Applications on traceable range proofs from fully regulatable privacy-preserving blockchains

Wulu Li[1], Lei Chen[1], Xin Lai[1], Xiao Zhang[1], and Jiajun Xin[1]

Shenzhen Onething Technologies Co., Ltd., Shenzhen, China
liwulu@onething.net

Abstract. Traceable range proofs enable regulators to trace the amounts of transactions in privacy-preserving blockchains, but it lacks adequate functionality in the application scenarios such as currency (assets) transfer, international trades and taxation, etc. In this paper, we give multiple modifications and applications on traceable Borromean range proof (T-BoRP) and traceable Bulletproofs range proof (TBuRP), which realize functionalities including multi-currency regulation, regulatable private assets transfer, auxiliary privacy calculation and secure joint regulation by usage of zero-knowledge proofs, homomorphic commitments and MPC protocols. Our work can help regulators to choose the regulatory policy as they wish and help users to transfer assets, compute private amounts under the regulation efficiently and independently. Our solutions are well suited for the future applications of regulatable privacy-preserving cryptocurrencies.

Keywords: Regulatable blockchains · Privacy preserving · Traceable range proofs · Applications.

1 Introduction

Blockchain technology has received extensive attention due to its functionalities such as open, transparentness, non-tamperability and traceability, which make it a potential candidate for cryptocurrency. Blockchain-based cryptocurrencies have been widely studied and developed since the introduction of Bitcoin[9] by Nakamoto in 2008. Ethereum [2], Monero[13], Zerocash[12] are the main representatives of blockchain-based cryptocurrencies. In June 2019, Facebook announced "Libra"[4], an international blockchain-based cryptocurrency, which will undoubtedly change the global financial systems and people's lives.

Traditional blockchain-based cryptocurrencies (such as Bitcoin and Ethereum) are publicly accounted and provide no privacy protection where the amounts of transactions are stored in plaintext and can be accessed by any user, making them restricted in various scenarios such as salary, donation, bidding, taxation, etc. To achieve privacy protection, there are various solutions including Confidential Transaction[7], Mimblewimble[5], Dash[3], Monero[13] and Zerocash[12], etc. Among them, Monero uses Cryptonote[13], Ring-CT[10] and Bulletproofs[1] as the key components and becomes one of the most promising solutions

for the future application of privacy-preserving blockchains. Nevertheless, the privacy-preserving blockchains have no regulatory functions, that would cause serious risks of illegal purposes such as illegal transactions, illegal assets transfer, money laundering, fraud, etc. To fill this gap, Li *et al.*[6] proposed traceable and linkable ring signatures (TLRS) and traceable range proofs (TBoRP and TBuRP) to achieve the traceability for identities, addresses and amounts in arbitrary transactions. Their construction is the first fully regulatable privacy-preserving blockchains against malicious regulators, where regulators can only trace signer identities, user addresses and amounts, while cannot double spend, corrupt users, slander users or escape from regulation, it is a proper solution to realize the regulatory function of blockchain-based cryptocurrency to keep the balance of decentralization, privacy protection and regulation.

In the application scenarios such as international trades, there are different currencies (currency A, currency B), different types of private data (unit price a, quantity of goods b and total price $c = a \times b$) regulated by different regulators (bank A, bank B and harbor), we may be face with requirements such as transferring currency A to currency B privately or computing the total price $c = a \times b$ by ourselves or other party privately, etc. As for blockchain-based cryptocurrencies, all of these operations and computations must be publicly verified and regulated by corresponding regulators. The original regulatable privacy-preserving blockchain introduced in [6] cannot provide such functionalities. In this paper, we introduce the modification of traceable range proofs and some cryptographic components to enhance the functionalities of regulatable privacy-preserving blockchains to achieve multi-currency regulation, regulatable private assets transfer, auxiliary privacy calculation and secure joint regulation.

1.1 Related Works

In this section we give the introduction of classic range proofs and traceable range proofs.

Range Proofs 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 2015, Neother *et al.*[10] gave the Borromean range proof, building from the Borromean ring signature[8], with linear proof size to the binary length of range. In 2018, Bünz *et al.*[1] introduced Bulletproofs, an efficient non-interactive zero-knowledge proof protocol with short proof size and without a trusted setup, the proof size is only logarithmic to the witness size and it is used in projects such as Monero, DERO, ZETHER.

Traceable Range Proofs Traceable range proof was first proposed by Li *et al.*[6] in 2019, they gave the construction of traceable Borromean range proof (TBoRP) and traceable Bulletproofs range proof (TBuRP) respectively. In their work, TBoRP is secure against malicious regulator, which means regulators can

only trace the amount of transactions and cannot break the soundness and traceability of TBoRP, meanwhile, TBuRP also has soundness against malicious regulators.

1.2 Our Contributions

In this paper, we modify the traceable range proofs and give several applications on them, including multi-currency regulation, regulatable private assets transfer, auxiliary privacy calculation and secure joint regulation.

Multi-currency Regulation and Regulatable Private Assets Transfer By making use of traceable range proof and Pedersen commitment, we introduce the multi-currency regulation that is suitable for different cryptocurrencies under different regulators, each regulator (say $\mathcal{R}_A$) can only trace the amount of specified currency (say $\mathcal{C}_A$). Users can transfer their money (from $\mathcal{C}_A$ to $\mathcal{C}_B$) freely and privately, the transfer is regulated by both $\mathcal{R}_A$ and $\mathcal{R}_B$ respectively.

We give a brief description of the scheme (with TBoRP) in the following:

- Setup. System sample a generator g of $\mathbb{G}$, each regulator $\mathcal{R}_i$ generate the trapdoor y_i for currency $\mathcal{C}_i$, and compute $h_i = g^{y_i}$ as the public parameter.
- Prove. For user Alice possesses money $\mathcal{C}_i$ for amount a_i, she run the TBoRP to give the range proof $\pi_{TBo}(c_i)$ for a_i from the commitment $c_i = g^x h_i^{a_i}$.
- Transfer. Assume Alice want to transfer her money $\mathcal{C}_i$ to money $\mathcal{C}_j$, she generates the switch proof $\pi_{Swit}(c_i, c_j, \gamma)$ and the new range proof $\pi_{TBo}(c_j)$.
- Verify. Any verifier in the blockchain can verify the proofs given by Alice, all the proofs are publicly verified.

Auxiliary Privacy Calculation Multi-currency regulation (or multiple regulation) scheme can be used in the scenario of auxiliary privacy calculation, in which users can calculate their private data and output the result commitment by themselves or the auxiliary party. For example, in a privacy-preserving blockchain of international trade, the unit price, number of goods and total price are private data which is only known by the owners and regulators, assume the unit price commitment already exists in the blockchain, when the goods arrive in port, the harbor manager (auxiliary party) can count the number of goods, publish the number commitment and calculate the commitment of total price, then the auxiliary party can directly publish the result on the blockchain without waiting for trading parties to calculate.

By usage of traceable range proof, Pedersen commitment and zero-knowledge proof, our scheme supports multiple calculations $(+, -, \times, \div)$ of private data from the commitments, and the results can be traced by regulators. In our construction, we borrow the homomorphic property of Pedersen commitment to realize the calculation of $+, -$, and construct new proof of product and proof of division for different basis to realize the calculation of $\times, \div$. It should be noted

that the scenario of auxiliary privacy calculation is available only for regulatable privacy-preserving blockchains, if the system is non-regulatable, no one in the blockchain can recover the final result before the auxiliary party exchange the private data with trading parties, then the trading parties can calculate the result commitment by themselves. The auxiliary privacy calculation can also be used in calculation of interests, calculation of taxes, etc.

Secure Joint Regulation Both TBoRP and TBuRP introduced before are in the form of one currency with one regulator, a potential risk is that when the regulator is attacked or malicious, the attacker (or malicious regulator) can trace all amounts of transactions for illegal purpose. To fill this gap, we modify the TBuRP to realize the functionality of secure joint regulation (which named JTBuRP), where there are multiple regulators for one currency. For any number of regulators (less than the regulatory threshold), they cannot recover any bit of the transaction amount, which can be traced only when sufficient regulators publish their results via MPC or secure conference. When the number of collusion parties not exceed the threshold, they get nothing about the transaction amount. This scheme is approach for decentralized regulation.

In our construction, we modify the TBuRP in the following aspects:

- Every regulator generates his own trapdoor and computes the corresponding public parameter, and uses the trapdoor in the regulation.
- All regulators reach a consensus about the bit partition of transaction amount, and output the partition result.
- When users generate the traceable range proof, they make merge operations of TK_is to ensure that any number (not exceed the threshold) of regulators cannot obtain any information about the amount, according to the bits partition result.
- All regulators (or exceed the number threshold) compute the tracing results T_i, and work together to recover the actual amount of transaction by MPC or secure conference, which depends on the regulatory policy or the organization of regulators.

1.3 Paper Organization

In section 2 we give some preliminaries, including description of TBoRP and TBuRP; in section 3 we introduce the application of multi-currency regulation and regulatable private assets transfer; in section 4 we introduce the application of auxiliary privacy calculation; The modification of TBuRP and application of secure joint regulation are given in section 5; in section 6 we give the conclusion.

