BPCEX: Towards Blockchain-based Privacy-preserving Currency Exchange

Wulu Li*, Lei Chen*, Xin Lai*, Xiao Zhang*, and Jiajun Xin*
*Onething Technologies Co., Ltd.,
Shenzhen, China,
liwulu@onething.net

***Abstract*—Privacy-preserving currency exchange between different cryptocurrencies on blockchain remains an open problem as the existing currency exchange schemes cannot provide anonymity of users or confidentiality of exchange amount. To solve this problem, we introduce BPCEX: a privacy-preserving currency exchange scheme which protects users' identities and the exchange amount, by usage of techniques including linkable ring signature, range proof, Diffie-Hellman key exchange, Pedersen commitment and UTXO swap. In BPCEX, the users' identities are hidden to verifiers and dealmakers, while the exchange amounts are hidden to the verifiers. BPCEX supports floating exchange rate, partial deal and public verification, without additional confirmation of traders, which improves the success rate and shortens the waiting time of the deal. Moreover, BPCEX is compatible with the regulatable privacy-preserving blockchain system, which realizes the traceability of users' identities and the exchange amount to prevent money laundering and illegal exchange, making BPCEX suitable in real-life applications, including currency market and stock market.**

***Index Terms*—blockchain, currency exchange, privacy-preserving, UTXO swap, partial deal, floating exchange rate, traceability**

I. Introduction

Since the introduction of Bitcoin [1] by Nakamoto in 2008, blockchain-based cryptocurrency has been developed rapidly with more applications and more advanced techniques. Thousands of altcoins are published to the blockchain field, among them, Ethereum [2], Monero [3] and Zerocash [4] are famous representatives that have been attracting increasing attentions for their technological breakthroughs. For Ethereum, smart contract is introduced to support Turing-complete languages and automatic transactions, and is widely used in decentralized applications including auction, lottery and business contract. For Monero, it uses linkable ring signature [3] to hide the identity of initiator, uses Diffie-Hellman key exchange to hide the identity of recipient, and uses range proof [5], [6] to hide the transaction amount. For Zerocash, it also realizes fully anonymous and confidential transactions by usage of the zero-knowledge succinct non-interactive argument of knowledge (zk-SNARKs). It should be emphasized that both Monero and Zerocash are built on UTXO model and not suited for smart contract.

Currency exchange (token exchange) between different cryptocurrencies is an important issue in real-life applications such as international trade, cross-border payment and stock trading. For stock trading, it can be regarded as the exchange between currency and stock (token), which is technically similar to the currency exchange in blockchain. Traditional currency exchange schemes, including [7]–[9], mainly use smart contract to realize this functionality. However, these schemes are built on account model, and have no privacy protection mechanisms to hide users' identities and the exchange amount, which brings about leakage of privacy. When someone wants to reach a deal for currency exchange with other users, without revealing the identity and amount, the smart contract based schemes no longer work. In 2016, Kosba *et al.* introduced Hawk [10]: a privacy-preserving smart contract system which has potential to realize the privacy-preserving currency exchange. However, Hawk uses zk-SNARKs as building blocks, which requires CRS (*common reference string*) of Gb size and is based on non-falsifiable assumptions, making Hawk inefficient for computation and storage (especially for mobile devices). Moreove, in Hawk, there exists a trusted manager who is able to recover the identities of users, as well as the exchange amount, which weakens its privacy protection as user's privacy will be leaked by malicious manager. So it is necessary to construct new currency exchange scheme to realize privacy protection with high efficiency, and support application functions including partial deal and floating exchange rate.

A. Our Contributions

1) Overview: In this paper, we introduce BPCEX (**B**lockchain-based **P**rivacy-preserving **C**urrency **EX**change), which is an exchange scheme between Monero-based cryptocurrencies with anonymity and confidentiality. In BPCEX, assume there are two types of Monero-based privacy-preserving cryptocurrencies ($\mathcal{C}_1$, $\mathcal{C}_2$) in the blockchain (with symbols £ and $), in which every UTXO has its own public-private keys, as well as the amount commitment and range proof. Ordinary transactions for each $\mathcal{C}_i$ are same as Monero. For currency exchange, there is a role of dealmaker in the blockchain, who is able to match the exchange quotations from users, and earn the reward. The BPCEX scheme mainly consists of three stages: Quote, Match and Deal. We give a brief introduction of each stage in the following:

Quote: For Alice who wants to exchange her coins in $\mathcal{C}_1$ to coins in $\mathcal{C}_2$ with floating exchange rate range, Alice generates a new UTXO and send a special transaction to the dealmaker,

with additional information including the exchange amount, exchange rate range and the reward amount. Notice that the secret key of new UTXO for exchange is possessed only by Alice, the dealmaker can recover the amount of the new UTXO, while cannot spend it as his own money, meanwhile, the dealmaker does not know the identity of Alice due to the privacy protection mechanisms of Monero.

Match: When a sufficient number of valid quotations have been received by the dealmaker, he recovers the corresponding exchange and reward amounts, then matches the matching quotations (between Alice and Bob), computes the final exchange rate and the blinding elements, then sends the matching result to the blockchain.

Deal: When verifiers receive the matching result from the dealmaker, they verify the validity of matching operation without any knowledge of the exchange amount and identities, except for the final exchange rate, then executes UTXO swap as the finish of the exchange, then Alice and Bob can receive the new money from the exchange deal respectively.

In BPCEX, the final exchange rate is publicly known, the verifier can only check the validity of quotations and matching outputs, cannot recover the exchange amount and users' identities, the dealmaker can recover the exchange amount to execute matching operation, but cannot recover users' identities as well, this is an effective approach to realize privacy-preserving currency exchange. Moreover, the efficiency of BPCEX exchange is similar to the ordinary Monero transactions, with a little more computations by the dealmaker to match the quotations, the users only need to wait for 2 blocks to receive the new money.

Note that there are several similarities between the dealmaker (this paper) and the manager (in [10]), as both of them can recover some privacy information from the quotation transaction (*or* private contract) and execute the matching task (*or* execute the code in private contract). Nevertheless, there are still some major differences between BPCEX and Hawk, a brief comparison is given in the following:

1. Hawk is based on Ethereum and Zerocash, while BPCEX is built on Monero system.
2. In Hawk, users' identities are known to the manager and will be leaked if the manager is malicious, while in BPCEX, the dealmaker does not know the users' identities, which realizes a higher level of anonymity.
3. In Hawk, after executing the codes in private contract, the manager need to generate a zk-SNARKs proof to prove the private contract is correctly executed, and this stage takes over 1 minute to compute with Gb-sized CRS, which means both users (Alice and Bob) have to wait for over 1 minute to receive the swapped coins. In BPCEX, after matching users' quotations, the dealmaker only need to publish the matching result including the final exchange rate and the blinding elements directly to the verifiers, without additional zero-knowledge proofs, this stage takes very little time, which helps to reduce the waiting time to receive their swapped coins for both users.

