

Division of Regulatory Power: Collaborative Regulation for Privacy-Preserving Blockchains

Tianyu Zhaolu, Zhiguo Wan, *Member, IEEE*, and Huaqun Wang

Abstract—Decentralized anonymous payment schemes may be exploited for illicit activities, such as money laundering, bribery and blackmail. To address this issue, several regulatory-friendly decentralized anonymous payment schemes have been proposed. However, most of these solutions lack restrictions on the regulator’s authority, which could potentially result in power abuse and privacy breaches. In this paper, we present a decentralized anonymous payment scheme with collaborative regulation (DAPCR). Unlike existing solutions, DAPCR reduces the risk of power abuse by distributing regulatory authority to two entities: Filter and Supervisor, neither of which can decode transactions to access transaction privacy without the assistance of the other one. Our scheme enjoys three major advantages over others: ① Universality, achieved by using zk-SNARK to extend privacy-preserving transactions for regulation. ② Collaborative regulation, attained by adding the ring signature with controllable linkability to the transaction. ③ Efficient aggregation of payment amounts, achieved through amount tags. As a key technology for realizing collaborative regulation in DAPCR, the ring signature with controllable linkability (CLRS) is proposed, where a user needs to specify a linker and an opener to generate a signature. The linker can extract pseudonyms from signatures and link signatures submitted by the same signer based on pseudonyms, without leaking the signer’s identity. The opener can recover the signer’s identity from a given pseudonym. The experimental results reflect the efficiency of DAPCR. The time overhead for transaction generation is 1231.2ms, representing an increase of less than 50% compared to ZETH. Additionally, the time overhead for transaction verification is only 1.2ms.

Index Terms—Ring Signature, Blockchain, Cryptocurrency, Regulation, Decentralized Finance.

I. INTRODUCTION

IN recent years, blockchain technology has had a substantial economic and social impact on the real world. One of the most widely adopted applications is the decentralized

The work was supported in part by the National Natural Science Foundation of China (No. U23B2002, 62272238, 62272425, 61972229); in part by the Foundation of State Key Laboratory of Public Big Data under Grant PBD2023-32; in part by the Key Research Project of Zhejiang Lab (No. 2022PD0AC01); and in part by the Major Basic Research Program of the Shandong Provincial Natural Science Foundation (ZR2020ZD01). (*Corresponding authors: Zhiguo Wan; Huaqun Wang.*)

T. Zhaolu is with the School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: zhaolty@aliyun.com).

Z. Wan is with the Zhejiang Lab, Hangzhou 310000, China (e-mail: wanzhiguo@zhejianglab.com).

H. Wang is with the School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China, and also with the State Key Laboratory of Public Big Data, Guizhou University, Guiyang, Guizhou 550025, China (e-mail: wanghuaqun@aliyun.com).

The full version of this paper: <https://ia.cr/2022/1634>.

payment system, also known as cryptocurrency. In 2021, the total volume of cryptocurrency transactions surged to \$15.8 trillion. However, in contrast to traditional centralized payment mechanisms, decentralized payment systems such as Bitcoin [1] and Ethereum [2] lack support for the privacy preservation of user identities and payment amounts. To address this privacy concern, researchers have proposed decentralized anonymous payment (DAP) systems like Monero, Zerocash and Zether [3]–[5]. In these solutions, the addresses of traders and the specific payment amount for each transaction are kept confidential from other users.

However, providing unconditional privacy in DAP may lead to an increase in criminal activities. Cryptocurrencies could potentially be used for bribery, blackmail, terrorist financing, and money laundering. Chainalysis¹ pointed out that in 2021, cryptocurrency-related criminal cases increased by 79% compared to 2020. Although during the same period, the overall transaction volume grew by over 550%, indicating a decrease in the proportion of illegal activities in the total transactions, this does not imply that regulation is unnecessary. FATF² and APG³ proposed that the absence of regulation has created significant loopholes for criminals, necessitating swift action to mitigate the risks of virtual assets being exploited by criminal and terrorist elements.

Numerous DAP schemes incorporating regulation have been proposed to combat illicit activities within decentralized payment systems. However, unrestricted regulation presents the risk of power abuse and privacy breaches. Therefore, our work focuses on achieving a delicate equilibrium between privacy preservation and regulation. Specifically, to mitigate the potential for regulatory power abuse, regulators should only have access to the sender’s address for suspicious transactions, while ensuring that the privacy information of compliant transactions remains confidential to regulators. Furthermore, it is essential to monitor the total payment amounts conducted by individual users during a designated transaction period. This monitoring becomes necessary as traders might choose to execute multiple smaller transactions rather than a single large transaction when transferring assets.

¹Chainalysis: <https://www.chainalysis.com/>

²Financial Action Task Force: <https://www.fatf-gafi.org/>

³Asia/Pacific Group on Money Laundering: <https://apgml.org/>

TABLE I: Properties of DAPCR and related works

Scheme	Collaborative Regulation	Non-interaction	Amount Aggregation	Universality
DAPCR	✓	✓	✓	✓
[6]	×	✓	×	×
[7]	×	✓	×	×
[8]	✓	✓	×	×
[9]	×	×	×	×
[10]	×	×	×	×

A. Advantages of DAPCR

In this paper, we propose a decentralized anonymous payment scheme with collaborative regulation (DAPCR). The advantages of DAPCR are as follows:

1. **Universality** means that DAPCR can be constructed based on any existing DAP scheme and is compatible with both UTXO and account models. To achieve this universality, we extended the original transactions in a DAP scheme with zk-SNARK. By introducing an additional regulated field to a DAP transaction, Filter and Supervisor can extract regulatory data from the additional field using the same set of algorithms, independently of the original DAP transactions.
2. **Collaborative regulation** means that the regulatory authority in DAPCR is decentralized to Filter and Supervisor, similar to the separation of powers in governments, requiring their cooperation to regulate transactions. In particular, Filter is responsible for linking the transactions submitted by the same signer and extracting the amount tag from each transaction. Supervisor can recover the user's public key from a given pseudonym. Both regulators must collaborate to regulate transactions.
3. **Payment aggregation** means that Filter can aggregate payment amounts from the same user's transactions and get a total payment tag, which is used to determine whether the total payment amount exceeds the user's limit, with Filter do not open transactions during this process.

To clarify the advantages of our scheme, the properties of DAPCR and related works are shown in Table I.

B. Paper Contributions

In summary, our contributions in this paper are as follows.

1. We propose the DAPCR scheme with the following advantages: ① Universality means that DAPCR can be constructed based on any existing DAP scheme and is compatible with both UTXO and account models. ② Collaborative regulation prevents abuse of power and privacy breaches. ③ Payment aggregation enables efficient screening of suspicious transactions. To our best knowledge, DAPCR is the first universal collaborative regulation scheme for DAP schemes.
2. We present security definitions of the DAPCR scheme and provide the security analysis for it.
3. We evaluate the performance of DAPCR on both local devices and the Fabric network. The time cost for transaction generation is about 1231.2 ms and that of transaction verification is about 12.5 ms. These experimental results indicate the effectiveness of DAPCR.
4. We also propose the ring signature with controllable linkability (CLRS), which is a key technology for enabling

collaborative regulation in DAPCR. It allows the designated user to link signatures from the same signer without revealing the signer's identity.

