Information Sciences 630 (2023) 713–726

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Enabling scalable and unlinkable payment channel hubs with oblivious puzzle transfer

Yilei Wang [a], Ming Liu [a], Huawei Ma [b], Shuyu Fan [a], Huiyu Zhou [c], Siqi Ju [d], Xiaoying Wang [e,*], Qintai Yang [e,*]

[a] *School of Computer Science, Qufu Normal University, Rizhao, China*
[b] *Institute of Artificial Intelligence and Blockchain, Guangzhou University, China*
[c] *School of Computing and Mathematical Sciences, University of Leicester, Leicester, UK*
[d] *School of Information Engineering, East University of Heilongjiang, China*
[e] *Third Affiliated Hospital of Sun Yat-sen University, Guangzhou 510630, China*

ARTICLE INFO

Keywords:
Payment channel network
Oblivious transfer
Anonymity
Unlinkability
Security

ABSTRACT

Payment channel networks (PCNs) are effective techniques for extending the scalability of cryptocurrencies. It achieves this by establishing a direct off-line channel from the sender to the receiver, going through one intermediary (aka. the *hubs*). In such scenarios, the hubs know the origin and destination of each transaction flowing through them, which jeopardizes the privacy of the underlying systems. Unfortunately, former efforts in ensuring transaction unlinkability either rely on trusted mixing services, are inefficient constructed (*e.g.,* constructed inefficient cryptographic primitives), or have limited applicability. In this paper, we present ObliHub, an efficient payment channel scheme that conceals transaction direction information to the hubs. The core technique of ObliHub in achieving unlinkability is our tailored oblivious puzzle transfer protocol (OPT), which enables puzzle solving among the payer, the hub, and the receiver to be conducted in an oblivious manner – the hub center neither knows where a puzzle hint came from nor who acquired it. The implementation of ObliHub only requires efficient cryptographic primitives, and compared with A^2L (a state-of-the-art Bitcoin-compatible PCH using homomorphic encryption), ObliHub saves 0.2 seconds in computation time over previous solutions and improves transfer throughput. Besides, our scheme is in accord with Universal Composability (UC) framework and we provide a comprehensive security analysis of it.

1. Introduction

In recent years, with the growing transaction demand for cryptocurrencies such as Bitcoin and Ethereum, higher orders have been placed on cryptocurrency throughput. However, as the underlying technology of these cryptocurrencies, the scalability of blockchain has become the bottleneck that restricts cryptocurrencies from realizing large-scale commercial applications on the ground. Therefore, scaling has become the most pressing issue for blockchain at present. In general, there are three methods for scalability: layer 0, layer 1, and layer 2 scaling techniques. Layer 0 is to expand the capacity through the relay network. Layer 1 is to change the capacity expansion by changing the blockchain (*e.g.,* consensus mechanism), such as a sharding mechanism. The last

* Corresponding authors.
E-mail addresses: wxying@mail.sysu.edu.cn (X. Wang), yangqint@mail.sysu.edu.cn (Q. Yang).

https://doi.org/10.1016/j.ins.2023.02.024
Received 17 October 2022; Received in revised form 7 December 2022; Accepted 8 February 2023

Available online 17 February 2023

layer is to expand the calculation to the chain, such as side chain [1], Plasma [2], and off-chain payment channel. Lightning network (LN) [3] is a type of off-chain payment channel that completes transactions through a two-way payment channel between two nodes in the network. As of September 12, 2022, the number of nodes participating in Lightning Network in the 1ML website is 17,813, the number of channels is 86,459, and the network capacity is 4,747.67 BTC (https://1ml.com/). The Lightning Network is gaining influence in terms of the number of nodes and transactions. Moreover, Samson Mow, Chief Strategy Officer of Blockstream, claims that the throughput of the Lightning Network exceeds that of any new generation blockchain, even surpassing payment systems such as Visa or Alipay. However, the Lightning Network protocol requires finding efficient paths between nodes, a process that can compromise private data such as trader identity information, transaction receipt times, and payment sizes. Just like the security risks that exist in the blockchain [4–6], the leakage of these private data, especially the trader's identity information, may cause security issues such as linkability, deanonymization, and nonatomicity of transactions [7,8]. Therefore, the privacy security of users is one of the issues that need to be addressed urgently during the expansion of Lightning Network.

As mentioned above, previous works resort to either trusted hubs or complex cryptography tools to circumvent linkable. Suppose a sender and a receiver complete a transfer via a hub,[1] solving a puzzle to complete the transfer where the puzzle is the transfer condition between the hub and receiver and gets its correct solution from the sender and hub, thus ensuring that only the sender completes the transfer first to get the correct solution and the receiver can complete the transfer in order. This also ensures that as long as the puzzle is unlocked, the transfer amount will flow from the sender to the hub and then to the receiver. Specifically, the first hub generates a puzzle and sends it to the receiver, which then sends the puzzle to the sender for the correct solution. Sender then uses the answer to the same puzzle as a prerequisite for transferring funds to the hub to obtain the correct solution. From the hub's point of view, the hub first completes the transfer by sending a solution to the sender to obtain the transfer amount. The sender then receives the solution and sends it to the receiver through a secure channel, and the receiver uses the solution to unlock the puzzle to obtain the same amount as the previous transfer. At this point, the sender and receiver complete the transfer via the hub. However, in the transfer process, the three nodes transmit, solving the same puzzle, then the hub sends to the sender and receives the same solution from the receiver; the hub will know the transaction relationship between sender and receiver, causing the trader identity information leakage. To avoid this problem, A^2L uses cryptographic primitives such as homomorphic encryption and adapter signature to blind the puzzle and its solution twice in turn so that the corresponding solution differs between the puzzle sent and receives by the hub. Consequently, the hub facilitates transaction transfer without revealing the linkage between sender and receiver. In addition, because the receiver does not respond to the request, the hub will be attacked by a denial of service attack (DoS), which eventually causes the hub's funds to be locked. To prevent attacks, A^2L introduces the registration process. However, this process will also increase the number of communication wheels in the agreement. Therefore, the blindization process of puzzle and solution and resistance to DoS attacks have become bottlenecks that affect the efficiency and safety of the hub transfers.

In fact, a hub cannot handle transfers between only one sender and one receiver. More generally, multiple senders and receivers transfer funds through the same hub. For each transfer, the hub generates a pair of puzzles and solutions, and the hub has various solutions, which can be used to complete the transfer with these multiple solutions and their numbers. To solve the problem of transferring different solutions, this paper proposes an oblivious transfer protocol, where the hub first sends the puzzle and its number to the receiver, and the receiver sends the number to the sender to find the correct solution instead of the previously sent puzzle. Then sender uses the number and hub for the oblivious transfer protocol. The protocol can perform the transfer function along with the oblivious transfer solution. More specifically, during the interaction between the hub and sender, only the correct solution is sent, but it is unknown which number corresponds to the sent solution. The sender sends the correct solution to the receiver to complete the transfer with the hub. In this case, although the transfer between receiver and hub can be completed with the correct solution, the hub does not know which sender this solution corresponds to. This is because, at the time of sending, hub does unknown which solution he is sending. This not only hides the relationship between the two parties of the transfer but also reduces the communication overhead and transmission time by transmitting a message with a short number instead of a puzzle. Further, to prevent DoS attacks, consider letting the receiver pledge a deposit through a smart contract and deducting the deposit once the receiver does not respond to the hub's transfer request within a specified time. This way, the receiver will suffer the consequences of launching the attack itself. As a consequence, no additional registration phase is required to prevent DoS attacks while reducing the number of communication rounds. Therefore, by improving the oblivious transfer (OT) protocol and the deposit smart contract, the number of rounds of the transfer protocol can be reduced, and its efficiency can be improved.

In this paper, we first construct an OT, an oblivious puzzle transfer protocol that can realize the transfer relationship between sender and hub by numbering bewilderingly to get a solution to hide the sender and receiver and complete the transfer simultaneously. Subsequently, we propose a novel protocol - ObliHub, which leverages oblivious transfer instead of adapter signature. In particular, the hub and receiver first generate some random puzzles. Then the sender sends the solution to the hub via oblivious transfer. Then, the hub learns the solution without linking the counterparties. Thus, the ObliHub can guarantee unlinkable, anonymization, and atomicity. The contributions are as follows. At the same time, its safety was proved under the Universal Composability framework [10,11].

1. We built an ObliHub payment channel scheme. First, create a puzzle, and then use the oblivious puzzle transfer protocol between the sender and hub to transfer the solution of the puzzle. The security of OPT can guarantee the security and privacy of both parties. At the same time, ObliHub can also guarantee the atomicity and unlinkability of transfer transactions (§3).

