Anonymous scheme for blockchain atomic swap based on zero-knowledge proof

Ling Cao[a], Zheyi Wan[b]
School of Software Engineering, Chongqing University of Posts and Telecommunications Chongqing 400065, China
[a]caoling@cqupt.edu.cn, [b]704681371@qq.com

***Abstract*—The blockchain's cross-chain atomic exchange uses smart contracts to replace trusted third parties, but atomic exchange cannot guarantee the anonymity of transactions, and it will inevitably increase the risk of privacy leakage. Therefore, this paper proposes an atom based on zero-knowledge proof. Improved methods of exchange to ensure the privacy of both parties in a transaction. The anonymous improvement scheme in this article uses the UTXO unconsumed model to add a new anonymous list in the blockchain. When sending assets to smart contracts, zero-knowledge proof is used to provide self-certification of ownership of the asset, and then the transaction is broken down. Only the hash value of the transaction is sent to the node, and the discarded list is used to verify the validity of the transaction, which achieves the effect of storing assets anonymously in the smart contract. At the same time, a smart contract is added when the two parties in the transaction communicate to exchange the contract address of the newly set smart contract between the two parties in the transaction. This can prevent the smart contract address information from being stolen when the two parties in the transaction communicate directly.**

Keywords—atomic exchange, zero-knowledge proof, smart contract

I. INTRODUCTION

Assuming a real trading scenario, Alice holds 1 BTC and Bob holds 100 ETH. Alice wants to use her 1 BTC to exchange the 100 ETH in Bob's account. Bob also agrees to this asset transaction. Previously they usually chose a large exchange as a third party for trading, and put your own money on the exchange to exchange assets. This method is possible, but there are corresponding risks. Two people now want to trade directly but are also afraid that the other party will terminate the transaction after receiving their own money, so whoever transfers the money first is bound to face greater risks. Finally, they decided to use atomic exchange to exchange. Alice first sent her 1BTC to a smart contract. The BTC is saved. When the smart contract receives B's public key and a random number x, the smart contract will send Alice's BTC to Bob. At the same time, there is a time limit on the smart contract. If you do not receive the unlocking information within this time, BTC will be returned to Alice. This time limit is called hash time lock.

After Alice creates the smart contract, she sends the contract and her public key to Bob. After Bob gets Alice's public key and sees that Alice has placed BTC in the smart contract, he repeats Alice's steps. Established a smart contract, which wrote that when A's public key and random number x were received, the ETH in the contract was transferred to Alice's account. This random number x is the same as the random number set by Alice, and set a hash time lock at the same time. This setting is 12 hours. When the time is up, ETH will be returned to Bob's account without unlocking. After Bob has set up the smart contract, he sends the set smart contract and his public key to Alice for viewing. At this time, both parties are ready to start the transfer.

Alice uses the random number generated by her and Bob's public key to unlock the smart contract set by Bob, and successfully receives the ETH that Bob placed in the contract. At this time, Bob can see the value of the random number X that unlocks the smart contract, and Bob can take This random number and the Alice's public key received previously to unlock the smart contract set by Alice, because the smart contract hash time lock set by A is 12 hours longer than Bob's contract, so Boy can go within these 12 hours. Unlock Alice's smart contract without worrying about Alice's withdrawal. So far, the two parties in the transaction have completed the atomic exchange.

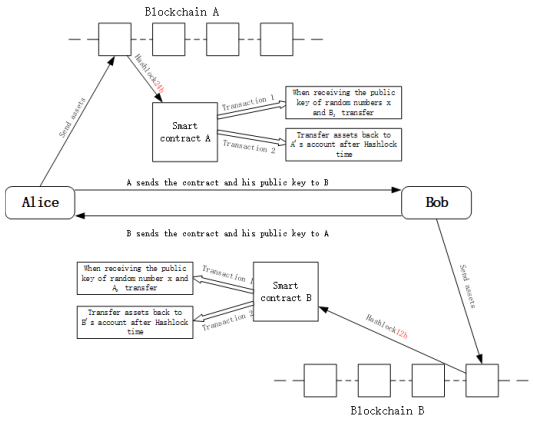

Figure 1 Atomic exchange process diagram

II. FUNDAMENTAL

A. Zero-knowledge proof

Let F_q be a finite field, the elliptic curve be E, Q be a point on the elliptic curve E, and $Q = nP$, and n be a secret. The public point P, Q and elliptic curve E, the proof and verification method of zero-knowledge proof are as follows:

Let P and V be the prover and verifier, respectively. P wants to prove to V that he knows the secret n, but doesn't want to expose it. They can do the following steps:

Step1: Randomly select an integer $r, r < q$. Calculation $P_1 = rP, P_2 = (n - r)P$,and send P_1, P_2 to validator V.

Step2: V randomly requests p to send r_i, i=1,2 , $r_1 = r, r_2 = n - r$.

Step3: After receiving r_i, V verifies whether $P_i = r_i P$ and $P_1 + P_2 = Q$ are true.

Repeat the above three steps m times until V believes P knows the secret n.

It can be proven that the probability that P can successfully deceive V each time is $1/2$, because if P does not know the secret n and P wants to successfully deceive V, P can perform the following process:

P selects an integer r, remembers $P_1 = rP, P_2 = Q - P_1$, and sends it to V as described above. Suppose V requires P to send $r_1 = r$, which happens to provide r. At this time, P successfully cheats V. Suppose that V requires the number r_2 corresponding to P_2, because P does not know the secret n, and the corresponding number r_2 obtained by P_2 is a discrete logarithm problem based on an elliptic curve, so P cannot get r_2 anyway. Therefore, the probability that P can successfully deceive V in each round is $1/2$, and after m rounds, the probability that P can successfully deceive V is $1/2^m$. So after a sufficiently large number of times, if P can answer correctly every time, V believes that P knows the secret n.

Through the above process, P proved to V that he knew the secret n without revealing any information about n to V.

B. Discrete logarithmic hypothesis

The discrete logarithm assumption applies to $\mathbb{G} =< g >$ and prime P, if all non-uniform polynomial time opponents $\mathcal{A}$ are:

$$Pr\left[g^a = h | h \xleftarrow{\$} \mathbb{z}_p; a \leftarrow \mathcal{A}(\mathbb{G}, g, h)\right] \approx 0 \quad (1)$$

C. Pedersen commitment plan

The Pedersen commitment scheme is a computationally bound commitment scheme. The commitment value of the scheme is uniformly distributed and does not depend on any difficulty assumptions, but its computational binding depends on the difficulty of finding discrete logarithms, that is, discrete logarithms. Assume that the commitment scheme can be divided into three phases.

1) Initialization: Suppose g, h are 2 generators of group $\mathbb{G}$ (order q). $log_g h$ is unknown, that is, the calculation of discrete logarithms is not feasible.

2) Commitment stage: The promiser wants to promise an integer $m \in Z_q$, first select a random number $r \in Z_q$, then calculate the commitment value $c = C(m, r) = g^m h^r$, and finally send the promiser c to the receiver By.

3) Opening phase: The promiser can choose an appropriate time to open the promise. When opening, the promiser sends the promised message m and the selected random number r to the receiver. The receiver calculates $c' = g^m h'$ and compares it with the promise value c he received before to see if they are equal. If they are equal, the commitment is opened correctly; otherwise, the commitment is rejected.

D. Elliptic curve cryptography

The addition operation on the elliptic curve E is defined as follows:

Suppose P, Q are two points on the curve, and $P = (x_1, y_1), Q = (x_2, y_2)$

1) $(x_1 = x_2)$ and $y_2 = -y_1$,then $P + Q = O$

2) $x_1 \neq x_2$, then $P + Q = R = (x_3, y_3)$, R is the intersection point of the PQ straight line and the elliptic curve at the X-axis symmetry point.

$$x_3 = \lambda^2 - x_1 - x_2 \quad (2)$$

$$y_3 = \lambda(x_1 - x_2) - y_1 \quad (3)$$

$$\lambda = \begin{cases} (y_2 - y_1)(x_2 - x_1)^{-1}, P \neq Q \\ (3x_1^2 + a)(2y_1)^{-1}, P = Q \end{cases} \quad (4)$$

For all $P \in E$, define $P + O = O + P = P$.

