tx}_{\mathsf{PQ}}.\mathsf{Out} := (\theta_0, \theta_1)$, and set $\mathsf{tx}_{\mathsf{PQ}}.\mathsf{In} := (\mathsf{ftx}.\mathsf{id}, \mathsf{ftx}.\alpha[0^0])$; Let $\mathsf{tx}_{\mathsf{QP}}$ denote another commitment transaction of ch, where $\mathsf{tx}_{\mathsf{QP}}.\mathsf{Out} := (\theta_2, \theta_3)$, and set $\mathsf{tx}_{\mathsf{QP}}.\mathsf{In} := \{\mathsf{ftx}.\alpha[0^0]\}$.
3. Return $(\mathsf{tx}_{\mathsf{PQ}}, \mathsf{tx}_{\mathsf{QP}})$.

Fig. 4: Transaction construction programs for SVC. Let Alice, Hub, and Bob be denoted by A, H, and B, respectively.

generated by Hub. Once Hub reveals the value $r^{\mathsf{pc}}_{\mathsf{H}}$ to Alice, where $(r^{\mathsf{pc}}_{\mathsf{H}}, R^{\mathsf{pc}}_{\mathsf{H}}) \in R$, and $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ has been recorded on-chain, Alice can immediately claim all the funds in $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}.\alpha[0]$.

The output $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}.\alpha[1]$ contains three spending conditions: (i) $\langle \mathsf{pk}_\mathsf{A} \wedge \mathsf{pk}_\mathsf{H} \wedge \mathsf{pk}_\mathsf{B} \wedge \tau\rangle^0$ locks the collateral of Alice and Hub; (ii) Once $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ is recorded on-chain, if Hub fails to publish $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$ within 2τ rounds, Alice can claim the funds in $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}.\alpha[1]$ via the condition $\langle \mathsf{pk}_\mathsf{A} \wedge +2\tau\rangle^1$; and (iii) If Hub broadcasts an old version of $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ (in which $r^{\mathsf{pc}}_{\mathsf{H}}$ has been revealed), Alice can spend the funds in $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}.\alpha[1]$ using the condition $\langle \mathsf{pk}_\mathsf{A} \wedge R^{\mathsf{pc}}_{\mathsf{H}}\rangle^2$.

In Step 3 of Fig. 5, upon receiving the message from Alice $(\mathsf{init\text{-}open}, \mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}, \gamma)$ (similarly, on the C_B side, Hub receives $(\mathsf{sid}, \mathsf{init\text{-}open}, \mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}, \gamma)$), Hub generates

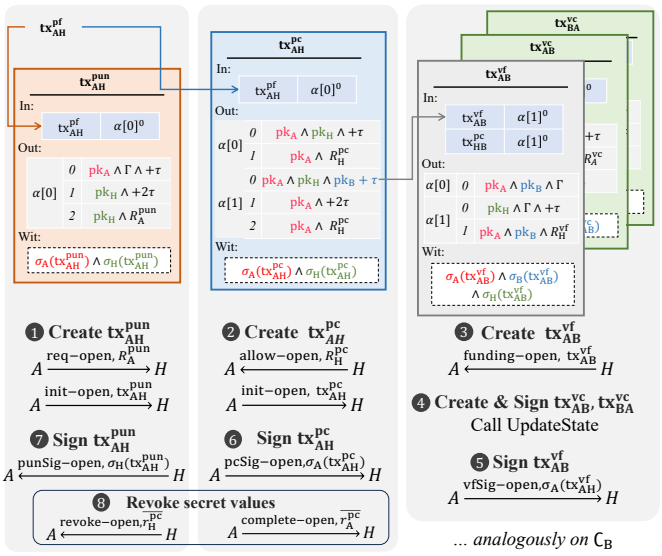


Fig. 5: Illustration of γ open phase on channel C_A.

$$\mathsf{st}^{\mathsf{vf}} := \{(\alpha[0], \langle \mathsf{MSig}(\mathsf{A}, \mathsf{B}) \wedge \mathsf{ATL}(\Gamma)\rangle^0), (\alpha[1], \langle \mathsf{Sig}(\mathsf{H}) \wedge \mathsf{ATL}(\Gamma) \wedge \mathsf{RTL}(\tau)\rangle^0, \langle \mathsf{MSig}(\mathsf{A}, \mathsf{B}) \wedge \mathsf{Wit}(R^{\mathsf{vf}}_{\mathsf{H}})\rangle^1)\} \tag{3}$$

and invokes $\texttt{GenTxVf}(\gamma, \mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}, R^{\mathsf{vf}}_{\mathsf{H}}, \mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}, \mathsf{st}^{\mathsf{vf}})$ to construct the transaction $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$. The transaction $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$ contains two outputs: (i) $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}.\alpha[0]^0$ serves as the fund output for virtual channel γ; and (ii) $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}.\alpha[1]$ contains Hub's collateral and fees. Hub then sends $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$ to Alice.

In Step 4 of Fig. 5, Alice calls `UpdateState` function with $(\mathsf{A}, \mathsf{B}, \mathsf{st}^{\mathsf{init}}, \gamma, \tau, b{=}0, \text{-}, \text{-})$. If it returns $(\mathsf{tx}^{\mathsf{vc}}_{\mathsf{AB}}, \mathsf{tx}^{\mathsf{vc}}_{\mathsf{BA}})$, Alice generates the signatures in Steps 5–6 as $\sigma_\mathsf{A}(\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}) \leftarrow \mathsf{Sign}(\mathsf{sk}_\mathsf{A}, \mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}})$ and $\sigma_\mathsf{A}(\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}) \leftarrow \mathsf{Sign}(\mathsf{sk}_\mathsf{A}, \mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}})$, and then sends $(\mathsf{vfSig\text{-}open}, \sigma_\mathsf{A}(\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}))$ and $(\mathsf{vfSig\text{-}open}, \sigma_\mathsf{A}(\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}))$ to Hub. In Step 7, Hub computes $\sigma_\mathsf{H}(\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}) \leftarrow \mathsf{Sign}(\mathsf{sk}_\mathsf{H}, \mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}})$ and sends it to Alice. Finally, all users exchange the revocation keys of the old commitment transactions $(\overline{\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}}, \overline{\mathsf{tx}^{\mathsf{pc}}_{\mathsf{HA}}})$. If no failures occur, the γ is considered successfully opened.

In the proposed SVC, only Hub possesses the fully signed transactions $(\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}, \mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}, \mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}})$ and is authorized to publish them on-chain. Alice (Bob), on the other hand, holds the fully signed $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}$ ($\mathsf{tx}^{\mathsf{pun}}_{\mathsf{HB}}$); if Hub fails to publish $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ ($\mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}$) and $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$ on the ledger $\mathcal{L}$ before $\Gamma - \tau$, Alice (Bob) can penalize Hub using $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}$ ($\mathsf{tx}^{\mathsf{pun}}_{\mathsf{HB}}$).

*2) **Update**:* Suppose Alice intends to pay α tokens to Bob. To achieve this payment, Alice and Bob jointly call `UpdateState` (shown in Fig. 6) to update the state of γ. Let $(\overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{AH}}}, \overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{HA}}})$ denote the commitment transactions generated for the previous payment. Alice then generates

$$\mathsf{st}^* := \{(\theta_0, \theta_1), (\theta_2, \theta_3)\}, \tag{4}$$

where

- $\theta_0 := (\alpha[0], \langle \mathsf{Sig}(\mathsf{A}) \wedge +\tau\rangle^0, \langle \mathsf{Sig}(\mathsf{B}) \wedge \mathsf{Wit}(R_\mathsf{A})\rangle^1)$,
- $\theta_1 := (\alpha[1], \langle \mathsf{Sig}(\mathsf{B})\rangle^0)$,
- $\theta_2 := (\alpha[0], \langle \mathsf{Sig}(\mathsf{B}) \wedge +\tau\rangle^0, \langle \mathsf{Sig}(\mathsf{A}) \wedge \mathsf{Wit}(R_\mathsf{B})\rangle^1)$,
- $\theta_3 := (\alpha[1], \langle \mathsf{Sig}(\mathsf{A})\rangle^0)$.

UpdateState$(\mathsf{P}, \mathsf{Q}, \mathsf{st}^*, \mathsf{ch}, \tau, b, \mathsf{tx}_{\mathsf{PQ}}, \mathsf{tx}_{\mathsf{QP}})$

P

1. Generate $(R_{\mathsf{P}}, r_{\mathsf{P}}) \leftarrow \mathsf{GenR}$
2. Send $(\mathsf{req\text{-}update}, \mathsf{st}^*, R_{\mathsf{P}}) \hookrightarrow \mathsf{Q}$.