[1] In [9], hub is also known as tumbler.

2. We propose an oblivious puzzle transfer protocol. In addition, we prove the security of the oblivious puzzle transfer protocol based on the general composability framework (§4).
3. We compare the performance of ObliHub and A^2L in terms of the number of wheels, computing time, *etc.* Among them, compared with A^2L, ObliHub is reduced from phase 3 to phase 2 and ensures the atomicity and unlinkability of payment. The simulation results show that the running time of ObliHub is 0.2 seconds less than that of A^2L (§5).

2. Preliminaries

Let's review some cryptographic primitives and basic concepts in this section.

2.1. Payment channel hub

Both payment channel hub (PCH) and PCN are off-chain payments, and the three operations of PCH payments are OpenChannel, CloseChannel, and Pay, of which OpenChannel and CloseChannel are two essential parts of all payment channels. In addition, the Pay operation differs from PCN mainly because it is a balance update operation between the sender, the receiver, and the intermediary. The following is an overview of its operation.

OpenChannel($P_i, P_j, cash_i, cash_j, f, state$): The channel node initiates a transaction on the blockchain to open a payment channel, and both parties or one party puts in money. P_i and P_i denote the node, and the payment channel between P_i and P_j can be denoted as $C(P_i, P_j)$, abbreviated as C. The amount deposited in the channel by the node is recorded as $cash$ (equal to the channel capacity). The f denotes the amount of transaction fee for using the channel, and the $state$ denotes the status of the channel: open or closed, and the record of the latest balance allocation.

CloseChannel(C, $state$): confirms the state of the channel: open and the final balance of both parties, initiates a transaction on the chain to close the channel and modifies the state of the channel to close. The $state$ is used as a list containing the results of the balance distribution after the transaction is completed.

Pay(P_S, P_T, P_R, V): updates the latest balance of both sides of the channel. V indicates the amount to be transferred. We note that the channel between P_S and intermediary P_T is $C(P_S, P_T)$. The channel between P_T and P_R is $C(P_T, P_R)$. The two nodes agree on the transfer intention, record the latest balance when the node is guaranteed to have enough money in the channel, and the transfer condition is satisfied, i.e., $cash_S \geqslant V$ and $cash_T \geqslant V$ are satisfied, then update in $C(P_S, P_T)$ to update $cash_S -= V$ and $cash_T += V$. Update in $C(P_T, P_R)$ to update $cash_T -= V$ and $cash_R += V$. Add to $C(P_S, P_T).state$ and $C(P_T, P_R).state$, respectively.

2.2. Commitment scheme

The two participants of the commitment scheme [12] are the committer and the receiver. Let the message space of the commitment be M. The scheme consists of two phases, commitment and verification. In the first stage, the committer computes the commitment algorithm P_{COM}, $(com(m; r), decom) \leftarrow P_{COM}$, where com is a commitment to a message $m \in M$ and a random number r. Then the commitment $com(m; r)$, referred to as c, is sent to the receiver. In another phase, first, the committer sends the $decom = (m; r)$, i.e., the message m and the random number r together to the receiver. Secondly, the receiver verifies the consistency with the previous commitment result by verifying the algorithm $\{0, 1\} := V_{COM}(com, m, r)$. In summary, the commitment scheme ensures that the promised message is always consistent for the promising party and that the receiver can verify it. In this paper, we use a hash function-based commitment scheme with perfectly binding [13] and perfectly hiding [14] properties. Perfectly binding means that the commitments of different messages must be different. Perfectly hiding means that the adversary cannot distinguish by two com messages respectively.

2.3. Non-interactive zero-knowledge proofs

Let R be an NP relation. There is $(x, w) \in R$, where x is a statement and w is a witness. Let L be the set of instantiations consisting of statements in R. The non-interactive proof system for a relation R consists of a CRS generation algorithm, a prover P, and a verifier V. The Non-interactive zero-knowledge [15] of the relation R consists of two algorithms, denoted NIZK$=(P_{NIZK}, V_{NIZK})$, where the prover outputs a proof π via the $\pi \leftarrow P_{NIZK}(x, w)$ algorithm. The verifier gets the proof result using the verification algorithm $\{0, 1\} := V_{NIZK}(x, \pi)$, where 1 represents a successful verification, and 0 illustrates a failed verification. NIZK ensures that the verifier does not get any additional information other than that the witness knows π, ensuring privacy and security for both.

2.4. Oblivious transfer

The oblivious transfer is a two-party protocol in which the participants are called sender and receiver. In the classical *1-out-of-2* protocol, the sender inputs two personal values x_0, x_1, and the receiver inputs a selection bit $\sigma \in \{0, 1\}$. At the end of the protocol, the receiver can only receive x_σ. For the other value, the receiver cannot obtain any related information. There is no output from the sender. Here we focus on the general construction framework of OT protocol with the concept of "dual-mode encryption" proposed by Peikert et al. [16], which can also be extended to *1-out-of-2* OT. The protocol includes chaotic decryption mode and normal decryption mode, where chaotic decryption mode does not have a private key that can be decrypted. The crs is a common string that will be generated in both modes. The protocol uses sid to indicate the session number to execute the protocol. The Functionality is defined as follows.

Definition 2.4 *(Protocol dm^{mode}).* Sender P_1 holds x_0, x_1, and receiver P_2 holds the selection bit $\sigma \in \{0, 1\}$.

(1) Both parties first return crs with ideal functionality F_{CRS}^{mode}, indicating whether this session mode is chaotic decryption mode or normal decryption mode. Suppose it is in normal decryption mode.

(2) P_2 calculates $(pk, sk) \leftarrow KeyGen(crs, \sigma)$ and sends (pk, sid) to P_1. The sid indicates the current session number.

(3) P_1 computation $y_i \leftarrow Enc(pk, i, x_i)$, $i \in \{0, 1\}$, and sends (y_i, sid) to P_2. P_2 can solve the solution α_b in y_b.

The paper also introduces the concept of encryption branches, which means that each encryption branch corresponds to a (pk, sk) pair, but the encryption branches in chaotic decryption mode cannot decrypt the ciphertext on that branch because the public key does not have a corresponding private key.

2.5. Elliptic curve cryptography

Elliptic Curve Cryptography (ECC) security relies on the Elliptic Curve Discrete Logarithm Problem (ECDLP). In this paper, we use an ECC-based encryption and decryption scheme. The elliptic curve E is defined over a finite field F_q.

Definition 2.5. The ECC scheme can be defined by a set of parameters (q, a, b, G, n, h) as

(1) The prime number q determines the range of the finite field.

(2) Two parameters $a, b \in F_q$ of the elliptic curve equation.

(3) A finite point $G \in \mathrm{E}(F_q)$ is also called the base point of the generating subgroup.

(4) n is the order of the subgroup.

(5) An auxiliary factor h.

The main parameters of the algorithm are defined as the above six-tuple, and a more comprehensive explanation of them has been provided in previous papers [17]. Here we write the encrypted ciphertext of message m as $c = Enc(m)$. Moreover, compared to other public key encryption schemes, ECC is easy to implement in our scheme as it requires a smaller key size for the same level of security, which has a small time cost.

3. Our protocol

In §3.1, we first describe the backbone of ObliHub, the oblivious puzzle transfer protocol, followed by a full description of ObliHub in §3.2. As we have seen, the oblivious puzzle transfer protocol is the indispensable second phase of ObliHub.

3.1. Oblivious puzzle transfer protocol

We propose a two-party protocol, oblivious puzzle transfer protocol, as the second phase of the offline payment scheme ObliHub. In OPT, suppose P_T has multiple solutions $(\alpha_i, i \in \{0, ..., n\})$, each solution has a number, and the participant P_S has a solution number $b \in \{0, ..., n\}$. The primary purposes of the protocol are: (1) Only P_S gets α_b. (2) P_T and P_S simultaneously obtain an ID q, indicating that OPT has been completed, which is used to assist the ObliHub in completing the transfer. The reason why the ID q is introduced is that P_T and P_S need to disclose α_b when confirming the transfer transaction in the second phase of ObliHub. At this time, P_T can be based on α_b knowing the linkability between P_S and P_R. To solve this problem, we use the idea of coin tossing to let both parties calculate an ID q in OPT. In this way, when confirming the transfer transaction in ObliHub, q instead of α_b will be disclosed to avoid linkability problems.

The specific process of OPT protocol is shown in protocol 1.