2) Floating Exchange Rate: BPCEX supports floating exchange rate during the Quote and Match stages, for a user Alice, she publishes the exchange rate range $\alpha : (\beta \pm \gamma)$ to the blockchain in the stage Quote, for example, it can range from $1 : 1.95$ to $1 : 2.05$ as Alice's accepted exchange range. The dealmaker extracts the exchange rate range from all quotations in the Match stage, compares and computes the corresponding amounts (including exchange amount, reward amount, accepted exchange range), then decides which two quotations to be matched and publishes the final exchange rate.

3) Partial Deal: Generally, in real-life currency exchange, the amounts of money between the two parties are not always matching exactly. For example, Alice has £1000 and wants to exchange it for \$1950→\$2050, with rate range $1 : 1.95 \rightarrow 1 : 2.05$, while Bob has \$200 and wants to exchange it for £99→£101 with range $0.99 : 2 \rightarrow 1.01 : 2$, their exchange rates is matching, but not for the amount, so the function of partial deal is a major requirement in construction of currency exchange schemes. In BPCEX, we introduce additional commitments and partition threshold to realize partial deal, which makes BPCEX also suitable for privacy-preserving stock exchange.

4) Towards Regulation: The full privacy protection in Monero-based cryptocurrency may cause abuse of privacy, and can commit crimes such as illegal exchanges and money laundering, which will become a negative factor for the application of BPCEX. In this paper, we construct a traceable BPCEX scheme by usage of TLRS [11] (traceable and linkable ring signature) and TRP [11] (traceable range proof) as the replacements to the original schemes (linkable ring signature, range proof) in Monero, to ensure all the exchanges are traceable by the regulator, while maintaining private to the dealmakers and verifiers.

B. Related Works

1) Privacy-preserving Techniques: Monero [3] is introduced in 2013 by Saberhagen who gave the initial construction of anonymous cryptocurrency by usage of linkable ring signature [12]. In 2016, Noether *et al.* [5] introduced Ring-CT protocol to realize the confidential transaction to hide the amount of transaction, by usage of Borromean range proof which is built directly from Borromean ring signature [13]. In 2018, Bünz *et al.* introduced Bulletproofs [6], an efficient non-interactive zero-knowledge proof protocol with short proofs and without a trusted setup, the proof size is only logarithmic to the witness size. In 2019, Yuen *et al.* [14] proposed Ring-CT 3.0, a new Bulletproofs-based confidential transaction system with smaller size and better efficiency.

In 2014, Ben-Sasson *et al.* [4] proposed Zerocash and zk-SNARKs, which is another type of privacy-preserving cryptocurrency with anonymity and confidentiality. Several follow-up works has been proposed, including libSNARK [15], zkSTARK [16], Aurora [17], Hyrax [18], Libra [19] and Supersonic [20].

2) Privacy-preserving Smart Contract: In 2016, Kosba *et al.* proposed Hawk [10], the first privacy-preserving smart contract system by usage of zk-SNARKs, has potential in applications such as sealed auction, crowdfunding and rock paper scissors, it also can be used in currency exchange (similar to sealed auction).

C. Organization

In section II we give some preliminaries; in section III we introduce UTXO swap and give the construction of BPCEX; in section IV we introduce the traceable BPCEX to realize the regulatory function; in section V we give the conclusion.

II. Preliminaries

In this paper, we use multiplicative cyclic group $\mathbb{G}$ to represent elliptic group with prime order $|\mathbb{G}| = q$, g is the generator of $\mathbb{G}$, group multiplication is $g_1 \cdot g_2$ and exponentiation is g^a. We use $H(\cdot)$ to represent hash function and $negl$ to represent negligible functions. For verifiers, 1 is for *accept* and 0 is for *reject*. For adversaries, PPT means probabilistic polynomial time. The DDH assumption means any PPT adversary cannot distinguish (g^a, h^a) from (g^a, r), where r is uniformly sampled from $\mathbb{G}$. The hardness of discrete logarithm problem means that any PPT adversary cannot compute x from g^x.

A. Linkable Ring Signature

Ring signature is a special type of signature scheme, in which signer can sign on behalf of a group chosen by himself, while maintaining anonymous within the group, ring signature was first proposed by Rivest, Shamir and Tauman [21] in 2001, several follow-up works including [22]–[24] have improvements in both efficiency and security.

Linkable ring signature is a variant of ring signature, which has the function of linkability, that is, when two ring signatures are signed by the same signer, they are linked by the algorithm Link, linkable ring signature is used in Monero to prevent double spending. The first linkable ring signature is proposed in 2004 [25], several improvements and modifications have proposed, including [12], [14], [26]. Monero uses MSLAG [5] as building blocks.

Monero has no regulatory functions as the anonymity of linkable ring signature is strong and the signer's identity is untraceable. To solve this problem, Li *et al.* proposed a traceable and linkable ring signature (TLRS) [11] to provide the traceability of signer's identity. We will use TLRS to construct the traceable BPCEX.

B. Pedersen Commitment

Pedersen commitment [27] was proposed in 1991, for elliptic curve $(\mathbb{G}, q = |\mathbb{G}|, g, h)$, where g is a generator of $\mathbb{G}$, h is a random element with discrete logarithm unknown to anyone.

Definition 1 (Pedersen commitment): The Pedersen commitment for a is $c = g^x h^a$, where $x \in \mathbb{Z}_q^*$ is a blinding element. Under the hardness of discrete logarithm, Pedersen commitment has the following properties:

1. (Hiding) Any (computational unbounded) adversary $\mathcal{A}$ cannot distinguish $c = g^x h^a$ from $c' = g^{x'} h^{a'}$.
2. (Binding) Any PPT adversary $\mathcal{A}$ cannot generate another secret a' binding with $c = g^x h^a = g^{x'} h^{a'}$.
3. (Homomorphic) Given $c_1 = g^x h^a, c_2 = g^y h^b$, then $c_1 \cdot c_2 = g^{x+y} h^{a+b}$ is a new commitment for $a + b$.

C. Range Proof

Range proof is a zero-knowledge proof to prove a committed hidden value lies within a certain range without revealing the value. The Pedersen-commitment-based range proofs are used in Monero system. In 2016, Neother *et al.* [5] gave the Borromean range proof, building from the Borromean ring signature [13], with linear proof size to the binary length of range. In 2018, Bünz *et al.* [6] introduced Bulletproofs, an efficient non-interactive zero-knowledge proof protocol with short proofs and without a trusted setup, the proof size is only logarithmic to the witness size.

For value $a \in [0, 2^n - 1]$ and the corresponding commitment $c = g^x h^a$, prover computes the binary expansion $a = a_0 + \cdots + 2^{n-1} a_{n-1}, a_i = 0, 1$ for $i = 0, \cdots, n-1$, and prove $a_i = 0, 1$ for every $i = 0, \cdots, n-1$, without leakage of the value. Both Borromean and Bulletproofs achieves completeness, soundness and zero-knowledge property, please refer to [5], [6] for detailed description.