Q upon $(\mathsf{req\text{-}update}, \mathsf{st}^*, R_{\mathsf{P}}) \hookleftarrow \mathsf{P}$

3. Generate $(R_{\mathsf{Q}}, r_{\mathsf{Q}}) \leftarrow \mathsf{GenR}$.
4. $(\mathsf{tx}^*_{\mathsf{PQ}}, \mathsf{tx}^*_{\mathsf{QP}}) \coloneqq \mathsf{GenTXc}(\mathsf{ch}, \mathsf{P}, \mathsf{Q}, \mathsf{st}^*, \tau)$
5. $\sigma_{\mathsf{Q}}(\mathsf{tx}^*_{\mathsf{PQ}}) \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Q}}, \mathsf{tx}^*_{\mathsf{PQ}})$.
6. Send $(\mathsf{init\text{-}update}, \mathsf{tx}^*_{\mathsf{PQ}}, \mathsf{tx}^*_{\mathsf{QP}}, \sigma_{\mathsf{Q}}(\mathsf{tx}^*_{\mathsf{PQ}})) \hookrightarrow \mathsf{P}$.

P upon $(\mathsf{init\text{-}update}, \mathsf{tx}^*_{\mathsf{PQ}}, \mathsf{tx}^*_{\mathsf{QP}}, \sigma_{\mathsf{Q}}(\mathsf{tx}^*_{\mathsf{PQ}})) \hookleftarrow \mathsf{Q}$

7. $\sigma_{\mathsf{P}}(\mathsf{tx}^*_{\mathsf{QP}}) \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{P}}, \mathsf{tx}^*_{\mathsf{QP}})$
8. Send $(\mathsf{allow\text{-}update}, \sigma_{\mathsf{P}}(\mathsf{tx}^*_{\mathsf{QP}})) \hookrightarrow \mathsf{Q}$.

Q upon $(\mathsf{allow\text{-}update}, \sigma_{\mathsf{P}}(\mathsf{tx}^*_{\mathsf{QP}})) \hookleftarrow \mathsf{P}$

9. If b=0, $\mathsf{ch}.\mathcal{CT} \coloneqq \{\mathsf{tx}^*_{\mathsf{PQ}}, \mathsf{tx}^*_{\mathsf{QP}}\}$ and return $(\mathsf{tx}^*_{\mathsf{PQ}}, \mathsf{tx}^*_{\mathsf{QP}})$ and stop.
10. If b=1, extract the revocation key r_{Q} corresponding to $\mathsf{tx}_{\mathsf{QP}}$, send $(\mathsf{revoke\text{-}update}, r_{\mathsf{Q}}) \hookrightarrow \mathsf{P}$.

P upon $(\mathsf{revoke\text{-}update}, r_{\mathsf{Q}}) \hookleftarrow \mathsf{Q}$

11. Extract revocation key r_{P} corresponding to $\mathsf{tx}_{\mathsf{PQ}}$, send $(\mathsf{complete\text{-}update}, r_{\mathsf{P}}) \hookrightarrow \mathsf{Q}$.
12. Let $\mathsf{ch}.\mathcal{CT} \coloneqq \{\mathsf{tx}^*_{\mathsf{PQ}}, \mathsf{tx}^*_{\mathsf{QP}}\}$ and add $((\mathsf{tx}_{\mathsf{PQ}}, r_{\mathsf{P}}), (\mathsf{tx}_{\mathsf{QP}}, r_{\mathsf{Q}}))$ to $\mathsf{ch}.\mathcal{CT}^{\mathsf{old}}$.

Q upon $(\mathsf{complete\text{-}update}, r_{\mathsf{P}}) \hookleftarrow \mathsf{P}$

13. Let $\mathsf{ch}.\mathcal{CT} \coloneqq \{\mathsf{tx}^*_{\mathsf{PQ}}, \mathsf{tx}^*_{\mathsf{QP}}\}$ and add $((\mathsf{tx}_{\mathsf{PQ}}, r_{\mathsf{P}}), (\mathsf{tx}_{\mathsf{QP}}, r_{\mathsf{Q}}))$ to $\mathsf{ch}.\mathcal{CT}^{\mathsf{old}}$.
14. Return $(\mathsf{tx}^*_{\mathsf{PQ}}, \mathsf{tx}^*_{\mathsf{QP}})$ and stop.

Fig. 6: State update process of SVC. Assuming this process is initiated by P (symmetrical for Q).

Next, Alice calls UpdateState with $(\mathsf{A}, \mathsf{B}, \mathsf{st}^*, \gamma, \tau, b{=}1, \overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{AB}}}, \overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{BA}}})$. Specifically, in Steps 1-2, Alice generates $(R_{\mathsf{A}}, r_{\mathsf{A}}) \leftarrow \mathsf{GenR}$ and sends R_{A} and st^* to Bob. In Steps 3-6, Bob generates $(R_{\mathsf{B}}, r_{\mathsf{B}}) \leftarrow \mathsf{GenR}$ and transactions $(\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{AB}}, \mathsf{tx}^{\mathsf{vc}*}_{\mathsf{BA}})$, where $\overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{AB}}}.\alpha[0] - \mathsf{tx}^{\mathsf{vc}*}_{\mathsf{AB}}.\alpha[0] = \alpha$ and $\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{BA}}.\alpha[0] - \overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{BA}}}.\alpha[0] = \alpha$. This means that Bob will receive α tokens in this payment. Then Bob generates $\sigma_{\mathsf{B}}(\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{AB}}) \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{B}}, \mathsf{tx}^{\mathsf{vc}*}_{\mathsf{AB}})$ and sends $(\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{AB}}, \mathsf{tx}^{\mathsf{vc}*}_{\mathsf{BA}})$ along with $\sigma_{\mathsf{B}}(\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{AB}})$ to Alice. In Steps 7-8, Alice computes $\sigma_{\mathsf{A}}(\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{BA}}) \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{A}}, \mathsf{tx}^{\mathsf{vc}*}_{\mathsf{BA}})$ and sends $\sigma_{\mathsf{A}}(\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{BA}})$ to Bob.

If b = 0, define the set of latest commitment transactions $\mathcal{CT} \coloneqq \{\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{AB}}, \mathsf{tx}^{\mathsf{vc}*}_{\mathsf{BA}}\}$ and return $(\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{AB}}, \mathsf{tx}^{\mathsf{vc}*}_{\mathsf{BA}})$ in Step 9; otherwise, in Steps 10-12, Alice and Bob exchange revocation secret values for $(\overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{AB}}}, \overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{BA}}})$ to prevent any malicious user from broadcasting old commitment transactions on-chain. Finally, if $(\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{AB}}, \mathsf{tx}^{\mathsf{vc}*}_{\mathsf{BA}})$ is returned, it indicates that Alice has successfully paid α tokens to Bob, and the virtual channel state is updated. In the case of malicious behavior during the update phase: (i) if the signature exchange for the new transactions $(\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{AB}}, \mathsf{tx}^{\mathsf{vc}*}_{\mathsf{BA}})$ fails, Alice and Bob do not initiate the exchange of revocation secret values for $(\overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{AB}}}, \overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{BA}}})$. Therefore, the honest user can safely publish $\overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{AB}}}$ or $\overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{BA}}}$ on-chain to safeguard funds; and (ii) if the exchange of revocation secret values for $(\overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{AB}}}, \overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{BA}}})$ fails, then both $(\overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{AB}}}, \overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{BA}}})$ and $(\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{AB}}, \mathsf{tx}^{\mathsf{vc}*}_{\mathsf{BA}})$ remain valid, each containing correct signatures. Even if the honest user has sent the revocation secret value of $\overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{AB}}}$ or $\overline{\mathsf{tx}^{\mathsf{vc}}_{\mathsf{BA}}}$ to the malicious user, the honest user can still publish $\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{AB}}$ or $\mathsf{tx}^{\mathsf{vc}*}_{\mathsf{BA}}$ on-chain to safeguard funds.