Protocol 1 Oblivious puzzle transfer protocol.

```
Input: $P_S$ inputs request message $mes$ and solution number $b$; $P_T$ inputs solution set $\{\alpha_0, \alpha_1, ..., \alpha_n\}$ and a commitment $c$.
Auxiliary input: Session ID $sid$.
Hybrid functionality: $F_{CRS}^{mode}$ receives $(sid, b)$ from $P_S$ and $(sid, \alpha_0, \alpha_1, ..., \alpha_n)$ from $P_T$ and returns to $P_S$ and $P_T$ the same $(sid, crs)$.
Output: $P_S$ receives $q = pk \oplus P$ and $\alpha_b$. $P_T$ receives $q = pk \oplus P$.
1: $P_S$ sends request message mes$=(sid$, C$(P_S, P_T)$.state, $V)$, $P_T$ verifies that C$(P_S, P_T)$.state $\neq$ open or $V > cash_T$, then abort. Otherwise, $P_T$ sends commitment $c = Com(P; r)$ and proof $\pi$ to $P_S$;
2: $P_S$ and $P_T$ input $(sid, b)$ and $(sid, \alpha_i)$, i$=\{0, ..., n\}$ to $F_{CRS}^{mode}$, and then receive $(sid, crs)$ from $F_{CRS}^{mode}$, respectively. Next, $P_S$ gets the public key $pk$, and private key $sk$ on encrypted branch $b$ by computing $KeyGen(crs, b)$, and sends $pk$ to $P_T$;
3: $P_T$ computes each solution $y_i = Enc(pk, i, \alpha_i)$, i$=\{0, ..., n\}$ and sends the computed $y_i$ and the unpacking commitment $decom(c)$ to $P_S$;
4: $P_S$ gets $\alpha_b$ by calculating $Dec(sk, y_b)$, and terminates the transfer agreement if it cannot be decrypted. Otherwise, $P_S$ calculates the output $q$; $P_T$ calculates the output $q$.
```

(1) The protocol starts with P_S sending the transfer request message mes (step 1 in Protocol 1), and P_T verifies the mes. If the state of channel $C(P_S, P_T)$ is not open or the transfer amount V is greater than the channel balance, then abort. Otherwise, P_T sends a commitment $c = Com(P; r)$ on random P to P_R, where r is a random number.

(2) P_S and P_T input b and set $\{\alpha_0, \alpha_1, ..., \alpha_n\}$, respectively. P_S uses Hybrid functionality F_{CRS}^{mode} [16] to select the normal decryption mode for number b. After receiving the returned crs, P_S generates a pair of public and private keys (pk, sk). P_S sends pk to P_T (step 2

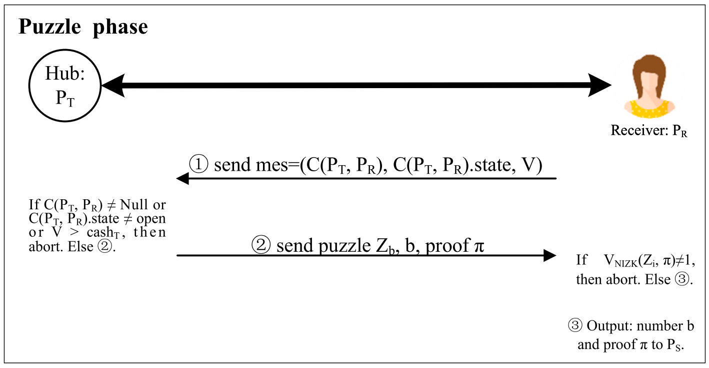
Puzzle phase
Hub: P_T
Receiver: P_R
① send mes=(C(P_T, P_R), C(P_T, P_R).state, V)
If C(P_T, P_R) ≠ Null or C(P_T, P_R).state ≠ open or V > cash_T, then abort. Else ②.
② send puzzle Z_b, b, proof π
If V_NIZK(Z_i, π)≠1, then abort. Else ③.
③ Output: number b and proof π to P_S.

Fig. 1. The diagram of the puzzle phase.

in Protocol 1). Here, the number of each solution is regarded as an encryption branch, and the number b is an encryption branch with a public key and corresponding private key. This ensures that P_S can only decrypt the solution corresponding to b when decrypting, and cannot solve other solutions.

(3) Then P_T encrypts its solutions in turn, and sends the settlement commitment to P_S together (step 3 in Protocol 1).

(4) P_S uses private key sk decryption α_b. If P_S decryption is wrong, then abort. Conversely, P_S and P_T can calculate the ID $q = pk \oplus P$ (step 4 in Protocol 1).

(5) When the protocol is completed, P_S receives the correct α_b, and by verifying that P_S and P_T have the same q value.

3.2. ObliHub

The transfer between P_S and P_R is realized through the PCH protocol, which includes three operations: OpenChannel, CloseChannel, and Pay. The first two operations in this PCH protocol are the same as those in literature [9], but the difference is the Pay operation. The Pay operation in this article is implemented through the ObliHub protocol.

Before introducing the protocol, the following assumptions are given.

(1) Since the transaction fee is optional during the transfer transaction, this paper assumes that the transaction fee is 0.

(2) Assume that all transfer amounts through the agreement are the same, which is recorded as amt.

(3) Suppose that there is a secure communication channel between the two parties of the transaction (for example, the anonymous communication channel F_{anon} [18] and the secure communication channel F_{smt} [19]).

Under the above assumptions, we propose the ObliHub protocol, in which Sender (P_S) transfers amt tokens to Receiver (P_R) through Hub (P_T). We follow the basic idea of solving the puzzle from the previous literature [9]. More specifically, ObliHub consists of two phases: the puzzle phase (between P_T and P_R) and the oblivious puzzle transfer phase (between P_S and P_T). In the puzzle phase, P_R sends a transfer request, and P_T gives P_R a puzzle Z_b (including the number b). If P_R solves the puzzle within the specified time, it can get tokens. Secondly, P_R sends the number to P_S through a secure channel between them. In the latter phase, P_T and P_S run the OPT protocol, and P_S gets the solution α_b of the puzzle Z_b from P_T, and P_T gets tokens from P_S.

At the same time, the calculation output $q = pk \oplus P$ is used as the identification of the end of the OPT protocol, avoiding the disclosure of α_b (refer to §3.1). Later, P_S will solve α_b is transmitted to P_R through the secure channel. In quest, P_R sends the puzzle solution α_b to P_T and gets tokens. So far, the transfer between P_S and P_R has been completed through P_T.

In the sequel, we present more details of these two phases in Fig. 1 and Fig. 2, respectively.

Puzzle phase: Since there is no direct connection between P_S and P_R, P_R sends mes and requests P_T to help complete the transfer. After receiving mes, P_T verifies whether channel ID $C(P_T, P_R)$ exists, whether channel status $C(P_T, P_R).state$ is open, and whether the remaining amount in the channel is lower than the transfer amount V. If the mes passes the verification, P_T generates the puzzle Z_b (including the number b) and its corresponding zero-knowledge proof π and sends it to P_R. More specifically, between P_T and P_R, P_T randomly selects a value r and a message M_b to generate a puzzle Z_b (P_T will pay amt token to P_R as long as P_R can give a solution to the puzzle), and uses non-interactive zero-knowledge proof to provide a proof π (proving that P_R, number b and Z_b are bound, and Z_b has a correct solution), in this way to avoid P_T not assisting in the transfer. After receiving the message, P_R first verifies whether P_R, number b, and Z_b are bound. If the verification is successful, P_R encrypts the number b and proof π, respectively, and sends them to P_S through the secure channel. Note that we still use the time lock in PCN, where the transfer will roll back if P_T and P_R do not solve the puzzle.