C. Paper Outline

Section II presents an overview of DAPCR. In Section III, we review the background materials associated with our work. Section IV presents the system framework and security definitions of CLRS and DAPCR. Next, we propose an efficient construction of CLRS, which is the building block of DAPCR in Section V. In Section VI, we present an efficient and generic construction of DAPCR. We also provide the security analysis and performance analysis for DAPCR in Section VII and VIII. Section IX reviews the recent literature relevant to DAP and ring signatures. Section X concludes our work.

II. OVERVIEW

To construct a decentralized anonymous payment scheme with regulation that satisfies universality, we extend the existing DAP scheme by adding additional regulated fields. Regulators can use a predefined set of algorithms to extract the necessary data from the regulated field, without considering the original structure of the DAP transaction. Furthermore, to ensure consistency between the regulatory data within a regulated field and the payment data within a DAP transaction, a user needs to generate a zero-knowledge proof to show this. As shown in Fig. 1, the regulated transaction rtx in our scheme consists of a DAP transaction tx , a proof π , and an additional regulated field reg .

To prevent the abuse of power and the leakage of privacy, we decentralize regulatory authority into two regulators: Filter and Supervisor. Both must cooperate to carry out the regulation effectively. Specifically, Filter can extract a pseudonym nym and a tag tag_v of payment amount from the regulated field in a transaction to assess whether the user with pseudonym nym violates the transaction policy, as described in Section IV-A2. If a violation is detected, Filter will submit the user's suspicious transactions and pseudonym nym to Supervisor, who can extract the sender's identity from nym .

As an example, Fig. 1 depicts the workflow of regulation. Initially, Filter retrieves transactions from the blockchain and extracts pseudonyms and amount labels from each transaction (Step ① in Fig. 1). To assess whether the user with pseudonym nym_1 complies with the transaction policy, Filter screens out all transactions from this user based on the pseudonym nym_1 and aggregates the amount labels of these transactions to obtain a total amount label tag_{sum} (Step ②). Filter can then determine whether the user complies with the transaction policy according to tag_v . Then Filter submits the user's pseudonym nym_1 and transactions to Supervisor if the user violates the transaction policy (Step ③). Supervisor will recover the user's identity from nym_1 (Step ④).

Note that Filter cannot obtain identities or payment amounts from pseudonyms and labels, and Supervisor cannot obtain pseudonyms without Filter's assistance. Therefore, if Filter and Supervisor do not collude, payment privacy within compliant

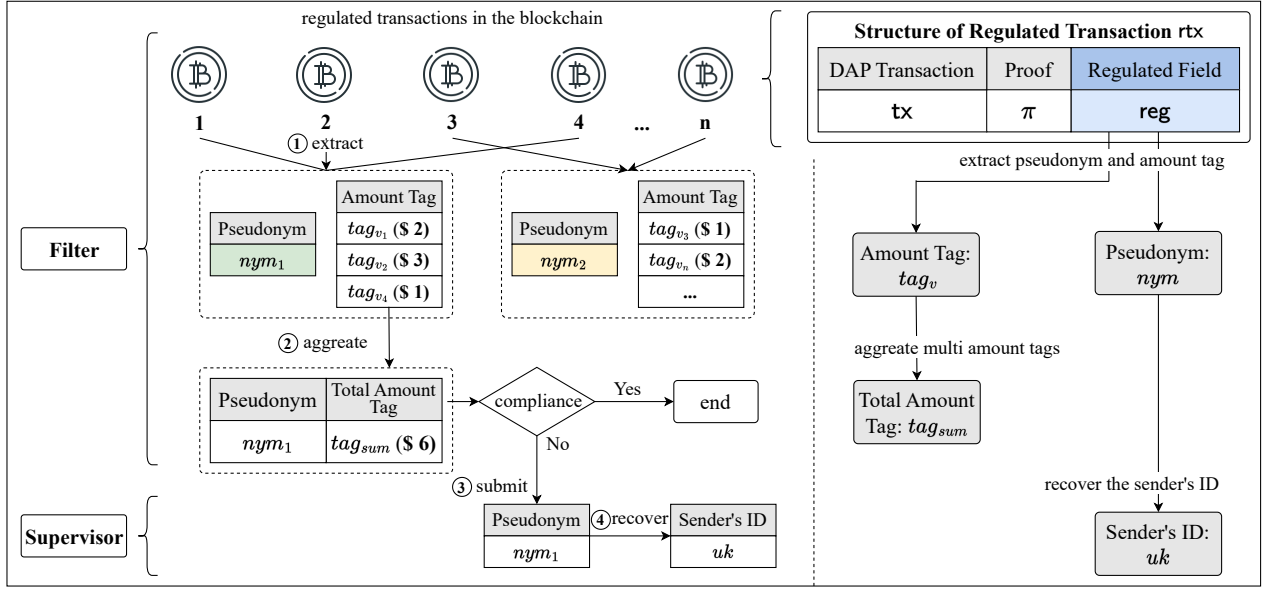


Fig. 1: Overview of DAPCR.

transactions will not be disclosed to either party. The threat model for our scheme is as follows.

Threat model: We assume that each user is selfish and potentially malicious, capable of launching active attacks such as double-spending attacks, malleability attacks, and over-spending attacks to steal assets or provide incorrect regulatory data to regulators. The malicious user also attempts to extract payment privacy within other users' transactions in the decentralized ledger. We also assume that Filter and Supervisor are honest-and-curious. They adhere to the DAPCR scheme and do not actively attack the system. Instead, they will attempt to obtain payment privacy within honest users' transactions. Furthermore, we also assume that Filter and Supervisor cannot collude. We believe this assumption holds in the consortium blockchain, as Filter and Supervisor can be two separate and independent entities participating in managing the consortium blockchain. In real-life scenarios, the law enforcement agency, such as the FBI, serves as Filter, while the judicial department, such as a court, serves as Supervisor. The law enforcement agency collects suspicious transactions from the decentralized ledger and submits them, along with evidence, to the judicial department. After verification, the judicial department removes the anonymity of these transactions and holds the relevant users accountable. It is important to note that these two entities operate independently and do not conspire together.

In the above threat model, an adversary cannot compromise both Filter and Supervisor. The DAPCR scheme prevents privacy leaks by distributing regulatory authority to Filter and Supervisor: ① If Filter is compromised, the adversary can extract users' pseudonyms from transactions but cannot recover public keys from these pseudonyms. ② If Supervisor is compromised, the adversary can only recover the senders' public keys from pseudonyms of suspicious transactions submitted by Filter, but cannot directly revoke the anonymity of transactions on the blockchain. Therefore, whether Filter or Supervisor is compromised, the privacy of transactions that

adhere to the transaction policy will not be breached.

III. PRELIMINARIES

A. Decentralized Anonymous Payments

A DAP scheme in the account model, such as Zether [5], can be highly simplified into algorithms as follows.