*3) **Release**:* In the release phase, the states in C_{A} and C_{B} are updated. Note that Alice and Bob must send a release request

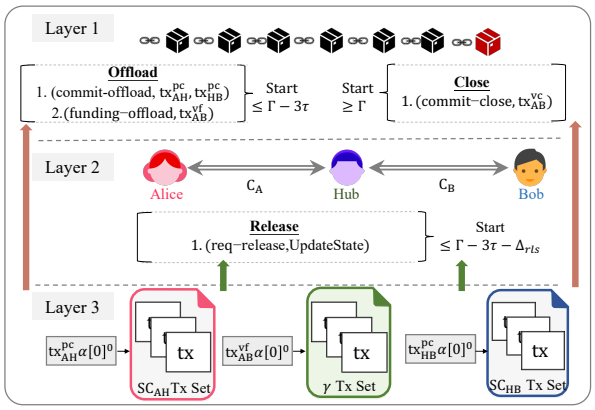


Fig. 7: Illustration of Release, Offload, and Close phases

to Hub before $\Gamma - 3\tau - \Delta_{\mathsf{rls}}$ rounds, where Δ_{rls} denotes the maximum number of rounds required to release the virtual channel. Note that Hub initiates the corresponding process only upon receiving each identical release instruction from both Alice and Bob within the time interval Δ_{op}; Otherwise, the process is terminated. As shown in Fig. 7, during the release phase, for C_{A} (analogous for C_{B}), the states of the segregated channels generated from transaction $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ (i.e., $\mathsf{SC}_{\mathsf{AH}}$ and γ) are transferred back to C_{A}. This is accomplished by generating new payment channel commitment transactions $\mathsf{tx}^{\mathsf{pc}^{\mathsf{rls}}}_{\mathsf{AH}}$ and $\mathsf{tx}^{\mathsf{pc}^{\mathsf{rls}}}_{\mathsf{HA}}$. Accordingly, Alice invokes UpdateState with $(\mathsf{A}, \mathsf{H}, \mathsf{st}^{\mathsf{rls}}, \mathsf{C}_{\mathsf{A}}, \tau, b{=}1, \mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}, \mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}})$ to generate $\mathsf{tx}^{\mathsf{pc}^{\mathsf{rls}}}_{\mathsf{AH}}$ and $\mathsf{tx}^{\mathsf{pc}^{\mathsf{rls}}}_{\mathsf{HA}}$ on C_{A}. When all users are honest and both C_{A}, this phase can be executed off-chain.

*4) **Offload**:* As shown in Fig. 7, if Hub fails to release the virtual channel γ before round $\Gamma - 3\tau$, Hub sends $(\mathsf{commit\text{-}offload}, \mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}, \mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}) \hookrightarrow \mathcal{L}$. Once these transactions are recorded on the ledger $\mathcal{L}$, Hub sends $(\mathsf{funding\text{-}offload}, \mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}) \hookrightarrow \mathcal{L}$ after τ rounds to retrieve fees and collateral; otherwise, the collateral locked in $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ and $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}$ will be forfeited. If a malicious Hub intentionally delays the offload phase, e.g., (i) delaying the publication of $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ ($\mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}$), Alice (Bob) can punish Hub by publishing $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}$ ($\mathsf{tx}^{\mathsf{pun}}_{\mathsf{HB}}$) after $\Gamma - \tau$ rounds; and (ii) delaying the publication of $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$, Alice and Bob can spend $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}.\alpha[1]^1$ or $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}.\alpha[1]^1$ respectively to punish Hub. Therefore, Alice and Bob can punish Hub and seize the relevant collateral and fees.

*5) **Close**:* Upon successful offloading, the virtual channel γ is converted into the payment channel C_{AB}, and the users, Alice and Bob, may choose to settle funds through the close phase. Suppose this phase is initiated by Alice (the process is symmetric for Bob) and achieves the closure of γ by sending $(\mathsf{commit\text{-}close}, \mathsf{tx}^{\mathsf{vc}}_{\mathsf{AB}}) \hookrightarrow \mathcal{L}$ after Γ rounds. For clarity of exposition, Fig. 8 illustrates the transactions and fund flows. We denote $\alpha^{\mathsf{op}}_{\mathsf{A}}$ and $\alpha^{\mathsf{op}}_{\mathsf{H}_1}$ as the funds locked by Alice and Hub in C_A, and $\alpha^{\mathsf{op}}_{\mathsf{H}_2}$ and $\alpha^{\mathsf{op}}_{\mathsf{B}}$ as the funds locked by Hub and Bob in C_B. For clarity, we include the fees paid by Alice and Bob to Hub in $\alpha^{\mathsf{op}}_{\mathsf{H}_1}$ and $\alpha^{\mathsf{op}}_{\mathsf{H}_2}$, respectively. After Hub publish the transaction $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$ to the ledger $\mathcal{L}$, it can retrieve the funds $\alpha^{\mathsf{op}}_{\mathsf{H}_1} + \alpha^{\mathsf{op}}_{\mathsf{H}_2}$. Simultaneously, Alice (resp. Bob) can close γ and settle the funds by publishing $\mathsf{tx}^{\mathsf{vc}}_{\mathsf{AB}}$ (resp. $\mathsf{tx}^{\mathsf{vc}}_{\mathsf{BA}}$).

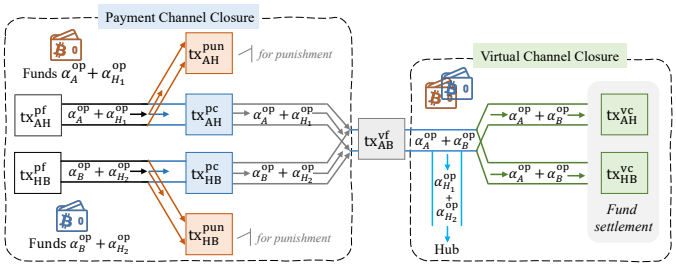


Fig. 8: Transactions and Fund Flows in SVC

6) Application of SVC in CPSS Payments: We introduce a concrete application scenario: micro-payments in smart grids. In this scenario, multiple energy providers and users need to frequently perform micro-payments to settle electricity usage. SVC shifts the micro-payment process off-chain and divides the payment channels into sub-payment channels and virtual channels. Through these segmented channels, energy providers and users can execute micro-payments concurrently while preserving the privacy of energy consumption, thereby meeting the CPSS requirements for high throughput and privacy [27].

V. Security Model

A. Models and Assumptions

1) Attack Model: We model the honest users as interactive Turing machines that can securely communicate with $\mathcal{F}_{svc}$. We also model the PPT adversary $\mathcal{A}$ as an interactive Turing machine. We consider static corruption [28], meaning the adversary must determine the corrupted users before the protocol begins. Once corruption is successful, $\mathcal{A}$ can control any information transfer between the corrupted users and route the corrupted users' incoming and outgoing communications.

2) Communication Model: The users of the channel communicate through the secure message transmission functionality $\mathcal{F}_{smt}$. A adversary $\mathcal{A}$ can delay messages between two honest users but cannot access or tamper with their contents [29]. Moreover, we model the synchronous communication network as $\mathcal{F}_{syn}$, where the protocol is executed in discrete rounds. $\mathcal{A}$ cannot change the order of messages sent between honest users. The definitions of $\mathcal{F}_{syn}$ and $\mathcal{F}_{smt}$ follow [30].

3) Global Ideal Ledger Functionality: We use the Global Universal Composability (GUC) framework [30] to model the blockchain as a global ledger, maintained by the global ideal ledger functionality $\mathcal{G}_{\mathcal{L}}$, which is parameterized by signature schemes and blockchain delay (Σ, τ). The information in $\mathcal{L}$ is public and can be updated using the ideal functionality.

B. Operations

We model the proposed SVC within the GUC framework [30], as illustrated in Fig. 9. We designate the payment channel hub (*i.e.*, Hub) as the intermediary between Alice and Bob, using the abbreviations $\{\mathsf{A}, \mathsf{H}, \mathsf{B}\}$ to represent users $\{\mathsf{Alice}, \mathsf{Hub}, \mathsf{Bob}\}$ respectively. We denote α_{A} and α_{H_1} as the balances of A and H in channel C_{A} before the virtual channel γ is opened, and α_{B} and α_{H_2} as the balances of B and H in channel C_{B} before γ is opened. Meanwhile, $\alpha_{\mathsf{A}}^{\mathsf{op}}$ and $\alpha_{\mathsf{B}}^{\mathsf{op}}$ denote the initial balances of the virtual channel. We use tx' to denote the transaction tx that contains the complete set of signatures. Each session is represented by a session identifier sid. $\mathcal{F}_{svc}$ internally maintains two sets: one for tracking the

Open

After receiving (sid, open-channel, γ,$\mathsf{st}_{\mathsf{A}}^{\mathsf{op}}$) from A (symm. for B), parse $\gamma.\mathsf{st} = \{(\alpha_{\mathsf{A}}^{\mathsf{op}}, \mathsf{Sig}_{\mathsf{A}}), (\alpha_{\mathsf{B}}^{\mathsf{op}}, \mathsf{Sig}_{\mathsf{B}})\}$, $\mathsf{st}_{\mathsf{A}}^{\mathsf{op}} = \{(\alpha_{\mathsf{A}} - \alpha_{\mathsf{A}}^{\mathsf{op}}, \mathsf{Sig}_{\mathsf{A}}), (\alpha_{\mathsf{H}_1} - \alpha_{\mathsf{B}}^{\mathsf{op}}, \mathsf{Sig}_{\mathsf{H}})\}$ and $\gamma.\mathsf{subchan} = (\mathsf{C}_{\mathsf{A}}, \mathsf{C}_{\mathsf{B}})$, and proceed as follows:

1. Send (sid,req-openVC,γ) to B and receive (sid,b) from B. Abort if $b = \perp$. Otherwise, for C_{A} (symm. for C_{B}), proceed as follows:
2. (i) Send (sid, vc.funding-query,γ) to A and H; (ii) receive (sid,b_{A}) from A and (sid,b_{H}) from H. Abort, if $b_{\mathsf{A}} = \perp$ or $b_{\mathsf{H}} = \perp$, do send (sid, vc.funding-abort, γ) to γ.users. Otherwise, send (sid, vc.funding-success, $\{\mathsf{A}, \mathsf{H}\}$) to C_{A}.users and continue.
3. (i) Send (sid, pc.state-query,$\mathsf{st}_{\mathsf{A}}^{\mathsf{op}}$) to A and H; (ii) receive (sid,b_{A}) from A and (sid,b_{H}) from H. Abort, if $b_{\mathsf{A}} = \perp$ or $b_{\mathsf{H}} = \perp$, do send (sid, pc.state-abort, γ) to γ.users. Otherwise, send (sid, pc.state-success, $\{\mathsf{A}, \mathsf{H}\}$) to C_{A}.users and continue.
4. (i) Send (sid, pc.pun-query,$\mathsf{st}_{\mathsf{A}}^{\mathsf{op}}$) to A and H; (ii) receive (sid,b_{A}) from A and (sid,b_{H}) from H. Abort, if $b_{\mathsf{A}} = \perp$ or $b_{\mathsf{H}} = \perp$, do send (sid, pc.pun-abort, γ) to γ.users. Otherwise, send (sid, pc.pun-success, $\{\mathsf{A}, \mathsf{H}\}$) to C_{A}.users and continue.

If receive similar messages from the channel C_{B} within Δ_{op} rounds, extract C_{A} from $\mathbb{PC}$, set $\mathsf{C}_{\mathsf{A}}.\mathsf{st} := \mathsf{st}_{\mathsf{A}}^{\mathsf{op}}$. Extract C_{B} from $\mathbb{PC}$, set $\mathsf{C}_{\mathsf{B}}.\mathsf{st} := \mathsf{st}_{\mathsf{B}}^{\mathsf{op}}$. Moreover, add C_{A} and C_{A} to $\mathbb{PC}$, and add γ to $\mathbb{VC}$, then send (sid, open-success, γ) to γ.users and $\mathcal{S}im$; otherwise, send (sid, open-abort, γ) to γ.users.

Update

1. Upon receiving (sid, update-channel, $\gamma.\mathsf{id}, \gamma.\mathsf{st}^{\mathsf{up}}$) from A (symetrical for B), send (sid, update-query, $\gamma.\mathsf{id}, \gamma.\mathsf{st}^{\mathsf{up}}$) to γ.endUsers.
2. Receive (sid, b_{A}) from A and (sid, b_{B}) from B. If $b_{\mathsf{A}} = \perp$ or $b_{\mathsf{B}} = \perp$, do send (sid, update-abort) to γ.endUsers.Otherwise, let $\gamma := \mathbb{VC}(\gamma.\mathsf{id})$, set $\gamma.\mathsf{st} := \mathsf{st}^{\mathsf{up}}$ and $\mathbb{VC}(\gamma.\mathsf{id}) := \gamma$ and send (sid, update-success) to γ.endUsers and $\mathcal{S}im$.

Release

1. Upon receiving (sid, release-channel, C_{A}, $\alpha_{\mathsf{A}}^{\mathsf{rls}}$, $\alpha_{\mathsf{H}_1}^{\mathsf{rls}}$) from A and (sid, release-channel, C_{B}, $\alpha_{\mathsf{B}}^{\mathsf{rls}}$,$\alpha_{\mathsf{H}_2}^{\mathsf{rls}}$) from B before $\Gamma - 3\tau - \Delta_{\mathsf{rls}}$ rounds, let $\gamma := \mathbb{VC}(\gamma.\mathsf{id})$, $\{(\alpha_{\mathsf{A}}^{l}, \mathsf{Sig}_{\mathsf{A}}), (\alpha_{\mathsf{B}}^{l}, \mathsf{Sig}_{\mathsf{B}})\} := \gamma.\mathsf{st}$. Check if $\alpha_{\mathsf{A}}^{\mathsf{rls}} = \alpha_{\mathsf{A}}^{l} + \alpha_{\mathsf{A}} - \alpha_{\mathsf{A}}^{\mathsf{op}}$, $\alpha_{\mathsf{H}_1}^{\mathsf{rls}} = \alpha_{\mathsf{B}}^{l} + \alpha_{\mathsf{H}_1} - \alpha_{\mathsf{B}}^{\mathsf{op}}$, $\alpha_{\mathsf{B}}^{\mathsf{rls}} = \alpha_{\mathsf{B}}^{l} + \alpha_{\mathsf{B}} - \alpha_{\mathsf{B}}^{\mathsf{op}}$, and $\alpha_{H^2}^{\mathsf{rls}} = \alpha_{\mathsf{A}}^{l} + \alpha_{\mathsf{H}} - \alpha_{\mathsf{A}}^{\mathsf{op}}$. If not, stop. Else proceed as follow.
2. Send (sid, release-query, γ) to γ.users and receive (sid, b_{A}), (sid, b_{H}) and (sid, b_{B}). If $b_{\mathsf{A}}, b_{\mathsf{H}}$ or $b_{\mathsf{B}} = \perp$, send (sid, release-abort) to γ.users. Otherwise, remove γ from $\mathbb{VC}$, remove $(\mathsf{C}_{\mathsf{A}}, \mathsf{C}_{\mathsf{B}})$ from $\mathbb{PC}$ and send (sid, release-success) to γ.users and $\mathcal{S}im$.

Offload

1. Upon receiving (sid, commitPC-offload, C_{A}, C_{B}, ($\mathsf{tx}_{\mathsf{AH}}^{\mathsf{pc}'}$,$\mathsf{tx}_{\mathsf{HB}}^{\mathsf{pc}'}$)) from H at $\Gamma - 3\tau$ rounds, proceed as follows:
2. Send (sid, pc-query, ($\mathsf{tx}_{\mathsf{AH}}^{\mathsf{pc}'}$,$\mathsf{tx}_{\mathsf{HB}}^{\mathsf{pc}'}$)) to $\mathcal{G}_{\mathcal{L}}$ and receive (sid, b_L) from $\mathcal{G}_{\mathcal{L}}$. If b_L=$\perp$, send (sid, commitPC-abort) to H. Otherwise, send (sid, commitPC-success) to H and continue.
3. Upon receiving (sid,commitvf-offload, γ, $\mathsf{tx}_{\mathsf{AB}}^{\mathsf{vf}'}$) from H, send (sid, vf-query, $\mathsf{tx}_{\mathsf{AB}}^{\mathsf{vf}'}$) to $\mathcal{G}_{\mathcal{L}}$.
4. Receive a message (sid, b_L) from $\mathcal{G}_{\mathcal{L}}$. If b_L=$\top$, send (sid, offload-success) to γ.users and $\mathcal{S}im$. Then remove γ from $\mathbb{VC}$. Otherwise, send (sid, offload-abort) to H and abort.

Close

Upon receiving (sid,commitVC-close,γ, $\mathsf{tx}_{\mathsf{AB}}^{\mathsf{vc}'}$) from A (symm. for B),

1. Send (sid,close-query,$\mathsf{tx}_{\mathsf{AB}}^{\mathsf{vc}'}$) to $\mathcal{G}_{\mathcal{L}}$ and receive (sid, b_L) from $\mathcal{G}_{\mathcal{L}}$. If b_L=$\top$, it sends (sid, close-success) to γ.endUsers and sends (sid, close-abort) to γ.endUsers otherwise.
2. Remove $(\mathsf{C}_{\mathsf{A}}, \mathsf{C}_{\mathsf{B}})$ from $\mathbb{PC}$, and forward (sid, close-success) to γ.users and $\mathcal{S}im$.

Fig. 9: Ideal Functionality $\mathcal{F}_{svc}$ for SVC.

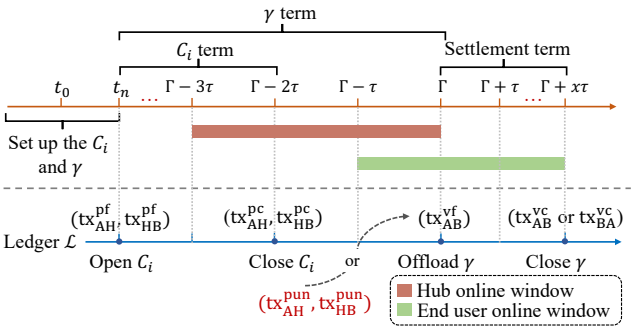


Fig. 10: Timeline of Transaction Recording.

participating payment channels, denoted by $\mathbb{PC}$; and another for tracking the opened virtual channels, denoted by $\mathbb{VC}$.