Oblivious puzzle transfer phase: This phase is consistent with the front (§3.1), and we make a detailed explanation here. As shown in Fig. 2, after P_S and P_T reach the payment intention, the P_S will receive the commitment $c = Com(P;r)$ sent by the P_T. P is the promised value, and r is a random number. Secondly, in this stage, P_S uses the serial number crs returned by F_{CRS}^{mod} to generate the public key pk and private key sk about encrypted branch b, and send the pk to P_T. Then, P_T encrypted its own solution set, and sent the promise $(P;r)$ to P_S. The second stage of the agreement is over, P_S solves α_b, and the payment can be completed when the $q = pk \oplus P$ of the two parties is the same. After P_S and P_T complete the payment, P_S can use ECC encryption [17] to be sent to P_R

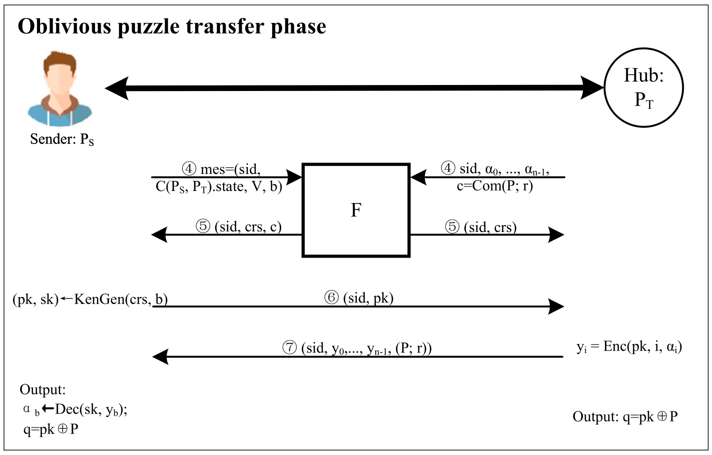
Oblivious puzzle transfer phase
Sender: P_S
Hub: P_T
④ mes=(sid, C(P_S, P_T).state, V, b)
④ sid, α_0, ..., α_{n-1}, c=Com(P; r)
F
⑤ (sid, crs, c)
⑤ (sid, crs)
(pk, sk)←KenGen(crs, b)
⑥ (sid, pk)
⑦ (sid, y_0,..., y_{n-1}, (P; r))
y_i = Enc(pk, i, α_i)
Output: α_b←Dec(sk, y_b); q=pk⊕P
Output: q=pk⊕P

Fig. 2. The diagram of the oblivious puzzle transfer phase.

through security channels. In the end, P_R decrypts and completes the payment with P_T. The transactions between the three people have been completed.

4. Security analysis

We demonstrate the proof of privacy and security of the OPT and ObliHub protocols in the universal composability framework. We first define the ideal functionality of F_{OPT} and $F_{ObliHub}$, who model the ideal behavior of OPT and ObliHub, respectively, including each party's possible inputs and outputs and the possible effects of adversary's operations on the execution results, in §4.1 and §4.2. And also the security proofs are given separately. Second, in §4.3, we discuss the security properties satisfied by ObliHub. Note that we provide the security of PCH depending on the safety of OPT and ObliHub protocols.

4.1. The security analysis of OPT

In this section, first, we specify the attacker model and define Universal Composability. Second, we describe the ideal operation of the OPT protocol. Finally, we prove step-by-step that our protocol is secure using the hybrid model approach; in other words, we prove that the protocol is secure by proving the correctness of Theorem 1.

Attacker model. We consider the nodes as interactive turning machines (ITM). Nodes interact with trusted functionality F through a secure communication channel. In the real world, we consider adversary A as a probabilistic polynomial time (PPT) machine that has access to the input of the corrupted node. In the ideal world, we use the simulator S to call the results of real-world adversary A's operations to facilitate access to the actual state of the corrupted node. We continue our previous considerations from the literature [20–22]. And in this paper, we only consider the static corruption model, *i.e.*, the adversary will only corrupt one node all the time.

In addition, let $EXEC_{OPT,A,\epsilon}$ denote the set of environment outputs ϵ when adversary A interacts with a user running the OPT protocol. We can write the set of environment outputs ϵ as $EXEC_{F,S,\epsilon}$ when the simulator S interacts with the trusted functionality F. Then the UC implementation concerning the global functionality is defined as follows.

Definition 1. (Global Universal Composability) A protocol ρ, UC realizes an ideal function F about global channels and global time. If there is a simulator S for any PPT adversary A, then for any environment ϵ, the output set $EXEC_{\rho,A,\epsilon}$ and $EXEC_{F,S,\epsilon}$ are computationally indistinguishable.

4.1.1. Ideal functionality F_{OPT}

We first present the definition of functionality F_{OPT}. In this protocol, P_S is the receiver, with the inputs of the session ID sid, the channel state $C(P_S, P_T).state$, the transfer amount size V, the selection bit $b \in \{0,...,n\}$ and the public key pk. The sender is P_T, with the input of the session ID sid, the solution set $(\alpha_i, i \in \{0,...,n\})$, and the random value P. The output of the P_S consists of the acknowledgment value $q = pk \oplus P$ and solution α_b. The output of the P_T is the acknowledgment value $q = pk \oplus P$. Finally, functionality F_{OPT} implements that P_S receives the solution corresponding to the number b and does not let P_T know exactly which one it is. Functionality F_{OPT} is described as follows.

Input: P_S inputs $(sid, C(P_S, P_T).state, V, b, pk)$, P_T inputs $(sid, \alpha_0, \alpha_1, ..., \alpha_n, P)$.

Output: P_S computes the output $q = pk \oplus P$ and a α_b. P_T computes the output q.

After receiving a request $mes = (sid, C(P_S, P_T).state)$ from P_S, F_{OPT} performs the following steps:

(1) Forward the message mes to P_T.

(2) If P_T does not respond or $C(P_S, P_T).state$ is not open then abort and tells P_S.

Ideal Functionality F_{COM}

- Commit: With P_0, P_1 interaction, if one side sends a Com(x; *sid*) request, if F_{COM} distinguishes that a commitment to x already exists then no feedback, otherwise store and return to the sender (receive, *sid*).
- Decommit: if inputs (decommit, *sid*) from P_1, if F_{COM} stored (x, *sid*) then send Dcom(x, *sid*) to P_2 and vice versa.

Fig. 3. Functionality commitment scheme.

(3) Otherwise, receive P_T inputs $(sid, \alpha_0, \alpha_1, ..., \alpha_n, P)$ and P_S inputs (sid, b, pk) and store them.
(4) Finally, sends (α_b, P, q) to P_S and (pk, q) to P_T.

4.1.2. The security proof of OPT

Theorem 1. *Assume that COM is a secure commitment scheme. The protocol OPT in Fig. 2 UC realizes the ideal functionality F_{OPT} in the F_{COM}-hybrid model.*

Proof. We discuss the security of OPT in the case of corrupted P_S and P_T, respectively. Then we construct a hybrid models and analyze them.

Hybrid h_0: The structure of h_0 is the same as that of the oblivious puzzle transfer protocol.

Hybrid h_1: Replace the steps in h_0 using the commitment scheme with calls to functionality F_{COM} of Fig. 3. Commitment has two essential properties: perfectly binding and perfectly hiding. Perfectly binding can be understood as the negligible probability of finding two different values x_1, x_2 (random number $r_1 \neq r_2$) satisfying $c_1 = c_2$ for any PPT adversary. Perfectly hiding: the adversary is the same as above, in that for $x_1 \neq x_2$, the commitment c_1, c_2 is indistinguishable.

Hybrid h_2: When P_S is corrupted, two behaviors of P_S will lead to simulator aborts. (1) The P_S may send the wrong pk instead of the pk generated by crs. This will result in P_T not getting the correct result q which will lead to P_S not being able to decrypt, then the simulator aborts. (2) The corrupted P_S always selects the chaotic mode which also leads to the absolute inability of the P_S to decrypt, then the simulator aborts.

On the other hand, when P_T is corrupted, P_T may change the output q, resulting in its q being inconsistent with P_S's q and unable to complete the payment, then simulator aborts.

Consequently, simulator S_{OPT} is defined as follows.

• **When P_T is corrupted, S_{OPT} does the following.**

① S sends $mes = (sid, C(P_S, P_T).state, V)$ to P_T by selecting the session ID sid that increases by 1.

② If A (on behalf of P_T) does not respond, the simulator S aborts. Otherwise, S receives a commitment $c = Com(P; r)$ from A.

③ S sends a random number b to F_{CRS}^{mode} and then randomly selects crs to P_T (sid, crs).

④ Computes $(pk, sk) \leftarrow KeyGen(crs, b)$, then sends pk to P_T.

⑤ S receive all $y_i = Enc(pk, i, \alpha_i)$ and decom $(P; r)$ computes by A.

⑥ Finally the output $q = pk \oplus P$ and $\alpha_b \leftarrow Dec(sk, y_b)$ can be computed.

• **When P_S is corrupted, S_{OPT} does the following.**

① S receives a request message mes = $(sid$, C$(P_S$, $P_T)$.state) from A (representing P_S). If $C(P_T, P_R)$ does not exist or $C(P_T, P_R)$.state $\neq$ open, the simulation is aborted. Otherwise, S selects a P and random number r to commit $c = com(P; r)$ to P_S.

② S performs a random selection of solutions to send (sid, α_0, α_1,...,α_n) to F_{CRS}^{mode}, and randomly selects a crs to send to P_T.

③ S receives pk from A (representing P_S), then computes all $y_i = Enc(pk, i, \alpha_i)$ and sends $(P; r)$ to P_S.

④ Finally, the output $q = pk \oplus P$ can be computed.