- $\text{Setup}(1^\lambda) \rightarrow pp$. This setup algorithm takes a security parameter λ as input and outputs a public parameter pp , which is an implicit input for other algorithms.
- $\text{AddrGen}(pp) \rightarrow (addr, s)$. A user executes this algorithm to generate his address $addr$ and secret key s .
- $\text{TxGen}(addr_S, addr_R, v, s, I_{pub}, I_{pri}) \rightarrow tx$. A user executes this algorithm to generate a DAP transaction tx . The algorithm takes as input a sender's address $addr_S$, a receiver's address $addr_R$, the payment amount v , a secret key s , I_{pub} (which represents additional public inputs) and I_{pri} (which represents additional private inputs), and outputs a transaction tx .
- $\text{TxFy}(tx, I_{pub}) \rightarrow 0/1$. Anyone can execute this algorithm to verify the validity of a transaction tx . The algorithm takes as input a transaction tx and outputs 1 if tx is valid or 0 otherwise.

A secure DAP scheme generally satisfies indistinguishability, non-malleability and balance.

1. **Indistinguishability.** The ledger discloses no information to any adversary attempting to access information beyond what is publicly available.
2. **Non-malleability.** No adversary possesses the capability to modify the information contained within a valid transaction tx .
3. **Balance.** The amount paid by any adversary cannot exceed its balance.

The DAP scheme in the UTXO model, such as Zerocash [4], can also be represented using the above algorithms if only one-to-one transactions are allowed. In Appendix B, we

discuss the minor differences between deploying DAPCR in the UTXO model and deploying it in the account model. In the construction of DAPCR, we overlook these minor differences and use the above algorithms to describe the DAP scheme in either model.

B. NIZK & SoK

1) *NIZK Protocol*: We first present an NP-relation \mathcal{R} defining the language $\mathcal{L}_{\mathcal{R}} = \{\phi | \exists \varpi : (\phi, \varpi) \in \mathcal{R}\}$ in which ϕ and ϖ are considered as a statement and a witness. Non-interactive zero-knowledge protocol, also known as NIZK protocol, for the relation \mathcal{R} is composed of three algorithms as follows [11].

- $\mathcal{G}(1^\lambda, \mathcal{R}) \rightarrow crs$. The setup algorithm takes a security parameter λ and an NP-relation \mathcal{R} as input, and outputs a common reference string crs .
- $\mathcal{P}(\phi, \varpi, crs) \rightarrow \pi$. The prover algorithm takes a statement ϕ , a witness ϖ and a common reference string crs as input, and outputs a proof π .
- $\mathcal{V}(\phi, \pi, crs) \rightarrow 0/1$. The verifier algorithm takes a statement ϕ , a proof π and a common reference string crs as input, and outputs 1 if π is valid or 0 otherwise.

A NIZK scheme satisfies completeness, zero-knowledge and knowledge soundness.

1. *Completeness*. For any $(\phi, \varpi) \in \mathcal{R}$,

$$Pr \left[\begin{array}{l} crs \leftarrow \mathcal{G}(1^\lambda, \mathcal{R}); \pi \leftarrow \mathcal{P}(\phi, \varpi, crs) \\ 1 \leftarrow \mathcal{V}(\phi, \varpi, crs) \end{array} \right] = 1.$$

2. *Zero-knowledge*. No information other than the truth of the statement is leaked. For any $(\phi, \varpi) \in \mathcal{R}$, any probabilistic polynomial-time (PPT) adversary \mathcal{A} and a polynomial-time simulator $\mathcal{S} = (\mathcal{G}_{sim}, \mathcal{P}_{sim})$,

$$\left| Pr \left[\begin{array}{l} (crs, \tau) \leftarrow \mathcal{G}_{sim}(1^\lambda, \mathcal{R}); \pi' \leftarrow \mathcal{P}_{sim}(\phi, crs, \tau) \\ \mathcal{A}(\pi', crs, \tau, \mathcal{R}) = 1 \end{array} \right] - Pr \left[\begin{array}{l} crs \leftarrow \mathcal{G}(1^\lambda, \mathcal{R}); \pi \leftarrow \mathcal{P}(\phi, \varpi, crs) \\ \mathcal{A}(\pi, crs, \tau, \mathcal{R}) = 1 \end{array} \right] \right| \leq \text{negl}(\lambda).$$

3. *Knowledge Soundness*. A secure NIZK scheme cannot prove a false statement. For any PPT adversary \mathcal{A} and a PPT extractor \mathcal{E} ,

$$Pr \left[\begin{array}{l} crs \leftarrow \mathcal{G}(1^\lambda, \mathcal{R}); \\ (\phi, \pi, \varpi) \leftarrow (\mathcal{A} \parallel \mathcal{E})(crs, \mathcal{R}) \\ 1 \leftarrow \mathcal{V}(\phi, \pi, crs) \wedge (\phi, \varpi) \notin \mathcal{R} \end{array} \right] \leq \text{negl}(\lambda).$$

Zk-SNARK, also known as the zero-knowledge succinct non-interactive argument of knowledge, is a special type of NIZK scheme. In addition to the above properties, zk-SNARK also satisfies succinctness: The runtime of a prover algorithm is polynomial in $|a| + \lambda$ and the size of π output by the prover algorithm is polynomial in λ .

2) *SoK protocols*: Signature of knowledge [12], also known as SoK, allows one to issue signatures on behalf of any NP statement. A valid signature of knowledge implies that the signer possesses a witness ϖ to the statement ϕ . A SoK protocol Π_{SoK} consists of three algorithms as follows.

- $\text{SoK.Setup}(1^\lambda, \mathcal{R}) \rightarrow pp$. The setup algorithm takes a security parameter λ and an NP-relation \mathcal{R} as input, and outputs a public parameter pp .
- $\text{SoK.Sign}(m, \phi, \varpi, pp) \rightarrow \pi$. The signing algorithm takes a message m , a statement ϕ , a witness ϖ and the public

parameter pp as input and outputs a signature of knowledge π if $(\phi, \varpi) \in \mathcal{R}$ or \perp otherwise.

- $\text{SoK.Vfy}(m, \phi, \pi, pp) \rightarrow 0/1$. The verifying algorithm takes a message m , a statement ϕ , a signature of knowledge π and the public parameter pp as input, and outputs 1 if π is valid, or 0 otherwise.

A secure SoK satisfies correctness, simulatability and extractability.

1. *Correctness*. For any $(\phi, \varpi) \in \mathcal{R}$,

$$Pr \left[\begin{array}{l} pp \leftarrow \text{Setup}(1^\lambda, \mathcal{R}); \pi \leftarrow \text{Sign}(m, \phi, \varpi, pp) \\ \text{Vfy}(m, \phi, \pi, pp) = 1 \end{array} \right] = 1 - \text{negl}(\lambda).$$

2. *Simulatability*. There are two PPT algorithms SimSetup and SimSign where $\text{SimSetup}(1^\lambda, \mathcal{R})$ outputs the public parameter pp and a trapdoor τ , and $\text{SimSign}(m, \phi, \tau, pp)$ outputs a signature π , such that for any PPT adversary \mathcal{A} ,

$$\left| Pr \left[(pp, \tau) \leftarrow \text{SimSetup}(1^\lambda, \mathcal{R}) : \mathcal{A}^{\mathcal{O}_{sim}}(\mathcal{R}, pp) = 1 \right] - Pr \left[pp \leftarrow \text{Setup}(1^\lambda, \mathcal{R}) : \mathcal{A}^{\text{Sign}}(\mathcal{R}, pp) = 1 \right] \right| \leq \text{negl}(\lambda),$$

where the oracle \mathcal{O}_{sim} receives (m, ϕ, ϖ) as input and outputs $\pi \leftarrow \text{SimSign}(m, \phi, \tau, pp)$ if $(\phi, \varpi) \in \mathcal{R}$ and \perp otherwise.