C. GUC Security

Let π be a protocol with access to $(\mathcal{F}_{syn}, \mathcal{F}_{smt}, \mathcal{G}_{\mathcal{L}})$. $\mathcal{Z}$ can receive output messages from parties in both the real world and the ideal world and provide input to them. We use $\mathbb{EXEC}^{\mathcal{F}_C}_{(\pi,\mathcal{A},Z)}$ to indicate the set of outputs from $\mathcal{Z}$ when the adversary $\mathcal{A}$ interacts with the protocol π, and use $\mathbb{EXEC}^{\mathcal{F}_C}_{(\mathcal{F}_{svc},Sim,Z)}$ to indicate the set of outputs from $\mathcal{Z}$ when $\mathcal{F}_{svc}$ interacts with the simulator $\mathcal{S}im$. We say that π GUC-realizes $\mathcal{F}_{svc}$ if, for any PPT environment $\mathcal{Z}$, it is unable to distinguish the output of the real world from that of the ideal world. The formal definition of security is expressed as:

Definition 2. *If for any PPT adversary $\mathcal{A}$ attacking a protocol π, there exists a PPT simulator $\mathcal{S}im$, such that no matter which protocol PPT environment $\mathcal{Z}$ selects for testing, we have*

$$\mathbb{EXEC}^{\mathcal{F}_C}_{\pi,\mathcal{A},Z} \approx \mathbb{EXEC}^{\mathcal{F}_C}_{\mathcal{F}_{svc},Sim,Z}, \tag{5}$$

where $\approx$ denotes computational indistinguishability, then we say that π GUC-realizes $\mathcal{F}_{svc}$ or π achieves GUC security.

D. Discussion on the Ideal Functionality

In this subsection, we will discuss how $\mathcal{F}_{svc}$ achieves the security and privacy goals as defined in Section IV-A.

Balance security. To protect the balance security of honest users, we incorporate multiple punishment mechanisms with different conditions to the transactions built in the open phase. For instance, $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ and $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}$ created in open phase can handle possible malicious behavior by H. As a trusted party, $\mathcal{F}_{svc}$ can accurately assess the messages sent by users. $\mathcal{F}_{svc}$ will query each user before starting the next step, and if any user refuses, it aborts the process of opening the virtual channel. In the open phase, if the virtual channel is successfully opened, $\mathcal{F}_{svc}$ sends a success message to all participating users and stores the opened channel in $\mathbb{VC}$. Otherwise, it sends a failed message. When all transactions are created, honest users can enforce the transactions on-chain in the case of malice.

Intuitively, as shown in Fig. 10, Hub only needs to access the ledger $\mathcal{L}$ at $\Gamma-3\tau$ round to publish the commitment transactions $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ and $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}$ of the payment channels $\mathsf{C}_i \in \{\mathsf{C_A}, \mathsf{C_B}\}$, thereby closing C_i. Here, τ denotes the maximum number of rounds required for a transaction to be recorded on the ledger. According to [7], τ is typically set to one day. Then, Hub waits for τ rounds before publishing the funding transaction $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$ of γ to offload it. Thereafter, Hub can claim the collateral and fees locked in $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}.\alpha[1]$ at any time.

For end users $\gamma.\mathsf{endUsers} \in \{\mathsf{Alice}, \mathsf{Bob}\}$, they only need to check the ledger $\mathcal{L}$ at $\Gamma-\tau$ round to verify whether Hub behaves maliciously in either of the following cases:

(i) Hub fails to close C_i at $\Gamma-\tau$ round. In this case, $\gamma.\mathsf{endUsers}$ can publish the punishment transactions $(\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}, \mathsf{tx}^{\mathsf{pun}}_{\mathsf{HB}})$ to punish Hub and spend $(\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}.\alpha[0], \mathsf{tx}^{\mathsf{pun}}_{\mathsf{HB}}.\alpha[0])$ to claim all funds in the payment channels;

(ii) Hub fails to publish $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$. (ii) That is, $\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$ fails to be recorded on $\mathcal{L}$ within 2τ rounds after $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}$ and $\mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}$ have been recorded on $\mathcal{L}$, then $\gamma.\mathsf{endUsers}$ can spend $(\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}.\alpha[1], \mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}.\alpha[1])$ to obtain Hub's collateral and fees.

Therefore, Hub only needs to stay online for at most 3τ time window to safely offload the virtual channel and protect funds, while the honest $\gamma.\mathsf{endUsers}$ have only a 3τ time window to respond and punish malicious Hub, which is significantly shorter than the entire channel lifetime Γ.

Atomicity. To achieve atomicity, $\mathcal{F}_{svc}$ queries confirmation from all participating users before initiating each step. The next step will proceed only after receiving confirmation from all channel users. For example, in the release phase, the release request for A and B is initiated only before $\Gamma-3\tau-\Delta_{\mathsf{rls}}$ rounds. In step 2 of the release phase, $\mathcal{F}_{svc}$ forwards the release request to $\gamma.\mathsf{users}$. After that, all participating users send a confirmation message to $\mathcal{F}_{svc}$ after $\mathsf{C_A}$ and $\mathsf{C_B}$ have released. Then $\mathcal{F}_{svc}$ receives confirmation messages from $\gamma.\mathsf{users}$. If all tokens from all participating users are $\top$, then $\mathcal{F}_{svc}$ notifies other users of the successful release message. Since each process is confirmed by all users, atomicity is guaranteed as long as there is at least one honest user.

Transaction privacy. After γ is opened, $\mathcal{F}_{svc}$ interacts exclusively with the end users of γ and constructs the virtual channel commitment transactions $(\mathsf{tx}^{\mathsf{vc}}_{\mathsf{AB}}, \mathsf{tx}^{\mathsf{vc}}_{\mathsf{BA}})$. Before $\Gamma-3\tau-\Delta_{\mathsf{rls}}$ round, $\gamma.\mathsf{endUsers}$ do not need to interact with the intermediary Hub. Consequently, Hub cannot learn any transaction information of γ, including the payment amounts or the number of payments, nor are any on-chain transactions generated, thereby effectively preserving transaction privacy.

VI. Security Analysis

A. Formal Analysis of SVC

According to Definition 2, a protocol π is called GUC secure if the environment $\mathcal{Z}$ cannot distinguish whether it is interacting with the ideal world or the real world, even in the presence of a PPT adversary $\mathcal{A}$. Since the execution of our protocol in the real world depends on the ideal functionalities $\mathcal{F}_C$, we define the proposed protocol in the hybrid world.

Theorem 1. *If a signature scheme adopted is EUF-CMA secure [31], [32], SVC GUC-realizes $\mathcal{F}_{svc}$ in the ($\mathcal{F}_{smt}$,$\mathcal{F}_{syn}$,$\mathcal{G}_{\mathcal{L}}$)-hybrid world.*

Proof. The security proof will describe the interaction between the simulator $\mathcal{S}im$ and the ideal functionality $\mathcal{F}_{svc}$ in the

ideal world and simulate an adversary $\mathcal{A}$ attack on hybrid protocol SVC in the ($\mathcal{F}_{smt}$,$\mathcal{F}_{syn}$,$\mathcal{G}_{\mathcal{L}}$)-hybrid world to prove that the protocol's execution is indistinguishable for $\mathcal{Z}$. Let $\mathcal{S}im$ interact with $\mathcal{F}_{svc}$ on behalf of an honest party, *i.e.*, every message that $\mathcal{F}_{svc}$ receives or sends from or to an honest party is directly forwarded to $\mathcal{S}im$.

Due to space limitations, we focus on the open phase, with the other phases following the same analytical principles. $\mathcal{S}im$ maintains the honest user's keys set $\mathbb{K}$ so that transactions can be signed on their behalf. If $\mathcal{S}im$ receives any error message from the PPT adversary $\mathcal{A}$, it aborts the operation. We assume the existence of the following two malicious scenarios:

Intermediary is corrupted. Suppose this phase is triggered by the honest γ.endUsers, $\mathcal{S}im$ continues as follows. First, $\mathcal{S}im$ receives (sid, req-openVC, γ) from $\mathcal{F}_{svc}$. Then, $\mathcal{S}im$ sends (req-open,$R^{\mathsf{pun}}_{\mathsf{sim}}$) representing an honest user, to $\mathcal{A}$. If $\mathcal{A}$ sends (sid, open-abort), $\mathcal{S}im$ sends (sid, $\perp$) to $\mathcal{F}_{svc}$ and abort the execution, otherwise sends (sid,$\top$) to $\mathcal{F}_{svc}$. After that, $\mathcal{S}im$ sequentially receives the messages (sid, vc.funding/pc.state/pc.pun-query) from $\mathcal{F}_{svn}$ and involves `GenTxPc`, `GenTxVf` and `GenTxPun` to construct transactions ($\mathsf{tx}^{\mathsf{pc}}, \mathsf{tx}^{\mathsf{vf}}, \mathsf{tx}^{\mathsf{pun}}, \mathsf{tx}^{\mathsf{vc}}$) with $\mathcal{A}$. After that, $\mathcal{S}im$ exchanges signatures with $\mathcal{A}$, and if either signature fails, sends $(\mathsf{sid}, \perp)$ to $\mathcal{F}_{svc}$ and abort; otherwise, sends (sid, $\top$) to $\mathcal{F}_{svc}$.