Next, we proceed to prove the indistinguishability of the neighboring experiments for environment ϵ.

Lemma 1. *For all PPT environment ϵ, it considers $EXEC_{h_0,A,\epsilon} \approx EXEC_{h_1,A,\epsilon}$.*

Proof. It is security proof that relies on the security of the commitment scheme COM. □

Lemma 2. *It considers $EXEC_{h_1,A,\epsilon} \approx EXEC_{h_2,A,\epsilon}$ for all PPT environment ϵ.*

Proof. We note that these two hybrids will differ if the experimental output fails, so limiting the probability of such events is sufficient. When an honest P_T interacts with a corrupted P_S, the P_S may enter the wrong number b as well as pk, causing the P_T to fail to calculate the q value correctly. But this will also cause the P_S to fail to encrypt and decrypt correctly, so the probability of the P_S doing so is negligible. □

Lemma 3. *For all PPT environment ϵ, it considers $EXEC_{h_2,A,\epsilon} \approx EXEC_{OPT,A,\epsilon}$.*

Proof. The two simulations and the final implementation are the same, only the statement expression changes. Therefore the distinguisher is indistinguishable. □

This concludes the proof of Theorem 1. □

4.2. The security analysis of $ObliHub$

Following, we analyze the security of the ObliHub scheme in a Universal Composability framework. We add the communication model to the attack model of the previous section. Then, the ideal operational procedure for each of the two phases of Oblihub is given. Finally, the protocol is shown to be secure by proving the correctness of Theorem 2.

Communication model. When communicating between the sender P_S and the receiver P_R, it is necessary to assume the communication channel, secure transmission function F_{smt}, and anonymous communication channel F_{anon} uses the definition in assumptions (§3). Secondly, the ideal functionality on the payment channel still selects the global channel functionality F_{GC} is generalized by the payment channel in PCN. For ease of understanding, the calculations in this paper are assumed to be transient. Continue using the ideal functionality F_{GDC} accepted by the formal communication channel in the previous literature [23]. Finally, we follow the concept of round formalization in A^2L through the global clock functionality F_{clock} representing time. Note that the Attacker model is consistent with §4.1.

Next, we start from the ideal $F_{ObliHub}$ and then simulate the impact of attacks or adversary input changes on the scheme's security. It turns out that Theorem 2 is safe in real execution.

4.2.1. Ideal functionality $F_{ObliHub}$

We define the ideal functional $F_{ObliHub}$ in the ObliHub scheme, which uses F_{GDC}, F_{smt}, and F_{anon}. Therefore, it can be regarded as an (F_{GDC}, F_{smt}, F_{anon}) - hybrid mode. $F_{ObliHub}$ manages a list $L(no, solution, id)$ that stores the receiver's number, solution, and identity id. $F_{ObliHub}$ is as follows:

Puzzle phase:

Input: P_R Inputs request message $mes = (C(P_T, P_R), C(P_T, P_R)$.state), P_T inputs m_b.

Output: gives P_R a puzzle Z_b.

After receiving the request mes from P_R, $F_{ObliHub}$ executes the following steps:

(1) Forward the message mes to P_T and simulator S.

(2) If P_T does not respond to or C (P_T, P_R).state ≠ open, $F_{ObliHub}$ aborts and tells P_R. Otherwise, $F_{ObliHub}$ receives a value m_b randomly selected by P_T.

(3) $F_{ObliHub}$ sends (Z_b, b, P_R) to P_T and P_R, notifies simulator S, and stores $L(b, m_b, P_R)$.

(4) The $F_{ObliHub}$ receives the request from P_R $(index, b, P_S)$, returns the $(index - req, b, P_R)$ to P_S, and notifies the simulator S.

Oblivious puzzle transfer phase:

Input: P_S inputs of the sender $(sid, b, C(P_S, P_T).state, V)$, P_T inputs $(sid, \alpha_0, \alpha_1 ... \alpha_n, P)$.

Output: P_S receives P and calculates output $q = pk \oplus P$ and α_b. P_T receives pk and calculates output q.

After receiving the request $mes = (sid, C(P_S, P_T).state)$ from P_S, $F_{ObliHub}$ executes the following steps:

(1) Forward the message mes to P_T and simulator S.

(2) If P_T does not respond or $C(P_S, P_T) =$ NULL or $C(P_S, P_T).state \neq$ open, abort, and tells P_S to abort. Otherwise, $F_{ObliHub}$ receives inputs $(sid, \alpha_0, \alpha_1 ... \alpha_n, P)$ and P_S inputs (sid, b).

(3) Then $F_{ObliHub}$ will send (pk, α_b, P) to P_S, (pk, P) to P_T, and sends (pk, α_b, P) to simulator S.

(4) The $F_{ObliHub}$ receives a request from the P_S $(index, \alpha_b, P_R)$, and then $F_{ObliHub}$ returns $(index, \alpha_b, P_S)$ to P_R and notifies simulator S.

4.2.2. The security proof of $ObliHub$

Theorem 2. *Assuming that the commitment COM is perfectly binding, NIZK is a non-interactive zero-knowledge scheme, the puzzle is based on the discrete logarithm puzzle, and the objective puzzle transfer protocol can safely transmit the solution. This ObliHub structure can be implemented in UC secure (F_{GDC}, F_{smt}, F_{anon}) - hybrid model functionality $F_{ObliHub}$.*

Proof. We construct the simulator S of the protocol in different cases and use the concept of the hybrid model to prove the security. Let's suppose there is a simulator S, which can simulate the implementation of this agreement in the ideal world and interact with functionality F.

Hybrid h_0: This corresponds to the original structure. That is, it is consistent with the protocol process of the ObliHub scheme.

Hybrid h_1: All calls to the commitment scheme are replaced by calls to the functionality F_{COM} in Fig. 3.

Hybrid h_2: All calls to the non-interactive zero-knowledge scheme NIZK are replaced by calls to functionality F_{NIZK} in Fig. 4. F_{NIZK} uses relation R.

Hybrid h_3: Based on hybrid h_2, replace OT operations related to $dual\text{-}mode$ encryption with calls to functionality F_{CRS}^{mode} in Fig. 5. In the oblivious puzzle transfer phase, you can ensure that crs is the value generated according to the input of both parties.

Hybrid h_4: In the puzzle phase, the honest P_T gives the corrupted P_R a puzzle Z_b and a proof π, and the P_R encrypts the proof π and solution number b to the P_S. If adversary A (for P_R) does not send a payment request, the aborting behavior of P_R is simulated.

Ideal Functionality F_{NIZK}

- P_1 has inputs (proof, sid, x, w). If (x, w) $\in$ R or sid has been used previously, then no feedback.
- Otherwise, feedback (proof, sid, x) to P_2.

Fig. 4. Functionality NIZK.

Ideal Functionality F_{CRS}^{mode}

- There are two participants P_1 and P_2. P_2 is the sender and P_1 is the receiver. P_1 chooses whether a mode is chaotic or normal encryption mode.
- Then they input (sid, x_0, x_1) and (sid, σ), $\sigma \in \{0, 1\}$, respectively, and F returns (sid, crs) to P_1 and P_2.

Fig. 5. Functionality F_{CRS}^{mode}.

If P_T is corrupted at this stage, no feedback is given to P_R's request. An unsolved puzzle is returned to P_R. If P_R's zero knowledge verification fails, abort will be simulated [24].

Hybrid h_5: In the oblivious puzzle transfer phase, when an honest P_S interacts with a corrupt P_T, the P_T may change the output $q = pk \oplus P$ (because of the previous hybrid model, it promises that the values of c and P will not change). However, this will lead to inconsistency between its q and the P_S's q in the payment, leading to the payment not being completed. Then abort simulation and output failure. In the oblivious puzzle transfer phase, when an honest P_T interacts with a corrupt P_S, the P_S always chooses the chaotic encryption mode or randomly sends a pk, which will cause it to be unable to decrypt normally. Then abort the simulation.

Simulator $S_{ObliHub}$: S simulates the same as the previous hybrid, except that it interacts with F for feedback. More specifically, we define the simulator $S_{ObliHub}$ as follows.

The simulator $S_{ObliHub}$ is in the puzzle phase:

• **P_R is the honest party, P_T is the corrupt party:**

① S sends $mes = (C(P_T, P_R), C(P_T, P_R).state)$ to P_T ($C(P_T, P_R).state$ is public).

② If P_T does not respond, then the simulation is aborted; otherwise, the S receives (Z_b, b, P_R, π) from the P_T.

③ S stores (Z_b, b, P_R, π) to list L and sends (Z_b, b, P_R, π) to P_S.

