MAPPCN: Multi-hop Anonymous and Privacy-Preserving Payment Channel Network

Somanath Tripathy and Susil Kumar Mohanty

Department of Computer Science and Engineering,
Indian Institute of Technology Patna, Bihta, Patna, India, 801106
som@iitp.ac.in, susil_1921cs05@iitp.ac.in

Abstract. Payment channel network (PCN) has become an indispensable mechanism to resolve scalability issues in blockchain-based cryptocurrencies. On the other hand, PCNs do not provide adequate security and privacy guarantee. Most of the existing payment schemes leak information about payer or payee to the nodes in the payment path. To address this issue, we propose a simple but effective, multi-hop, anonymous, and privacy-preserving PCN *(MAPPCN)*. MAPPCN construction is based on Elliptic curve cryptography (ECC) and is proved to be secure while achieving payment path privacy, sender, and receiver anonymity. MAPPCN can be performed in $(3 \cdot n + 5)$ Elliptic curve scalar multiplication (ECSM) operations for an off-chain payment operation.

Keywords: ETLC: Elliptic-curve Time Lock Contract · HTLC: Hashed Time-Lock Contract · PCN: Payment channel network · Off-chain PCN · Layer-2 PCN · Blockchain · Security · Privacy · Anonymity.

1 Introduction

Blockchain-based cryptocurrencies including Bitcoin [18], Ethereum [21] are increasing their popularity. The number of transactions in Bitcoin and Ethereum are 256,253 K (lowest) TPD, 351,791 K (highest) TPD[1], and 571,769 K (lowest) TPD, 751,354 K (highest) TPD[2] in 30^{th} Oct, 2019 respectively. This rapid growth of transactions would lead to scalability and privacy issues. A payment channel is a suitable alternative and widely deployed solution at the present system, in which a sequence of the transaction(s) can be performed between two users without updating each on the blockchain.

Two users can establish a payment channel by depositing a fixed amount and creating a 2-party ledger ($2PL$). Later on, they can use this $2PL$ to transfer within themselves, without involving blockchain for every payment. After completion of the transactions, they set the last state into the blockchain to get the corresponding coins. This idea can be extended for two users to execute

[1] Bitcoin transactions chart per day. https://www.blockchain.com/en/charts/n-transactions

[2] Ethereum transactions chart per day. https://etherscan.io/chart/tx

a transaction through intermediate hops by establishing a Payment Channel Network (PCN). Several PCNs are proposed, but most of them fail to provide adequate security and privacy guarantee. Therefore, there is a call for developing an anonymous multi-hop and privacy-preserving PCN.

Lightning network [5] and Raiden network [4] are the most popular PCN widely deployed on Bitcoin and Ethereum, respectively. There are many practical deployments of PCNs, including [1–4, 16, 17], and some survey work on layer-2 or off-chain scalability solutions are proposed in [9, 11]. Privacy and security of PCNs are crucial and need to be addressed carefully. Recently many privacy-preserving mechanisms for PCNs including [8, 10, 14, 15, 22], are presented with their advantages and limitations.

This work proposes a simple but effective, anonymous multi-hop and privacy-preserving PCN ($MAPPCN$). Like AMHL [15], $MAPPCN$ uses Elliptic curve group-based operation. Unlike, $AMHL$ the sender in our proposed scheme does not require to communicate directly with intermediate nodes other than its immediate neighbor. It is showed that the proposed mechanism achieves consumer privacy and resistant to different attacks.

The remaining part of this paper is organized as follows. Section 2 describes the related work. The background concept including payment channel network, Hash Time Lock Contract ($HTLC$) are presented in Section 3. The adversary model and security requirements are discussed in Section 4 and the proposed $MAPPCN$ technique is described in Section 5. Security analysis is presented in Section 6. Section 7 discusses conclusions and future scope of the paper.

2 Related work

Recently, many payment channel networks [1–3] are based on Lightning network [5] in Bitcoin, while [4, 16, 17] are based on Ethereum. Recently, the authors in [8] and [14] observed that the techniques presented in [1–3] are not privacy-preserving and prone to security attacks. Further, different privacy-preserving PCNs have been proposed recently.

Blind Off-chain Lightweight Transactions ($BOLT$) [8] consists of a set of techniques for constructing privacy-preserving payment channels for a decentralized currency. These techniques ensure that multiple payments on a single channel are unlinkable to each other, and the channels are funded with anonymized capital. It imposes a communication overhead of around 5 MB upon each node, which hinders its deployability. TumbleBit [10] is an unidirectional unlinkable anonymous off-chain payment hub protocol that uses untrusted intermediary called *Tumbler*. It is fully compatible with Bitcoin and provides atomicity. Security of TumbleBit follows from the standard assumption of RSA and ECDSA.

Zhang *et al.* [23] proposed an anonymous off-blockchain micropayment mechanism, in which the payee receives payment from an *"honest-but-curious"* intermediary T by solving (RSA assumption based) puzzles. It achieves strong unlinkability and unforgeability. A secure, privacy-preserving and interoperable payment channel hub (PCH) has been proposed in [20] called A^2L. A^2L has been built on the three-party protocol for conditional transactions, where the intermediary pays the receiver, only if the latter solves the assigned cryptographic challenge with the help of the sender. Malavolta et al. . [14] presented an alternative security and privacy approach for payment channel networks called *Multi-HTLC* (MHTLC), but requires more communication overhead. In [15], they reported the *wormwhole attack* in *HTLC* based payment channel network, and proposed a privacy-preserving mechanism called anonymous multi-hop locks (AMHL). Also, they demonstrated the practicality of their approach and its impact on the security, privacy, interoperability, and scalability of today's cryptocurrencies. Recently, authors in [22] proposed a Chameleon Hash Time-Lock Contract (*CHTLC*), for addressing the payment path privacy issues in PCNs. *CHTLC* outperforms than *MHTLC [14]* as, five times faster for payment data initialization, and reduced communication overhead from $17,000$ KB to just 7.7 KB.