2 Preliminaries

2.1 Notations

In this paper, in order to be consistent with Bulletproofs, 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.

2.2 Zero-knowledge proofs

Zero-knowledge proof system is a proof system (P, V) in which a prover proves to the verifier that he has a certain knowledge but does not reveal the knowledge itself. The formal definition is that given language L and relation R, for $\forall x \in L$, there exists a witness w such that $(x, w) \in R$, to prove $x \in L$ without disclosing w. The transcript between prover and verifier is $\langle P(x, w), V(x)\rangle$, the proof is correct (or wrong) if $\langle P(x, w), V(x)\rangle = 1$(or 0). The security notions of zero-proof system contains *completeness*, *soundness*, *zero-knowledge*:

Definition 1 (Completeness) (P, V) *has* ***completeness*** *for any non-uniform polynomial time adversary* $\mathcal{A}$*,*

$$\mathsf{Pr}[(x, w) \leftarrow \mathcal{A}(1^\lambda) : (x, w) \notin R \text{ or } \langle P(x, w), V(x)\rangle = 1] = 1 - negl.$$

When the probability equals 1*, then* (P, V) *has perfect completeness.*

Definition 2 (Soundness) (P, V) *has* ***soundness*** *for any non-uniform polynomial time adversary* $\mathcal{A}$ *and* $x \notin L$*,*

$$\mathsf{Pr}[(x, s) \leftarrow \mathcal{A}(1^\lambda) : \langle P(x, w), V(x)\rangle = 1] = negl.$$

In Σ *protocols with Fiat-Shamir transformation in the random oracle model, we use the notion of* ***special soundness****, that is, for a* 3*-round interactive proof protocol, if a non-uniform polynomial time adversary* $\mathcal{A}$ *can generate* 2 *valid proofs* $(x, c, e_1, s_1), (x, c, e_2, s_2)$*, then there exists a extraction algorithm* Ext *which can extract a witness* $(x, w) \in R$*, where* c *represents the commitment,* $e_i s$ *are challenges and* $s_i s$ *are responses.*

Definition 3 (Zero-knowledge) (P, V) *has* ***perfect*** *(or* ***computational****)* ***zero-knowledge****, for any non-uniform polynomial time (or PPT) adversary* $\mathcal{A}$*,*

$$\mathsf{Pr}[(x, w) \leftarrow \mathcal{A}(1^\lambda); tr \leftarrow \langle P(x, w), V(x, \rho)\rangle : (x, w) \in R \text{ and } \mathcal{A}(tr) = 1] = (or \approx_c)$$

$$\mathsf{Pr}[(x, w) \leftarrow \mathcal{A}(1^\lambda); tr \leftarrow S(x, \rho) : (x, w) \in R \text{ and } \mathcal{A}(tr) = 1].$$

In Fiat-Shamir-based protocol, the randomness of ρ *is from the output of hash function, it is said to be* ***public coin*** *and the protocol is* ***honest-verifier zero-knowledge****.*

Pedersen Commitment Pedersen commitment[11] 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. Pedersen commitment is the key component to construct a series of zero-knowledge proofs, including proof of equality (switch proof), proof of product.

Definition 4 (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:*

- *(Hiding) Any (computational unbounded) adversary $\mathcal{A}$ cannot distinguish $c = g^x h^a$ from $c' = g^{x'} h^{a'}$.*
- *(Binding) Any PPT adversary $\mathcal{A}$ cannot generate another secret a' binding with $c = g^x h^a = g^{x'} h^{a'}$.*
- *(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 $a + b$.*

Switch Proof For two Pedersen commitments $c_1 = g^x h_1^a$ and $c_2 = g^y h_2^a$, we can prove the equality of hidden value ($a = a$) by switch proof (h_1 switch to h_2):

1. Prover generates $r_1, r_2, r \in \mathbb{Z}_q^*$ uniformly at random, computes $e = H(g^{r_1} h_1^r, g^{r_2} h_2^r)$.
2. Prover computes $z_1 = r_1 + ex, z_2 = r_2 + ey, z_3 = r + ea$, output the proof $\pi(c_1, c_2) = (z_1, z_2, z_3, e)$.
3. Verifier checks $e \stackrel{?}{=} H(g^{z_1} h_1^{z_3} / c_1^e, g^{z_2} h_2^{z_3} / c_2^e)$.

The switch proof also has perfect completeness, special soundness and honest verifier zero-knowledge.

Proof of Product For three Pedersen commitments $c_1 = g^x h^a$, $c_2 = g^y h^b$ and $c_3 = g^z h^c$, we can prove the product of hidden value ($a \times b = c$), this protocol has been introduced in [14]:

1. Prover generates $r_1, r_2, r_3, r_4, r_5 \in \mathbb{Z}_q^*$ uniformly at random, computes $e = H(g^{r_1} h^{r_2}, g^{r_3} h^{r_4}, g^{r_5} c_1^{r_4})$.
2. Prover computes $z_1 = r_1 + ex, z_2 = r_2 + ea, z_3 = r_3 + ey, z_4 = r_4 + eb, z_5 = r_5 + e(z - xb)$, output the proof $\pi(c_1, c_2, c_3) = (z_1, z_2, z_3, z_4, z_5, e)$.
3. Verifier checks $e \stackrel{?}{=} H(g^{z_1} h^{z_2} / c_1^e, g^{z_3} h^{z_4} / c_2^e, g^{z_5} c_1^{z_4} / c_3^e)$.

The proof of product also has perfect completeness, special soundness and honest verifier zero-knowledge. Specially, when the validity proof of commitments $c_1 = g^x h^a$, $c_2 = g^y h^b$ already exists, then we can omit the generation of r_1, r_2, z_1, z_2 to cut down the proof size. In section 4, we propose new proof of product and division with different basis to realize privacy calculation under different regulators.

2.3 Traceable Range proofs

We introduce Traceable Borromean Range Proof (TBoRP) and Traceable Bulletproofs Range Proof (TBuRP) in appendix A, these works come from [6]. The classic range proofs used in Monero are omitted here for brevity, readers may refer to [10, 1] for the detailed descriptions. It should be mentioned that both

TBoRP and TBuRP have completeness, soundness, zero-knowledge and traceability. For TBoRP, it can be modified to achieve security against malicious regulator by usage of mirror commitment (TBoRP'), which is also introduced in appendix A.

3 Multi-currency Regulation and Regulatable Private Assets Transfer

In this section, we give the applications of multi-currency regulation and regulatable private assets transfer, by usage of TBoRP and switch proof.

Assume there are m different regulatable cryptocurrencies $\mathcal{C}_1, \cdots, \mathcal{C}_m$ (using Pedersen commitment to hide amount) in the blockchain, as well as m different regulators $\mathcal{R}_1, \cdots, \mathcal{R}_m$ to trace the amount of each cryptocurrency respectively. We give the construction of multi-currency regulation in the following:

- $\mathsf{Par} \leftarrow \mathsf{Setup}(\lambda)$: system chooses elliptic curve $\mathbb{G}$ and a generator $g \in \mathbb{G}$. For each cryptocurrency $\mathcal{C}_i$, $i = 1, \cdots, m$, regulator $\mathcal{R}_i$ generates $y_i \in \mathbb{Z}_q^*$ as the trapdoor, computes $h_i = g^{y_i}$, system outputs $(\mathbb{G}, q, g, h_1, \cdots, h_m)$ as the public parameters.
- $(c_i, \pi_{TBo}(c_i)) \leftarrow \mathsf{Gen}(\mathsf{Par}, a_i)$: for user Alice who possesses money $\mathcal{C}_i$ with amount a_i, she generates the blinding element x_i and computes the corresponding commitment $c_i = g^{x_i} h_i^{a_i}$, and runs TBoRP to get the traceable range proof $\pi_{TBo}(c_i)$ for c_i.
- $(c_j, \pi_{TBo}(c_j), \pi_{Swit}(c_i^\gamma, c_j)) \leftarrow \mathsf{Transfer}(c_i, \gamma)$: Assume Alice wants to transfer her money from $\mathcal{C}_i$ to $\mathcal{C}_j$, with ratio $1 : \gamma$, Alice finishes the transfer as follows:
  1. Alice computes the new amount $a_j = a_i \cdot \gamma$;
  2. Alice generates the new blinding element x_j and computes the new commitment $c_j = g^{x_j} h_j^{a_j}$;
  3. Alice computes the switch proof $\pi_{Swit}(c_i^\gamma, c_j)$ between c_i^γ and c_j;
  4. Alice runs TBoRP to get the traceable range proof $\pi_{TBo}(c_j)$ for c_j.
- $1/0 \leftarrow \mathsf{Verify}(c_i, c_j, \pi_{TBo}(c_j), \pi_{Swit}(c_i^\gamma, c_j))$:
  1. Verifier computes c_i^γ and checks the validity of switch proof $\pi_{Swit}(c_i^\gamma, c_j)$;
  2. Verifier checks the validity of traceable range proof $\pi_{TBo}(c_j)$;
  3. If all passed then outputs 1, removes the money $(c_i, \pi_{TBo}(c_i))$ and adds the new money $(c_j, \pi_{TBo}(c_j))$ to the blockchain, otherwise outputs 0.
- $a_j^* \leftarrow \mathsf{Trace}(c_j, y, \pi_{TBo}(c_j))$: the regulator $\mathcal{R}_j$ runs the Trace algorithm to trace the amount of the new money c_j.