• **P_T is the honest party and P_R is the corrupt party:**

① After S receiving P_R's request message ($C(P_T, P_R)$, $C(P_T, P_R)$.state), perform the following operations:

② If $C(P_T, P_R)$.state $\neq$ open, then the simulation is aborted; otherwise, S randomly chooses a α_b to compute the puzzle Z_b, proves π, and (α_b, Z_b, b, π) is stored in the list L. Finally, S sends (Z_b, b, π) to the P_R.

The simulator $S_{ObliHub}$ is in the oblivious puzzle transfer phase:

• **P_S is the honest party, P_T is the corrupt party:**

① S randomly selects the session ID sid, and sends $mes = (sid, C(P_S, P_T).state)$ to P_T.

② If the P_T does not respond, abort the simulation. Otherwise, S receives a commitment $c = Com(P; r)$ from P_T.

③ S sends b to F_{CRS}^{mode} for storage in the previous stage then selects (sid, crs) and sends it to P_T.

④ S calculates $(pk, sk) \leftarrow KeyGen(crs, b)$, and then sends pk to P_T.

⑤ S receives all $y_i = Enc(pk, i, \alpha_i)$ settlement commitment $(P; r)$ from P_S.

⑥ If $V_{COM}(c, P, r) \neq 1$, then the aborting behavior of P_S is simulated. Otherwise, the final output $q = pk \oplus P$ and $\alpha_b \leftarrow Dec(sk, yb)$.

• **P_T is the honest party and P_S is the corrupted party:**

① S receives request message $mes = (sid, C(P_S, P_T).state)$ from P_S, if $C(P_S, P_T)$=Null or $C(P_S, P_T).state \neq$ open, then the aborting behavior of P_T is simulated. Otherwise, S randomly selects a P to make a commitment c to P_S.

② S finds α_b according to the list L. Other solutions are chosen randomly by the simulator. P_T gives $(sid, \alpha_0, \alpha_1, ..., \alpha_n)$ to F_{CRS}^{mode}. Then F_{CRS}^{mode} returns a random crs to send to P_T.

③ S receives pk from P_S, then S computes all $y_i = Enc(pk, i, \alpha_i)$ and sends it to P_S along with decom $(P; r)$.

④ Finally, the output $q = pk \oplus P$ can be computed.

Next, we proceed to prove the indistinguishability of the neighboring experiments for the environment ϵ.

Lemma 4. *For any environment ϵ, it holds that $EXEC_{h_0,A,\epsilon} \approx EXEC_{h_1,A,\epsilon}$.*

Proof. Since the difference between the two models is only the invocation of the functionality F_{COM}. The security of the commitment scheme COM determines the proof. □

Lemma 5. *For an arbitrary environment ϵ, it holds that $EXEC_{h_1,A,\epsilon} \approx EXEC_{h_2,A,\epsilon}$.*

Proof. The comparison reveals that the difference between the two models is only the invocation of the functionality F_{NIZK}. Thus their security arises from the security of the non-interactive zero-knowledge scheme. □

Lemma 6. *For any environment ϵ, it holds that $EXEC_{h_2,A,\epsilon} \approx EXEC_{h_3,A,\epsilon}$.*

Proof. These two models differ only in the invocation of the functionality F_{OPT}. So the security from the OPT protocol is proved. □

Lemma 7. *For an arbitrary environment ϵ, it holds that$EXEC_{h_3,A,\epsilon} \approx EXEC_{h_4,A,\epsilon}$.*

Proof. The difference between models h_3 and h_4 is the puzzle phase. We can see that when malicious rivals do evil, they will end the transfer before the transaction (unfinished puzzle phase). Thus, the view and output simulated by Simulator S in h_4 are irresistible when it is terminated in h_3. □

Lemma 8. *For an arbitrary environment ϵ, it considers that $EXEC_{h_4,A,\epsilon} \approx EXEC_{h_5,A,\epsilon}$.*

Proof. The difference between these two models lies in the oblivious puzzle transfer phase stage. We can exactly call the F_{OPT} function as well as the deposit contract to simulate it. Then its security and indistinguishability of view and output can depend on the OPT protocol. Thus, the result of the simulation in the oblivious puzzle transfer phase is the same as the one returned by adversary A. □

Lemma 9. *For an arbitrary environment ϵ, it considers $EXEC_{h_5,A,\epsilon} \approx EXEC_{F,A,\epsilon}$.*

Proof. Since the two simulations are identical, only the writing style differs. Therefore, the two are indistinguishable. □

This concludes the proof of Theorem 2. □

4.3. Discussion

As mentioned above, we discuss the UC security for OPT protocol and ObliHub. Subsequently, we discussed other security issues.

Atomicity. Atomicity is a very effective way of securing the balances of the participants. This property ensures that the transactions between the parties are completed or not and that the participants do not gain or lose additional tokens. The case where atomicity cannot be completed is when one of the two channels does not complete the transfer, i.e., only the P_T receives the money, or only the P_R receives the money. For the first case, it is the case that the P_R does not receive the solution, or the solution received by the P_R, cannot solve the puzzle. On the one hand, it is possible that the P_R did not receive the solution due to the delay, but the P_S is the payment initiator who can try to send it again. On the other hand, if the solution given by the P_T is not correct, then the oblivious puzzle transfer protocol phase will not be completed, so the first case will not occur. For the second case, the P_R can solve the puzzle itself and complete the transfer, which also implies that the P_R has to solve the discrete logarithm (DLOG) problem, but this has been proven difficult by the academic community.

Unlinkability. Unlinkability refers to the fact that the hub node does not have access to information about the payment direction of the originator and receiver of the transaction when both sides of the transaction are transacting through the payment channel hub node. According to our premise assumption, the number of transaction values transferred in PCH is the same, so P_T does not derive the relationship between transaction parties based on the transaction value size. Secondly, we use the idea of oblivious transfer to ensure that the private information (*e.g.,* identity, transaction direction) of each other is not learned between the participants during the interaction. In particular, we use multiple puzzles so that hub nodes cannot know the correspondence between the sender and the receiver of each transfer. This also further enhances the unlinkability of senders and receivers.

Defending against DoS attacks. In this paper, we consider having the P_R pledge a deposit through a smart contract and deduct the deposit once the P_R does not respond to the P_T's transfer request within a specified time. Making the P_R the primary bearer of the attack's consequences stifles this phenomenon's occurrence. In addition, to avoid P_T refusing to send the correct solution, this article considers that P_T calls the deposit contract before the second stage to suppress the occurrence of such malicious behaviors.

5. Performance analysis

The device environment for our simulation experiments is inter CORE i7 computer with Ubuntu 18.04. Our simulation experiments are modifications of the code for oblivious transfer and the code for commitment scheme, etc. [25]. We also simulated the reproduction of the A^2L code in the same environment. The premise of our simulation experiments is to simulate the same number of public key operations as A^2L, so the simulation experiments for ObliHub are done using the classical *1-out-of-2* OT, which can also be easily extended to the *1-out-of-n* transfer of puzzle solutions. The simulation results show that the total running time of the scheme in this paper is reduced by 0.2 seconds compared to the total running time of the previous scheme, reducing the communication cost between the parties' interactions and improving the transfer throughput of the PCH. We refer to transfer

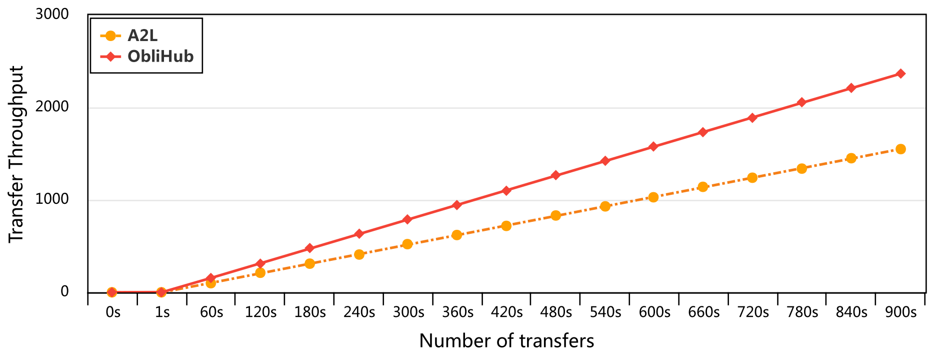

Fig. 6. Transfer throughput comparison.