TEEChain [13] is another off-chain payment protocol that utilizes trusted execution environments (*TEEs*) to perform secure, efficient, and scalable payments on the top of a blockchain, with asynchronous blockchain access. *AMCU* [7] is the first protocol for achieving atomic multi-channel updates and *state privacy*. Moreover, the reduced collateral mitigates the consequences of griefing attacks in PCNs. Meanwhile, the (multi-payment) atomicity achieved by AMCU opens the door to new applications such as credit re-balancing and crowdfunding that are not possible otherwise. *Sprites* [17] is an alternative PCN technique that supports partial withdrawals and deposits in the payment channel without interruption. It reduces the delay to the expiry timeout period, and funds (collateral cost) locked at intermediate payment channels, and it improves throughput and decentralization. *PISA* [16] is a generic state channel which can be used to build applications like payments, auctions, boardroom voting, gaming, using a new third party agent called *custodian*. The custodian is designed to help alleviate a new assumption in state channels, which require every participating party to remain online (synchronized with the blockchain).

3 Background

The section delivers the background information related to payment channel network (PCN). For the sake of readability, the frequently used notations are presented in Table 1.

3.1 Payment Channel Network (PCN)

As shown in Fig.1, a payment channel locks up the collateral cost between two parties in a *multisig* address on the blockchain to facilitate the participants to send or withdraw in the future, through off-chain transactions. A payment channel has three states; these are ❶ open channel, ❷ update channel (off-chain transactions) and ❸ close channel as marked with shaded region in Fig.1. Two on-chain transactions (to open and to close the channel) are required to access the blockchain. Between these two transactions, users can make a vast number of transactions by exchanging signatures on updated states of the collateral cost. The deposited funds can be settled by any party according to the last updated state while closing the channel.

Table 1: Notations

Notation	Description	Notation	Description
F_p	Finite field	v	Actual amount payer wants to pay payee
E_p	Elliptic curve Group	γ_i	Balanced amount or Collateral fund with user u_i
P	Base point on E_p	v_i	Updated payment amount after deducting transaction fee of user u_i
r	Secret or random number	$c_{<u_i,u_j>}$	Payment channel (between users u_i and u_j) identifier
$f(u_i)$	Transaction fee of user u_i	t_i	Expiration time of the transaction corresponding to user u_i
u_i	User or Intermediate user	S & R	Sender or Payer (u_0) & Receiver or Payee (u_{n+1})

A PCN is a path of payment channels, used to perform off-chain transactions between two users (the payer or sender S and payee or receiver R), without existing of direct payment channel between them. In Fig.1, Sender S wants to pay $\boxed{v = \$30}$ to Receiver R through the payment path $\boxed{S \to u_1 \to u_2 \to u_3 \to R}$ (we assume the transaction fees for all the intermediate users are set to $f = \$0.1$). The payer S sends a payment request $\boxed{\text{Pay(S, } u_1, v_1, \text{s, } t_1)}$, where, $\boxed{v_1 = v + \sum_{i=1}^{n} f(u_i)}$ (i.e., \$30.3 = \$30 + $\sum_{i=1}^{3} 0.1$) to the next neighbor u_1. For a successful payment, each intermediate user u_i in the path must have a collateral cost (γ_i) not less than the payment amount to be forwarded next, *i.e.*, $\boxed{\gamma_i \geq v_{i+1}}$ (γ_1 = \$80 and v_2 = \$30.2), where $\boxed{v_i = v_1 - \sum_{j=1}^{i-1} f(u_j)}$, otherwise *aborts* the payment request. In Fig.1, the intermediate user u_1 deducts transaction fee $f = \$0.1$ and forwards $\boxed{v_2 = \$30.2}$ to user u_2 and so on. Thus, R receives correctly v coins from that sender S initiates the payment with the amount v_1.

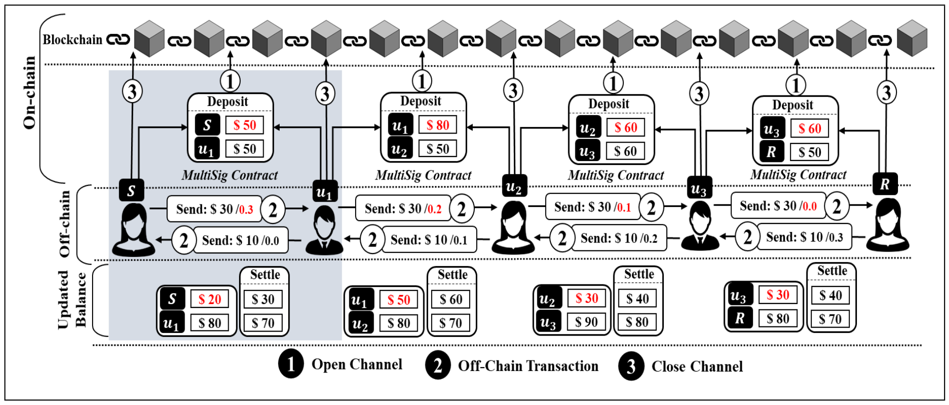


Fig. 1: PCN: Payment Channel Network

3.2 Routing in PCNs

The foremost essential task to construct a payment path with enough capacity between the payer and payee is an interesting problem. We are assuming the network topology is known to every user, and a gossip protocol between users can be carried out to broadcast the existence of any payment channel [19]. Furthermore, the fees charged by every user can be made public by similar means. Under these conditions, the sender can locally calculate the path to the receiver.

3.3 Atomicity in Multi-hop Payments

One of the fundamental properties in a multi-hop payment is *atomicity*, which means either all channel's capacity (each user's collateral cost of the corresponding channel $c_{<u_i,u_{i+1}>}$) in the path will update or none. A user would lose some coins if it transfers an amount to its next neighbor but never receives the corresponding coins from the preceding user in the same path. To avoid such issues, Lightning Network introduced a solution called Hash Time-Lock Contract *HTLC* [5] to achieve this property.

Here the Contract is executed between two users. In Fig. 2, S pays \$$v$ to R through the payment channel network. For simplicity, we assume that each user u_i charges a transaction fee of $f(u_i)$ coins. At the beginning, the receiver R sends a hashed random secret s, $\boxed{s = H(r)}$ to sender S. In the second step, sender S creates a payment contract $\boxed{HTLC(S, u_1, v_1, s, t_1)}$ where $\boxed{v_1 = v + \sum_{i=1}^{n} f(u_i)}$ with the right neighbor u_1 asking to forward the payment to user u_2, otherwise the payment is refunded back to sender S if time t elapsed. After receiving the payment, the user u_1, first deducts its transaction fee from the payment amount and creates a similar payment between himself and the next neighbor user u_2. When the *HTLC* condition meets (i.e., the secret r is reached to the sender) in reverse direction, the sender releases the locked v coins. The HTLC condition is based on a one-way hash function H, a hash value $\boxed{s = H(r)}$, the amount of coins v, and timeout t.