In 2019, Li *et al.* proposed the first traceable range proof (TBP) [11] to realize the traceability of the hidden amount by usage of trapdoors, tracing keys and zero-knowledge proofs. The hidden amount in TRPs remains private to verifiers, but can be traced by the regulators with possession of the trapdoors. We will use traceable range proof to construct the traceable BPCEX.

D. Monero

In Monero system, every UTXO U_α has its public-private keys (PK_α, SK_α), the SK_α is known only by the owner of U_α, and the amount of U_α is hidden by the Pedersen commitment. When spending U_α, the owner (initiator) chooses the receiver, generates the output of transaction, including the output amount, new UTXO and its public key, then the initiator chooses another n irrelevant UTXOs (denoted by $U_1, \cdots, U_n$), together with U_α, as the inputs of the transaction, uses linkable ring signature to sign the transaction, with public key set $L_{PK} = \{PK_\alpha, PK_1, \cdots, PK_n\}$ (with randomized arrangement). The receiver can recover the output amount and the private key (only known by the receiver) of the new UTXO by Diffie-Hellman key exchange, then the receiver can receive the money to his wallet. During the transaction, range proofs is used for proving the validity of the hidden amount (in range $[0, 2^{n-1}]$), linkable ring signature is used for anonymity of initiator, Diffie-Hellman key exchange is used for anonymity of receiver. For detailed description of Monero, please refer to [3], [5].

III. Introduction of BPCEX

In this section we give the construction of BPCEX, first we introduce the UTXO swap technique, which is the key

procedure to realize currency exchange in the Deal stage, then we give the detailed description of BPCEX (for exact matching) and proof of correctness and security. In the last we give the modification of BPCEX to achieve partial deal.

A. UTXO Swap

Assume there are two types of Monero-based cryptocurrencies $(\mathcal{C}_1, \mathcal{C}_2)$ in the blockchain, for UTXO U_1 in $\mathcal{C}_1$ (owned by Alice) and U_2 in $\mathcal{C}_2$ (owned by Bob), the algorithm of UTXO swap is to swap the corresponding public-private keys between U_1 and U_2 without changing the hidden amounts and Pedersen commitments of them. After UTXO swap, the new UTXO U_1' is owned by Bob and U_2' is owned by Alice, which means Alice and Bob exchange their UTXOs successfully. Fig.1 shows the operation in UTXO swap, denoted by $(U_1', U_2') \leftarrow \mathsf{Swap}(U_1, U_2)$, where $U_1, U_1' \in \mathcal{C}_1$, $U_2, U_2' \in \mathcal{C}_2$.

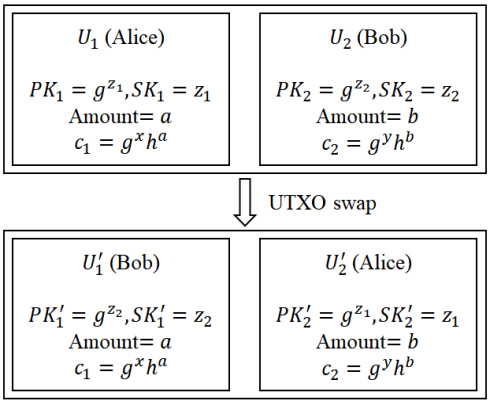

Fig. 1. UTXO swap.

It should be emphasized that if Alice does not know the hidden amount b and the blinding element y in $c_2 = g^y h^b$, she still cannot spend U_2', which is the main challenge to construct currency exchange scheme directly from UTXO swap. A simple solution is to let the dealmaker send the encryption of b, y to Alice in the Match stage (using PK_2' to encrypt), but when the dealmaker is malicious, he gives the wrong ciphertext to Alice, the verifiers cannot find any error during verification, as all the information are hidden to the verifiers. In the construction of BPCEX, we solve this problem by publishing the final exchange rate (by dealmaker) which has no effects on the privacy of the scheme, but can help both users to recover the hidden amounts and blinding elements correctly without any additional encryption and decryption, and can prevent malicious dealmaker from maliciously matching (including incorrect matching, incomplete matching, detailed description of malicious dealmaker is in III.D).

B. Construction

In this subsection we give the construction of BPCEX for exact matching, which is a simpler scheme can help readers have a more direct understanding of the techniques in BPCEX, including usage of exchange rate, usage of Pedersen commitment in Match stage, privacy recovery by users. Then we give the modified scheme to achieve partial deal in III.E.

The BPCEX consists of six algorithms: Setup, Quote, Match, Deal, Verify and Receive:

$(\mathsf{Par}, \mathsf{Add}s, \mathsf{Key}s) \leftarrow \mathsf{Setup}(\lambda)$:

1. System chooses elliptic curve $\mathbb{G}$ with prime order q and a generator $g \in \mathbb{G}$, then generates another independent element $h \in \mathbb{G}$ whose discrete logarithm is unknown to anyone (optional, use hash to point to compute h), system outputs $(\mathbb{G}, q, g, h)$ as the public parameters;
2. Alice and Bob generate their long-term addresses for each cryptocurrency respectively, similar to Monero;
3. The dealmaker generates his public-private key pair $(PK_d, SK_d) = (g^{x_d}, x_d)$ and his long-term address Add_d.

Alg. 1. Setup

Assume Alice wants to exchange her UTXO U_α (with amount a_α, commitment $c_\alpha = g^{x_\alpha} h^{a_\alpha}$, public-private keys (PK_α, SK_α)) in $\mathcal{C}_1$ to coins in $\mathcal{C}_2$, she needs to submit an exchange order to the dealmaker, that is the Quote stage.

$(L_U, E_A, R_A, \sigma_A, \pi_A, U_A, U_A') \leftarrow \mathsf{Quote}(U_\alpha, SK_\alpha)$:

1. Alice computes the amount for exchange a and the amount of reward a', satisfying $a_\alpha = a + a'$, then samples random blinding elements $x, x' \in \mathbb{Z}_q^*$ and computes the new commitments $c_A = g^x h^a, c_A' = g^{x'} h^{a'}$, then generates the range proof $\pi_A = (\pi_{RP}(c_A), \pi_{RP}(c_A'))$. Notice that c_A is the commitment of the UTXO U_A for exchange (pending), and c_A' is the commitment of the UTXO U_A' for reward (pending), and π_{RP} refers to range proof;
2. Alice uniformly generates the public-private key pair $(PK_A, SK_A) = (g^{x_A}, x_A)$ (with x_A only known by Alice) for U_A, and generates the public key PK_A' for U_A'. It should be emphasized that the generation of PK_A' is same as in Monero, only the dealmaker (receiver) can recover the corresponding secret key SK_A'. Then Alice outputs $U_A = (PK_A, c_A)$, $U_A' = (PK_A', c_A')$;
3. Alice computes $E_A = \mathsf{Enc}_K(a, a', x, x')$ by standard symmetric encryption algorithm, such as AES, where $K = PK_d^{x_A} = g^{x_A x_d}$ is the Diffie-Hellman key exchange;
4. Alice chooses another n irrelevant UTXOs: $U_1, \cdots, U_n$ (with commitments $c_1, \cdots, c_n$), as the hiding elements, then computes the set of input UTXOs $L_U = \{U_\alpha, U_1, \cdots, U_n\}$ and the corresponding public key sets $L_{PK}^1 = \{PK_\alpha, PK_1, \cdots, PK_n\}$, $L_{PK}^2 = \{c_\alpha (c_A c_A')^{-1}, c_1 (c_A c_A')^{-1}, \cdots, c_n (c_A c_A')^{-1}\}$ (with randomized arrangement), this step is same as Monero;
5. Alice decides the exchange rate range R_A, for example $1 : 1.95 \to 1 : 2.05$;
6. Alice runs the multilayered linkable ring signature Rsign: $\sigma_A = \mathsf{Rsign}(SK_\alpha, SK_\alpha'; L_U, E_A, R_A, \pi_A, U_A, U_A')$, outputs the quotation $Q_A = (L_U, E_A, R_A, \sigma_A, \pi_A, U_A, U_A')$.