3. *Extractability*. In addition to algorithms SimSetup and SimSign , there exists an extractor algorithm Ext such that for any PPT adversary \mathcal{A} ,

$$Pr \left[\begin{array}{l} (pp, \tau) \leftarrow \text{SimSetup}(1^\lambda, \mathcal{R}); \\ (m, \phi, \pi) \leftarrow \mathcal{A}^{\mathcal{O}_{sim}}(\mathcal{R}, pp); \varpi \leftarrow \text{Ext}(m, \phi, \pi, \tau, pp) \\ (\phi, \varpi) \in \mathcal{R} \vee (m, \phi, \pi) \in L \vee \text{Vfy}(m, \phi, \pi, pp) = 0 \end{array} \right] = 1 - \text{negl}(\lambda),$$

where L is a list of queries to SimSign .

Note that NIZK and SoK protocols in the random oracle model can be efficiently realized by applying the Fiat-Shamir transform [13] to Σ -protocols.

C. Notations

Notations used in this paper and their descriptions are shown in Table II.

IV. SYSTEM FRAMEWORK AND SECURITY DEFINITIONS

In this section, the system framework and security definitions of DAPCR are going to be introduced.

A. DAP with Collaborative Regulation

1) *System Framework*: An entity that independently regulates the anonymous payment system may abuse regulatory power. To avoid the disadvantage, we divide the regulatory power into two authorities called Filter and Supervisor. Figure 2 depicts the system framework of workflow DAPCR, which consists of five entities as follows:

1. *Blockchain*. The DAPCR scheme is based on a permissioned blockchain, which is only accessed by users with permissions. Supervisor is responsible for registering users who are allowed to join the permissioned blockchain.
2. *User*. A new user needs permission from all consensus nodes to join the consortium blockchain [14]. Each user is

TABLE II: Notations and descriptions

Notation	Description	Notation	Description
\mathcal{F}	Filter	\mathcal{S}	Supervisor
\mathcal{U}	User	(uk, sk)	user's public and private keys
(pk_F, sk_F)	Filter's public and private keys	(pk_O, sk_O)	Supervisor's public and private keys
tx	anonymous transaction in DAP	rtx	regulated transaction in DAPCR
nym	user's pseudonym	tag_μ	tag of user's payment limit
μ	user's payment limit	v_{sum}	user's total payment amount
reg	user's registration data	rpt	report of suspicious transactions
(pk_L, sk_L)	linker's public and private keys	(pk_O, sk_O)	opener's public and private keys
R	set of public keys	σ	signature in CLRS

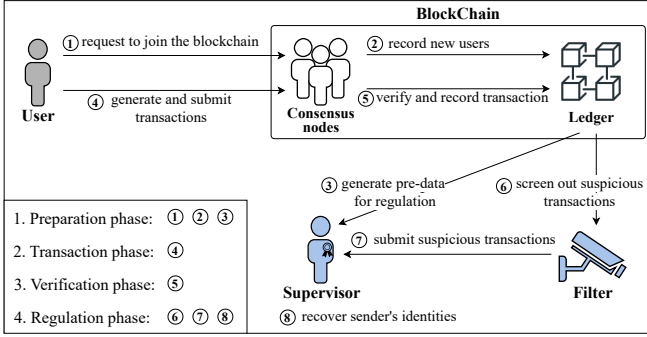


Fig. 2: System framework and workflow.

assigned a payment limit μ , and his total payment amount of anonymous transactions within each trading period should not exceed this limit.

3. *Consensus Nodes* are responsible for checking the validity of transactions and sealing the valid transactions into new blocks.
4. *Filter* has the ability to link transactions from the same sender and determine whether they comply with the transaction policy.
5. *Supervisor* is an entity responsible for registering users and identifying the sender's public key of suspicious transactions with the assistance of \mathcal{S} .

As shown in Fig. 2, the DAPCR scheme consists of four phases. In the preparation phase, the public parameter is published. Entities generate their public and private keys, and new users obtain permission from consensus nodes to join the blockchain. In the transaction phase, users generate regulated transactions to transfer their assets. In the verification phase, consensus nodes verify the validity of regulated transactions, and only valid transactions are sealed into blocks. In the regulation phase, Filter screens out suspicious transactions and submits the report on suspicious transactions to Supervisor. Once the report is received, Supervisor recovers the sender's public key of suspicious transactions.

2) *Transaction Policy*: Each user is assigned a payment limit, and his total payment amount of anonymous transactions within each trading period should not exceed the limit. To simplify the design and enhance regulatory efficiency, the DAPCR scheme requires that a user's total payment amount of anonymous transactions within a trading period equals his payment limit. If it does not match, Filter will mark these transactions as suspicious and include them in regulation. If the total payment amount does not reach the payment

limit, the user can submit a transaction, of which the receiver is himself, with a payment amount equal to the shortfall, to make the total payment amount equal to the payment limit. With each anonymous transaction the user publishes, an equivalent amount is deducted from his payment limit. If the user exhausts his payment limit but still wants to transact, he can make a public transaction, which doesn't affect his payment limit.

3) *Formal Definition*: A DAPCR scheme is composed by the following PPT algorithms.

- $param \leftarrow \text{Setup}(1^\lambda)$. This algorithm takes as input a security parameter λ and outputs a public parameter $param$, which is implicit input to other algorithms.
- $(pk_F, sk_F) \leftarrow \text{FInit}(param)$. \mathcal{F} executes this algorithm to generate their public-private key pair (pk_F, sk_F) .
- $(pk_S, sk_S) \leftarrow \text{SInit}(param)$. \mathcal{S} executes this algorithm to generate their public-private key pair (pk_S, sk_S) .
- $(uk, sk) \leftarrow \text{KeyGen}(pk_S)$. This algorithm takes pk_S as input and outputs a public-private key pair (uk, sk) for a user.
- $req \leftarrow \text{Join}(uk, \mu, pk_S)$. A new user executes this algorithm to generate a request req to join the blockchain for consensus nodes. This algorithm takes uk , pk_S , and the user's payment limit μ as input and outputs a request req . If the new user obtains permission from all nodes, req will be recorded on the blockchain.
- $(tag_\mu, nym) \leftarrow \text{Reg}(req, pk_S, sk_S)$. \mathcal{S} reads req from the decentralized ledger and executes this algorithm to generate the data required for regulation. This algorithm takes req , sk_S and pk_S as input and outputs the user's limit tag tag_μ and his pseudonym nym .
- $rtx \leftarrow \text{RtxGen}(addr_S, addr_R, v, s, I_{pub}, I_{pri}, R, sk, pk_F)$. The algorithm takes as input a tuple $(addr_S, addr_R, v, s, I_{pub}, I_{pri}, R, sk, pk_F)$ and outputs a regulated transaction rtx , where $R = \{uk_0, uk_1, \dots, uk_{n-1}\}$, $uk = uk_j \in R$ and sk is a private key corresponding to uk .
- $0/1 \leftarrow \text{Verify}(rtx, pk_F, R, I_{pub})$. This algorithm, which verify the validity of a regulated transaction rtx , takes rtx , pk_F , R and I_{pub} as input, and outputs 1 if rtx is valid or 0 otherwise.
- $(tag_v, nym) \leftarrow \text{Extract}(rtx, sk_F)$. The algorithm takes rtx and sk_F as input and extracts an amount tag tag_v and the sender's pseudonym nym from rtx . Since pseudonyms are uniquely associated with user identities, they can be used to link transactions from the same sender.
- $(S, \text{susp}/\perp) \leftarrow \text{Detect}(nym, tag_\mu, sk_F, ledger)$. This algorithm takes as input a pseudonym nym , a tag tag_μ of