4 Adversary Model and Security Requirements

We assume that attacker $\mathcal{A}$ is computational efficient to corrupt a subset of existing users in PCN. $\mathcal{A}$ can extract the internals states of those corrupted users and all the information flow through them. The corrupted users can cooperate to compromise the security of PCN. $\mathcal{A}$ can send arbitrary message impersonating any corrupted user. But, the communication among the honest (not-corrupted) users occur through secure channels and thus confidential to $\mathcal{A}$.

4.1 Security Model

We consider the ideal functionality $\mathcal{F}$ defined in *MHTLC* [14] to prove the security and privacy of *MAPPCN*. $\mathcal{F}$ is a trusted functionality interact with users (u_i) and maintains two lists $\mathcal{L}$ and $\mathcal{C}$. Input $\mathcal{F}$ consists of the following three operations:

1. ***OpenChannel***: The *open channel* operation accepts $(open, c_{<u_i,u_j>}, \gamma_i, u_j, t, f(u_j))$ from user u_i, where γ_i is the collateral cost of user u_i of the channel $c_{<u_i,u_j>}$, t and $f(u_j)$ are the expiration time and transaction fee respectively. $\mathcal{F}$ verifies whether $c_{<u_i,u_j>}$ is unique and valid identifier, and then, sends $(c_{<u_i,u_j>}, \gamma_i, t, f(u_j))$ to u_j. If u_j authorizes the operation, $\mathcal{F}$ chooses a random number h and appends tuple $(c_{<u_i,u_j>}, \gamma_i, t, f(u_j))$ and $(c_{<u_i,u_j>}, v, t, h)$ respectively to $\mathcal{B}$ and $\mathcal{L}$. Finally $\mathcal{F}$ returns h to u_i and u_j.
2. ***CloseChannel***: Consequent upon receiving the inputs $(close, c_{<u_i,u_j>}, h)$ from either user u_i or u_j, $\mathcal{F}$ verifies $\mathcal{B}$ for an entry $(c_{<u_i,u_j>}, \gamma_i, t, f(u_j))$ and $\mathcal{L}$ for an entry $(c_{<u_i,u_j>}, \gamma_i', t', h)$, where $(h \neq \perp)$. If $c_{<u_i,u_j>} \in \mathcal{C}$ or $t \geq |\mathcal{B}|$ or $t' > |\mathcal{B}|$, $\mathcal{F}$ aborts, otherwise $\mathcal{F}$ appends $(c_{<u_i,u_j>}, u_i, v', t')$ to $\mathcal{B}$ and $c_{<u_i,u_j>}$ to $\mathcal{C}$. Then $\mathcal{F}$ informs both u_i, u_j with message $(c_{<u_i,u_j>}, \perp, h)$.
3. ***Payment***: In this operation, after receiving $(pay, v, (c_{<S,u_1>}, \cdots, c_{<u_n,R>}), (t_0, \cdots, t_n))$ from S, $\mathcal{F}$ performs the following protocol:
   (a) $\mathcal{F}$ randomly picks h_i for each $i \in \{0, \cdots, n\}$ and searches for an entry $(c_{<u_{i-1},u_i>}, v_i, t_i', f(u_j))$ in $\mathcal{B}$. If found, $\mathcal{F}$ sends $\langle h_i, h_{i+1}, c_{<u_{i-1},u_i>}, c_{<u_i,u_{i+1}>}, v - \sum_{j=i}^{n} f(u_j), t_{i-1}, f(u_j)\rangle$ to user u_i and $\langle h_n, c_{u_n,R}, v, t_n\rangle$ to R through a separate secure channel. Then $\mathcal{F}$ checks if all the entries of the form $(c_{<u_{i-1},u_i>}, v_i')$ exists in $\mathcal{L}$ and the corresponding $\delta_i = v_i' - (v - \sum_{j=i}^{n} f(u_j)) \geq 0$ and $(t_{i-1} \geq t_i)$. Then, $\mathcal{F}$ adds $d_i = (c_{<u_{i-1},u_i>}, \delta_i, t_i, \perp)$ to $\mathcal{L}$, where $(c_{<u_{i-1},u_i>}, v_i') \in \mathcal{L}$ is the entry with the lowest u_i'. If any of the conditions fails $\mathcal{F}$ removes all d_i entries added in this phase to $\mathcal{L}$ and aborts.
   (b) $\mathcal{F}$ queries each user u_i $\forall i \in \{(n+1), \cdots, 1\}$ with (h_i, h_{i+1}) through secure channel. Each user can replies with either $\perp$ or $\top$. Let j be the index of user that returns $\perp$, s.t. $\forall i > j$, u_i returned $\top$. If no user returned $\perp$, then set $j = 0$.
   (c) $\forall i \in \{(j+1), \cdots, n\}$, $\mathcal{F}$ updates $d_i \in \mathcal{L}$ to $(\_, \_, \_, h_i)$ and informs the user with the message $(success, h_i, h_{i+1})$, $\forall i \in \{0, \cdots j\}$ $\mathcal{F}$ removes d_i from $\mathcal{L}$ and notifies the user with the message $(\perp, h_i, h_{i+1})$.

4.2 Security and Privacy Requirements

The security and privacy requirements in payment channel networks are summarised as follows:

1. **Balance Privacy**: No honest user loses coins even if some involving participants are corrupted.

2. **Privacy (Off-Path)**: For a pay operation, involving only honest users, no user outside the payment path learn about the payment related information.
3. **Privacy (On-Path)**: For a payment operation, the honest but curious users (not colluding) inside the payment path, learn no information about the user who is sending or receiving party.
4. **Hash-lock Decorrelation**: In the payment path $\boxed{S \rightarrow u_1 \rightarrow u_2 \rightarrow u_3 \rightarrow R}$ shown in Fig.2, as the hash-lock is same through out the route, the users can know that this is the same payment they got from S and get more information on that.
5. **Stealing Transfer Fee Attack**: Let us consider a payment path used by $\boxed{S}$ $\boxed{\rightarrow u_1 \rightarrow u_2 \rightarrow u_3 \rightarrow R}$, to pay v coins to R. Thus, the sender S needs to send total v_1 coins, where $\boxed{v_1 = v + \sum_{i=1}^{n} f(u_i)}$. Suppose user u_1 and user u_3 are colluded on this payment path, to eliminate intermediate user u_2 from participating in the successful completion of payment, thereby stealing the payment forwarding charges $f(u_2)$, which was contracted for honest nodes of the payment path. An illustrative example of *stealing transfer fee attack* is depicted in Fig. 2 with red coloured arrow (step 7 and 8) during lock releasing phase.