End user is corrupted. Suppose this phase is triggered by the corrupted user, $\mathcal{S}im$ continues as follows. First, $\mathcal{S}im$ receives (sid, req-open,$R^{\mathsf{pun}}_{\mathcal{A}}$) from $\mathcal{A}$ and sends (sid, open-channel,γ,$\mathsf{st}^{\mathsf{op}}_{\mathsf{sim}}$) to $\mathcal{F}_{svc}$, then receives (sid,req-openVC,γ). If $\mathcal{A}$ sends (sid, open-abort) on behalf of the corrupted users, $\mathcal{S}im$ sends (sid, $\perp$) and abort the execution. Otherwise, send (sid,$\top$). When $\mathcal{F}_{svc}$ receives a message (sid,$\top$) from $\mathcal{S}im$ in response to req-openVC, it sequentially sends (sid, vc.funding/pc.state/pc.pun-query) to $\mathcal{S}im$. Then $\mathcal{S}im$ creates transactions ($\mathsf{tx}^{\mathsf{pc}}, \mathsf{tx}^{\mathsf{vf}}, \mathsf{tx}^{\mathsf{pun}}, \mathsf{tx}^{\mathsf{vc}}$) and exchanges signatures with $\mathcal{A}$ on behalf of the honest users neighboring of $\mathcal{A}$. If an invalid signature appears, $\mathcal{S}im$ forwards (sid,$\perp$) as a response to vc.funding/pc.state/pc.pun-query to $\mathcal{F}_{svc}$ and abort, otherwise forward (sid,$\top$). Finally, employ $\mathcal{F}_{syn}$ to progress to the next round. □

If $\mathcal{A}$ cannot sign transactions on behalf of corrupted users, then the execution sets of the ideal world and the hybrid world are indistinguishable to the environment $\mathcal{Z}$. Therefore, the open phase of the hybrid protocol SVC GUC-realizes the open phase of $\mathcal{F}_{svc}$ in the ($\mathcal{F}_{smt}$,$\mathcal{F}_{syn}$,$\mathcal{G}_{\mathcal{L}}$)-hybrid world. Since the security proofs for the update, release, offload, and close phases of SVC follow similar principles, Theorem 1 is proven. If $\mathcal{F}_{svc}$ achieves the security and privacy goals defined in Section IV-A, then according to the GUC security definition described in Definition 2, the SVC also achieves these security and privacy goals.

VII. Performance Evaluation

The experimental platform uses an Intel Core i7-8565U CPU (1.99 GHz), 16 GB memory, and Windows 10. The code is written in Python and utilizes the python-bitcoin-utils library and Bitcoin scripts for transaction and channel construction. To meet the requirements of payment efficiency and privacy in CPSS, we conducted experiments on two LN snapshots taken in January 2022 [33] and May 2022 [34]. Each experiment is repeated 100 times, and we plot the mean and standard deviation. We also test compatibility by deploying the created transactions on the Bitcoin *testnet* and confirm that the transactions function as expected.

In this section, we compare the proposed SVC with the BCVC-V, LNVC-V [10], and Donner [11]. Among them, BCVC-V is the state-of-the-art validity-based virtual channel construction built over generalized channels [26], employing a punish-then-split payment paradigm. LNVC-V is another validity-based virtual channel construction built over lightning-style channels. Donner is suitable for multi-hop payment channel networks, offering the capability to enforce updates via a global transaction. To ensure a fair comparison, we implement these virtual channel constructions based on a payment channel hub, where the sender and receiver construct a payment channel network through a single intermediary.

A. Comparison of Features

As shown in Table II, in terms of continuous online days, SVC only requires the intermediary Hub to remain online at $\Gamma - 3\tau$ round and publish transactions ($\mathsf{tx}^{\mathsf{pc}}_{\mathsf{AH}}, \mathsf{tx}^{\mathsf{pc}}_{\mathsf{HB}}, \mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$) on-chain to complete the release phase of the virtual channel, after which it does not need to remain online. In the worst-case scenario, SVC only requires the intermediary Hub to remain online continuously for $3\cdot\tau$ to publish transactions. In contrast, BCVC-V, LNVC-V, and Donner require all users to stay online to offload the virtual channel, with continuous online durations of $5\cdot\tau$, $3\cdot\tau$, and $3\cdot\tau$, respectively, to publish transactions and monitor the blockchain. All compared constructions complete the payment by directly creating transactions off-chain, so there is no need for trusted third-party assumptions (such as watchtower). Except for Donner, the other compared constructions can lock the end-users' collateral for bidirectional payments in a virtual channel.

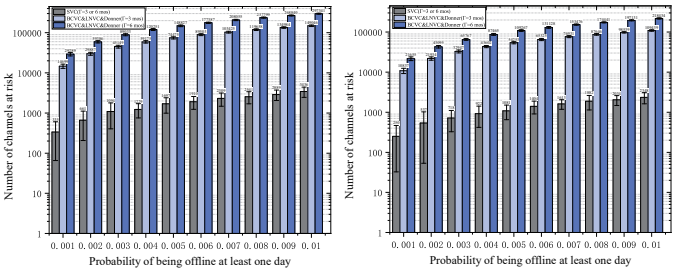
In a reopen operation, an SVC user only needs to update the transaction set ($\mathsf{tx}^{\mathsf{vf}}_{\mathsf{AB}}$, $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{AH}}$, $\mathsf{tx}^{\mathsf{pun}}_{\mathsf{HB}}$, $\mathsf{tx}^{\mathsf{vc}}_{\mathsf{AB}}$, $\mathsf{tx}^{\mathsf{vc}}_{\mathsf{BA}}$) to reopen the virtual channel and reset its lifetime and fund allocation. In comparison, BCVC, LNVC, and Donner require $8 \cdot n$, $14 \cdot n$, and $20 \cdot n$ transactions, respectively. Although all schemes have linear overhead $\Theta(n)$ in the number of reopen operations, SVC exhibits the smallest constant factor. For punishment, end users can publish $\mathsf{tx}^{\mathsf{pun}}$ when the intermediary delays offloading. BCVC, LNVC, and Donner incur $\Theta(n)$ on-chain transactions and time delay per punishment operation, whereas SVC requires only one on-chain transaction with unit delay, independent of network size. For the punishment operation, when the intermediary delays the offload phase, each end user in BCVC, LNVC, and Donner incurs $\Theta(n)$ on-chain transactions along with a corresponding $\Theta(n)$ time delay. In contrast, each end user in SVC needs to publish only a single $\mathsf{tx}^{\mathsf{pun}}$ on-chain, and the time delay per user is one time unit. In terms of scalability, the proposed SVC supports parallel payments, i.e., payments can be executed simultaneously via SC and γ. In addition, SVC incurs lower overhead for both reopen and punishment operations, further improving the scalability of the system.

TABLE II: Comparison with other virtual channel schemes.

	BCVC-V [10]	**LNVC-V** [10]	**Donner** [11]	**SVC**
Continuous online days	$5 \cdot \tau$	$3 \cdot \tau$	$3 \cdot \tau$	$3 \cdot \tau$
No third party	✓	✓	✓	✓
Atomicity	✓	✓	✓	✓
Transaction privacy	✓	✓	✓	✓
Bidirectional payment	✓	✓	×	✓
Reopen: txs off-chain	$8 \cdot n$	$14 \cdot n$	$20 \cdot n$	$5 \cdot n$
Punishment: on-chain overhead per user	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	1
Punishment: on-chain time delay per user	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	1
Scalibility	×	×	×	✓

B. Risk of Failing to Go Online on Time

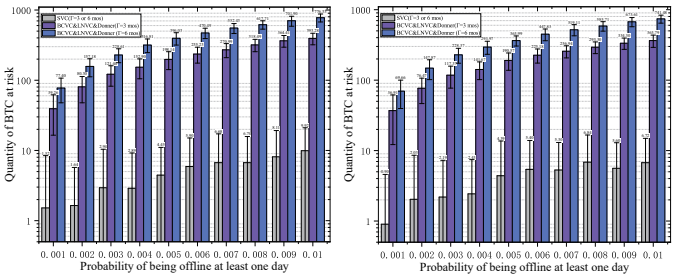
We test the potentially risky channels and risky BTC (keeping two decimal places) when paying with different virtual channel constructions in the LN snapshots from January 2022 and May 2022. We assume that τ equals one day, and the horizontal axis represents the probability P that users cannot come online within a day. We also assume that the lifetime Γ of the virtual channel is set to 3 months and 6 months.