Alg. 2. Quote (*Alice*)

Notice that the Quote algorithm is similar to the ordinary transactions in Monero, with difference in the generation of (PK_A, SK_A), where SK_A is only known by Alice, not the receiver (in Monero), another difference is in Quote, extra information R_A is added for matching the quotation. Moreover, in step 3, one can also use asymmetric encryption $E_A = \mathsf{Enc}_{PK_d}(a, a', x, x')$ which can be recovered by deal-

maker, the choice of encryption is not restricted; in step 6, one of the signing key is $SK'_\alpha = x_\alpha - x - x'$, the multilayered linkable ring signature (MSLAG [5]) is same as Monero.

Similar to Alice, Bob also submits an exchange order to the dealmaker, assume Bob wants to exchange his UTXO U_β (with amount a_β and commitment c_β) in $\mathcal{C}_2$ to coins in $\mathcal{C}_1$, he runs the Quote algorithm to generate the quotation output $Q_B = (L'_U, E_B, R_B, \sigma_B, \pi_B, U_B, U'_B) \leftarrow \mathsf{Quote}(U_\beta, SK_\beta)$, where $U_B = (PK_B, c_B)$, $U'_B = (PK'_B, c'_B)$, $c_B = g^y h^b, c'_B = g^{y'} h^{b'}$ and $E_B = \mathsf{Enc}_{K'}(b, b', y, y')$.

For the dealmaker, he can extract all the valid quotations from the blockchain, then match the quotations to earn the reward. In the following we assume that the quotations from Alice and Bob are matching, we call the dealmaker David:

$(\sigma_d, z, c_d, r_d, Q_A, Q_B) \leftarrow \mathsf{Match}(Q_A, Q_B, SK_d)$:

1. David receives all the valid quotations (Q_A, Q_B) from the blockchain, computes $K = PK_A^{x_d}, K' = PK_B^{x_d}$ by his secret key SK_d, and recovers the corresponding privacy information $(a, a', x, x') \leftarrow \mathsf{Dec}_K(E_A)$, $(b, b', y, y') \leftarrow \mathsf{Dec}_{K'}(E_B)$;
2. David checks $c_A \stackrel{?}{=} g^x h^a, c'_A \stackrel{?}{=} g^{x'} h^{a'}, c_B \stackrel{?}{=} g^y h^b, c'_B \stackrel{?}{=} g^{y'} h^{b'}$, if all passed then continues, otherwise aborts;
3. David decides to match the quotation Q_A, Q_B, if $a : b$ is in $R_A \cap R_B$ (the amounts of rewards (a', b') are also taken into consideration by David to make the decision);
4. David computes the final exchange rate $r_d = \gamma : \theta = a : b$, for example when $a = £1000, b = \$1980$, $r_d = \gamma : \theta = 50 : 99$ with $\gamma = 50, \theta = 99$, and r_d is publishes as (γ, θ);
5. David computes $c_d = c_A^\theta / c_B^\gamma$ and $z = x\theta - y\gamma$;
6. David signs $\sigma_d = \mathsf{Sign}(SK_d; z, c_d, r_d, Q_A, Q_B)$ and outputs the matching result $M_d = (\sigma_d, z, c_d, r_d, Q_A, Q_B)$.

Alg. 3. Match

The verifiers are responsible to check the validity of all the transactions and exchanges in the blockchain. For ordinary transactions, verifications are same as in Monero; for currency exchanges, additional verifications for Match are needed, while the verifications for Quote are similar as in Monero:

$1/0 \leftarrow \mathsf{Verify}(Q_A)$:

1. For the verification of Quote, similar to Monero, the verifiers check the validity of L^1_{PK} and L^2_{PK};
2. Then the verifiers check the validity of σ_A and π_A;
3. If all passed then outputs 1, otherwise outputs 0.

$1/0 \leftarrow \mathsf{Verify}(M_d)$:

1. For the verification of Match, verifiers check whether r_d lies in $R_A \cap R_B$;
2. Then the verifiers check $c_A^\theta / c_B^\gamma \stackrel{?}{=} c_d$ and $g^z \stackrel{?}{=} c_d$;
3. Then the verifiers check the validity of σ_d;
4. If all passed then outputs 1, otherwise outputs 0.

Alg. 4. Verify

The Deal is executed after all the verifications are passed:

$(U^*_A, U^*_B) \leftarrow \mathsf{Deal}(Q_A, Q_B, M_d)$:

1. When all the verifications are passed, verifiers run UTXO swap algorithm to get the swapped UTXOs $(U^*_B, U^*_A) \leftarrow \mathsf{Swap}(U_A, U_B)$;
2. Verifiers revoke the old UTXOs (U_A, U_B) and add new UTXOs $(U^*_A, U^*_B, U'_A, U'_B)$ to the system, where $U^*_A \in \mathcal{C}_2$ is owned by Alice, $U^*_B \in \mathcal{C}_1$ is owned by Bob and $U'_A \in \mathcal{C}_1, U'_B \in \mathcal{C}_2$ are owned by David.

Alg. 5. Deal

All participants (Alice, Bob, David) are able to receive the UTXOs when Deal is done, then they can spent the new money as they wish.

$(b^*, y^*) \leftarrow \mathsf{Receive}(a, x, M_d, U^*_A)$ (*Alice*):

1. For a valid Deal output of the exchange $U^*_A = (PK_A, c_B)$, Alice computes $b^* = a\theta\gamma^{-1}$;
2. Alice computes $y^* = (x\theta - z)\gamma^{-1}$;
3. Alice checks $g^{y^*} h^{b^*} \stackrel{?}{=} c_B$, if passed then receives U^*_A to her wallet.

$(a^*, x^*) \leftarrow \mathsf{Receive}(b, y, M_d, U^*_B)$ (*Bob*):

1. For a valid Deal output of the exchange $U^*_B = (PK_B, c_A)$, Bob computes $a^* = b\gamma\theta^{-1}$;
2. Bob computes $x^* = (y\gamma + z)\theta^{-1}$;
3. Alice checks $g^{x^*} h^{a^*} \stackrel{?}{=} c_A$, if passed then receives U^*_B to his wallet.