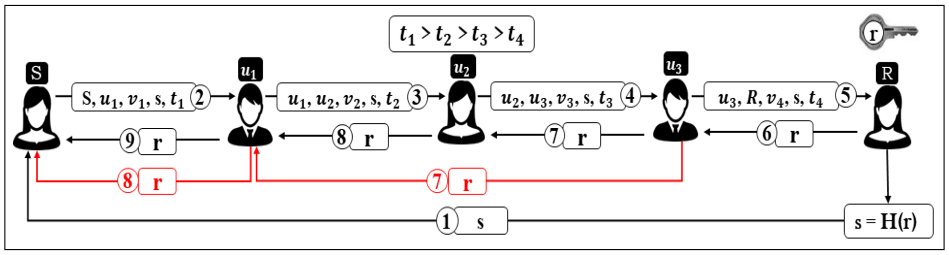


Fig. 2: HTLC & Stealing Transfer Fee Attack

5 The Proposed MAPPCN

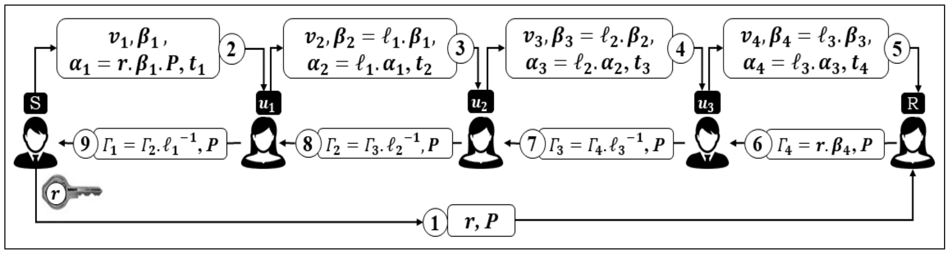
This section presents our proposed simple, but effective Multi-hop Anonymous and Privacy-Preserving Payment Channel Network named (*MAPPCN*). *MAPPCN* uses Elliptic curve cryptography (*ECC*) [12]. Let E_p be an additive elliptic curve group over a finite field F_p. The scalar multiplication on the group E_p is defined as $c \cdot P = P + P + \cdots + P(c - times)$, where $c \in z_p$ and P is a base point on E_p. The Elliptic curve discrete logarithmic problem (*ECDLP*) is defined as given P and $c \cdot P$, it is computationally infeasible to find c. The security of *MAPPCN* relies on *ECDLP* problem.

5.1 Assumptions

We focus on the design of a privacy-preserving PCN mechanism, but the efficiency of the routing protocols in PCNs is beyond the scope of this work. We assume that each pair of users communicate through a secure and authenticated channel for sharing the payment channel. Unlike *MHTCL* [14] and *CHTLC* [22], we are not considering the existence of a secure channel between the sender and each intermediate user for *setup* phase. Each intermediate node has only knowledge about its previous neighbor and next neighbor of the corresponding payment path. But sender has complete information about each and every node information; like user identifier (public key), lock-time (t_i), transaction fee ($f(u_j)$), channel identifier ($c_{<u_i,u_{i+1}>}$), channel capacity or collateral cost (γ_i) of user u_i of the corresponding channel ($c_{<u_i,u_{i+1}>}$).

5.2 MAPPCN Construction Overview

As shown in Fig. 3, consider a payment channel network in which the payment to be occurred between user S (sender or payer) and R (receiver or payee) through three intermediate users u_1, u_2 and u_3. Thus the payment path of five users exists as $\boxed{S \rightarrow u_1 \rightarrow u_2 \rightarrow u_3 \rightarrow R}$.


$v_1, \beta_1,$
$\alpha_1 = r.\beta_1.P, t_1$
2
$v_2, \beta_2 = \ell_1.\beta_1,$
$\alpha_2 = \ell_1.\alpha_1, t_2$
3
$v_3, \beta_3 = \ell_2.\beta_2,$
$\alpha_3 = \ell_2.\alpha_2, t_3$
4
$v_4, \beta_4 = \ell_3.\beta_3,$
$\alpha_4 = \ell_3.\alpha_3, t_4$
5
S
u_1
u_2
u_3
R
9
$\Gamma_1 = \Gamma_2.\ell_1^{-1}, P$
8
$\Gamma_2 = \Gamma_3.\ell_2^{-1}, P$
7
$\Gamma_3 = \Gamma_4.\ell_3^{-1}, P$
6
$\Gamma_4 = r.\beta_4, P$
r
1
r, P
Fig. 3: MAPPCN: Anonymous Multi-hop Privacy-preserving PCN

At first, sender S generates a random number r and base point P in Elliptic curve group E_p. Sender S sends $\boxed{< r, P >}$ to receiver R through a secure channel. Next S sends $\boxed{S, u_1, v_1, \beta_1, \alpha_1, t_1}$ to its next neighbor u_1. Each user u_i generates a random number ℓ_i and computes $\boxed{\beta_{i+1} = \ell_i \cdot \beta_i}$ and $\boxed{\alpha_{i+1} = \ell_i \cdot \alpha_i}$, sends $\boxed{< \beta_{i+1}, \alpha_{i+1} >}$ to its next neighbor u_{i+1}. Consequent upon receiving the tuple $\boxed{< \beta_{n+1}, \alpha_{n+1} >}$ from its neighbor u_n, the receiver R verifies if $\boxed{r \cdot \beta_{n+1} \cdot P \overset{?}{=} \alpha_{n+1}}$ then computes $\boxed{\Gamma_{n+1} = r \cdot \beta_{n+1}}$ and returns to its previous neighbor. The neighbor u_n upon receiving Γ_{n+1} from R, verifies if $\boxed{\Gamma_{n+1} \cdot P \overset{?}{=} \alpha_{n+1}}$ and releases the locked coins accordingly, and sends $\boxed{\Gamma_n = \ell_n{}^{-1} \cdot \Gamma_{n+1}}$ to its previous neighbor. Each intermediate neighbor u_i performs the similar operations and releases the locked amount and returns $\boxed{\Gamma_i = \ell_i{}^{-1} \cdot \Gamma_{i+1}}$ to its previous neighbor. Thus each user in the path receives the committed coins.