(a) LN snapshot Jan. 2022 (b) LN snapshot May 2022

Fig. 11: Number of channels at risk.

In all evaluated constructions, the intermediary can offload virtual channels in advance. As shown in Figs. 11a to 12b, under different LN snapshots, the SVC consistently maintains a minimal number of risky channels and risky BTC. For instance, in the LN snapshot of May 2022 ($\Gamma = 6$ months), when P=1%, the risky channels and risky BTC in SVC are approximately 2348 and 6.72. In contrast, BCVC-V, LNVC-V, and Donner have approximately 218434 channels at risk and 741.48 BTC at risk when P=1%. This is because the SVC does not assume that users are continuously online every day after the virtual channel is offloaded. This is due to the inclusion of an absolute time lock for the funding transaction of the virtual channel. If there is no request to adjust or release the virtual channel, γ.endUsers only need to monitor the blockchain at $\Gamma - \tau$ round. In contrast, the BCVC-V, LNVC-V, and Donner constructions all exhibit a high number of risky channels and risky BTC across different LN snapshots, as these constructions require continuous online operation and daily blockchain monitoring after the virtual channel is offloaded. This is because, in these constructions, the spending conditions for virtual channel funding transactions include only signature conditions. These conditions can also be satisfied by old state transactions, so if the time interval exceeds τ, they may lose funds. Therefore, across different LN snapshots, SVC has fewer risky channels and less risky BTC than the compared schemes, effectively protecting the fund security of users.



(a) LN snapshot Jan. 2022 (b) LN snapshot May 2022

Fig. 12: Number of BTC at risk.

C. Overhead

As shown in Table III, when opening a virtual channel, the proposed SVC requires creating 7 transactions (including 2 payment channel state transactions, 1 funding transaction, 2 virtual channel state transactions, and 2 punishment transactions) off-chain, resulting in a total of 2957 bytes. In contrast, BCVC-V is constructed over generalized channels and requires the creation of 8 transactions (including 4 payment channel state transactions, 1 funding transaction, 1 refund transaction, and 2 virtual channel state transactions), resulting in a total overhead of 2951 bytes; LNVC-V requires 14 transactions (including 4 payment channel state transactions, 2 funding transactions, 4 refund transactions, and 4 virtual channel state transactions), totaling 5914 bytes; Donner requires the additional construction of a virtual channel to facilitate reverse payments, resulting in the highest overhead during the open phase. Specifically, Donner requires the generation of 16 transactions, totaling 3848 bytes.

In the update phase, SVC needs to update 2 virtual channel commitment transactions to update the virtual channel state, resulting in 680 bytes. In contrast, BCVC-V, LNVC-V, and Donner generate 2, 4, and 4 transactions, respectively, resulting in total overheads of 695, 1360, and 1360 bytes. To release the virtual channel, SVC needs to update 2 payment channel commitment transactions in each payment channel, totaling 1360 bytes. In the offload phase, the intermediary Hub needs to publish the funding transaction of the virtual channel to the ledger $\mathcal{L}$, which requires SVC to make 3 on-chain transactions totaling 1567 bytes. In contrast, during the offload phase, BCVC-V, LNVC-V, and Donner incur overheads of 2256 bytes (6 txs), 1901 bytes (4 txs), and 406 bytes (2 txs), respectively. During the close phase, SVC, BCVC-V, LNVC-V, and Donner incur overheads of 1907 bytes (4 txs), 2951 bytes (8 txs), 2241 bytes (5 txs), and 3228 bytes (14 txs), respectively.

In the punishment operation, users can publish 2 punishment transactions (totaling 710 bytes) to punish the intermediary and obtain all funds corresponding to the payment channels. BCVC-V requires publishing 6 transactions (including 2 commit transactions, 2 split transactions, and 2 claim transactions totaling 3156 bytes) to punish the intermediary. Consequently, compared with LNVC-V, the SVC reduces the communication

TABLE III: Overhead comparison of virtual channel constructions using PCH.

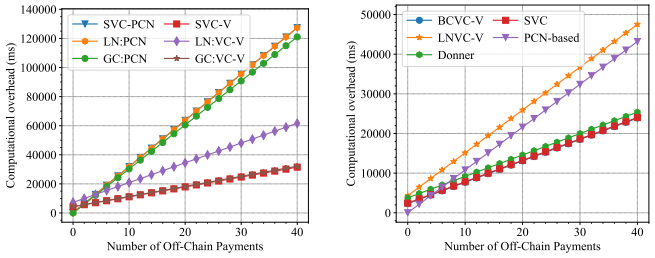
	LNVC-V		BCVC-V		Donner		SVC	
	on-chain	off-chain	on-chain	off-chain	on-chain	off-chain	on-chain	off-chain
	#txs/size	#txs/size	#txs/size	#txs/size	#txs/ size	#txs/ size	#txs/size	#txs/size
Open	0/0	14/5914 B	0/0	8/2951 B	0/0	16/3848 B	0/0	7/2957 B
Update	0/0	4/1360 B	0/0	2/695 B	0/0	4/1360 B	0/0	2/680 B
Release	0/0	4/1360 B	0/0	4/1390 B	0/ 0	4/1360 B	0/0	4/1360 B
Offload	4/1901 B	0/0	6/2256 B	0/0	2/406 B	0/ 0	3/1567 B	0/0
Close	5/2241 B	0/0	8/2951 B	0/0	14/3228 B	0/ 0	4/1907 B	0/0
Punishment	4/1056 B	0/0	6/3156 B	0/0	4/912 B	0/0	2/710 B	0/0

TABLE IV: Comparison of signature computation.

	User	BCVC-V	LNVC-V	Donner	SVC
Open	Alice	5	8	8	6
	Hub	7	14	12	6
	Bob	5	10	8	6
Update	Alice	$2n$	$4n$	$2n$	$2n$
	Hub	0	0	0	0
	Bob	$2n$	$4n$	$2n$	$2n$
Release	Alice	2	2	2	2
	Hub	4	4	4	4
	Bob	2	2	2	2

"n" indicates the number of payments.

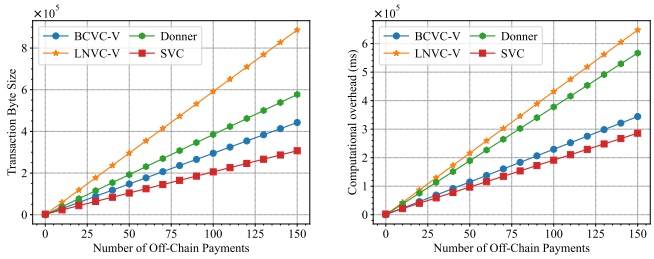
overhead by approximately 50% during the open phase and 15% during the close phase. SVC and BCVC-V have similar overhead during the open phase; however, SVC reduces the communication overhead by approximately 35% when closing the channel. Table IV presents the signature computation required for constructing different virtual channels at various phases. The proposed SVC achieves a relatively low signature computation overhead in the open phase. Compared with Donner, the signature computation in the open phase is reduced by 36%.



(a) Communication overhead (b) Computational overhead

Fig. 13: Comparison of overhead under different schemes

Next, we simulate and compare the overhead incurred by different constructions in building payments off-chain. As shown in Fig. 13a, we compare SVC with BCVC-V, LNVC-V, and the corresponding PCN-based schemes. Fig. 13a shows that when the number of off-chain transactions exceeds 2, SVC and BCVC-V incur lower communication overhead than their corresponding PCN-based schemes. Therefore, once the virtual channel is successfully constructed, as the number of off-chain transactions increases, the advantages of the virtual channel will become apparent, effectively reducing overhead consumption. The proposed SVC incurs an overhead of 2957 bytes when opening a virtual channel, compared to 2951 bytes for BCVC-V. As the number of payments increases, the overhead of SVC is lower than that of BCVC-V. When the number of off-chain payments is 20, the overhead of SVC is 324 bytes less than BCVC-V. This advantage becomes more pronounced as the number of payments increases.



(a) Communication overhead (b) Computational overhead

Fig. 14: Comparison of overhead of Reopen.

As shown in Fig. 13b, we further compare the computational overhead of different constructions in multiple payment scenarios. It can be observed that as the number of payments increases, the computational overhead of all constructions grows linearly with the number of payments. When the number of payments exceeds 7, SVC shows a significant advantage over the PCN-based schemes. Overall, SVC demonstrates higher computational efficiency and consistently outperforms both Donner and LNVC-V. Finally, we conducted experimental evaluations of the computational and communication overhead incurred during the reopen operation of a virtual channel. In SVC, users only need to regenerate a set of transactions (tx^{vf}_{AB}, tx^{pun}_{AH}, tx^{pun}_{HB}, tx^{vc}_{AB}, tx^{vc}_{BA}) to reopen the virtual channel and adjust its lifetime. As shown in Fig. 14, SVC consistently outperforms BCVC-V, LNVC-V, and Donner throughout the entire process, exhibiting lower computational and communication overhead. Regarding communication overhead, SVC reduces the cost by 65% compared to Donner, and by 22% compared to BCVC-V. In terms of computational overhead, SVC reduces costs by 66% and 23% compared with Donner and BCVC-V, respectively.