III. ANONYMOUS SCHEME DESIGN

A. Designing an anonymous hash list

Traditional atomic swap technology uses smart contracts on two chains to set up transactions, but the information on the smart contract is open and transparent. Enter the address of the smart contract in the browser, and you can clearly see all the code information in the smart contract. Therefore, the traditional atomic exchange is information secure but does not have anonymity. You can lock both sides of the transaction by looking up the basic information of the smart contract. Therefore, an anonymous atomic swap scheme is set up to protect the privacy of both parties in the transaction. In the blockchain, the hash value of each transaction is stored in the form of a Merkel Tree. The immutability of the blockchain is also because these hash values cannot be changed. We set the storage of the hash value to two lists, one called a full hash list and one called an nullifier list. The full hash list contains the hash values of all transactions in the chain, while the discard list contains the hash values of transactions that have already been performed. We represent the original input-output structure of the transaction as a Note structure, and H represents the hash value of the Note structure, $HASH\ (Note)$. The existence of all transactions in the blockchain is shown in the following table. The left side of the table is the hash value of all transactions, and the right side of the table is the transaction that has been spent.

TABLE I. BLOCKCHAIN HASH LIST

Full hash list	Nullifier hash list
$H_1 = HASH(Note_1)$	$N_1 = HASH(Note_2)$
$H_2 = HASH(Note_2)$	
$H_3 = HASH(Note_3)$	

The specific scheme is as follows:

1) Alice finds one or more unspent notes, that is, UTXO output, and each note has a corresponding value. Use zero-knowledge proof to prove that you own the asset of $Note_1$:

Initial stage: use libsnark to implement zkSNARKs. It requires a trusted setup phase to generate a common reference string, or a pair of ek and vk. When performing trusted settings, select some random numbers to calculate ek and vk, but these random numbers cannot be known by the prover and verifier. Otherwise, any party who knows these numbers will break the security of the protocol. The trusted setting should be done by a trusted third party. zkSNARK needs to specify a circuit in advance. This circuit is limited to

computing a fixed number of hashes. This setup generates a pair of keys(ek, vk), where ek is used to generate the S proof, and vk is used to verify the R proof. The number of hashes that can be calculated during the period. We use the symbol $m_{[1,q][1,n][0,s]}$ to represent the asset of $Note_1$, and the initialization program randomly generates $s+1$ group elements:

$$U_{[0,s]} = (U_o, U_1, U_2, \dots, U_s), U_j \xleftarrow{\$} \mathbb{G}, j \in [0,s] \quad (5)$$

The initializer calculates (n, q) validators:

$$\sigma_{li=Com(m_{li1},\dots,m_{lis};m_{li0})=u_0^{m_{li0}}\cdot\prod_{j=1}^{s}u_j^{m_{lij}},l\in[1,q],i\in[1,n]} \quad (6)$$

The matrix of $\sigma_{[1,q][1,n]}$ is a common input and is shared between the sender and the receiver. The size of the matrix is also $1/s$ of the data file. Another job of the initialization program is to set zkSNARK with the circuit C, which calculates the hash value:

$$(ek, vk) = zkSetup(C, 1^{\lambda}) \quad (7)$$

ek is the evaluation key of the sender, and vk is the authentication key of the receiver.