Theorem 5 (Correctness) *Alice can run the above scheme to transfer her money from $\mathcal{C}_i$ to $\mathcal{C}_j$, with ratio $1 : \gamma$, and the regulator can recover her amount correctly.*

Proof. According to the original amount a_i of $\mathcal{C}_i$ and transfer ratio γ, we know

$$c_i^\gamma = (g^{x_i} h_i^{a_i})^\gamma = g^{x_i \gamma} h_i^{a_i \gamma} = g^{x_i \gamma} h_i^{a_j}.$$

Then Alice can run the switch proof to prove that $c_j = g^{x_j} h_j^{a_j}$ and $g^{x_i \gamma} h_i^{a_j}$ have the same hidden value ($a_j = a_j$), then Alice can compute the traceable range proof for c_j and finishes the transfer.

According to the correctness of switch proof and TBoRP, and the traceability of TBoRP, we know that the proofs can pass the verification, and the regulator can trace the amount. □

Theorem 6 (Security) *The scheme of multi-currency regulation and regulatable private assets transfer is secure (completeness, soundness and zero-knowledge) for any PPT adversary (without possession of trapdoors).*

Proof. The security of the scheme follows from the completeness, soundness and zero-knowledge of switch proof and TBoRP, we omit it for brevity. □

It should be noted that our scheme is also secure against malicious regulator when use TBoRP' as component in construction, we give the detailed description of the modified scheme in appendix B. Meanwhile, the above scheme is also suitable for TBuRP, with different trapdoor generation algorithm, which will be introduced in appendix C.

4 Auxiliary Privacy Calculation

In this section we introduce the application on calculation of privacy amounts from Pedersen commitments. First, we give the construction to realize calculation for $(+, -, \times, \div)$ of Pedersen commitments with different commitment basis, that is to say, for $c_1 = g^x h_1^a$ and $c_2 = g^y h_2^b$, we give the new commitments $c_3 = g^x h_1^c$ and proofs for $a \pm b$, $a \times b$ and $a \div b$, where $h_1 \neq h_2$. Moreover, we give the construction of auxiliary privacy calculation in which the calculation is done by auxiliary party who does not know the calculation results, the auxiliary privacy calculation is suitable for scenarios such as international trades, taxation and calculation of interests.

4.1 Privacy Calculation From Commitments

For Alice's different commitments $c_1 = g^x h_1^a$ and $c_2 = g^y h_2^b$ with hidden amounts a and b regulated by $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively, Alice wants to computes the new commitments for $a \pm b, a \times b$ and $a \div b$, we give the proof of $+, -, \times, \div$ with different basis, where the $+, -, \times, \div$ is calculations in $\mathbb{Z}_q$.

Proof of Sum (Difference) For $c_1 = g^x h_1^a$ and $c_2 = g^y h_2^b$, Alice needs to generate a new commitment $c_3 = g^z h_1^c$, where $c = a + b$ and give the zero-knowledge proof π_{Sum} of the calculation, she does as follows:

1. Alice samples $z \leftarrow \mathbb{Z}_q^*$ and computes $c_3 = g^z h_1^c$, where $c = a + b$;
2. Alice generates the switch proof $\pi_{Sum}(c_1, c_2, c_3) = \pi_{Swit}(c_2, c_3/c_1)$ for c_2 and c_3/c_1;

3. Verifier checks the validity of switch proof, if passed then outputs 1, otherwise outputs 0.

It is easy to see that $c_3/c_1 = g^{z-x}h_1^{c-a} = g^{z-x}h_1^b$, then we get the correctness and security of the above scheme from the completeness, soundness and zero-knowledge of switch proof. The construction of proof of difference is the same as proof of sum, we omit it for brevity.

Theorem 7 *The proof of sum (difference) has correctness, completeness, soundness, zero-knowledge for any PPT adversary.*

Proof of Product For $c_1 = g^x h_1^a$ and $c_2 = g^y h_2^b$, Alice needs to generate a new commitment $c_3 = g^z h_1^c$, where $c = ab$ and give the zero-knowledge proof of the calculation, we extend the proof of product in section 2.2 to realize the multiplication between different basis:

1. Alice samples $z \leftarrow \mathbb{Z}_q^*$ and computes $c_3 = g^z h_1^c$, where $c = ab$;
2. Alice generates $r_1, r_2, r_3, r_4, r_5 \in \mathbb{Z}_q^*$ uniformly at random, computes $e = H(g^{r_1}h_1^{r_2}, g^{r_3}h_2^{r_4}, g^{r_5}c_1^{r_4})$;
3. Alice computes $z_1 = r_1 + ex, z_2 = r_2 + ea, z_3 = r_3 + ey, z_4 = r_4 + eb, z_5 = r_5 + e(z - xb)$, output the proof $\pi_{Prod}(c_1, c_2, c_3) = (z_1, z_2, z_3, z_4, z_5, e)$;
4. Verifier checks $e \stackrel{?}{=} H(g^{z_1}h_1^{z_2}/c_1^e, g^{z_3}h_2^{z_4}/c_2^e, g^{z_5}c_1^{z_4}/c_3^e)$, if passed then outputs 1, otherwise outputs 0.

Similar to section 2.2, when the validity proof of commitments $c_1 = g^x h_1^a$, $c_2 = g^y h_2^b$ already exists, then we can omit the generation of r_1, r_2, z_1, z_2 to cut down the proof size. The correctness and security of π_{Prod} easily follows the discussion of [14].

Theorem 8 *The proof of product has correctness, completeness, soundness, zero-knowledge for any PPT adversary.*

Proof of Quotient We construct proof of quotient scheme by usage of proof of product for twice. For $c_1 = g^x h_1^a$ and $c_2 = g^y h_2^b$, Alice needs to generate a new commitment $c_3 = g^z h_1^c$, where $c = a/b$ and give the zero-knowledge proof π_{Quot} of the calculation.

1. Alice samples $z \leftarrow \mathbb{Z}_q^*$ and computes $c_3 = g^z h_1^c$, where $c = a/b = ab^{-1}$;
2. Alice samples $w, v \leftarrow \mathbb{Z}_q^*$, computes $c_4 = g^w h_2^{b^{-1}}$ and $c_5 = g^v h_2^1$;
3. Alice computes the proof of product $\pi_{Prod}(c_2, c_4, c_5)$;
4. Alice generates $r_1 \leftarrow \mathbb{Z}_q^*$ and computes $e = H(g^{r_1})$;
5. Alice computes $z_1 = r_1 + ev$;
6. Alice computes the proof of product $\pi_{Prod}(c_1, c_4, c_3)$;
7. Alice outputs the proof of quotient $\pi_{Quot}(c_1, c_2, c_3) = (c_4, c_5, \pi_{Prod}(c_2, c_4, c_5), \pi_{Prod}(c_1, c_4, c_3), z_1, e)$;

8. Verifier checks the validity of $\pi_{Prod}(c_2, c_4, c_5)$ and $\pi_{Prod}(c_1, c_4, c_3)$, then checks $e \stackrel{?}{=} H(g^{z_1}/(c_5 h_2^{-1})^e)$, if all passed then outputs 1, otherwise outputs 0.

Since $b \cdot b^{-1} = 1$, and $a \cdot b^{-1} = ab^{-1}$, it clear that $\pi_{Prod}(c_2, c_4, c_5), \pi_{Prod}(c_1, c_4, c_3)$ give the right proof of the commitments. The correctness and security of the above scheme can be proved easily from the correctness and security of product proof π_{Prod}.

Theorem 9 *The proof of quotient has correctness, completeness, soundness, zero-knowledge for any PPT adversary.*

It should be noted that for proofs of $+, -, \times, \div$, Alice should compute the traceable range proof $\pi_{TRP}(c_3)$ after she gets the calculation results to ensure the traceability of the calculations. Alice may choose TBoRP (TBoRP' or TBuRP) as π_{TRP} in the above schemes, and TBoRP' can make the schemes secure against malicious regulators.

Theorem 10 *By adding traceable range proof, the π_{Sum}, π_{Diff}, π_{Prod}, π_{Quot} realize traceability for any PPT adversary.*

4.2 Auxiliary Privacy Calculation

In this subsection we introduce another type of privacy calculation, the auxiliary privacy calculation. The main difference between ordinary privacy calculation and auxiliary privacy calculation is that in stead of Alice, the calculation is operated by an auxiliary party $\mathcal{A}$, who possesses part of the private value in the calculation, such as number of goods, exchange ratio and interest rate, $\mathcal{A}$ can directly compute the result without waiting for communication with Alice.