VIII. Concluding Remarks and Future Work

Blockchain payments are considered an ideal payment solution in intelligent CPSS environments. However, it currently faces issues such as privacy leakage and limited scalability. To address these issues, this work proposes a scalable virtual channel construction compatible with the UTXO model, named SVC. SVC achieves a trustworthy and efficient payment process through contracts, allowing users in CPSS to

receive immediate compensation upon completing computational tasks. Additionally, SVC introduces parallel construction by dividing the payment channel into a sub-payment channel and a virtual channel, ensuring payment efficiency and privacy. The experimental analysis demonstrates that SVC can protect the users' funds while maintaining low overhead.

Due to the limitation of payment channel capacity, high-value payments cannot be delivered in the payment channel network. In future work, we will investigate lightweight virtual channel constructions based on a multi-channel payment network to improve the success rates of high-valued payments.

ACKNOWLEDGMENT

This work was partially supported by the National Key Research and Development Program of China under Grant 2021YFA1000600, the National Natural Science Foundation of China (Grant No. U2468205), the National Natural Science Foundation of China (Grant No. 62472168), the National Natural Science Foundation of China (Grant Nos. U25A20429 and 62332018), and the Sichuan Provincial Natural Science Foundation (Grant No. 2025ZNSFSC0497).

REFERENCES

[1] X. Zhou, W. Liang, K. I.-K. Wang, and S. Shimizu, "Multi-modality behavioral influence analysis for personalized recommendations in health social media environment," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 888–897, 2019.

[2] Y. Maleh, S. Lakkineni, L. Tawalbeh, and A. A. AbdEl-Latif, *Blockchain for Cyber-Physical Systems: Challenges and Applications*. Cham: Springer International Publishing, 2022, pp. 11–59.

[3] X. Zhou, W. Liang, J. She, Z. Yan, and K. I.-K. Wang, "Two-layer federated learning with heterogeneous model aggregation for 6g supported internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5308–5317, 2021.

[4] Y. Wu, C. Zhang, and L. Zhu, "Privacy-preserving and traceable blockchain-based charging payment scheme for electric vehicles," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21 254–21 265, 2023.

[5] J. He, W. Qiu, S. Zhuo, M. Xu, Q. Zhang, Z. Xiong, and Z. Zheng, "An efficient multi-party payment protocol for iot micro-payments," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[6] J. Cai, W. Liang, X. Li, K. Li, Z. Gui, and M. K. Khan, "Gtxchain: A secure iot smart blockchain architecture based on graph neural network," *IEEEInternet of ThingsJournal*, vol. 10, no. 24, pp. 21 502–21 514, 2023.

[7] L. Aumayr, P. Moreno-Sanchez, A. Kate, and M. Maffei, "Blitz: Secure {Multi-Hop} payments without {Two-Phase} commits," in *30th USENIX Security Symposium*, 2021.

[8] W. Liang, S. Xie, K.-C. Li, X. Li, X. Kui, and A. Y. Zomaya, "Mcdsc: A dynamic secure resource configuration scheme based on medical consortium blockchain," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 3525–3538, 2024.

[9] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016, https://lightning.network/lightningnetwork-paper.pdf.

[10] L. Aumayr, M. Maffei, O. Ersoy, A. Erwig, S. Faust, S. Riahi, K. Hostáková, and P. Moreno-Sanchez, "Bitcoin-compatible virtual channels," in *IEEE Symposium on Security and Privacy*, 2021, pp. 901–918.

[11] L. Aumayr, P. Moreno-Sanchez, A. Kate, and M. Maffei, "Breaking and fixing virtual channels: Domino attack and donner," in *Network and Distributed System Security Symposium*, 2023.

[12] Y. Liu, W. Liang, K. Xie, S. Xie, K. Li, and W. Meng, "Lightpay: A lightweight and secure off-chain multi-path payment scheme based on adapter signatures," *IEEE Transactions on Services Computing*, no. 01, pp. 1–14, nov 5555.

[13] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "Perun: Virtual payment hubs over cryptocurrencies," in *IEEE Symposium on Security and Privacy*, 2019, pp. 106–123.

[14] S. Xie, L. Xiao, D. Han, K. Xie, X. Li, and W. Liang, "Hcvc: A high-capacity off-chain virtual channel scheme based on bidirectional locking mechanism," *IEEE Transactions on Network Science and Engineering*, pp. 1–12, 2023.

[15] X. Zhou, X. Zheng, T. Shu, W. Liang, K. I.-K. Wang, L. Qi, S. Shimizu, and Q. Jin, "Information theoretic learning-enhanced dual-generative adversarial networks with causal representation for robust ood generalization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 2, pp. 2066–2079, 2025.

[16] B. K. Sovacool and D. D. Furszyfer Del Rio, "Smart home technologies in europe: A critical review of concepts, benefits, risks and policies," *Renewable and Sustainable Energy Reviews*, vol. 120, p. 109663, 2020.

[17] X. Zhou, W. Liang, Z. Luo, and Y. Pan, "Periodic-aware intelligent prediction model for information diffusion in social networks," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 894–904, 2021.

[18] A. I. Sanka and R. C. Cheung, "A systematic review of blockchain scalability: Issues, solutions, analysis and future research," *Journal of Network and Computer Applications*, vol. 195, p. 103232, 2021.

[19] X. Zhou, J. Wu, W. Liang, K. I.-K. Wang, Z. Yan, L. T. Yang, and Q. Jin, "Reconstructed graph neural network with knowledge distillation for lightweight anomaly detection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 9, pp. 11 817–11 828, 2024.

[20] 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*, pp. 1–13, 2025.

[21] D. Han, Y. Zhu, D. Li, W. Liang, A. Souri, and K.-C. Li, "A blockchain-based auditable access control system for private data in service-centric iot environments," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3530–3540, 2022.

[22] M. Jourenko, M. Larangeira, and K. Tanaka, "Lightweight virtual payment channels," in *Cryptology and Network Security*, S. Krenn, H. Shulman, and S. Vaudenay, Eds. Cham: Springer International Publishing, 2020, pp. 365–384.

[23] E. Tairi, P. Moreno-Sanchez, and M. Maffei, "A2l: Anonymous atomic locks for scalability in payment channel hubs," in *IEEE Symposium on Security and Privacy*, 2021, pp. 1834–1851.

[24] W. Liang, Y. Liu, C. Yang, S. Xie, K. Li, and W. Susilo, "On identity, transaction, and smart contract privacy on permissioned and permissionless blockchain: A comprehensive survey," *ACM Computing Surveys*, 2024.

[25] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 281–308, 1988.

[26] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, and S. Riahi, "Generalized channels from limited blockchain scripts and adaptor signatures," in *Advances in Cryptology – ASIACRYPT 2021*, M. Tibouchi and H. Wang, Eds. Cham: Springer International Publishing, 2021, pp. 635–664.

[27] X. Zhou, W. Liang, J. Ma, Z. Yan, and K. I.-K. Wang, "2d federated learning for personalized human activity recognition in cyber-physical-social systems," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 6, pp. 3934–3944, 2022.

[28] C. Egger, P. Moreno-Sanchez, and M. Maffei, "Atomic multi-channel updates with constant collateral in bitcoin-compatible payment-channel networks," in *ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 801–815.

[29] X. Zhou, W. Liang, K. I.-K. Wang, K. Yada, L. T. Yang, J. Ma, and Q. Jin, "Decentralized federated graph learning with lightweight zero trust architecture for next-generation networking security," *IEEE Journal on Selected Areas in Communications*, vol. 43, no. 6, pp. 1908–1922, 2025.

[30] R. Canetti, Y. Dodis, R. Pass, and S. Walfish, "Universally composable security with global setup," in *Theory of Cryptography*, S. P. Vadhan, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 61–85.

[31] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 281–308, 1988.

[32] W. Liang, Y. Yang, C. Yang, Y. Hu, S. Xie, K.-C. Li, and J. Cao, "Pdpchain: A consortium blockchain-based privacy protection scheme for personal data," *IEEE Transactions on Reliability*, vol. 72, no. 2, pp. 586–598, 2023.

[33] "Snapshots: Lightning network," Jan, 2022, https://ln.fiatjaf.com/.

[34] "Snapshots: Lightning network," May, 2022, https://github.com/CosimoSguanci/Mass-Exit-Attacks-on-LN/tree/main.