R first uses the key bound to the key promise to verify that the authenticated encrypted data is correct:

$$\prod_{i=1}^{n}\sigma_{li}^{(c^i)}\cdot k_{li} \stackrel{?}{=} \prod_{i=1}^{n}\prod_{j=0}^{s}u_j^{\overline{m}_{lij}}; l\epsilon[1,q] \quad (8)$$

Then the knowledge of the R verification key proves the key promise of $w\cdot r\cdot t$:

$$\prod_{i=0}^{n}{k_{li}}^{c^i} \stackrel{?}{=} \prod_{j=0}^{s}u_j^{z_{lj}}; l\epsilon[1,q] \quad (9)$$

Finally, R verification proves that each column of keywords is a proof derived from the image of the hash h_ω:

$$zkVerify_{vk}\left(\left(l, j, c, z_{lj}, h_\omega\right), 1, \pi_{lj}\right); l\epsilon[1,q];$$
$$j \in [0,s] \quad (10)$$

2) After Alice proves her ownership of the transaction $Note_1$, she decrypts $Note_1$ with her private key sk, obtains the data in $Note_1$, and then creates two new $Note_2$ and $Note_3$. $Note_2$ is set to send to the smart contract, and $Note_3$ is set to send the balance in $Note_1$ to your account address. If $Note_1$ is exactly the same as the asset that needs to be exchanged, you do not need to set $Note_3$.

3) Alice sends the hash value H_1 of $Note_1$ to the node network of the blockchain. After receiving H_1, the node will determine whether the hash value exists in the discard list. If it exists, then $Note_1$ is double spent . Otherwise, the hash value is recorded in the discard list. At the same time, Alice also sends the hash values of $Note_2$ and $Note_3$ to the node, so that the node can only know H_1 of $Note_1$and H_2, H_3 of $Note_2, Note_3$. The transaction sent by Alice to the smart contract is hidden, and the address of the smart contract cannot be found by looking up the transaction index value of $Note_1$, thereby achieving anonymity in the process of storing assets to the smart contract. At this time, the hash table becomes as follows form:

TABLE II. UPDATE BLOCKCHAIN HASH LIST

Full hash list	Nullifier hash list
$H_1 = HASH(Note_1)$	$N_1 = HASH(Note_2)$
$H_2 = HASH(Note_2)$	$N_2 = HASH(Note_1)$
$H_3 = HASH(Note_3)$	
$H_4 = HASH(Note_4)$	
$H_5 = HASH(Note_5)$	

B. Newly added smart contract as a contract address interaction method

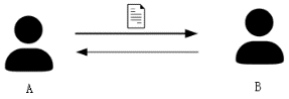

Figure 2 Original exchange contract address

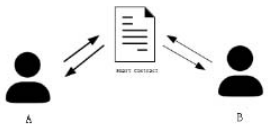

Figure 3 Improved protocol exchange

The improved protocol includes three parties: the transaction initiator (Alice), the transaction receiver (Bob), and the smart contract (Julia). Alice's private key sk, Bob owns the public key pk corresponding to Alice's private key sk. Bob needs to verify that the private key owned by Alice is the correct private key corresponding to the public key, and that Alice and Bob can exchange the contract address of the assets stored by both parties through the smart contract.

Alice first generates a random number, $r < rand()$, and then Alice uses the elliptic curve to calculate the mapping of R on the elliptic curve R $(R = r * G)$. At the same time, she also maps the contract address k where her asset is stored to $K = k * G$, send R and K to Bob, after receiving K and R, Bob generates a random number c and sends it to Alice, Alice encrypts the address k, generates a random number p, and let $E(k) = p + k$, Calculate $Z = r + c * sk$. Alice sends $E(k)$ and Z to Bob, while sending p to the smart contract Julia. Bob verifies whether $z * G = R + c * pk$, if they are equal, it proves that the private key sk owned by Alice is the private key corresponding to the public key pk. After the verification is successful, Bob sends his contract address to Julia, and at the same time Obtain the p on the contract. With the random number p, Bob can obtain the k value by solving the encrypted address information. Alice and Bob can complete the exchange of the address information through the smart contract.