MAPPCN Construction: The primary three operations of *MAPPCN* are discussed here as follows:

Algorithm 1: MAPPCN Payment Protocol

```
Sender's Payment Routine:
1: Generate a random number $r$ and a Elliptic curve base point $P$
2: Send $\langle r, P \rangle$ to receiver R via a secured channel
3: $v_1 = v + \sum_{i=1}^{n} f(u_i)$
4: if $(v_1 \le \gamma_0)$ then
5:     $\gamma_0 = \gamma_0 - v_1$
6:     $t_0 = t_{now} + \Delta \cdot n$
7:     Generate a random value $\beta_1$
8:     Compute $\alpha_1 = r \cdot \beta_1 \cdot P$
9:     ETLC(S, $u_1$, $v_1$, $\beta_1$, $\alpha_1$, $t_1$)
10: else
11:     Abort
12: end if

Receiver's Payment Routine:
1: if $((r \cdot \beta_{n+1} \cdot P == \alpha_{n+1})$ and $(t_{n+1} > t_{now} + \Delta))$ then
2:     Compute $\Gamma_{n+1} = r \cdot \beta_{n+1}$
3:     Send $\langle \Gamma_{n+1}, P \rangle$ to user $u_n$
4: else
5:     Abort
6: end if

Intermediate's Payment Routine:
1: if $((v_{i+1} \le \gamma_i)$ and $(t_{i+1} == t_i - \Delta))$ then
2:     $\gamma_i = \gamma_i - v_{i+1}$
3:     Generate a random value $\ell_i$
4:     Compute $\beta_{i+1} = \ell_i \cdot \beta_i$ and $\alpha_{i+1} = \ell_i \cdot \alpha_i$
5:     ETLC($u_i$, $u_{i+1}$, $v_{i+1}$, $\beta_{i+1}$, $\alpha_{i+1}$, $t_{i+1}$)
6: else
7:     Abort
8: end if
9: if $(\Gamma_{i+1} \cdot P == \alpha_{i+1})$ then
10:     $\Gamma_i = \Gamma_{i+1} \cdot {\ell_i}^{-1}$
11:     Send $\langle \Gamma_i, P \rangle$ to user $u_{i-1}$
12: else
13:     Abort
14: end if

/*$\Delta$: For some positive value, $u_0$ = S, $u_{n+1}$ = R*/
```

1. *OpenChannel*$(u_i, u_j, \gamma, t, f(u_j))$: This operation is to open a payment channel between users u_i and u_j. For this u_i, u_j create a joint-wallet depositing collateral cost. Both the users agreed upon the initial capacity of the channel (γ), channel expiration timeout (t), the fee charged to use the channel ($f(u_j)$) and a channel identifier ($c_{<u_i,u_j>}$). Subsequently the transaction is added to the blockchain, and the operation returns *true*. If any of the previous steps is not carried out, the operation returns $false$.
2. *CloseChannel*$(c_{<u_i,u_j>}, v)$: This operation is used by two users (u_i, u_j) sharing an open payment channel ($c_{<u_i,u_j>}$) to close it at the state defined by v and accordingly update their balances in the blockchain. It returns *true* if it is successfully included in to the blockchain, otherwise $false$.
3. *Pay*$((c_{<S,u_1>}, \cdots, c_{<u_n,R>}), v) \rightarrow \{false, true\}$: This operation takes a list of payment channels $(c_{<S,u_1>}, c_{<u_1,u_2>}, \cdots, c_{<u_n,R>})$ to the corresponding payment path from payer S to payee R and the payment amount v. In each payment channel $c_{<u_i,u_i+1>}$ in the path, it has at least a current balance $\gamma_i \ge v_i$, where $v_i = v - \sum_{j=1}^{i-1} f(u_j)$, for each intermediate user u_i, the pay operation deducts it's transaction fee from the amount v_i' and returns *true*. Otherwise, none of the balances of payment channel be updated and the *pay* operation returns $false$.

Algorithm 1 depicts *MAPPCN* payment protocol. The detailed protocol steps for all the parties (sender, receiver, and intermediate users) are discussed as under.

Sender's payment routine: We assume that there exists a payment channel network or payment path denoted as $\boxed{\mathcal{P} = \{S, u_1, u_2, \cdots, u_n, R\}}$. Payer $\boxed{S}$ generates a random number r and a base point P in an Elliptic curve group E_p and sends it to payee $\boxed{R}$ in a secured channel. Further, $\boxed{S}$ generates a random value β_1 and computes $\boxed{\alpha_1 = r \cdot \beta_1 \cdot P}$ (Steps 7-8). Then, it creates an Elliptic curve based time-lock commitment or contract i.e., $\boxed{\text{ETLC}(\boxed{S}, \boxed{u_1}, v_1, \beta_1, \alpha_1, t_1)}$ to the next neighbor $\boxed{u_1}$ (Step 9) and promising that if $\boxed{u_1}$ can provide a pair of value $\boxed{< \mathit{\Gamma}_1, P >}$ within t_1 time such that $\boxed{\mathit{\Gamma}_1 \cdot P \overset{?}{=} \alpha_1}$, user $\boxed{S}$ pays v_1 coins to $\boxed{u_1}$.

Intermediate's payment routine: At the time of locking or commitment phase, each intermediate user $\boxed{u_i}$ receives the payment request from its previous neighbor $\boxed{u_{i-1}}$, then it checks (i) if the commitment request of user $\boxed{u_{i-1}}$ is fulfilled, and it has enough amount of coins (γ_i) i.e., $\boxed{\gamma_i \geq v_{i+1}}$ $\big(v_{i+1}$: after deducting transaction fee from v_i, $\boxed{v_i = v_1 - \sum_{j=1}^{i-1} f(u_j)}\big)$. (ii) Correctness of the lock-time commitment t_{i+1} i.e.,$\boxed{t_{i+1} \overset{?}{=} t_i - \mathit{\Delta}}$, otherwise it aborts. Then, the intermediate user $\boxed{u_i}$ generates a random value ℓ_i $(1 \leq i \leq n)$ and computes $\boxed{\beta_{i+1} = \ell_i \cdot \beta_i}$ and $\boxed{\alpha_{i+1} = \ell_i \cdot \alpha_i}$. Then, it creates a time-lock commitment i.e., $\boxed{\text{ETLC}(\boxed{u_i}, \boxed{u_{i+1}}, v_{i+1}, \beta_{i+1}, \alpha_{i+1}, t_{i+1})}$ to the next neighbor $\boxed{u_{i+1}}$ (Step 5). During lock release, user $\boxed{u_i}$ waits for a pair of value $\boxed{< \mathit{\Gamma}_{i+1}, P >}$ from user $\boxed{u_{i+1}}$, to fulfill the condition $\boxed{\mathit{\Gamma}_{i+1} \cdot P \overset{?}{=} \alpha_i}$ and claims the coins from user $\boxed{u_i}$ (Step 9). With the help of $\mathit{\Gamma}_{i+1}$, user $\boxed{u_i}$ computes $\boxed{\mathit{\Gamma}_i = \mathit{\Gamma}_{i+1} \cdot \ell^{-1}}$ for releasing left-lock, and sends $\boxed{< \mathit{\Gamma}_i, P >}$ to user $\boxed{u_{i-1}}$ and claims the coins.