In the construction, assume Alice possesses private value a and publishes $c_1 = g^x h_1^a$ in the blockchain, with c_1's traceable range proof $\pi_{TRP}(c_1)$, while $\mathcal{A}$ possesses another private value b (regulated by a different regulator) and publishes $c_2 = g^y h_2^b$ in the blockchain, with c_2's traceable range proof $\pi_{TRP}(c_2)$. $\mathcal{A}$ need to computes commitment for $a \pm b$, ab, a/b and gives the proof of calculation respectively. We introduce the construction of auxiliary privacy calculation in the following.

Proof of Auxiliary Add (Subtraction) For $c_1 = g^x h_1^a$ (a is possessed by Alice) and $c_2 = g^y h_2^b$ (b is possessed by $\mathcal{A}$), $\mathcal{A}$ needs to generate a new commitment c_3 for $c = a + b$ and give the zero-knowledge proof π_{Add} of the calculation, he does as follows (proof of auxiliary subtraction is similar and is omitted):

1. $\mathcal{A}$ samples $z \leftarrow \mathbb{Z}_q^*$ and computes $c_4 = g^z h_1^b$;
2. $\mathcal{A}$ computes the switch proof $\pi_{Swit}(c_2, c_4)$;
3. $\mathcal{A}$ computes $c_3 = c_1 c_4$;
4. $\mathcal{A}$ outputs the proof of auxiliary add $\pi_{Add}(c_1, c_2, c_3) = \pi_{Swit}(c_2, c_3/c_1)$;

5. Verifier checks the validity of switch proof, if passed then outputs 1, otherwise outputs 0.

Since $c_3 = c_1 c_4 = g^{x+z} h_1^{a+b}$, then we get the correctness and security of the above scheme from the correctness and security of π_{Swit}.

Theorem 11 *The proof of auxiliary add has correctness, completeness, soundness, zero-knowledge for any PPT adversary.*

Proof of Auxiliary Multiplication For $c_1 = g^x h_1^a$ (a is possessed by Alice) and $c_2 = g^y h_2^b$ (b is possessed by $\mathcal{A}$), $\mathcal{A}$ needs to generate a new commitment $c_3 = g^z h_1^c$ for $c = ab$ and give the zero-knowledge proof π_{Mult} of the calculation, he does as follows:

1. $\mathcal{A}$ samples $z \leftarrow \mathbb{Z}_q^*$ and computes $c_3 = g^z c_1^b$;
2. $\mathcal{A}$ computes the switch proof for c_2 and c_3 (with basis h_2 and c_1), that is, $\mathcal{A}$ generates $r_1, r_2, r_3 \in \mathbb{Z}_q^*$ uniformly at random, computes $e = H(g^{r_1} h_2^{r_2}, g^{r_3} c_1^{r_2})$;
3. $\mathcal{A}$ computes $z_1 = r_1 + ey, z_2 = r_2 + eb, z_3 = r_3 + ez$, output the proof of auxiliary multiplication $\pi_{Mult}(c_1, c_2, c_3) = \pi_{Swit}(c_2, c_3) = (z_1, z_2, z_3, e)$;
4. Verifier checks $e \stackrel{?}{=} H(g^{z_1} h_2^{z_2} / c_2^e, g^{z_3} c_1^{z_2} / c_3^e)$, if passed then outputs 1, otherwise outputs 0.

Since $c_3 = g^z c_1^b = g^{z+xb} h_1^{ab}$, then we get the correctness and security of π_{Mult} from the correctness and security of π_{Swit}.

Theorem 12 *The proof of auxiliary multiplication has correctness, completeness, soundness, zero-knowledge for any PPT adversary.*

Proof of Auxiliary Division We construct proof of auxiliary division scheme by usage of proof of product π_{Prod} and proof of auxiliary multiplication π_{Mult}. For $c_1 = g^x h_1^a$ (a is possessed by Alice) and $c_2 = g^y h_2^b$ (b is possessed by $\mathcal{A}$), $\mathcal{A}$ needs to generate a new commitment $c_3 = g^z h_1^c$ for $c = a/b$ and give the zero-knowledge proof π_{Div} of the calculation, he does as follows:

1. $\mathcal{A}$ samples $w, v \leftarrow \mathbb{Z}_q^*$, computes $c_4 = g^w h_2^{b^{-1}}$ and $c_5 = g^v h_2^1$;
2. $\mathcal{A}$ computes the proof of product $\pi_{Prod}(c_2, c_4, c_5)$;
3. $\mathcal{A}$ generates $r_1 \leftarrow \mathbb{Z}_q^*$ and computes $e = H(g^{r_1})$;
4. $\mathcal{A}$ computes $z_1 = r_1 + ev$;
5. $\mathcal{A}$ samples $z \leftarrow \mathbb{Z}_q^*$ and computes $c_3 = g^z c_1^{b^{-1}}$;
6. $\mathcal{A}$ computes the proof of auxiliary multiplication $\pi_{Mult}(c_1, c_4, c_3)$;
7. $\mathcal{A}$ outputs the proof of auxiliary division $\pi_{Div}(c_1, c_2, c_3) = (c_4, c_5, \pi_{Prod}(c_2, c_4, c_5), \pi_{Mult}(c_1, c_4, c_3), z_1, e)$;
8. Verifier checks the validity of $\pi_{Prod}(c_2, c_4, c_5), \pi_{Mult}(c_1, c_4, c_3)$, then checks $e \stackrel{?}{=} H(g^{z_1} / (c_5 h_2^{-1})^e)$, if all passed outputs 1, otherwise outputs 0.

Since $b \cdot b^{-1} = 1$ and $c_3 = g^z c_1^{b^{-1}} = g^{z+xb^{-1}} h_1^{ab^{-1}}$, then we get the correctness and security of π_{Div} from the correctness and security of π_{Prod} and π_{Mult}.

Theorem 13 *The proof of auxiliary division has correctness, completeness, soundness, zero-knowledge for any PPT adversary.*

It should be noted that, for proofs of auxiliary calculations (add, subtraction, multiplication, division), the auxiliary party $\mathcal{A}$ does not know the hidden value a from c_1, and he does not know the calculation result c. In the application of regulatable blockchain, the hidden values a and b are known by the designated regulators (a for $\mathcal{R}_A$ and b for $\mathcal{R}_B$), then the calculation results can be traced by MPC protocol or other secure channel between $\mathcal{R}_A$ and $\mathcal{R}_B$, which realize the functionality of regulation.

5 Secure Joint Regulation

The original traceable range proofs (TBoRP and TBuRP[6]) only support single regulator for each cryptocurrency, the transaction amounts will be leaked if the regulator is attacked or malicious, a natural solution to fill the gap is to use joint regulation by multiple regulators to trace the transaction amounts jointly, any number of regulators (less than the threshold) cannot trace any bit of the amount, that brings more solid security to the traceable range proof and regulatable blockchains. In this section we introduce a modification for TBuRP (which named JTBuRP) to realize the functionality of joint regulation.

5.1 Construction of JTBuRP

To describe the JTBuRP scheme more clearly, we use a example of parameters in the construction, in fact, the number of amount bits, number of regulators, number of trapdoors, number of threshold are not restricted, anyone can change the parameter selection due to different applications and regulatory policies. In this paper, we set the amount bits to be $n = 32$, assume there are 4 regulators $\mathcal{R}_A, \mathcal{R}_B, \mathcal{R}_C, \mathcal{R}_D$, each regulator generates 2 trapdoors, each trapdoor is related with 4 bits, the regulation threshold is 4 (that is, for regulators with number less than 4, they cannot trace any bit of the amount). Our example is equally distributed for regulators and trapdoors with bit partition $P = ((4,4),(4,4),(4,4),(4,4))$, it should be noted that unequal distribution also applies to our scheme, we omit it due to its complicated expression.

- $\mathsf{Par} \leftarrow \mathsf{Setup}(\lambda)$:
  1. System chooses elliptic curve $\mathbb{G}$ and generators $g, h, g_0, \cdots, g_{n-1} \in \mathbb{G}$, where $n = 32$;
  2. $\mathcal{R}_A$ generates $y_0, y_1 \in \mathbb{Z}_q^*$, $\mathcal{R}_B$ generates $y_2, y_3 \in \mathbb{Z}_q^*$, $\mathcal{R}_C$ generates $y_4, y_5 \in \mathbb{Z}_q^*$, $\mathcal{R}_D$ generates $y_6, y_7 \in \mathbb{Z}_q^*$ as their trapdoors respectively;
  3. All regulators compute $h_i = g_i^{y_{\lfloor i/4 \rfloor}}$ for $i = 0, \cdots, n-1$;
  4. System outputs $(\mathbb{G}, q, g, h, \mathbf{g}, \mathbf{h}, P)$ as the public parameters, where $\mathbf{g} = (g_0, \cdots, g_{n-1}) \in \mathbb{G}^n, \mathbf{h} = (h_0, \cdots, h_{n-1}) \in \mathbb{G}^n$ and $P = ((4,4),(4,4),(4,4),(4,4))$ is the partition of bits.