$(SK'_A, SK'_B, a', b', x', y') \leftarrow \mathsf{Receive}(Q_A, Q_B, SK_d)$ (*David*):

1. David recovers (a', b', x', y') by usage of his secret key SK_d, then checks $c'_A \stackrel{?}{=} g^{x'} h^{a'}, c'_B \stackrel{?}{=} g^{y'} h^{b'}$, this step has been done in Match;
2. David computes the secret keys (SK'_A, SK'_B) of the reward UTXO (U'_A, U'_B), which is same as in Monero;
3. David receives the (U'_A, U'_B) to his wallet.

Alg. 6. Receive

In the stage Match, if the ciphertext E_A, E_B is invalid, David will abort and sign a cheat message for the corresponding quotations, then the verifiers will directly revoke the UTXOs in the invalid quotations as the punishment, this setting can prevent dishonest users from spamming quotations. Meanwhile, in the generation of $r_d = (\gamma, \theta)$, David samples $u \in \mathbb{Z}^*_{2^{\lfloor \log q \rfloor - N}}$ uniformly at random, then computes $\gamma = au, \theta = bu$ to ensure the computation of (γ, θ) does not exceed q (no boundary crossing occurs), where N is the maximum amount length in the cryptocurrency ($N = 32$ in Monero).

For Alice and Bob, they can withdraw the quotations when there are no suitable matching quotations by signing the withdraw information, then they receive (U_A, U_B) to their wallets respectively, the dealmaker can still get the reward but cannot make any matching after the quotation is withdrawn. For (U_A, U_B), it cannot be spent before it is withdrawn, that is to say, (U_A, U_B) is in a pending state and cannot be transacted.

For a complete currency exchange in BPCEX, the waiting time for both Alice and Bob is at least 2-block generation time, where in block Block_{m-1} they submit the quotations, in block Block_m the dealmaker finishes Match and in block Block_{m+1} the verifiers finish the UTXO swap and Deal.

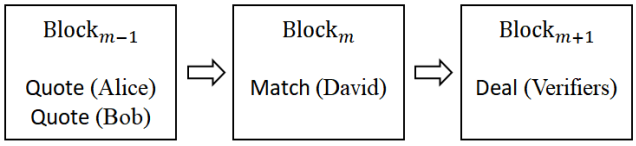

Fig. 2. Waiting blocks.

C. Correctness

Theorem 2 (Correctness of BPCEX): For honest users Alice and Bob (with matching quotations), and honest dealmaker David, they can successfully finish the currency exchange: Alice and Bob can receive the swaped UTXOs and David can receive the reward correctly.

Proof: If all the messages in Quote, Match are correctly generated, then we get $r_d = \gamma : \theta = a : b$ and $z = x\theta - y\gamma$, then $c_A^{\theta}/c_B^{\gamma} = (g^x h^a)^{\theta}/(g^y h^b)^{\gamma} = g^{x\theta - y\gamma} h^{a\theta - b\gamma} = g^z = c_d$, then they can pass the verification. Meanwhile, when the deal is done, Alice computes $b = a\theta\gamma^{-1}$, $y = (x\theta - z)\gamma^{-1} = y\gamma\gamma^{-1}$, Bob computes $a = b\gamma\theta^{-1}$, $x = (y\gamma + z)\theta^{-1} = x\theta\theta^{-1}$. Then Alice and Bob can recover the hidden amounts and blinding elements respectively, then they can receive the new UTXOs to their wallets.

For David, he can recover (a', b', x', y') by usage of his secret key SK_d, and can generate the private keys for U'_A and U'_B correctly (from the correctness of Monero), then he can receive the rewards to his wallet. □

D. Security

1) Security Requirements: The security requirements of BPCEX contains anonymity, confidentiality, double-spending resistance and malicious dealmaker resistance. We introduce these requirements respectively:

1. **Anonymity** of BPCEX is similar to Monero, it is required that the identity of user in the exchange cannot be recovered by other parties, including the dealmaker, veirfiers and the counterparty (for Alice, her counterparty is Bob), this feature makes BPCEX a completely anonymous currency exchange scheme.
2. **Confidentiality** of BPCEX is that the exchange amounts (a, b) and blinding elements (x, y) are hidden to other parties (excluding the dealmaker and the counterparty), the definition of confidentiality of BPCEX is weaker than Monero, as the amounts and blinding elements of the swaped UTXOs can be recovered by the dealmaker. Meanwhile, the final exchange rate is published as $r_d = (\gamma, \theta)$, as well as $z = x\theta - y\gamma$, which may bring about leakage of the privacy if one of a, b, x, y is disclosed.
3. **Double spending Resistance** of BPCEX is similar to Monero that any user cannot double spend his UTXOs in $\mathcal{C}_1$ or $\mathcal{C}_2$, an extra requirement of BPCEX is that during the exchange, any user cannot spend his original UTXOs. For example, in the stages of Quote, Match, Deal, Alice cannot spend U_α, U_A, after the stage Deal, she can only spend the swaped UTXO U_A^*.
4. **Malicious Dealmaker Resistance** of BPCEX means that the malicious dealmaker cannot spend the UTXOs (U_A, U_B) in the exchange, and he cannot break the anonymity of users and anonymity of UTXOs. Moreover, he cannot make incorrect matching and incomplete matching, which means he cannot make a match that is incorrect $(a : b \neq \gamma : \theta)$, nor make a match that is incomplete (Alice cannot recover the amount or blinding element of U_A^* from the output of a valid Deal).

2) Proof of Anonymity:

Theorem 3: BPCEX is an anonymous currency exchange scheme with UTXO anonymity and addresses anonymity.

Proof: In BPCEX, user Alice makes a special transaction to the dealmaker (and herself) in the Quote stage, the anonymity of the transaction is same as Monero: using linkable ring signature to hide the identity of the input UTXO (U_α) among other UTXOs ($\{U_1, \cdots, U_n\}$). The anonymity of input UTXO is derived from the anonymity of linkable ring signature. For the uniformly sampled public-private key pair (PK_A, SK_A) of U_A, it has no relationship with Alice's long-term address, then any adversary cannot recover Alice's long-term address from the exchange, then we get the UTXO anonymity and addresses anonymity of BPCEX. □

3) Proof of Confidentiality:

Theorem 4: BPCEX has $1/2^{\lfloor \log q \rfloor - N}$ amount confidentiality and blinding element confidentiality for any PPT adversary except for the dealmaker and the exchange parties.

Proof: For the range proofs $\pi_A = (\pi_{RP}(c_A, c'_A)), \pi_B = (\pi_{RP}(c_B, c'_B))$ in the exchange, we know that the proofs π_A, π_B will not leak any information of the exchange amounts and the blinding elements, this feature is similar to Monero. The additional messages related with the exchange amount in BPCEX contains the ciphertext E_A, E_B, the exponent z and the final exchange rate r_d, where E_A, E_B will not leak any information according to the semantic security of the encryption algorithm. Moreover, $r_d = (\gamma, \theta) = (au, bu)$ with uniformly sampled $u \leftarrow \mathbb{Z}^*_{2^{\lfloor \log q \rfloor - N}}$ by the dealmaker, there are $2^{\lfloor \log q \rfloor - N}$ possible amounts of (a, b) if r_d is published. The possibility of correctly guessing the right amount of (a, b) is $1/2^{\lfloor \log q \rfloor - N}$, then we get the $1/2^{\lfloor \log q \rfloor - N}$ amount confidentiality of BPCEX for any PPT adversary except for the dealmaker and the exchange parties.