TABLE III: Properties of CLRS and related signatures

Scheme	Controllable Linkability	Irrevocable-iff-linked ¹	Collaborative Revocability ²	No Issuer
CLRS	✓	✓	✓	✓
[15]	×	✓	×	✓
[16]	✓	✓	×	✓
[17], [18]	×	×	×	✓
[19], [20]	✓	×	×	✓
[21]	×	✓	×	✓
[22]	✓	✓	×	×
[23], [24]	×	×	×	✓

¹ means that a linker cannot revoke the anonymity of signatures, even if these signatures are linked.

² means that the opener can only revoke the anonymity of signatures with the assistance of the linker.

nym 's payment upper limit, Filter's private key sk_F and a decentralized ledger ledger, which denoted all transactions sealed in blocks during a trading period, and outputs a set S of all transactions submit by nym . The algorithm further outputs $susp$ if nym 's total payment amount exceeds their upper limit, else outputs \perp .

- $rpt \leftarrow \text{Report}((S, susp), nym, pk_F, sk_F)$. This algorithm takes as input a tuple $(S, susp)$, a pseudonym nym and a public-private key pair (pk_F, sk_F) , and outputs a report rpt proving that nym published suspicious transactions in a trading period.
- $(uk, v_{sum})/\perp \leftarrow \text{Recover}(rpt, pk_F, sk_S)$. This algorithm takes a report rpt , Filter's public key pk_F and Supervisor's private key sk_S as input and check the validity of rpt . If rpt is valid, it outputs the sender's public key uk and the total payment amount v_{sum} else outputs \perp .

B. Ring Signature with Controllable Linkability

We propose a ring signature with controllable linkability, which is the building block of DAPCR. When signing a message, the signer needs to specify both a linker and an opener.

The CLRS scheme offers controllable linkability and collaborative revocability. The former means that only a designated linker can link signatures from the same sender. The latter means that the designated opener can revoke the anonymity of a signature with the assistance of the linker, but cannot do so independently, preventing the abuse of revocation authority and protecting privacy. Specifically, the linker can extract the pseudonym from each signature, and signatures with the same pseudonym come from the same signer. The opener can recover the signer's identity from any pseudonym but cannot directly open signatures. Table III compares the CLRS scheme with related works to show the advantage of our scheme.

The linker and opener are two semi-honest entities, which means they adhere to the CLRS protocol, do not actively sign messages, but can access ring signatures published in the system and attempt to identify the signers. We also assume that linker and opener do not collude.

A CLRS scheme consists of the following algorithms:

- $pp \leftarrow \text{Setup}(1^\lambda)$. The algorithm takes a security parameter λ as input, and generates a public parameter pp which is implicitly input to other algorithms.
- $(pk_L, sk_L) \leftarrow \text{LKGen}(pp)$. The algorithm outputs a public-private key pair (pk_L, sk_L) for a linker.

- $(pk_O, sk_O) \leftarrow \text{OKGen}(pp)$. The algorithm outputs a public-private key pair (pk_O, sk_O) for an opener.
 - $(uk, sk) \leftarrow \text{UKGen}(pk_O)$. The algorithm takes an opener's public key pk_O as input and outputs a user's public-private key pair (uk, sk) .
 - $\sigma/\perp \leftarrow \text{Sign}(R, m, pk_L, sk)$. The algorithm takes a ring $R = \{uk_0, uk_1, \dots, uk_{n-1}\}$, a message m , a linker's public key pk_L and a user's private key sk as input. If sk is the private key corresponding to $uk_j \in R$ and $j \in \{0, 1, \dots, n-1\}$, the algorithm outputs a signature σ of (R, m) else outputs \perp .
 - $0/1 \leftarrow \text{Vfy}(R, m, \sigma, pk_L)$. The algorithm takes a ring R , a message m , a linker's public key pk_L and a signature σ as input and outputs a bit b . If $b = 1$, σ is valid otherwise is invalid.
 - $nym \leftarrow \text{Ext}(\sigma, sk_L)$. The algorithm takes a valid signature σ and a linker's private key sk_L as input and outputs the signer's pseudonym nym .
 - $\text{link/unlink} \leftarrow \text{Link}(\sigma_0, \sigma_1, sk_L)$. A linker executes the algorithm CLRS.Ext to extract pseudonyms nym_0 and nym_1 from σ_0 and σ_1 . If $nym_0 = nym_1$, this algorithm outputs link else outputs unlink.
 - $uk \leftarrow \text{Open}(nym, sk_O)$. The algorithm takes a pseudonym nym and an opener's private key sk_O as input and outputs the user's public key uk .
- In addition, a linker can prove that the same pseudonyms nym are extracted from multiple signatures $\sigma_i|_{i=0}^{n-1}$. In other words, these signatures are signed by the same user with pseudonym nym .
- $\pi \leftarrow \text{Prove}(\sigma_i|_{i=0}^{n-1}, nym, pk_L, sk_L)$. This algorithm takes several signatures $\sigma_i|_{i=0}^{n-1}$, a pseudonym nym and a linker's public-private key pair (pk_L, sk_L) as input, and outputs a proof π of correct extraction.
 - $0/1 \leftarrow \text{Judge}(\sigma_i|_{i=0}^{n-1}, nym, \pi, pk_L)$. This algorithm takes several signatures $\sigma_i|_{i=0}^{n-1}$, a pseudonym nym , a proof π and a linker's public key pk_L as input, and outputs a bit b . If $b = 1$, π is valid else is invalid.

C. Security Definition

1) *Security Definition of CLRS*: First, we introduce two adversaries: \mathcal{A}_1 , who has compromised the linker and obtained the linking key sk_L , and \mathcal{A}_2 , who has compromised the opener and obtained the opening key sk_O . However, we assume that an adversary cannot compromise both the opener and the linker at the same time, because with sk_L and sk_O , an