– $(A, S, c, \{TK_i\}_{i=0,\cdots,19}, \pi(TK_0, \cdots, TK_{19}, A)) \leftarrow \mathsf{Gen}(\mathsf{Par}, a)$:
  1. According to the public parameters $(\mathbb{G}, q, g, h, \mathbf{g}, \mathbf{h}, P)$ and amount $a \in [0, 2^n - 1]$, prover Alice samples $x \in \mathbb{Z}_q^*$ uniformly, computes $c = h^x g^a$ as the commitment;
  2. Alice computes the binary expansion $a = a_0 + \cdots + 2^{n-1}a_{n-1}, a_i = 0, 1$ for $i = 0, \cdots, n-1$, sets $\mathbf{a}_L = (a_0, \cdots, a_{n-1})$;
  3. Alice computes $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n = (a_0 - 1, \cdots, a_{n-1} - 1)$;
  4. Alice samples $\alpha \in \mathbb{Z}_q$ uniformly at random, computes
$$A = h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} = h^\alpha g_0^{a_0} \cdots g_{n-1}^{a_{n-1}} h_0^{a_0 - 1} \cdots h_{n-1}^{a_{n-1} - 1};$$
  5. Alice samples $\mathbf{s}_L, \mathbf{s}_R \in \mathbb{Z}_q^n, \rho \in \mathbb{Z}_q$ uniformly at random, computes $S = h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}$;
  6. For every $j = 0, \cdots, 15$, Alice computes $TK_j = g_{2j}^{\alpha - a_{2j}} g_{2j+1}^{\alpha - a_{2j+1}}$, then computes $TK_{16+k} = \prod_{i=0}^{3} (h_{8i+2k}^{-\alpha - a_{8i+2k}+1} h_{8i+2k+1}^{-\alpha - a_{8i+2k+1}+1})$ for $k = 0, \cdots, 3$, the number of TK_is is 20;
  7. Alice gives the validity proof $\pi(TK_0, \cdots, TK_{19}, A)$ for all TK_is that TK_j is a production of g_{2j}'s power and g_{2j+1}'s power for $j = 0, \cdots, 15$, TK_{16+k} is a production of $\{h_{8i+2k}\}_{i=0,\cdots,3}$'s power and $\{h_{8i+2k+1}\}_{i=0,\cdots,3}$'s power for $k = 0, \cdots, 3$, and $A \cdot \prod_{i=0}^{19} TK_i = (h \prod g_i / \prod h_i)^\alpha$ is a power of $h \prod g_i / \prod h_i$;
  8. Alice outputs $(A, S, c, \{TK_i\}_{i=0,\cdots,19}, \pi(TK_0, \cdots, TK_{19}, A))$.

– $(T_1, T_2, \tau_x, \mu, t, \mathbf{l}, \mathbf{r}) \leftarrow \mathsf{Prove}(A, S, c, \{TK_i\}_{i=0,\cdots,19}, \pi(TK_0, \cdots, TK_{19}, A))$:
  1. Prover sends $(A, S, c, \{TK_i\}_{i=0,\cdots,19}, \pi(TK_0, \cdots, TK_{19}, A))$ to verifier;
  2. Verifier samples $y, z \in \mathbb{Z}_q$ uniformly at random, and sends them to prover;
  3. Prover computes T_1, T_2 and sends them to verifier;
  4. Verifier samples $x \in \mathbb{Z}_q$ uniformly at random, and sends it to prover;
  5. Prover computes $\tau_x, \mu, t, \mathbf{l}, \mathbf{r}$ and sends them to verifier.

– $1/0 \leftarrow \mathsf{Verify}$: we only introduce the verification of $\pi(TK_0, \cdots, TK_{19}, A)$:
  1. For every $i = 0, \cdots, 19$, verifier checks the validity of TK_i;
  2. Verifier computes $A \cdot \prod_{i=0}^{19} TK_i$ and checks the validity of $A \cdot \prod_{i=0}^{19} TK_i$;
  3. Verifier continues the rest verification of Bulletproofs;
  4. If all passed then outputs 1, otherwise outputs 0.

– $a^* \leftarrow \mathsf{Trace}(\{TK_i\}_{i=0,\cdots,19}, y_0, \cdots, y_7)$:
  1. For every $j = 0, \cdots, 15$, all regulators compute $T_j = TK_j^{y_{\lfloor j/2 \rfloor}}$;
  2. All regulators compute and search for $d_i \in \{-1, 1\}$ such that $TK_{16} \cdot T_0 T_4 T_8 T_{12} = h_0^{d_0} h_1^{d_1} h_8^{d_8} h_9^{d_9} h_{16}^{d_{16}} h_{17}^{d_{17}} h_{24}^{d_{24}} h_{25}^{d_{25}}$, output $a_i^* = \frac{1}{2} - \frac{1}{2} d_i$ for $i = 0, 1, 8, 9, 16, 17, 24, 25$;
  3. All regulators compute and search for $d_i \in \{-1, 1\}$ such that $TK_{17} \cdot T_1 T_5 T_9 T_{13} = h_2^{d_2} h_3^{d_3} h_{10}^{d_{10}} h_{11}^{d_{11}} h_{18}^{d_{18}} h_{19}^{d_{19}} h_{26}^{d_{26}} h_{27}^{d_{27}}$, output $a_i^* = \frac{1}{2} - \frac{1}{2} d_i$ for $i = 2, 3, 10, 11, 18, 19, 26, 27$;
  4. All regulators compute and search for $d_i \in \{-1, 1\}$ such that $TK_{18} \cdot T_2 T_6 T_{10} T_{14} = h_4^{d_4} h_5^{d_5} h_{12}^{d_{12}} h_{13}^{d_{13}} h_{20}^{d_{20}} h_{21}^{d_{21}} h_{28}^{d_{28}} h_{29}^{d_{29}}$, output $a_i^* = \frac{1}{2} - \frac{1}{2} d_i$ for $i = 4, 5, 12, 13, 20, 21, 28, 29$;

5. All regulators compute and search for $d_i \in \{-1, 1\}$ such that $TK_{19} \cdot T_3T_7T_{11}T_{15} = h_6^{d_6}h_7^{d_7}h_{14}^{d_{14}}h_{15}^{d_{15}}h_{22}^{d_{22}}h_{23}^{d_{23}}h_{30}^{d_{30}}h_{31}^{d_{31}}$, output $a_i^* = \frac{1}{2} - \frac{1}{2}d_i$ for $i = 6, 7, 14, 15, 22, 23, 30, 31$;
6. All regulators output $a^* = a_0^* + \cdots + 2^{n-1}a_{n-1}^*$.

It should be noted that in the Trace algorithm, all regulators must participate in the computation, they can use MPC protocol, secure conference or simply send T_is to the regulatory center to fulfill this task.

Since $T_j = TK_j^{y_{\lfloor j/2 \rfloor}} = g_{2j}^{(\alpha - a_{2j})y_{\lfloor j/2 \rfloor}} g_{2j+1}^{(\alpha - a_{2j+1})y_{\lfloor j/2 \rfloor}} = h_{2j}^{\alpha - a_{2j}} h_{2j+1}^{\alpha - a_{2j+1}}$, we know that for $k = 0, 1, 2, 3$,

$$TK_{16+k} \cdot \prod_{i=0}^{3} T_{4i+k} = \prod_{i=0}^{3} (h_{8i+2k}^{-\alpha - a_{8i+2k} + 1 + \alpha - a_{8i+2k}} h_{8i+2k+1}^{-\alpha - a_{8i+2k+1} + 1 + \alpha - a_{8i+2k+1}})$$

$$= \prod_{i=0}^{3} (h_{8i+2k}^{-2a_{8i+2k}+1} h_{8i+2k+1}^{-2a_{8i+2k+1}+1}) = \prod_{i=0}^{3} (h_{8i+2k}^{d_{8i+2k}} h_{8i+2k+1}^{d_{8i+2k+1}}).$$

Then we have $a_j = \frac{1}{2} - \frac{1}{2}d_j \in \{0, 1\}$ for $j = 0, \cdots, n-1$, and we get the correctness of JTBuRP.

Theorem 14 (Correctness) *The privacy amounts in JTBuRP scheme can be correctly traced by regulators jointly.*

The proof security for JTBuRP is similar to TBuRP in [6], and the proof of zero-knowledge is in the next subsection.

Theorem 15 (Security) *The JTBuRP scheme has completeness, soundness for any PPT adversary and has traceability for any PPT adversary without possession of trapdoors.*

5.2 Threshold Analysis of JTBuRP

In this subsection we consider the regulatory threshold of JTBuRP, that is, any number (less than the threshold) of regulators can not recover any bit of the amount from JTBuRP, this strengthen the security of JTBuRP when regulators are attacked, corrupted or collusive. Meanwhile, this feature also applies to scenarios such as multi-regulator committee, international organization, joint regulation between branches, etc. It is a new approach to realize the decentralized regulation.