For the blinding elements (x, y), we know that any PPT adversary cannot recover x, y by usage of $z = x\theta - y\gamma, \gamma, \theta$, then we get the blinding element confidentiality of BPCEX for any PPT adversary except for the dealmaker and the exchange parties. □

4) Proof of Double spending Resistance:

Theorem 5: Double spending is prevented in BPCEX for any PPT adversary, assuming the majority of verifiers are honest (the consensus is valid).

Proof: In the Quote stage, the double spending prevention of U_α, U_β is from the linkability of linkable ring signature, which is same as Monero. Moreover, for UTXOs U_A, U_B which are pending to exchange, they can only be spent after the quotation is withdrawn, according to the honesty of the

verifiers, who will not let any pending UTXO appear in a valid transaction. Moreover, when the Deal is done, U_A, U_B is removed and U_A^*, U_B^* is added into the system, the double spending prevention of U_A^*, U_B^* is also same as Monero, then we finish the double spending prevention of BPCEX. □

5) Proof of Malicious Dealmaker Resistance:

Theorem 6: BPCEX can prevent malicious dealmaker from spending user's UTXO, recovering identity of user and identity of input UTXO. BPCEX can also prevent malicious dealmaker from making a match that is incorrect or incomplete.

Proof: Since the public-private key pair (PK_A, SK_A) is generated by Alice, the dealmaker cannot learn any information about SK_A, then he cannot spend the swaped UTXO U_A^* after the Deal is done. Moreover, the dealmaker cannot recover the identity of Alice and the identity of U_α, according to the anonymity of BPCEX in *Theorem 3*.

For the prevention of incorrect match and incomplete match, we prove that, for any valid Match result $(\sigma_d, z, c_d, r_d, Q_A, Q_B)$ passed by the verifiers, we have $a : b = \gamma : \theta$ and users can recover the exchange amounts (a, b) and blinding elements (x, y) correctly. In fact, for $c_A = g^x h^a$ and $c_B = g^y h^b$, if $g^z = c_A^\theta / c_B^\gamma$, then we know $g^z = g^{x\theta - y\gamma} h^{a\theta - b\gamma}$. If $a\theta - b\gamma \neq 0$, then the dealmaker gets a nontrivial relation between g and h, which contradicts with the hardness of discrete logarithm problem, then we get $a\theta - b\gamma = 0$ with $a : b = \gamma : \theta$, and both users can recover the exchange amount (a, b) with the help of (γ, θ). Moreover, $g^z = g^{x\theta - y\gamma} h^{a\theta - b\gamma} = g^{x\theta - y\gamma}$, then $z = x\theta - y\gamma$ and both users can recover x, y with the help of z. □

E. Modification

The BPCEX introduced in III.B only supports exact deal, which means the exchange amounts (a, b) can reach a deal satisfying $a : b \in R_A \cap R_B$. However, in real-life currency exchange, the amounts in the quotation are not always matching for a full deal, then partial deal is needed in BPCEX to increase the success rate of exchange. We give the modified Quote', Match' and Deal' to realize this functionality.

In the modified scheme, we introduce a new concept called UTXO partition: for a UTXO U with amount a and commitment $c = g^x h^a$, we partition U into U_1, U_2 by using proportion of deal $\tau \in [0, 1]$, with the amount in U_1 being τa and amount in U_2 being $(1 - \tau)a$, and the corresponding commitments of U_1 and U_2 become $c_1 = g^{x_1} h^{\tau a}$ and $c_2 = g^{x_2} h^{(1-\tau)a}$. For example, when $c = g^x h^{1000}$ with $\tau = 0.1$ (10% of the amount is partitioned), then $c_1 = g^{x_1} h^{100}$ and $c_2 = g^{x_2} h^{900}$ is the partition results. In the construction of modified Match', proportion of deal τ is added into the scheme to realize partial deal.

$(L_U, E_A, R_A, \sigma_A, \pi_A, U_A, U'_A, S_A) \leftarrow$ Quote'(U_α, SK_α)

1. Alice computes the amount for exchange a and the amount of reward a', satisfying $a_\alpha = a + a'$, then samples random blinding elements $x, x' \in \mathbb{Z}_q^*$ and computes the new commitments $c_A = g^x h^a, c'_A = g^{x'} h^{a'}$, then generates the range proof $\pi_A = (\pi_{RP}(c_A), \pi_{RP}(c'_A))$. Notice that c_A is the commitment of the UTXO U_A for exchange (pending), and c'_A is the commitment of the UTXO U'_A for reward (pending), and π_{RP} refers to range proof;
2. Alice uniformly generates the public-private key pair $(PK_A, SK_A) = (g^{x_A}, x_A)$ (with x_A only known by Alice) for U_A, and generates the public key PK'_A for U'_A. It should be emphasized that the generation of PK'_A is same as in Monero, only the dealmaker (receiver) can recover the corresponding secret key SK'_A. Then Alice outputs $U_A = (PK_A, c_A)$, $U'_A = (PK'_A, c'_A)$;
3. Alice computes $E_A = \mathsf{Enc}_K(a, a', x, x', r_A)$ by standard symmetric encryption algorithm, such as AES, where $K = PK_d^{x_A} = g^{x_A x_d}$ is the Diffie-Hellman key exchange, r_A is uniformly sampled in $\mathbb{Z}_q^*$;
4. Alice chooses another n irrelevant UTXOs: $U_1, \cdots, U_n$ (with commitments $c_1, \cdots, c_n$), as the hiding elements, then computes the set of input UTXOs $L_U = \{U_\alpha, U_1, \cdots, U_n\}$ and the corresponding public key sets $L_{PK}^1 = \{PK_\alpha, PK_1, \cdots, PK_n\}$, $L_{PK}^2 = \{c_\alpha (c_A c'_A)^{-1}, c_1 (c_A c'_A)^{-1}, \cdots, c_n (c_A c'_A)^{-1}\}$ (with randomized arrangement), this step is same as Monero;
5. Alice decides the exchange rate range R_A;
6. Alice computes $s_i = g^{H(r_A, i)}$ for $i = 1, \cdots, M_A$, where M_A is the maximum number of UTXO partitions (set by Alice), then gets $S_A = \{s_1, \cdots, s_{M_A}\}$;
7. Alice runs the multilayered linkable ring signature Rsign: $\sigma_A = \mathsf{Rsign}(SK_\alpha, SK'_\alpha; L_U, E_A, R_A, \pi_A, U_A, U'_A, S_A)$, outputs $Q_A = (L_U, E_A, R_A, \sigma_A, \pi_A, U_A, U'_A, S_A)$.