<p>Anonymity Experiment $\text{Anon}_{\mathcal{A}_i}(\lambda)$:</p> <ol style="list-style-type: none"> 1: $pp \leftarrow \text{Setup}(1^\lambda)$ 2: $(pk_L, sk_L) \leftarrow \text{LKGen}(pp)$ 3: $(pk_O, sk_O) \leftarrow \text{OKGen}(pp)$ 4: $(R, m, uk_0, uk_1) \leftarrow \mathcal{A}_i(sk_{\mathcal{A}_i}, pk_L, pk_O)$ 5: $b \leftarrow_{\\$} \{0, 1\}$ 6: $\sigma_b \leftarrow \text{Sign}(R, m, pk_L, sk_b)$ 7: $b' \leftarrow \mathcal{A}_i(\sigma_b, sk_{\mathcal{A}_i}, pk_L, pk_O)$ 8: output $b' = b$ <p>Nym-soundness Experiment $\text{NymSound}_{\mathcal{A}_i}(\lambda)$:</p> <ol style="list-style-type: none"> 1: $pp \leftarrow \text{Setup}(1^\lambda)$ 2: $(pk_L, sk_L) \leftarrow \text{LKGen}(pp)$ 3: $(pk_O, sk_O) \leftarrow \text{OKGen}(pp)$ 4: $(R, m, \sigma, nym_0, nym_1, \pi_0, \pi_1) \leftarrow \mathcal{A}_i(sk_{\mathcal{A}_i}, pk_L, pk_O)$ 5: output $\bigwedge \text{Vfy}(R, m, \sigma, pk_L) = 1$ $\wedge \text{Judge}(\sigma, nym_0, \pi_0, pk_L) = 1$ $\wedge \text{Judge}(\sigma, nym_1, \pi_1, pk_L) = 1$ $\wedge nym_0 \neq nym_1 \wedge (nym_i^{sk_O}, \cdot) \in R$ 	<p>Unforgeability Experiment-I $\text{Unforge}_{\mathcal{A}_i}^1(\lambda)$:</p> <ol style="list-style-type: none"> 1: $pp \leftarrow \text{Setup}(1^\lambda)$ 2: $(pk_O, sk_O) \leftarrow \text{OKGen}(pp)$ 3: $(pk_L, sk_L) \leftarrow \text{LKGen}(pp)$ 4: $(R, m, \sigma) \leftarrow \mathcal{A}_i^{\mathcal{O}_{\text{UKGen}}, \mathcal{O}_{\text{Sign}}, \mathcal{O}_{\text{Reveal}}}(pp, pk_O, pk_L)$ 5: output $R \subset L_{\text{UKGen}} \setminus L_{\text{Reveal}}$ $\wedge (\sigma, R, m, pk_L, \cdot) \notin L_{\text{Sign}}$ $\wedge \text{Vfy}(R, m, \sigma, pk_L) = 1$ <p>Nym-extractability Experiment $\text{NymExt}_{\mathcal{A}_i}(\lambda)$:</p> <ol style="list-style-type: none"> 1: $pp \leftarrow \text{Setup}(1^\lambda)$ 2: $(pk_L, sk_L) \leftarrow \text{LKGen}(pp)$ 3: $(pk_O, sk_O) \leftarrow \text{OKGen}(pp)$ 4: $(R, m, \sigma) \leftarrow \mathcal{A}_i(sk_{\mathcal{A}_i}, pk_L, pk_O)$ 5: $nym \leftarrow \text{Ext}(\sigma, sk_L)$ 6: $\pi \leftarrow \text{Prove}(\sigma, nym, pk_L, sk_L)$ 7: output $\bigwedge \text{Vfy}(R, m, \sigma, pk_L) = 1$ $\wedge \text{Judge}(\sigma, nym, \pi, pk_L) = 0$ 	<p>Unforgeability Experiment-II $\text{Unforge}_{\mathcal{A}_1}^2(\lambda)$:</p> <ol style="list-style-type: none"> 1: $pp \leftarrow \text{Setup}(1^\lambda)$ 2: $(pk_L, sk_L) \leftarrow \text{LKGen}(pp)$ 3: $(pk_O, sk_O) \leftarrow \text{OKGen}(pp)$ 4: $(uk, R, m, \sigma) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{UKGen}}, \mathcal{O}_{\text{Sign}}, \mathcal{O}_{\text{Reveal}}}(pp, pk_O, pk_L)$ 5: output $uk \in L_{\text{UKGen}} \setminus L_{\text{Reveal}}$ $\wedge (\sigma, R, m, pk_L, uk) \notin L_{\text{Sign}}$ $\wedge \text{Vfy}(R, m, \sigma, pk_L) = 1$ $\wedge nym \leftarrow \text{Ext}(\sigma, sk_L)$ $\wedge uk \leftarrow \text{Open}(nym, sk_O)$ <p>Correctness Experiment $\text{Correct}_{\mathcal{A}_i}(\lambda)$:</p> <ol style="list-style-type: none"> 1: $pp \leftarrow \text{Setup}(1^\lambda)$ 2: $uk \leftarrow \text{UKGen}(sk)$ 3: $(pk_O, sk_O) \leftarrow \text{OKGen}(pp)$ 4: $(pk_L, sk_L) \leftarrow \text{LKGen}(pp)$ 5: $\sigma \leftarrow \text{Sign}(R, m, pk_L, pk_O, sk)$ 6: output $uk \in R \wedge \text{Vfy}(R, m, \sigma, pk_L) = 1$
---	--	---

Fig. 3: Security experiments for CLRS.

adversary would be able to trace the signer of any valid signature. We believe that this assumption is realistic.

Definition 1. We say that a CLRS scheme is secure if it satisfies correctness, unforgeability, anonymity, nym-extractability and nym-soundness.

1. *Correctness.* CLRS satisfies correctness if $\text{Correct}_{\mathcal{A}_i}(\lambda) \leq \text{negl}(\lambda)$.
2. *Unforgeability.* For $i \in \{1, 2\}$, \mathcal{A}_i without any ring member's private key cannot forge a ring signature on behalf of the ring. In addition, \mathcal{A}_1 cannot forge a signature from which the pseudonym extracted is associated with an honest user. A CLRS scheme satisfies unforgeability if for any PPT adversary \mathcal{A}_i , $\text{Unforge}_{\mathcal{A}_i}^1(\lambda) \leq \text{negl}(\lambda)$, and for any PPT adversary \mathcal{A}_1 , $\text{Unforge}_{\mathcal{A}_1}^2(\lambda) \leq \text{negl}(\lambda)$.

In these two experiments,

- $\mathcal{O}_{\text{UKGen}}$ is an oracle that runs $(uk, sk) \leftarrow \text{UKGen}(pk_O)$. L_{UKGen} is a list of users' public keys generated by this oracle;
 - $\mathcal{O}_{\text{Reveal}}$ is an oracle which returns the corresponding secret key when queried on a user's public key uk . L_{Reveal} is a list of public keys of which the secret key has been revealed;
 - $\mathcal{O}_{\text{Sign}}$ is an oracle that on query (R, m, pk_L, uk) returns $\sigma \leftarrow \text{Sign}(R, m, pk_L, sk)$. L_{Sign} is a list of the queries and responses (σ, R, m, pk_L, uk) .
3. *Anonymity.* CLRS satisfies anonymity if for any PPT adversary \mathcal{A}_i , $\text{Anon}_{\mathcal{A}_i}(\lambda) \leq \text{negl}(\lambda)$.
 4. *Nym-extractability.* A linker can always extract the signer's pseudonym from a signature and generate a proof for correct extraction. CLRS satisfies nym-extractability if for any PPT adversary \mathcal{A}_i , $\text{NymExt}_{\mathcal{A}_i}(\lambda) \leq \text{negl}(\lambda)$ where $sk_{\mathcal{A}_1} = sk_L$ and $sk_{\mathcal{A}_2} = sk_O$.
 5. *Nym-soundness.* Nym-soundness ensures that a linker cannot extract pseudonyms of two different signers from a signature, even if users in the ring are fully corrupt. CLRS satisfies nym-soundness if for any PPT adversary \mathcal{A}_i , $\text{NymSound}_{\mathcal{A}_i}(\lambda) \leq \text{negl}(\lambda)$.