From the example of JTBuRP in section 5.1, we know that $\mathcal{R}_A$ possesses $T_0 \to T_3$, $\mathcal{R}_B$ possesses $T_4 \to T_7$, $\mathcal{R}_C$ possesses $T_8 \to T_{11}$, $\mathcal{R}_D$ possesses $T_{12} \to T_{15}$, then in the tracing stage, without loss of generality, assume $\mathcal{R}_A$ is absent, $\mathcal{R}_B$, $\mathcal{R}_C$ and $\mathcal{R}_D$ want to recover at least 1 bit of the amount, they run the Trace algorithm to compute d_j for $j = 0, \cdots, n-1$ from $TK_{16+k} \cdot \prod_{i=0}^{3} T_{4i+k}$ which they cannot compute successfully without help of $\mathcal{R}_A$ ($\mathcal{R}_A$ possesses trapdoor y_0, y_1 and can compute $T_0 \to T_3$), then we know that the example scheme has threshold

4 and has zero-knowledge property for any number (< 4) of regulators. Moreover, as mentioned before, the parameters (number of bits, number of regulators, number of trapdoors, amount bit partition and threshold) of JTBuRP can be adjusted to adapt various applications.

Theorem 16 *JTBuRP in section 5.1 has regulatory threshold 4, the transaction amounts are leaked iff all regulators collude. In another word, JTBuRP is zero-knowledge for any PPT adversary (not corrupt all regulators).*

6 Conclusion

In this paper, we give several applications on traceable range proofs (TBoRP, TBuRP) including multi-currency regulation, regulatable private assets transfer, auxiliary privacy calculation and secure joint regulation. We summarize them in the following:

- For the application of multi-currency regulation and regulatable private assets transfer, different regulators generate different trapdoors and public parameters for different cryptocurrencies in one blockchain, users can transfer their money via switch proof and traceable range proof to ensure the correctness, privacy and regulation of the transfer.
- For the application of auxiliary privacy calculation, any user can calculate $(+, -, \times, \div)$ the private amounts from commitments not only by themselves, but also by the auxiliary party who only possesses part of the amounts. We give the zero-knowledge proof protocol to support these calculations, which has potential in international trades, taxation and calculation of interests.
- For the application of secure joint regulation, by modification of TBuRP, we get JTBuRP which ensures any number of regulators (less than the regulatory threshold) cannot recover any bit of transaction amount in the blockchain. JTBuRP realizes the functionality of decentralized regulation, which is suitable for applications such as multi-regulator committee, international organization, etc.

Future Works In the future, we need to study and research in the following aspects:

1. Improve the efficiency of traceable range proof: reduce the proof size, reduce the number of tracing keys, reduce the running time for proving, verifying and tracing;
2. Study new approaches to realize traceability of transaction amount in privacy-preserving blockchains;
3. Research for solutions to transfer assets or exchange money between different cryptocurrencies more efficiently and privately, while under regulation;
4. Research for solutions to realize privacy calculation from commitments by the third party, who acts as the calculation proxy and possesses no privacy values.

References

1. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 315–334. IEEE (2018)
2. Buterin, V., et al.: A next-generation smart contract and decentralized application platform. white paper **3**, 37 (2014)
3. Duffield, E., Diaz, D.: Dash: A privacycentric cryptocurrency. No Publisher (2015)
4. Facebook: Libra white paper. URl: https://libra.org/en-US/white-paper/ (2019)
5. Jedusor, T.E.: Mimblewimble (2016)
6. Li, W., Chen, L., Lai, X., Zhang, X., Xin, J.: Traceable and linkable ring signatures, traceable range proofs and applications on regulatable privacy-preserving blockchains. Cryptology ePrint Archive, Report 2019/925 (2019), https://eprint.iacr.org/2019/925
7. Maxwell, G.: Confidential transactions. URL: https://people. xiph. org/˜ greg/confidential_values. txt (Accessed 09/05/2016) (2015)
8. Maxwell, G., Poelstra, A.: Borromean ring signatures (2015)
9. Nakamoto, S., et al.: Bitcoin: A peer-to-peer electronic cash system (2008)
10. Noether, S., Mackenzie, A., et al.: Ring confidential transactions. Ledger **1**, 1–18 (2016)
11. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual International Cryptology Conference. pp. 129–140. Springer (1991)
12. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE (2014)
13. Van Saberhagen, N.: Cryptonote v 2.0 (2013)
14. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-efficient zksnarks without trusted setup. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 926–943. IEEE (2018)

A Traceable Range Proofs

A.1 Traceable Borromean Range Proof

Traceable Borromean range proof provides the validity proof and traceability of transaction amount ($a \in [0, 2^n - 1]$) by making use of Borromean ring signature and Pedersen commitment:

- $\mathsf{Par} \leftarrow \mathsf{Setup}(\lambda)$: system chooses elliptic curve $\mathbb{G}$ and a generator $g \in \mathbb{G}$, the regulator generates $y \in \mathbb{Z}_q^*$ as the trapdoor, computes $h = g^y$, system outputs $(\mathbb{G}, q, g, h)$ as the public parameters.
- $(L_{PK}, SK, c, \{TK_i\}, \pi(\{c_i\}, \{TK_i\}, \{e_i\})) \leftarrow \mathsf{Gen}(\mathsf{Par}, a)$:
  1. According to the public parameters and amount $a \in [0, 2^n - 1]$, prover Alice samples $x \in \mathbb{Z}_q^*$ uniformly, computes $c = g^x h^a$ as the commitment;
  2. Alice computes the binary expansion $a = a_0 + \cdots + 2^{n-1} a_{n-1}, a_i = 0, 1$ for $i = 0, \cdots, n-1$, samples $x_0, \cdots, x_{n-1}$ uniformly, satisfying $x_0 + \cdots + x_{n-1} = x$;

3. For every $i = 0, \cdots, n-1$, Alice computes $c_i = g^{x_i} h^{2^i a_i}, c'_i = g^{x_i} h^{2^i a_i - 2^i}$, outputs $L^i_{PK} = (c_i, c'_i)$;
4. For every $i = 0, \cdots, n-1$, Alice computes $TK_i = h^{x_i - 2^i a_i}$ and $e_i = H(c_0, \cdots, c_{n-1}, TK_0, \cdots, TK_{n-1}, i)$, gives all TK_i's validity proof $\pi(\{c_i\}, \{TK_i\}, \{e_i\})$ that $\prod_{i=0}^{n-1} TK_i^{e_i}$ is a power of h and $\prod_{i=0}^{n-1}(TK_i \cdot c_i)^{e_i}$ is a power of gh;
5. Alice outputs $(L_{PK} = \{L^0_{PK}, \cdots, L^{n-1}_{PK}\}, c, \{TK_i\}_{i=0,\cdots,n-1}, \pi)$ and retains $(a, SK = (x_0, \cdots, x_{n-1}))$.

– $\sigma \leftarrow \mathsf{Prove}(SK, c, L_{PK})$: Alice runs the Borromean ring signature for $L_{PK} = \{(c_0, c'_0), \cdots, (c_{n-1}, c'_{n-1})\}$, outputs $\sigma \leftarrow \mathsf{Rsign}(SK, c, L_{PK})$.

– $1/0 \leftarrow \mathsf{Verify}(\sigma, c, L_{PK}, \{TK_i\}, \pi(\{c_i\}, \{TK_i\}, \{e_i\}))$:
1. Verifier computes $e_0, \cdots, e_{n-1}$, checks the validity of $\pi(\{c_i\}, \{TK_i\}, \{e_i\})$;
2. Verifier checks $\prod c_i \stackrel{?}{=} c$;
3. For every $i = 0, \cdots, n-1$, verifier checks $c_i / c'_i \stackrel{?}{=} h^{2^i}$;
4. Verifier checks the validity of Borromean ring signature σ, if all passed then outputs 1, otherwise outputs 0.

– $a^* \leftarrow \mathsf{Trace}(\sigma, L_{PK}, y, \{TK_i\}_{i=0,\cdots,n-1})$:
1. For every $i = 0, \cdots, n-1$, regulator computes c_i^y;
2. For every $i = 0, \cdots, n-1$, if $c_i^y = TK_i$ then outputs $a^*_i = 0$, otherwise outputs $a^*_i = 1$;
3. Regulator outputs $a^* = a^*_0 + \cdots + 2^{n-1} a^*_{n-1}$ as the tracing result.

The TK_is validity proof $\pi(\{c_i\}, \{TK_i\}, \{e_i\})$ works as follows:

1. Let $P_1 = \prod_{i=0}^{n-1} TK_i^{e_i}$ and $P_2 = \prod_{i=0}^{n-1}(TK_i \cdot c_i)^{e_i}$, prover generates $r_1, r_2 \in \mathbb{Z}_q^*$, computes $f = H(h^{r_1}, (gh)^{r_2})$, then computes $z_1 = r_1 + f \sum_{i=0}^{n-1} e_i(x_i - 2^i a_i)$, $z_2 = r_2 + f \sum_{i=0}^{n-1} e_i x_i$, outputs the proof is $\pi = (z_1, z_2, f)$.
2. Verifier checks $f \stackrel{?}{=} H(h^{z_1}/(P_1)^f, (gh)^{z_2}/(P_2)^f)$.