Alg. 7 Quote' (*Alice*)

Bob also submits an order to exchange his U_β (with amount a_β and commitment c_β) in $\mathcal{C}_2$ to coins in $\mathcal{C}_1$, he runs the Quote' algorithm to generate the quotation output $Q_B = (L'_U, E_B, R_B, \sigma_B, \pi_B, U_B, U'_B, S_B) \leftarrow$ Quote'(U_β, SK_β), where $U_B = (PK_B, c_B)$, $U'_B = (PK'_B, c'_B)$, $c_B = g^y h^b, c'_B = g^{y'} h^{b'}$, $E_B = \mathsf{Enc}_{K'}(b, b', y, y', r_B)$ and $S_B = \{t_1, \cdots, t_{M_B}\}$ with $t_i = g^{H(r_B, i)}$.

After receiving quotations from both sides (Alice and Bob), the dealmaker David will make a match if the intersection of their exchange rate ranges is not empty. Without loss of generality, we assume Bob's quotation reaches a full deal and Alice's quotation reaches a partial deal with proportion τ_A.

$(\sigma_d, z, c_d, r_d, Q_A, Q_B, U_1, U'_1, \tau_A) \leftarrow$ Match'(Q_A, Q_B, SK_d):

1. David receives all the valid quotations (Q_A, Q_B) from the blockchain, computes $K = PK_A^{x_d}, K' = PK_B^{x_d}$ by his secret key SK_d, and recovers the corresponding privacy information $(a, a', x, x', r_A) \leftarrow \mathsf{Dec}_K(E_A)$, $(b, b', y, y', r_B) \leftarrow \mathsf{Dec}_{K'}(E_B)$;
2. David checks $c_A \stackrel{?}{=} g^x h^a, c'_A \stackrel{?}{=} g^{x'} h^{a'}, c_B \stackrel{?}{=} g^y h^b, c'_B \stackrel{?}{=} g^{y'} h^{b'}$, then checks the validity of S_A and S_B, if all passed then continues, otherwise aborts;
3. If $R_A \cap R_B \neq \varnothing$, then David decides to match the quotation Q_A, Q_B (the amounts of rewards (a', b') are also taken into consideration by David to make the decision);

4. David computes $0 < a_1 < a$ satisfying $a_1 : b \in R_A \cap R_B$, then computes the final exchange rate $r_d = \gamma : \theta = a_1 : b$ and r_d is published as (γ, θ), then computes the deal proportion of Alice's quotation $\tau_A = a_1 : a = \lambda_1/\lambda$ and τ_A is published as (λ_1, λ);
5. David computes the partitioned UTXO (U_1, U_1'), with commitment $c_1 = s_1 \cdot c_A^{\lambda_1\lambda^{-1}}$ and $c_1' = c_A^{(\lambda-\lambda_1)\lambda^{-1}}$;
6. David sets the public key of U_1 as $PK_1 = PK_A \cdot s_1$, sets public key of U_1' as $PK_1' = PK_A$, then we get $U_1 = (c_1, PK_1)$ and $U_1' = (c_1', PK_1')$;
7. David computes $c_d = c_1^{\theta}/c_B^{\gamma}$ and $z = (H(r_A, 1) + x\lambda_1\lambda^{-1})\theta - y\gamma$;
8. David signs $\sigma_d = \mathsf{Sign}(SK_d; z, c_d, r_d, Q_A, Q_B, U_1, U_1', \tau_A)$ and outputs $M_d = (\sigma_d, z, c_d, r_d, Q_A, Q_B, U_1, U_1', \tau_A)$.

Alg. 8. Match'

The verifiers can check the validity of the Match' result and then executes the UTXO swap, then both users can receive the swaped UTXO to their wallets respectively:

$1/0 \leftarrow$ Verify'(M_d):
1. For the verification of Match', verifiers check whether r_d lies in $R_A \cap R_B$;
2. Then the verifiers check $c_1 \stackrel{?}{=} s_1 \cdot c_A^{\lambda_1\lambda^{-1}}$, then check $c_1^{\theta}/c_B^{\gamma} \stackrel{?}{=} c_d$ and $g^z \stackrel{?}{=} c_d$;
3. Then the verifiers check $PK_1 \stackrel{?}{=} PK_A \cdot s_1$;
4. Then the verifiers check the validity of σ_d;
5. If all passed then outputs 1, otherwise outputs 0.

$(U_A^*, U_B^*, U_1') \leftarrow$ Deal'(Q_A, Q_B, M_d):
1. When all the verifications are passed, verifiers run UTXO swap algorithm to get the swapped UTXOs $(U_B^*, U_A^*) \leftarrow \mathsf{Swap}(U_1, U_B)$;
2. Verifiers revoke the old UTXOs (U_A, U_B) and add new UTXOs $(U_A^*, U_B^*, U_B', U_1')$ to the system, where $U_A^* \in \mathcal{C}_2$ is owned by Alice, $U_B^* \in \mathcal{C}_1$ is owned by Bob, $U_B' \in \mathcal{C}_2$ is owned by David, and $U_1' \in \mathcal{C}_1$ is still in the quotation for exchange.

$(b^*, y^*) \leftarrow$ Receive'(a, x, M_d, U_A^*) (*Alice*):
1. For a valid Deal output of the exchange $U_A^* = (PK_1, c_B)$, Alice computes $a_1^* = a\lambda_1\lambda^{-1}$, $b^* = a_1^*\theta\gamma^{-1}$;
2. Alice computes $y^* = ((H(r_A, 1) + x\lambda_1\lambda^{-1})\theta - z)\gamma^{-1}$;
3. Alice checks $g^{y^*}h^{b^*} \stackrel{?}{=} c_B$, if passed then receives U_A^* to her wallet.

$(a_1^*, x^*) \leftarrow$ Receive'(b, y, M_d, U_B^*) (*Bob*):
1. For a valid Deal output of the exchange $U_B^* = (PK_B, c_1)$, Bob computes $a_1^* = b\gamma\theta^{-1}$;
2. Bob computes $x^* = (y\gamma + z)\theta^{-1}$;
3. Bob checks $g^{x^*}h^{a_1^*} \stackrel{?}{=} c_1$, if passed then receives U_B^* to his wallet.

Alg. 9. Verify', Deal' and Receive'

Notice that the modified scheme above only describe the first partition of U_A for partial deal, when it is partitioned for the second time (if necessary), then s_2 will be used in the generation of the new UTXO. For the dealmaker, he can receive the reward from Alice until all the partitions of U_A reaches a exchange deal, then U_A' is added into the system by the verifiers, and the dealmaker can receive it to his wallet.