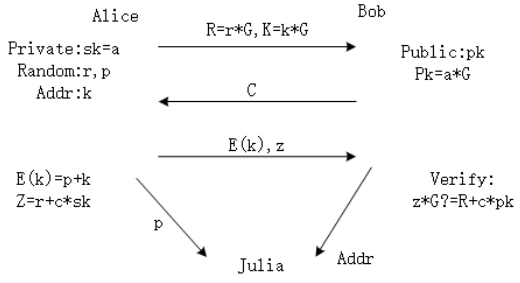

Figure 4 Improved protocol process

IV. SCHEME PERFORMANCE ANALYSIS

A. Security analysis

The zero-knowledge proof used in this paper is designed and implemented based on the discrete logarithm problem of the elliptic curve. After m times of protocol interaction, the probability that the transaction initiator can trick the verification node in the blockchain is $1/2^m$. With the increase of records, the length of the block is increasing, and the number of verification nodes is also increasing. The value of the transaction is stored in the blockchain in a fixed and tamperable manner, so the output and input of each transaction have passed the node's confirmation can be initiated. Each transaction can be initiated after the node confirms that it is true and valid. The scheme adopts the scheme of anonymously storing assets in smart contracts. According to transaction records, it cannot be obtained which smart contract the transaction initiator has stored assets in. At the same time, during the communication between the two parties in the transaction, a new smart contract was set up as an interaction method for the transfer address, which avoided the contract address information being captured by others during the communication between the two parties in the transaction, so the anonymous solution was secure.

B. Reliability analysis

In the process of transactions, transactions conform to the basic characteristics of atomic swaps, namely atomicity. A hash time lock is set on each smart contract, and the transaction is automatically canceled when the transaction times out. The smart contract that exchanges addresses will only send the P value set by the prover to the verifier when it receives the contract address information sent by the verifier, and the transaction will be cancelled if it expires. At the same time, the hash time lock set by the smart contract set by the transaction initiator and receiver for asset storage has a time difference, preventing one party from directly canceling the transaction after receiving the asset. Therefore this anonymous solution is reliable.

V. CONCLUSION

This article proposes an anonymous mechanism for the atomic exchange scheme in blockchain cross-chain transactions. In the traditional atomic exchange scheme smart contract, all the information of both parties of the transaction can be found. The two parties of the transaction do not have anonymity. Aiming at the problem of openness and transparency of smart contract addresses, this article sets up a new scheme to ensure that transaction initiators store assets anonymously in smart contracts, and exchange smart contract address anonymity between trading parties. This scheme has certain advantages in protecting the privacy of both parties in a transaction.

REFERENCES

[1] Bensasson E , Chiesa A , Tromer E , et al. Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture[J]. Sasson, 2014.

[2] Watanabe H , Fujimura S , Nakadaira A , et al. [IEEE 2016 IEEE International Conference on Consumer Electronics (ICCE) - Las Vegas, NV, USA (2016.1.7-2016.1.11)] 2016 IEEE International Conference on Consumer Electronics (ICCE) - Blockchain contract: Securing a blockchain applied to smart contracts[J]. 2016:467-468.

[3] Kshetri N. Blockchain's roles in strengthening cybersecurity and protecting privacy[J]. Telecommunications Policy, 2017, 41(10): 1027-1038.

[4] Zhu J, Yonggui F U, Information S O. Progress in blockchain application research[J]. Science & Technology Review, 2017.

[5] Christidis K, Devetsikiotis M. Blockchains and smart contracts for the internet of things[J]. Ieee Access, 2016, 4: 2292-2303.

[6] Campanelli M , Gennaro R , Goldfeder S , et al. Zero-Knowledge Contingent Payments Revisited: Attacks and Payments for Services[C]// the 2017 ACM SIGSAC Conference. ACM, 2017.

[7] Mallikarjun Reddy Dorsala, V N Sastry, Sudhakar chapram. Fair Protocols for Verifiable Computations Using Bitcoin and Ethereum[C]// 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). IEEE, 2018.

[8] M. Saravanan and A. Priya (2019). An Algorithm for Security Enhancement in Image Transmission Using Steganography. Journal of the Institute of Electronics and Computer, 1, 1-8.

[9] A. Lumini, L. Nanni, A. Codogno and F. Berno (2019). Learning morphological operators for skin detection. Journal of Artificial Intelligence and Systems, 1, 60–76.