Security experiments for CLRS are presented in Fig. 3.

2) *Security Definition of DAPCR:* We also present two adversaries for DAPCR: \mathcal{A}'_1 is an adversary that compromises Filter, while \mathcal{A}'_2 is an adversary that compromises Supervisor. However, we assume that adversaries cannot simultaneously compromise both Filter and Supervisor because they could collaborate to obtain private information from suspicious transactions. We believe this assumption is reasonable.

Definition 2. We say that a DAPCR scheme is secure if it satisfies indistinguishability, \mathcal{F} -extractability, \mathcal{F} -soundness, balance and non-malleability.

1. *Indistinguishability.* For \mathcal{A}'_2 , the regulated transaction reveals no information about transaction privacy. For \mathcal{A}'_1 , they can link regulated transactions from the same user but cannot obtain any additional information. We say that DAPCR satisfies indistinguishability if for any PPT adversary \mathcal{A}'_i , $\text{Ind}_{\mathcal{A}'_i}(\lambda) \leq \text{negl}(\lambda)$.
2. *\mathcal{F} -extractability.* Filter can extract the sender's pseudonym and the tag of payment amount from any valid transaction. DAPCR satisfies \mathcal{F} -extractability if for any PPT adversary \mathcal{A}'_i , $\text{FExt}_{\mathcal{A}'_i}(\lambda) \leq \text{negl}(\lambda)$ where $\text{amount}(\text{rtx})$ is the transaction rtx 's payment amount, and $\text{nym}(uk)$ is the pseudonym of the public key uk .
3. *\mathcal{F} -soundness.* For a valid regulated transaction, \mathcal{F} cannot extract two different pseudonyms or tags from the transaction. DAPCR satisfies \mathcal{F} -soundness if for any PPT adversary \mathcal{A}'_i , $\text{FSound}_{\mathcal{A}'_i}(\lambda) \leq \text{negl}(\lambda)$.
4. *Balance.* The amount paid by any PPT adversaries cannot exceed their balance.
5. *Non-malleability.* No PPT adversary possesses the capability to modify the information contained within a regulated transaction rtx .

Security experiments for DAPCR are presented in Fig. 4.

V. RING SIGNATURE WITH CONTROLLABLE LINKABILITY

First, we introduce three NIZK protocols Π_{enc} , Π_{dec} and Π_{mem} that are used to construct CLRS.

Indistinguishability Experiment $\text{Ind}_{\mathcal{A}'_i}(\lambda)$:	\mathcal{F} -extractability Experiment $\text{FExt}_{\mathcal{A}'_i}(\lambda)$:	\mathcal{F} -soundness Experiment $\text{FSound}_{\mathcal{A}'_i}(\lambda)$:
1: $param \leftarrow \text{Setup}(1^\lambda)$ 2: $(pk_F, sk_F) \leftarrow \text{FInit}(param)$ 3: $(pk_S, sk_S) \leftarrow \text{SInit}(param)$ 4: $(T_0, T_1, R) \leftarrow \mathcal{A}'_i(sk_{\mathcal{A}'_i}, pk_S, pk_F)$ where $T_j = (addr_{S_j}, addr_{R_j}, v_j, uk_j)$ $\wedge uk_0, uk_1 \in R$ 5: $rtx_b \leftarrow \text{RtxGen}(addr_{S_b}, addr_{R_b}, v_b, uk_b, sk_b, R, \cdot)$ 6: $b' \leftarrow \mathcal{A}'_i(rtx_b, sk_{\mathcal{A}'_i}, pk_S, pk_F)$ 7: output $b' = b$	1: $param \leftarrow \text{Setup}(1^\lambda)$ 2: $(pk_F, sk_F) \leftarrow \text{FInit}(param)$ 3: $(pk_S, sk_S) \leftarrow \text{SInit}(param)$ 4: $(R, rtx, v, r) \leftarrow \mathcal{A}'_i(sk_{\mathcal{A}'_i}, pk_S, pk_F)$ 5: $(nym, tag_v) \leftarrow \text{Extract}(rtx, sk_F)$ 6: output $v \neq \text{amount}(rtx)$ $\vee (\forall uk \in R, nym \neq nym(uk))$ $\wedge \text{Verify}(rtx, pk_F, R, I_{\text{pub}}) = 1$	1: $param \leftarrow \text{Setup}(1^\lambda)$ 2: $(pk_F, sk_F) \leftarrow \text{FInit}(param)$ 3: $(pk_S, sk_S) \leftarrow \text{SInit}(param)$ 4: $(R, rtx, T_0, T_1) \leftarrow \mathcal{A}'_i(sk_{\mathcal{A}'_i}, pk_S, pk_F)$ where $T_j = (nym_j, tag_j, \pi_j^{nym}, \pi_j^{tag})$ 5: output $V_{nym}(rtx, nym_0, \pi_0^{nym}) = 1$ $\vee V_{nym}(rtx, nym_1, \pi_1^{nym}) = 1$ $\vee V_{tag}(rtx, tag_0, \pi_0^{tag}) = 1$ $\vee V_{tag}(rtx, tag_1, \pi_1^{tag}) = 1$ $\wedge \text{Verify}(rtx, pk_F, R, I_{\text{pub}}) = 1$

Fig. 4: Security experiments for DAPCR.

A. NIZK Protocols in CLRS

- $\Pi_{mem} = (\mathcal{G}_{mem}, \mathcal{P}_{mem}, \mathcal{V}_{mem})$ represents a NIZK protocol for the relation

$$\mathcal{R}_{mem} = \left\{ \begin{array}{l} ((c_0, c_1, \dots, c_{n-1}), (l, r), (g, h)) : \\ \forall i, c_i \in \mathbb{G} \wedge c_i = g^0 h^r \wedge \\ l \in \{0, 1, \dots, n-1\} \wedge r \in \mathbb{Z}_q^* \end{array} \right\}.$$

Σ -protocols for the above relation called *one-out-of-many proofs* [25] that can be used to prove that one out of many commitments can be opened to 0 without requiring the prover to possess knowledge of the openings of the other commitments. Applying the Fiat-Shamir transform to this Σ -protocol yields the NIZK protocol Π_{mem} . Moreover, the protocol Π_{mem} requires no trusted setup, and the proof size is logarithmic in the number of all commitments.

- $\Pi_{enc} = (\mathcal{G}_{enc}, \mathcal{P}_{enc}, \mathcal{V}_{enc})$ represents a NIZK protocol for the relation

$$\mathcal{R}_{enc} = \left\{ \begin{array}{l} ((c, u), (\alpha, \beta), (g, h, pk)) : \\ c = g^\alpha h^\beta \wedge u = pk^\beta \wedge pk, g, h \in \mathbb{G} \wedge \alpha, \beta \in \mathbb{Z}_q^* \end{array} \right\}.$$

The protocol allows one to prove the correctness of an ElGamal ciphertext $ct = (c, u)$ given a specific public key pk .