F. Efficiency Estimation

The BPCEX is deeply related with Monero system, where Quote is a special transaction in $\mathcal{C}_1$ (or $\mathcal{C}_2$), with slightly more message of the exchange rate range, which makes the size and verification time for Quote very close to the transactions in Monero. Meanwhile, computations in Match is efficient as it only consists of computations in $\mathbb{G}$ (with 12 exponentiations, 5 multiplications), computations in $\mathbb{Z}_q^*$ (with 5 multiplications), AES decryption and signature. The verification of Match only consists of computations in $\mathbb{G}$ (with 3 exponentiations, 1 multiplications) and verification of signature, which is also efficient. Moreover, The Receive is similar to the receive algorithm in Monero, with slightly more computations in $\mathbb{Z}_q^*$ (compute $b^* = a\theta\gamma^{-1}, y^* = (x\theta - z)\gamma^{-1}$). To summarize, the overall storage and computations of BPCEX is very close to the transactions in Monero system.

IV. Towards Regulation

A. Motivation and Necessity

Privacy-preserving currency exchange is something to be taken seriously, as it may facilitate crimes such as money laundering, transfer of illegal assets, where criminals can easily escape from legal sanctions due to the privacy protection mechanisms, making the legitimacy of BPCEX seriously threatened. Meanwhile, privacy protection is also the need of honest users (enterprises) to isolate the privacy from the public (competitors). So it is necessary to construct a BPCEX with the regulatory functions to achieve crime prevention.

B. Construction

A recent work [11] proposed the first fully regulatable privacy-preserving blockchains against malicious regulators, in their constructions, there exists a regulator who can trace all the identities and amounts of transactions, while in the verifier's view the scheme remains anonymous and confidential.

In the construction of regulatable BPCEX, we use TLRS (traceable and linkable ring signature) to replace MLSAG (multilayered linkable ring signature) and use TPR (traceable range proof) to replace RP (range proof), we only describe the differences between BPCEX and regulatable BPCEX:

$(\mathsf{Par}, \mathsf{Add}s, \mathsf{Key}s) \leftarrow \mathsf{Setup}(\lambda)$:
1. System chooses elliptic curve $\mathbb{G}$ with prime order q and a generator $g \in \mathbb{G}$, the regulator $\mathcal{R}$ generates a trapdoor $y \in \mathbb{Z}_q^*$, then computes $h = g^y \in \mathbb{G}$, system outputs $(\mathbb{G}, q, g, h)$ as the public parameters.

$(L_U, E_A, R_A, \sigma_A, \pi_A, U_A, U_A') \leftarrow \mathsf{Quote}(U_\alpha, SK_\alpha)$:
1. Alice computes the amount for exchange a and the amount of reward a', satisfying $a_\alpha = a + a'$, then samples random blinding elements $x, x' \in \mathbb{Z}_q^*$ and computes the new commitments $c_A = g^x h^a, c_A' = g^{x'} h^{a'}$, then generates the range proof $\pi_A = (\pi_{TRP}(c_A), \pi_{TRP}(c_A'))$. Where π_{TRP} refers to traceable range proof;
6. Alice runs the traceable and linkable ring signature TLRS: $\sigma_A = \mathsf{TLRS}(SK_\alpha, SK_\alpha'; L_U, E_A, R_A, \pi_A, U_A, U_A')$, outputs the quotation $Q_A = (L_U, E_A, R_A, \sigma_A, \pi_A, U_A, U_A')$.

$(U_\alpha, U_\beta, a, b) \leftarrow \mathsf{Trace}(Q_A, Q_B, M_d, y)$:

1. The regulator $\mathcal{R}$ uses his trapdoor y to trace the input UTXO (U_α, U_β) from the TLRS signatures (σ_A, σ_B);
2. $\mathcal{R}$ uses y to trace the exchange amounts (a, b) from the traceable range proofs (π_A, π_B).

Alg. 10. Traceable BPCEX

V. Conclusion

In this paper we give the construction of BPCEX: a blockchain-based privacy-preserving currency exchange between two types of Monero-based cryptocurrencies in the UTXO model. BPCEX achieves anonymity, confidentiality, double spending resistance and malicious dealmaker resistance. BPCEX has the functionality of floating exchange rate and can be modified to realize partial deal, which makes BPCEX suitable in real-life applications. Moreover, BPCEX can be further modified to achieve regulatory functions to prevent money laundering and illegal assets transfer.

References

[1] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, p. 37, 2014.

[3] N. Van Saberhagen, "Cryptonote v 2.0," *URL: https://cryptonote.org/whitepaper.pdf*, pp. 04–13, 2013.

[4] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 459–474.

[5] S. Noether, A. Mackenzie *et al.*, "Ring confidential transactions," *Ledger*, vol. 1, pp. 1–18, 2016.

[6] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 315–334.

[7] Y. V. L. Luu, "Kybernetwork: A trustless decentralized exchange and payment service," *URl: https://kyber.network*.

[8] D. Wang, J. Zhou, A. Wang, and M. Finestone, "Loopring: A decentralized token exchange protocol," *URL: https://github.com/Loopring/whitepaper*, 2018.

[9] W. Warren and A. Bandeali, "0x: An open protocol for decentralized exchange on the ethereum blockchain," *URL: https://github.com/0xProject/whitepaper*, 2017.

[10] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 839–858.

[11] W. Li, L. Chen, X. Lai, X. Zhang, and J. Xin, "Fully regulatable privacy-preserving blockchains against malicious regulators," Cryptology ePrint Archive, Report 2019/925, 2019.

[12] A. Back, "Ring signature efficiency," *Bitcointalk (accessed 1 May 2015) https://bitcointalk.org/index.php*, 2015.

[13] G. Maxwell and A. Poelstra, "Borromean ring signatures," 2015.

[14] T. H. Yuen, S.-f. Sun, J. K. Liu, M. H. Au, M. F. Esgin, Q. Zhang, and D. Gu, "Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security," 2019.

[15] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von neumann architecture," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 781–796.

[16] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity." *IACR Cryptology ePrint Archive*, vol. 2018, p. 46, 2018.

[17] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward, "Aurora: Transparent succinct arguments for r1cs," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2019, pp. 103–128.

[18] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish, "Doubly-efficient zksnarks without trusted setup," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 926–943.

[19] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song, "Libra: Succinct zero-knowledge proofs with optimal prover computation." *IACR Cryptology ePrint Archive*, vol. 2019, p. 317, 2019.

[20] B. Bünz, B. Fisch, and A. Szepieniec, "Transparent snarks from dark compilers," 2019.

[21] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2001, pp. 552–565.

[22] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup, "Anonymous identification in ad hoc groups," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2004, pp. 609–626.

[23] N. Chandran, J. Groth, and A. Sahai, "Ring signatures of sub-linear size without random oracles," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2007, pp. 423–434.

[24] J. Groth and M. Kohlweiss, "One-out-of-many proofs: Or how to leak a secret and spend a coin," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 253–280.

[25] J. K. Liu, V. K. Wei, and D. S. Wong, "Linkable spontaneous anonymous group signature for ad hoc groups," in *Australasian Conference on Information Security and Privacy*. Springer, 2004, pp. 325–335.

[26] J. K. Liu, M. H. Au, W. Susilo, and J. Zhou, "Linkable ring signature with unconditional anonymity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 157–165, 2013.

[27] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Annual International Cryptology Conference*. Springer, 1991, pp. 129–140.