Receiver's payment routine: Once the payee $\boxed{R}$, receives the ETLC commitment from its previous neighbor $\boxed{u_n}$, it checks the validity of the commitment $\boxed{r \cdot \beta_{n+1} \cdot P \overset{?}{=} \alpha_{n+1}}$ and $\boxed{t_{n+1} \overset{?}{>} t_{now} + \mathit{\Delta}}$ i.e., it can meet the condition within time t_{n+1} (Step 1). Then it computes $\boxed{\mathit{\Gamma}_{n+1} = r \cdot \beta_{n+1}}$ and sends $\boxed{< \mathit{\Gamma}_{n+1}, P >}$ to $\boxed{u_n}$ to claim the coins (Steps 2-3). It aborts otherwise.
Since each intermediate user $\boxed{u_i}$ $(1 \leq i \leq n)$, redeems the amount by generating the collision with the arguments $\boxed{< \mathit{\Gamma}_i, P >}$ it received from the next neighbors, and waits for the next neighbor to redeem the amount. Thus, all the users are paid with the promised amount after completion of protocol.

5.3 Discussion

MAPPCN achieves the following properties:

1. **Atomicity**: Every user in the path ($\boxed{u_1}$, $\boxed{u_2}$, $\boxed{u_3}$) would be able to release its left lock by computing $\boxed{\mathit{\Gamma}_i = {\ell_i}^{-1} \cdot \mathit{\Gamma}_{i+1}}$, thus by releasing the right lock $\mathit{\Gamma}_{i-1}$ only.
2. **Consistency**: No user ($\boxed{u_1}$ or $\boxed{u_2}$ or $\boxed{u_3}$) can release the lock i.e., compute $\mathit{\Gamma}_i$ without its corresponding right lock $\mathit{\Gamma}_{i-1}$.
3. **Relationship anonymity**: Intermediate user has no information about the set of users in PCN path except its right neighbor. Moreover, unlike *MHTLC* [14] and *CHTLC* [22], in *MAPPCN* sender/ receiver does not send any information to all intermediate users except its direct neighbor.
4. **Balance Privacy**: Let $\boxed{u_i}$ be an intermediate user in a payment $\boxed{pay((c_{<S,u_1>}, c_{<u_1,u_2>}, \cdots, c_{<u_n,R>}), v))}$. In lock releasing phase, if the user $\boxed{u_{i+1}}$ multiply some value (or send the same value which was received) rather than ℓ_{i+1}^{-1} (step 10, Intermediate Node Payment Routine of Algorithm 1) maliciously, then the user $\boxed{u_i}$ gets the incoming message $\boxed{< \mathit{\Gamma}'_{i+1} \neq \mathit{\Gamma}_{i+1}, P >}$, thus fails during verification so does not release the lock.
5. **Privacy (Off-Path)**: In the off-chain or layer-2 payment channel network, all the communications that happened between the users is in the secured channel. Therefore any intermediate user $\boxed{u_i}$ involved in the payment does not learn about the payment value except the neighbor information.
6. **Privacy (On-Path)**: In the state-of-the-art off-chain payment channel network like MHTLC [14] and CHTLC [22], the sender sends some path specific secret information to each user of the payment. So, each user knows the sender. But, in the proposed MAPPCN payment network, each intermediate user $\boxed{u_i}$ has only information about its previous and next neighbor information. Therefore any (corrupted or honest) user cannot determine the correct sender-receiver pair, thus MAPPCN and achieves sender and receiver anonymity.

The proposed *MAPPCN* payment protocol gives the same level of security as that of *MHTLC* [14] and *CHTLC* [22] without requiring zero-knowledge proofs (*ZKP*) or chameleon hash (which require exponential operations). The security of *MAPPCN* follows the security model used by *MHTLC* [14] and *CHTLC* [22] according to the universal composable (*UC*) security paradigm [6] and relies on ECDLP. On the top, MAPPCN achieves following interesting properties.

1. **No Setup Phase**: The proposed payment protocol, we does not opt setup phase. In MHTLC, the sender computes zero-knowledge proofs (π takes 309 ms per NIZK proof) for communication (1.65 MB data per NIZK proof) between the intermediate users to ensure the correctness of the received message. At the same time, in CHTLC, the sender computes chameleon hash (256 bytes, $\approx$ 55 ms per user) for each user and sends it in a private channel. MAPPCN protocol is efficient; as, the sender does not compute any intensive operations. Therefore the computational overhead of the sender is very less as compared to the aforementioned *atomic swap* protocols.
2. **Sender and Receiver Anonymity**: In both the MHTLC and CHTLC, the sender sends some path specific secret information to each intermediate user.

Table 2: Computation overhead comparison

Schemes	Setup Phase	Commitment Phase	Releasing Phase	Total Operations
MAPPCN	$\cdots$	$\mathbf{(n+1)}$ **ECSM**	$\mathbf{(2 \cdot n+4)}$ **ECSM**	$\mathbf{(3 \cdot n+5)}$ **ECSM**
MHTLC [14]	$(n+1)$ H, n NIZK	$(k \cdot n+1)$ H	$\cdots$	$\mathbf{((k+1) \cdot n+2)}$ **H, n NIZK**
CHTLC [22]	$(n+2)$ CH	$(n+1)$ CH	$(3 \cdot n+3)$ CH	$\mathbf{(5 \cdot n+6)}$ **CH**

CH: Chameleon Hash 1 CH = 2× Exponentiation operations H: Hash (SHA256)
n: # intermediate users/ hops NIZK: Non-Interactive Zero-Knowledge
ECSM: Elliptic Curve Scalar Multiplication k: Constant

So each intermediate user knows the sender, thus loses sender anonymity. But in the proposed *atomic swap* protocol, the sender does not send any information to each intermediate user except its next neighbor, therefore, achieves strong *sender* and *receiver* anonymity.