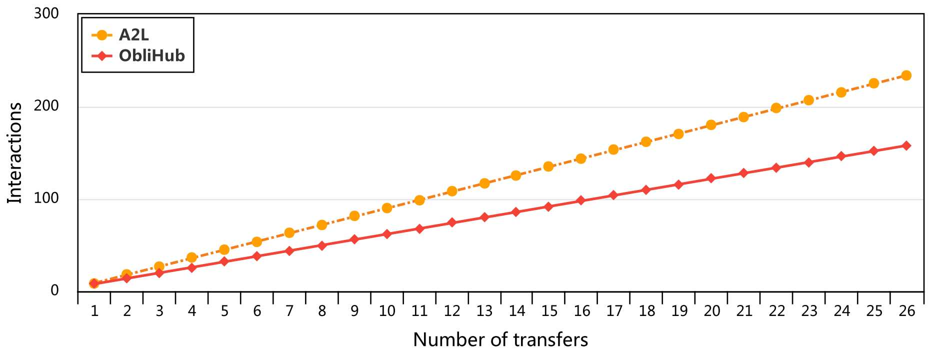

Fig. 7. Comparison of the number of interactions between parties.

throughput as the number of transfers that can be made per unit of time. Increasing transfer throughput means increasing the number of transfers that can be performed simultaneously, ultimately achieving the purpose of off-chain payment scaling. Fig. 6 shows the throughput statistics of the parties.

Number of Rounds. In our protocol, the transfer of sender, hub, and receiver can be done in two phases. First, in the puzzle phase, the receiver completes the agreement of the puzzle with the payment channel hub P_T, the sender then performs the oblivious puzzle transfer phase to obtain the solution and complete the transaction. The receiver receives the solution from the sender. Finally, the receiver receives the solution from the sender and solves the puzzle to complete the transaction with the P_T. As shown in Fig. 7, when the sender and receiver have m transactions to complete, only 6m + 2 interactions are needed, while 9m interactions are needed for the previous solution. Overall, the ObliHub uses several different puzzles to represent the identifier of the current round and requires only two rounds to complete the transfer payment.

Communication Overhead. We denote the total amount of information exchanged between the parties in the protocol as the communication overhead. Fig. 8 shows the total size of the messages passed between the parties in this paper. In this paper, we use the *1-out-of-2* OT as an example for our simulations, as long as we guarantee that the P_T can make an unlinkable atomic transfer by sending only two puzzles while guaranteeing the ideal communication overhead and five public key operations.

Computation Time. ObliHub does not require each user to compute digital signatures other than the two payment channels opening and closing. Only the linear computational complexity of the oblivious puzzle transfer needs to be completed. In Table 1 we compared it with the A^2L article. The number of operations column before the semicolon indicates the number of A^2L operations, and after the semicolon indicates the number of our operations, which can be clearly seen [26]. We both use the Commitment scheme and Non-interactive zero-knowledge. Still, instead of using complex cryptographic primitives such as the Adaptor signature scheme, we use OT and ECC to finish answering the puzzle.

6. Related work

Similar to the lightning network, the payment channel network establishes multiple two-way payment channels between multiple nodes and uses Hash Time Lock Contracts (HTLC) to complete multiple transactions [28]. However, the efficiency of HTLC has been criticized by the industry. To reduce the running time and communication burden, Zhang X et al. [33] proposed to optimize multi-broadcast traffic. And Malavolta G et al. [21] proposed a Multi-Hop HTLC contract, the core of Fulgor and Rayo so that the payment channel network can be deployed in practice. However, in the work of Poon J. et al. and Malavolta G et al. [21,28], all nodes use the same value when completing the contract, which may lead to the theft of the transaction costs of intermediate routing nodes, thus triggering Wormhole attack. To solve this problem, Malavolta G et al. proposed Anonymous Multi-Hop Locks (AMHLs) [29], which introduced an additional communication phase. During contract execution, different two-way payment channels use different

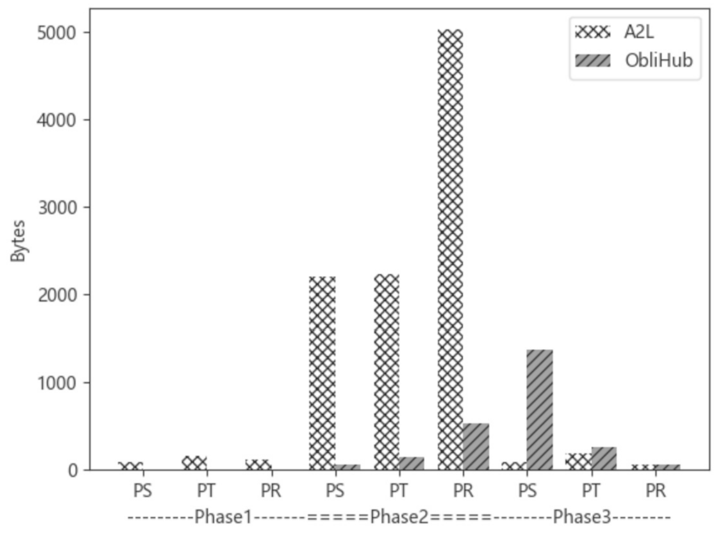

Fig. 8. Comparison Chart of the Implementation Results of Two Schemes.

Table 1
Technical comparison with A^2L.

Cryptographic primitives	A^2L	ObliHub	Numbers of operation
Commitment scheme	✓	✓	1; 2
Non-interactive zero-knowledge	✓	✓	2; 2
Homomorphic encryption scheme	✓	×	1; -
Digital signature scheme	✓	×	6; -
Adaptor signature scheme	✓	×	4; -
(Blinded) Randomizable signature scheme	✓	×	2; -
Oblivious Transfer	×	✓	-; 1
Ecc	×	✓	-; 2

Table 2
A comparison between off chain transfer services and hybrid services. (✓ indicates implementation, × indicates no implementation. - indicates no comparison.)

	Experimental blockchain	Functionality	Atomicity	Unlinkability	Number of Phase	Resist DoS attacks
Perun [27]	Ethereum	Generic	✓	×	-	×
Raiden [20]	Ethereum	PCN	✓	×	-	×
LN [28]	Bitcoin	PCN	×	×	-	×
Multi-HTLC [21]	Bitcoin	PCN	✓	×	-	×
AMHL [29]	Bitcoin	PCN	✓	×	-	×
BOLT [30]	ZCash	PCH	✓	✓	3	×
TumbleBit [22]	Bitcoin	PCH	✓	✓	3	×
A^2L [9]	Bitcoin	PCH	✓	✓	3	✓
Mixing [31]	Bitcoin or Ethereum	Generic	✓	✓	-	×
CoinJoin [32]	Bitcoin	Generic	✓	✓	-	×
ObliHub	Bitcoin	PCH	✓	✓	2	✓

values to prevent any two nodes in the path from conspiring to steal the node transaction fees between them, thus effectively resisting the Wormhole attack. However, AMHLs does not solve the problem of identity linkability related to payment. For example, when multiple nodes pay off the chain, the routing node knows the identity, amount, transaction direction, and other sensitive information of the parties to the transaction through it. Ferenc Beres et al. pointed out that because LN is a small world, even if both parties pay through onion coins, the identity information of both parties can be inferred, and the relationship between them can be linked through the payment direction [34]. Ferenc Beres et al. proposed a method to inject extra hops into the routing path, which can strengthen the privacy protection of both sides of the transaction with very little extra transaction cost.

A virtual Payment Channel (VPC) [27] is set up between sender and receiver such that most transactions are complete without the hub node. However, the hub node is still aware of the opening and closing of VPC. Consequently, the hub node infers the identity of sender and receiver according to the balance in the last transaction. Recently, the PCH has proposed an ideal solution to guarantee unlinkable between the sender and receiver. TumbleBit [22] is the first PCH scheme to achieve the effect of online transactions by solving puzzles off the chain. Similar solutions include BOLT [30] and A^2L [9]. Among them, A^2L transfers funds through non-trusted third-party nodes to achieve the unlinkability, reliability, and atomicity of transactions. Although the problem of unlinkability has been solved, A^2L uses a variety of complex cryptographic primitives, such as commitment, homomorphic encryption [35], zero knowledge, randomized puzzles, and adapter signature, resulting in its unsatisfactory complexity. In addition to PCH, token hybrid services can also be guaranteed unlinkable, such as Mixing [31], CoinJoin [32], and STSIIML [36]. The CoinJoin service can effectively avoid token theft and achieve anonymity of transactions. In STSIIML, hybrid services are further elaborated through the combination of machine learning and game theory. And Table 2 shows the functional comparison between some off-chain transfer services and hybrid services. We found that compared with PCN, PCH can better realize the unlinkability between the three nodes. The scheme in this paper is developed under the background of PCH and can realize the atomicity and unlinkability of payment.