A.2 Modification of TBoRP

We introduce TBoRP', a modification of TBoRP to realize soundness, zero-knowledge against malicious regulators, by adding the mirror commitment into the proof.

– $\mathsf{Par} \leftarrow \mathsf{Setup'}(\lambda)$: system chooses elliptic curve $\mathbb{G}$ and generators $g_1, g_2 \in \mathbb{G}$, the regulator generates $y \in \mathbb{Z}_q^*$ as the trapdoor, computes $h = g_2^y$, system outputs $(\mathbb{G}, q, g_1, g_2, h)$ as the public parameters.

– $(L_{PK}, SK, c, d, \{d_i\}, \{TK_i\}, \pi_1, \pi_2) \leftarrow \mathsf{Gen'}(\mathsf{Par}, a)$:
1. According to the public parameters $(\mathbb{G}, q, g_1, g_2, h)$ and amount $a \in [0, 2^n - 1]$, prover Alice samples $x \in \mathbb{Z}_q^*$ uniformly, computes commitment $c = g_1^x h^a$ and the mirror commitment $d = g_2^x h^a$;
2. Alice computes the binary expansion $a = a_0 + \cdots + 2^{n-1} a_{n-1}, a_i = 0, 1$ for $i = 0, \cdots, n-1$, samples $x_0, \cdots, x_{n-1}$ uniformly, satisfying $x_0 + \cdots + x_{n-1} = x$;

3. For every $i = 0, \cdots, n-1$, Alice computes $(c_i = g_1^{x_i} h^{2^i a_i}, c_i' = g_1^{x_i} h^{2^i a_i - 2^i})$ and $d_i = g_2^{x_i} h^{2^i a_i}$, outputs $(L_{PK}^i = (c_i, c_i'), d_i)$;
4. For every $i = 0, \cdots, n-1$, Alice computes $TK_i = h^{x_i - 2^i a_i}$ and $e_i = H(c_0, \cdots, c_{n-1}, d_0, \cdots, d_{n-1}, TK_0, \cdots, TK_{n-1}, i)$, gives validity proof of all TK_i and d_i: $(\pi_1, \pi_2) = \pi(\{c_i\}, \{d_i\}, \{TK_i\}, \{e_i\})$ that π_1 proves the validity of $\{d_i\}$ and π_2 proves the validity of $\{TK_i\}$;
5. Alice outputs $(L_{PK} = \{L_{PK}^0, \cdots, L_{PK}^{n-1}\}, c, d, \{d_i\}, \{TK_i\}, \pi_1, \pi_2)$ and retains $(a, SK = (x_0, \cdots, x_{n-1}))$.

– $\sigma \leftarrow \mathsf{Prove'}(SK, c, L_{PK}, d)$: Alice runs the Borromean ring signature for $L_{PK} = \{(c_0, c_0'), \cdots, (c_{n-1}, c_{n-1}')\}$, outputs $\sigma \leftarrow \mathsf{Rsign}(SK, c, L_{PK}, d)$.
– $1/0 \leftarrow \mathsf{Verify'}(\sigma, c, d, \{d_i\}, L_{PK}, \{TK_i\}, \pi_1, \pi_2)$:
1. Verifier computes $e_0, \cdots, e_{n-1}$ and checks the validity of $\pi_1(\{c_i\}, \{d_i\}, \{e_i\})$;
2. Verifier checks the validity of $\pi_2(\{c_i\}, \{TK_i\}, \{e_i\})$;
3. Verifier checks $\prod c_i \stackrel{?}{=} c$, $\prod d_i \stackrel{?}{=} d$;
4. For every $i = 0, \cdots, n-1$, verifier checks $c_i / c_i' \stackrel{?}{=} h^{2^i}$;
5. Verifier checks the validity of Borromean ring signature σ, if all passed then outputs 1, otherwise outputs 0.

– $a^* \leftarrow \mathsf{Trace'}(\sigma, \{d_i\}, y, \{TK_i\}_{i=0,\cdots,n-1})$:
1. For every $i = 0, \cdots, n-1$, regulator computes d_i^y;
2. For every $i = 0, \cdots, n-1$, if $d_i^y = TK_i$ then outputs $a_i^* = 0$, otherwise outputs $a_i^* = 1$;
3. Regulator outputs $a^* = a_0^* + \cdots + 2^{n-1} a_{n-1}^*$.

Note that $\pi_1(\{c_i\}, \{d_i\}, \{e_i\})$ proves the validity of $\{d_i\}$:

1. Prover computes $R = \frac{\prod c_i^{e_i}}{\prod d_i^{e_i}} = (\frac{g_1}{g_2})^{\sum e_i x_i}$, generates $r \in \mathbb{Z}_q^*$ and computes $f = H((\frac{g_1}{g_2})^r)$;
2. Prover computes $z = r + f \sum e_i x_i$, outputs the proof $\pi_1 = (z, f)$;
3. Verifier checks $f \stackrel{?}{=} H((\frac{g_1}{g_2})^z / R^f)$, if passed then output 1, otherwise outputs 0.

And $\pi_2(\{c_i\}, \{TK_i\}, \{e_i\})$ proves the validity of $\{TK_i\}$, same as the original TBoRP in the appendix A.1.

A.3 Traceable Bulletproofs Range Proof

Traceable Bulletproofs range proof (TBuRP) provides the same functionality as TBoRP, with different construction and more flexible application in joint regulation:

– $\mathsf{Par} \leftarrow \mathsf{Setup}(\lambda)$: system chooses elliptic curve $\mathbb{G}$ and generators $g, h, g_0, \cdots, g_{n-1} \in \mathbb{G}$, the regulator generates $y_0, \cdots, y_{n/2-1} \in \mathbb{Z}_q^*$ as the trapdoors, computes $h_{2i} = g_{2i}^{y_i}, h_{2i+1} = g_{2i+1}^{y_i}, i = 0, \cdots, n/2-1$, system outputs $(\mathbb{G}, q, g, h, \mathbf{g}, \mathbf{h})$ as the public parameters, where $\mathbf{g} = (g_0, \cdots, g_{n-1}) \in \mathbb{G}^n, \mathbf{h} = (h_0, \cdots, h_{n-1}) \in \mathbb{G}^n$.

– $(A, S, c, \{TK_i\}_{i=0,\cdots,n-1}, \pi(TK_0, \cdots, TK_{n-1}, A)) \leftarrow \mathsf{Gen}(\mathsf{Par}, a)$:
  1. According to the public parameters $(\mathbb{G}, q, g, h, \mathbf{g}, \mathbf{h})$ and amount $a \in [0, 2^n - 1]$, prover Alice samples $x \in \mathbb{Z}_q^*$ uniformly, computes $c = h^x g^a$ as the commitment (consistent with Bulletproofs);
  2. Alice computes the binary expansion $a = a_0 + \cdots + 2^{n-1} a_{n-1}, a_i = 0, 1$ for $i = 0, \cdots, n-1$, sets $\mathbf{a}_L = (a_0, \cdots, a_{n-1})$;
  3. Alice computes $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n = (a_0 - 1, \cdots, a_{n-1} - 1)$;
  4. Alice samples $\alpha \in \mathbb{Z}_q$ uniformly at random, computes
$$A = h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} = h^\alpha g_0^{a_0} \cdots g_{n-1}^{a_{n-1}} h_0^{a_0 - 1} \cdots h_{n-1}^{a_{n-1} - 1};$$
  5. Alice samples $\mathbf{s}_L, \mathbf{s}_R \in \mathbb{Z}_q^n, \rho \in \mathbb{Z}_q$ uniformly at random, computes $S = h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}$;
  6. For every $j = 0, \cdots, n/2 - 1$, Alice computes $TK_{2j} = g_{2j}^{\alpha - a_{2j}} g_{2j+1}^{\alpha - a_{2j+1}}$, $TK_{2j+1} = h_{2j}^{-\alpha - a_{2j}+1} h_{2j+1}^{-\alpha - a_{2j+1}+1}$, the number of TK_is is n;
  7. Alice gives the validity proof $\pi(TK_0, \cdots, TK_{n-1}, A)$ of all TK_is that TK_{2j} is a production of g_{2j}'s power and g_{2j+1}'s power, TK_{2j+1} is a production of h_{2j}'s power and h_{2j+1}'s power, and $A \cdot \prod_{i=0}^{n-1} TK_i = (h \prod g_i / \prod h_i)^\alpha$ is a power of $h \prod g_i / \prod h_i$;
  8. Alice outputs $(A, S, c, \{TK_i\}_{i=0,\cdots,n-1}, \pi(TK_0, \cdots, TK_{n-1}, A))$.
– $(T_1, T_2, \tau_x, \mu, t, \mathbf{l}, \mathbf{r}) \leftarrow \mathsf{Prove}(A, S, c, \{TK_i\}_{i=0,\cdots,n-1}, \pi(TK_0, \cdots, TK_{n-1}, A))$:
  1. Alice sends $(A, S, c, \{TK_i\}_{i=0,\cdots,n-1}, \pi(TK_0, \cdots, TK_{n-1}, A))$ to verifier;
  2. Verifier samples $y, z \in \mathbb{Z}_q$ uniformly at random, sends them to Alice;
  3. Alice computes T_1, T_2 and sends them to verifier;
  4. Verifier samples $x \in \mathbb{Z}_q$ uniformly at random, and sends it to Alice;
  5. Alice computes $\tau_x, \mu, t, \mathbf{l}, \mathbf{r}$ and sends them to verifier.
– $1/0 \leftarrow \mathsf{Verify}$: we only introduce the verification of $\pi(TK_0, \cdots, TK_{n-1}, A)$:
  1. For every $i = 0, \cdots, n-1$, verifier computes $A \cdot \prod_{i=0}^{n-1} TK_i$ and checks the validity of $A \cdot \prod_{i=0}^{n-1} TK_i$ and TK_i;
  2. Verifier continues the rest verification of Bulletproofs;
  3. If all passed then outputs 1, otherwise outputs 0.
– $a^* \leftarrow \mathsf{Trace}(\{TK_i\}_{i=0,\cdots,n-1}, y_0, \cdots, y_{n/2-1})$:
  1. For every $j = 0, \cdots, n/2 - 1$, regulator computes $TK_{2j+1} \cdot TK_{2j}^{y_j}$;
  2. If $TK_{2j+1} \cdot TK_{2j}^{y_j} = h_{2j} h_{2j+1}$, then outputs $(a_{2j}^*, a_{2j+1}^*) = (0, 0)$;
  3. If $TK_{2j+1} \cdot TK_{2j}^{y_j} = h_{2j}^{-1} h_{2j+1}$, then outputs $(a_{2j}^*, a_{2j+1}^*) = (1, 0)$;
  4. If $TK_{2j+1} \cdot TK_{2j}^{y_j} = h_{2j} h_{2j+1}^{-1}$, then outputs $(a_{2j}^*, a_{2j+1}^*) = (0, 1)$;
  5. If $TK_{2j+1} \cdot TK_{2j}^{y_j} = h_{2j}^{-1} h_{2j+1}^{-1}$, then outputs $(a_{2j}^*, a_{2j+1}^*) = (1, 1)$;
  6. Regulator outputs $a^* = a_0^* + \cdots + 2^{n-1} a_{n-1}^*$.