3. **Stealing Transfer Fee Attack Resistance**: Let $\mathcal{P} = \{$S$, \cdots,$ u_{i-1}, u_i, u_{i+1}, $\cdots,$ R$\}$ be a payment path. Suppose user u_{i-1} and u_{i+1} colluded to steal the transaction fees of honest user u_i. But, neither of them can derive the secret used ℓ_i or release the lock that they took part in the corresponding payment without the intervention of the user u_i.

6 Security and Performance Analysis

6.1 Security Analysis

Theorem 1. MAPPCN *UC-realizes the ideal functionality* $\mathcal{F}$*, if* ECDLP *is computationally difficult.*

Proof. Consider a simulator $\mathcal{S}$ which simulates the (real world) execution *MAPPCN*, while interacting with the ideal functionality $\mathcal{F}$. $\mathcal{S}$ also handles the users corrupted by the adversary $\mathcal{A}$. The following PCN operations are to be simulated.

OpenChannel$(c_{<u_i,u_j>}, \gamma, t, f(u_j))$: Let the request be initiated by u_i, there would be two cases arise:

- *Corrupted* u_i: $\mathcal{A}$ impersonates u_i and sends a request $(c_{<u_i,u_j>}, \gamma, t, f(u_j))$ to $\mathcal{S}$. Consequently, $\mathcal{S}$ initiates an agreement protocol with $\mathcal{A}$ and sends $(open, c_{<u_i,u_j>}, \gamma, t, f(u_j))$ to $\mathcal{F}$. $\mathcal{F}$ returns $(c_{<u_i,u_j>}, h)$.
- *Corrupted* u_j: $\mathcal{F}$ sends message $(c_{<u_i,u_j>}, v, t, f(u_j))$ to $\mathcal{S}$. Upon receiving this message, $\mathcal{S}$ initiates an agreement protocol with $\mathcal{A}$, on behalf of u_i, for opening a channel. If it is successful $\mathcal{S}$ sends a successful message to $\mathcal{F}$, which returns $(c_{<u_i,u_j>}, h)$. Then $\mathcal{S}$ initializes the list $\mathcal{L}_{c_{<u_i,u_j>}}$ and appends $(h, v, \perp, \perp)$, otherwise it outputs $\perp$.

CloseChannel$(c_{<u_i,u_j>}, v)$: Like open channel, two cases would arise:

- *Corrupted* u_i: $\mathcal{A}$ (impersonates u_i) sends close channel request. Consequently S retrieves $\mathcal{L}_{c_{<u_i,u_j>}}$ for some value $(h, v, \_, \_)$ if could not found aborts. Otherwise, $\mathcal{S}$ sends close $(c_{<u_i,u_j>}, v, h)$ to $\mathcal{F}$.
- *Corrupted* u_j: $\mathcal{F}$ sends $(c_{<u_i,u_j>}, h, \perp)$ to $\mathcal{S}$. $\mathcal{S}$ informs $\mathcal{A}$ as closing of channel $c_{<u_i,u_j>}$.

Pay$(c_{S,u_1}, c_{u_1,u_2}, \cdots, c_{u_n,R}, v)$: Each user acts according to its role in the protocol as defined below:

- *Sender*: To initiate the payment, $\mathcal{A}$ provides $\boxed{m_1 = (c_{<S,u_1>}, v_1, t_1, \beta_1, \alpha_1)}$ to u_1 notifying the path information i.e., the next hop address to be forwarded. If $\boxed{t_0 \geq t_1}$, then simulator $\mathcal{S}$ sends pay $\boxed{(v_i, c_{<u_{i-1},u_i>}, t_i)}$ to $\mathcal{F}$, otherwise aborts. For each u_i, the simulator $\mathcal{S}$ confirms the payment only when it receives Γ_i from u_i such that $\boxed{\Gamma_{i+1} \cdot \ell_i^{-1} \cdot P = \alpha_i}$, and thus, for receiver R if $\boxed{\ell_n \cdot \beta_n \cdot r \cdot P = \alpha_n}$. For each confirmed payment, the entry in $\mathcal{L}_{c_{<u_i,u_j>}}$ containing $(h_i^*, v^*, \_, \_)$ with lowest v^* is updated by $(h_i, v^* - v_i, \Gamma_i)$.
- *Receiver*: $\mathcal{F}$ sends $\boxed{(h, c_{<u_n,R>}, v, t_n)}$ to simulator $\mathcal{S}$. Consequently simulator $\mathcal{S}$ chooses ℓ, r randomly and sends a triple as $\boxed{(\alpha, \beta, \beta \cdot r \cdot P)}$. If $\mathcal{A}$ returns $\boxed{< \Gamma, P >}$ s.t. $\boxed{\Gamma \cdot P = \alpha}$, then $\mathcal{S}$ returns $\top$ to $\mathcal{F}$, otherwise it sends $\perp$.
- *Intermediate user*: $\mathcal{F}$ informs $\mathcal{S}$ about the corrupted users in the payment with a message of the form $\boxed{m = (h_i, h_{i+1}, c_{<u_{i-1},u_i>}, c_{<u_i,u_i+1>}, v, t_{i-1}, t_i)}$. Simulator S chooses randomly ℓ_i and sends $\boxed{\langle c_{<u_{i-1},u_i>}, c_{<u_i,u_{i+1}>}, \ell_i, \alpha_i = r \cdot \ell_i \cdot P, \ell_{i+1}, \alpha_{i+1} = r \cdot \ell_{i+1} \cdot P, v, t_{i-1}, t_i \rangle}$ to $\mathcal{A}$. If $\mathcal{A}$ outputs r^* such that $\boxed{r^* \cdot \ell_i \cdot P = \alpha_i}$ then aborts.

Now it is trivial to observe the *indistinguishability* in both *OpenChannel* and *CloseChannel*. Moreover, the payment chain never stops at the honest node, so the simulation does not abort. Therefore, $\mathcal{A}$ could succeed to interrupt the payment or abort the simulation outputting the correct Γ_i such that $\boxed{\Gamma_{i+1} \cdot \ell_i^{-1} \cdot P = \alpha_i}$. In such case $\mathcal{A}$ could break *ECDLP* as communication in channel is private.
Thus MAPPCN is secure as long as ECDLP assumption holds.