7. Conclusion

Off-chain payment is a panacea to solve the problem of blockchain expansion. This paper introduces the offline payment scheme ObliHub, its design, and specific protocol content. It can solve the problem of secure transfer when the intermediary hub node is untrusted, learn about the relationship between two transfer parties, and solve the atomic payment problem. Secondly, we define the protocol process in our scheme according to the ideal functionality and prove that our scheme is secure using the universal composability framework. Finally, we provide a theoretical performance analysis of the scheme, which shows that the scheme can complete the transfer between three nodes in two rounds. At the same time, this scheme has good benefits for promoting the solution of blockchain scalability.

Declaration of competing interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled, “Enabling Scalable and Unlinkable Payment Channel Hubs with Oblivious Puzzle Transfer”.

Data availability

No data was used for the research described in the article.

Acknowledgement

This study is supported by the Foundation of National Natural Science Foundation of China (Grant No.: 62072273, 72111530206, 61962009); The Major Basic Research Project of Natural Science Foundation of Shandong Province of China (ZR2019ZD10); Natural Science Foundation of Shandong Province (ZR2019MF062); Shandong University of Science and Technology Program Project (J18A326); Guangxi Key Laboratory of Cryptography and Information Security (No.: GCIS202112); Foundation of Guizhou Provincial Key Laboratory of Public Big Data (No. 2019BD-KFJJ009); This work was supported by the Key-Area Research and Development Program of Guangdong Province (No. 2020B0101130015).

References

[1] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, P. Wuille, Enabling blockchain innovations with pegged sidechains 72 (2014) 201–224, http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains, 2014.
[2] J. Poon, V. Buterin, Plasma: scalable autonomous smart contracts, White Paper (2017) 1–47.
[3] S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system, Decentralized Business Review (2008) 21260.
[4] T. Li, Y. Chen, Y. Wang, Y. Wang, M. Zhao, H. Zhu, Y. Tian, X. Yu, Y. Yang, Rational protocols and attacks in blockchain system, Secur. Commun. Netw. 2020 (2020) 8839047, https://doi.org/10.1155/2020/8839047.
[5] T. Li, Z. Wang, G. Yang, Y. Cui, Y. Chen, X. Yu, Semi-selfish mining based on hidden Markov decision process, Int. J. Intell. Syst. 36 (7) (2021) 3596–3612, https://doi.org/10.1002/int.22428.
[6] T. Li, Z. Wang, Y. Chen, C. Li, Y. Jia, Y. Yang, Is semi-selfish mining available without being detected?, Int. J. Intell. Syst. 37 (12) (2022) 10576–10597, https://doi.org/10.1002/int.22656.
[7] J. Li, X. Hu, P. Xiong, W. Zhou, et al., The dynamic privacy-preserving mechanisms for online dynamic social networks, IEEE Trans. Knowl. Data Eng. (2020), https://doi.org/10.1109/TKDE.2020.3015835.
[8] F. Yuan, S. Chen, K. Liang, L. Xu, Research on the coordination mechanism of traditional Chinese medicine medical record data standardization and characteristic protection under big data environment, 1st Edition, vol. 1 of 1, Shandong People’s Publishing House, Shandong, 2021, No. 517 Shungong Road, Shizhong District, Jinan, Shandong Province, China.
[9] E. Tairi, P. Moreno-Sanchez, M. Maffei, A^{2}l: anonymous atomic locks for scalability in payment channel hubs, in: 42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24–27 May 2021, IEEE, 2021, pp. 1834–1851.
[10] R. Canetti, Universally composable security: a new paradigm for cryptographic protocols, in: Proceedings 42nd IEEE Symposium on Foundations of Computer Science, 2001, pp. 136–145.
[11] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, Y.-a. Tan, Secure multi-party computation: theory, practice and applications, Inf. Sci. 476 (2019) 357–372.
[12] V. Kolesnikov, R. Kumaresan, Improved ot extension for transferring short secrets, in: Annual Cryptology Conference, Springer, 2013, pp. 54–70.
[13] A. Biryukov, S. Tikhomirov, Deanonymization and linkability of cryptocurrency transactions based on network analysis, in: IEEE European Symposium on Security and Privacy, Euro S&P 2019, Stockholm, Sweden, June 17–19, IEEE, 2019, pp. 172–184.
[14] M. Möser, K. Soska, E. Heilman, K. Lee, H. Heffan, S. Srivastava, K. Hogan, J. Hennessey, A. Miller, A. Narayanan, N. Christin, An empirical analysis of traceability in the monero blockchain, Proc. Priv. Enh. Technol. 2018 (3) (2018) 143–163, https://doi.org/10.1515/popets-2018-0025.

[15] M. Blum, P. Feldman, S. Micali, Non-interactive zero-knowledge and its applications, in: O. Goldreich (Ed.), Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali, ACM, 2019, pp. 329–349.
[16] C. Peikert, V. Vaikuntanathan, B. Waters, A framework for efficient and composable oblivious transfer, in: D.A. Wagner (Ed.), Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17–21, 2008. Proceedings, in: Lecture Notes in Computer Science, vol. 5157, Springer, 2008, pp. 554–571.
[17] D. Hankerson, A.J. Menezes, S. Vanstone, Guide to Elliptic Curve Cryptography, Springer Science & Business Media, 2006.
[18] R. Canetti, Y. Dodis, R. Pass, S. Walfish, Universally composable security with global setup, in: S.P. Vadhan (Ed.), Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings, in: Lecture Notes in Computer Science, Springer, 2007, pp. 61–85.
[19] J. Camenisch, A. Lysyanskaya, A formal treatment of onion routing, in: V. Shoup (Ed.), Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings, in: Lecture Notes in Computer Science, vol. 3621, Springer, 2005, pp. 169–187.
[20] Raiden Network, https://raiden.network/.
[21] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, S. Ravi, Concurrency and privacy with payment-channel networks, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 455–471.
[22] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, S. Goldberg, 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, The Internet Society, 2017.
[23] S. Dziembowski, L. Eckey, S. Faust, J. Hesse, K. Hostáková, Multi-party virtual state channels, in: Y. Ishai, V. Rijmen (Eds.), Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I, in: Lecture Notes in Computer Science, Springer, 2019, pp. 625–656.
[24] T. Li, J. Li, X. Chen, Z. Liu, W. Lou, Y.T. Hou, Npmml: a framework for non-interactive privacy-preserving multi-party machine learning, IEEE Trans. Dependable Secure Comput. 18 (6) (2020) 2969–2982.
[25] https://github.com/archit-p/simplest-oblivious-transfer.
[26] J. Li, Y. Huang, Y. Wei, S. Lv, Z. Liu, C. Dong, W. Lou, Searchable symmetric encryption with forward search privacy, IEEE Trans. Dependable Secure Comput. 18 (1) (2019) 460–474.
[27] S. Dziembowski, L. Eckey, S. Faust, D. Malinowski, Perun: virtual payment hubs over cryptocurrencies, in: 2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019, IEEE, 2019, pp. 106–123.
[28] J. Poon, T. Dryja, The bitcoin lightning network: Scalable off-chain instant payments, 2016.
[29] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, M. Maffei, 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, The Internet Society, 2019.
[30] M. Green, I. Miers, Bolt: anonymous payment channels for decentralized currencies, in: B. Thuraisingham, D. Evans, T. Malkin, D. Xu (Eds.), Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017, ACM, 2017, pp. 473–489.
[31] S. Meiklejohn, R. Mercer, Möbius: trustless tumbling for transaction privacy, Proc. Priv. Enh. Technol. 2018 (2) (2018) 105–121, https://doi.org/10.1515/popets-2018-0015.
[32] R. Canetti, Universally composable security: a new paradigm for cryptographic protocols, in: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA, IEEE Computer Society, 2001, pp. 136–145.
[33] X. Zhang, Y. Wang, G. Geng, J. Yu, Delay-optimized multicast tree packing in software-defined networks, IEEE Trans. Serv. Comput., Early Access (2021) 1–14, https://doi.org/10.1109/TSC.2021.3106264.
[34] I.A. Seres, A.A. Benczúr, A cryptoeconomic traffic analysis of bitcoin's lightning network, https://doi.org/10.21428/58320208.d4cd697e, 2021.
[35] G. Castagnos, F. Laguillaumie, Linearly homomorphic encryption from {DDH}, in: Cryptographers' Track at the RSA Conference, Springer, 2015, pp. 487–505.
[36] Y. Wang, T. Li, M. Liu, C. Li, H. Wang, Stsiiml: study on token shuffling under incomplete information based on machine learning, Int. J. Intell. Syst. (2022), https://doi.org/10.1002/int.23033.