The computation of $T_1, T_2, \tau_x, \mu, t, \mathbf{l}, \mathbf{r}$ as well as the verification algorithm is same as Bulletproofs, and is omitted for brevity, please refer to [1] for detailed description. The number of trapdoors and tracing keys can be selected flexibly in different scenarios. The description of TBuRP and JTBuRP in this paper are interactive protocols, the non-interactive version of TBURP and JTBuRP can be easily derived by Fiat-Shamir transformation, similar to Bulletproofs.

B Multi-currency Regulation From TBoRP'

We give the introduction of multi-currency regulation scheme from TBoRP', which makes the scheme secure against malicious regulator.

- $\mathsf{Par} \leftarrow \mathsf{Setup}(\lambda)$: system chooses elliptic curve $\mathbb{G}$ and generators $g_1, g_2 \in \mathbb{G}$, for each cryptocurrency $\mathcal{C}_i$, $i = 1, \cdots, m$, regulator $\mathcal{R}_i$ generates $y_i \in \mathbb{Z}_q^*$ as the trapdoor, computes $h_i = g_2^{y_i}$, system outputs $(\mathbb{G}, q, g_1, g_2, h_1, \cdots, h_m)$ as the public parameters.
- $(c_i, d_i, \pi_{TBo'}(c_i, d_i)) \leftarrow \mathsf{Gen}(\mathsf{Par}, a_i)$: for user Alice who possesses money $\mathcal{C}_i$ with amount a_i, she generates the blinding element x_i and compute the corresponding commitment $c_i = g_1^{x_i} h_i^{a_i}$ and $d_i = g_2^{x_i} h_i^{a_i}$, then runs TBoRP' to get the traceable range proof $\pi_{TBo'}(c_i, d_i)$ for c_i and d_i.
- $(c_j, d_j, \pi_{TBo'}(c_j, d_j), \pi_{Swit}(c_i^\gamma, c_j)) \leftarrow \mathsf{Transfer}(c_i, d_i, \gamma)$: Assume Alice wants to transfer her money from $\mathcal{C}_i$ to $\mathcal{C}_j$, with ratio $1 : \gamma$, Alice finishes the transfer as follows:
  1. Alice computes the new amount $a_j = a_i \cdot \gamma$;
  2. Alice generates the new blinding element x_j and computes the new commitment $c_j = g_1^{x_j} h_j^{a_j}$ and $d_j = g_2^{x_j} h_j^{a_j}$;
  3. Alice computes the switch proof $\pi_{Swit}(c_i^\gamma, c_j)$ between c_i^γ and c_j;
  4. Alice runs TBoRP' to get the traceable range proof $\pi_{TBo'}(c_j, d_j)$.
- $1/0 \leftarrow \mathsf{Verify}(c_i, d_i, c_j, d_j, \pi_{TBo'}(c_j, d_j), \pi_{Swit}(c_i^\gamma, c_j))$:
  1. Verifier computes c_i^γ and checks the validity of switch proof $\pi_{Swit}(c_i^\gamma, c_j)$;
  2. Verifier checks the validity of traceable range proof $\pi_{TBo'}(c_j, d_j)$;
  3. If all passed then outputs 1, removes the money c_i and adds the new money $(c_j, d_j, \pi_{TBo'}(c_j, d_j))$ to the blockchain, otherwise outputs 0.
- $a_j^* \leftarrow \mathsf{Trace}(d_j, y_j, \pi_{TBo'}(c_j, d_j))$: the regulator $\mathcal{R}_j$ runs the Trace algorithm to trace the amount of the new money c_j.

C Multi-currency Regulation From TBuRP

We can also use TBuRP to construct multi-currency regulation scheme, with more flexible regulatory policy choices, including the joint regulation.

- $\mathsf{Par} \leftarrow \mathsf{Setup}(\lambda)$: system chooses elliptic curve $\mathbb{G}$ and generators $g, g_0, \cdots, g_{n-1}$, $h \in \mathbb{G}$, for each cryptocurrency $\mathcal{C}_i$, $i = 1, \cdots, m$, regulator $\mathcal{R}_i$ generates $\mathbf{Tr}_i = (y_i^{(0)}, \cdots, y_i^{(n-1)}) \in (\mathbb{Z}_q^*)^n$ as the trapdoors, computes $\mathbf{h}_i = (h_i^{(0)}, \cdots, h_i^{(n-1)}) = (g_0^{y_i^{(0)}}, \cdots, g_{n-1}^{y_i^{(n-1)}})$, system outputs $(\mathbb{G}, q, g, g_0, \cdots, g_{n-1}, h, \mathbf{h}_1, \cdots, \mathbf{h}_m)$ as the public parameters.
- $(c_i, \pi_{TBu}(c_i)) \leftarrow \mathsf{Gen}(\mathsf{Par}, a_i, \mathbf{h}_i)$: for user Alice who possesses money $\mathcal{C}_i$ with amount a_i, she generates the blinding element x_i and compute the corresponding commitment $c_i = h^{x_i} g^{a_i}$, and runs TBuRP to get the traceable range proof $\pi_{TBu}(c_i)$ with basis $\mathbf{h}_i$.
- $(c_j, \pi_{TBu}(c_j)) \leftarrow \mathsf{Transfer}(c_i, \gamma)$: Assume Alice wants to transfer her money from $\mathcal{C}_i$ to $\mathcal{C}_j$, with ratio $1 : \gamma$, Alice finishes the transfer as follows:

1. Alice computes the new amount $a_j = a_i \cdot \gamma$;
2. Alice computes the new blinding element $x_j = x_i \cdot \gamma$ and computes the new commitment $c_j = c_i^{\gamma} = h^{x_j} g^{a_j}$;
3. Alice runs TBuRP to get the traceable range proof $\pi_{TBu}(c_j)$ for c_j with basis $\mathbf{h}_j$.

– $1/0 \leftarrow \mathsf{Verify}(c_i, c_j, \pi_{TBu}(c_j), \gamma)$:

1. Verifier checks $c_i^{\gamma} \stackrel{?}{=} c_j$;
2. Verifier checks the validity of traceable range proof $\pi_{TBu}(c_j)$;
3. If all passed then outputs 1, removes the money $(c_i, \pi_{TBu}(c_i))$ and adds the new money $(c_j, \pi_{TBu}(c_j))$ to the blockchain, otherwise outputs 0.

– $a_j^* \leftarrow \mathsf{Trace}(c_j, \mathbf{Tr}_j, \pi_{TBu}(c_j))$: the regulator $\mathcal{R}_j$ runs the Trace algorithm to trace the amount of the new money c_j.