6.2 Performance Analysis

Here, we discuss the performance of *MAPPCN* payment protocol in terms of computation overhead. Table 2 presents the comparison of different phases (Setup phase, Commitment or Locking phase and Lock releasing phase) of the multi-hop payment schemes.

MHTLC [14] requires $(n+1)$ hash (SHA256) operations and n zero-knowledge proof (NIZK) operations, while CHTLC [22] requires $(n + 2)$ chameleon hash (CH) operations (where n is the number of intermediate users, 1 CH = 2 $\times$ Exponentiation operations). On the other hand, MAPPCN protocol does not need setup phase. Also, zero-knowledge proofs and chameleon hash operations are the most expensive operations as compared to Elliptic curve point multiplication. In commitment or locking phase MHTLC requires constant operations (as reported by the authors, sender requires 309 ms to compute a NIZK proof $\pi_{\mathtt{i}}$ of size 1.65 MB and to verify $\pi_{\mathtt{i}}$ intermediate user takes 130 ms time), CHTLC requires $(n+1)$ chameleon hash operations and MAPPCN requires $(n+1)$ Elliptic curve scalar multiplication (ECSM) operations. At the time of lock releasing phase MHTLC, CHTLC, and MAPPCN requires $(3 \cdot n + 3)$ chameleon hash operations, and $(2 \cdot n + 4)$ Elliptic curve scalar multiplication operations respectively.

Thus, the overall computation overheads are $\big((k+1) \cdot n + 2\big)$ hash (SHA256) operations and n zero-knowledge proof (NIZK) operations are required to execute an instance of MHTLC, while $(5 \cdot n + 6)$ chameleon hash operations required for CHTLC. On the other hand, MAPPCN requires only $(3 \cdot n + 5)$ Elliptic curve scalar multiplication operations. Therefore, MAPPCN would achieve better performance.

7 Conclusion

In this paper, we proposed a novel anonymous privacy-preserving payment protocol called *MAPPCN* to address the payment path privacy and sender anonymity issues in PCNs. Security of *MAPPCN* relies on ECDLP and analysed using universal composable paradigm. *MAPPCN* requires lesser computation i.e., $(3 \cdot n + 5)$ Elliptic curve scalar multiplication operations for preserving sender and receiver anonymity. The implementation of *MAPPCN* is under progress.

Acknowledgement

We acknowledge the Ministry of Human Resource Development, Government of India for providing fellowship under Ph. D. programme to complete this work.

References

1. A scala implementation of the lightning network. `https://github.com/ACINQ/eclair`
2. c-lightning – a lightning network implementation in c. `https://github.com/ElementsProject/lightning`
3. Lightning network daemon. `https://github.com/lightningnetwork/lnd`
4. Raiden Network. `https://raiden.network/`
5. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. `https://lightning.network/lightning-network-paper.pdf`
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proceedings of the 42^{nd} IEEE Symposium on Foundations of Computer Science. pp. 136–. FOCS '01 (2001)
7. Egger, C., Moreno-Sanchez, P., Maffei, M.: Atomic multi-channel updates with constant collateral in bitcoin-compatible payment-channel networks. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019. pp. 801–815 (2019)

8. Green, M., Miers, I.: BOLT: anonymous payment channels for decentralized currencies. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 473–489. ACM (2017)
9. Gudgeon, L., Moreno-Sanchez, P., Roos, S., McCorry, P., Gervais, A.: SoK: off the chain transactions. IACR Cryptology ePrint Archive p. 360 (2019)
10. Heilman, E., Alshenibr, L., Baldimtsi, F., Scafuro, A., Goldberg, S.: Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In: 24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017 (2017)
11. Jourenko, M., Kurazumi, K., Larangeira, M., Tanaka, K.: SoK: A taxonomy for layer-2 scalability related protocols for cryptocurrencies. IACR Cryptology ePrint Archive **2019**, 352 (2019)
12. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation (Jan 1987)
13. Lind, J., Naor, O., Eyal, I., Kelbert, F., Sirer, E.G., Pietzuch, P.: Teechain: A secure payment network with asynchronous blockchain access. In: 27th ACM Symposium on Operating Systems Principles. pp. 63–79. SOSP (2019)
14. Malavolta, G., Moreno-Sanchez, P., Kate, A., Maffei, M., Ravi, S.: Concurrency and privacy with payment-channel networks. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017. pp. 455–471 (2017)
15. Malavolta, G., Moreno-Sanchez, P., Schneidewind, C., Kate, A., Maffei, M.: Anonymous multi-hop locks for blockchain scalability and interoperability. In: 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019 (2019)
16. McCorry, P., Bakshi, S., Bentov, I., Meiklejohn, S., Miller, A.: Pisa: Arbitration outsourcing for state channels. In: Proceedings of the 1st ACM Conference on Advances in Financial Technologies, AFT 2019, Zurich, Switzerland, October 21-23, 2019. pp. 16–30 (2019)
17. Miller, A., Bentov, I., Bakshi, S., Kumaresan, R., McCorry, P.: Sprites and state channels: Payment networks that go faster than lightning. In: Financial Cryptography and Data Security - 23rd International Conference, FC. pp. 508–526 (2019)
18. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf
19. Prihodko, P., Sakhno, K., Ostrovskiy, A., Zhigulin, S., Osuntokun, O.: Flare: An approach to routing in lightning network
20. Tairi, E., Moreno-Sanchez, P., Maffei, M.: A^2L: Anonymous atomic locks for scalability and interoperability in payment channel hubs. IACR Cryptology ePrint Archive p. 589 (2019)
21. Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger, https://ethereum.github.io/yellowpaper/paper.pdf
22. Yu, B., Kermanshahi, S.K., Sakzad, A., Nepal, S.: Chameleon hash time-lock contract for privacy preserving payment channel networks. In: Steinfeld, R., Yuen, T.H. (eds.) Provable Security (2019)
23. Zhang, D., Le, J., Mu, N., Liao, X.: An anonymous off-blockchain micropayments scheme for cryptocurrencies in the real world. IEEE Transactions on Systems, Man, and Cybernetics: Systems pp. 1–11 (2018). https://doi.org/10.1109/TSMC.2018.2884289