- $\Pi_{dec} = (\mathcal{G}_{dec}, \mathcal{P}_{dec}, \mathcal{V}_{dec})$ represents a NIZK protocol for the relation

$$\mathcal{R}_{dec} = \left\{ \begin{array}{l} ((ct_i = (c_i, u_i))_{i=0}^{n-1}, m, pk), (sk, h) : \\ \forall i, c_i = m \cdot u_i^{\frac{1}{sk}} \wedge pk = h^{sk} \wedge \\ m, pk, h, c_i, u_i \in \mathbb{G} \wedge sk \in \mathbb{Z}_q^* \end{array} \right\}$$

where $ct_i|_{i=0}^{n-1}$ are n ElGamal ciphertexts. This protocol can prove that the decryption result of multiple ciphertexts is the same plaintext m .

Similar to the protocol Π_{mem} , Π_{enc} and Π_{dec} are respectively transformed from interactive protocols Σ_{enc} and Σ_{dec} , of which the details are presented in Appendix A.

B. Construction of CLRS

An efficient construction of CLRS consists of the following algorithms:

- Setup.** Consider two big prime numbers p and q . Elliptic curve cryptography (ECC) is based on the use of non-singular elliptic curves E on F_p . g is a generator of group \mathbb{G} , which is a cyclic group with order q . ECC can be given by a tuple (\mathbb{G}, g, p, q) . The public parameter used in our construction is denoted by $pp = (g, h, \mathcal{H})$ where $g, h \in \mathbb{G}$

and the hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$. pp is implicitly input to other algorithms.

- LKGen.** A linker randomly samples private key $sk_L \in \mathbb{Z}_q^*$ and computes $pk_L = h^{sk_L}$.
- OKGen.** An opener randomly samples private key $sk_O \in \mathbb{Z}_q^*$ and computes $pk_O = g^{sk_O}$.
- UKGen.** A user randomly samples private key $sk \in \mathbb{Z}_q^*$ and computes $pk = pk_O^{sk}$. Then the user randomly samples $r \in \mathbb{Z}_q^*$ and computes $c = g^{sk} h^r$. Lastly, the user generates a proof $\pi_{enc} \leftarrow \mathcal{P}_{enc}((c, pk), (r, sk), (h, g, pk_O))$. The user outputs $uk = (pk, c, \pi_{enc})$. One calculates $b \leftarrow \mathcal{V}_{enc}((c, pk), \pi_{enc}, (h, g, pk_O))$ to verify the validity of uk . If $b = 1$, uk is valid else is invalid. When implementing this scheme in the blockchain, a smart contract can be deployed as a bulletin board, which is responsible for verifying the validity of public keys and recording valid public keys.
- Sign.** To generate a signature σ for m , a user with the key pair (uk, sk) chooses a ring $R = \{uk_0, \dots, uk_{n-1}\}$ that satisfies $uk = uk_j \in R$. Initially, the user randomly samples $k \in \mathbb{Z}_q^*$ and computes $com = g^{sk} h^k$ and $K = pk_L^k$. Subsequently, the user extracts c_i from each public key $uk_i \in R$, computes $c'_i = c_i / com$ and gets a new ring $R' = \{c'_0, c'_1, \dots, c'_{n-1}\}$ where $c'_j = g^0 h^{r-k}$. The user also calculates a proof $\pi_{mem} \leftarrow \mathcal{P}_{mem}(R', (j, r - k), (g, h))$. Then the user randomly chooses $x_1, x_2 \in \mathbb{Z}_q^*$ and computes

$$\begin{aligned} com' &= g^{x_1} h^{x_2}, \\ K' &= pk_L^{x_2}, \\ e &= \mathcal{H}(R|m|com|com'|K|K'), \\ y_1 &= x_1 + e \cdot sk, \\ y_2 &= x_2 + e \cdot k. \end{aligned} \tag{1}$$

$\pi = (com', K', y_1, y_2)$ is a signature of knowledge proving $((com, K), (sk, k)) \in \mathcal{R}_{enc}$. The formula 1 is a SoK scheme transformed from the interactive protocol Σ_{enc} presented in Appendix A-A. Finally, the user outputs $\sigma = (com, K, \pi_{mem}, \pi)$.

- Vfy.** Given a signature σ of (R, m) , an opener's public key pk_O and a linker's public key pk_L , a verifier checks the validity of σ . Initially, the verifier computes R' and $b \leftarrow \mathcal{V}_{mem}(R', \pi_{mem}, (g, h))$. If $b = 0$, σ is invalid. Otherwise, the verifier computes $e = \mathcal{H}(R|m|com|com'|K|K')$ checks

whether the following equations hold.

$$\begin{aligned} g^{y_1} h^{y_2} &\stackrel{?}{=} com' com^e \\ pk_L^{y_2} &\stackrel{?}{=} K' K^e \end{aligned}$$

If both of the above equations hold, σ is valid else is invalid.

- **Ext.** Given a valid signature $\sigma = (com, K, \pi_{mem}, \pi)$ and a linker's private key sk_L , the linker computes the pseudonym $nym = K^{-\frac{1}{sk_L}} com = g^{sk}$ of the signer.
 - **Link.** Given two signatures σ_0, σ_1 and a linker's private key sk_L , a linker executes the algorithm CLRS.Ext to extract $nym_0 = g^{sk_0}$ and $nym_1 = g^{sk_1}$ from σ_0 and σ_1 . If $nym_0 = nym_1$ the algorithm outputs link else outputs unlink.
 - **Open.** Given a pseudonym nym and an opener's private key sk_O , an opener recovers $pk = nym^{sk_O} = pk_O^{sk}$ from the pseudonym and outputs $uk = (pk, \cdot)$.
- In addition, a linker can execute the algorithm CLRS.Prove to prove that a pseudonym nym is extracted from σ without revealing the private key sk_L . One can verify whether nym is extracted from σ .
- **Prove.** Given several signatures $\sigma_i = (com_i, K_i, \cdot)$ for $i \in \{0, 1, \dots, n-1\}$, a pseudonym nym and a linker's private key sk_L , the linker computes

$$\pi_{dec} \leftarrow \mathcal{P}_{dec}(((com_i, K_i)|_{i=0}^{n-1}, nym, pk_L), sk_L, h)$$

to prove correct decryption of $(com_i, K_i)|_{i=0}^{n-1}$ and that the decryption result of these ciphertexts is a same message nym . In other words, π_{dec} is a proof for that several signatures are published by the same signer with pseudonym nym .

- **Judge.** Given several signatures $\sigma_i = (com_i, K_i, \cdot)$ for $i \in \{0, 1, \dots, n-1\}$, a pseudonym nym , a linker's public key pk_L and a proof π_{dec} , one can compute

$$b \leftarrow \mathcal{V}_{dec}(((com_i, K_i)|_{i=0}^{n-1}, nym, pk_L), \pi_{dec}, h).$$

If $b = 1$, π_{dec} is valid else is invalid.

Theorem 1. *The proposed CLRS scheme satisfies correctness.*

PROOF. For a ring $R = \{uk_0, uk_1, \dots, uk_{n-1}\}$, the user with public key $uk_j = (pk_j, c_j, \pi_{enc}) = (pk_O^{sk}, g^{sk} h^r, \pi_{enc})$ calculates $\sigma = (com, K, \pi_{mem}, \pi)$ of (R, m) that satisfies $\sigma \leftarrow \text{CLRS.Sign}(R, m, pk_L, sk)$.

To verify the validity of σ , the verifier first computes $c'_i = c_i/com$ for $i \in \{0, 1, \dots, n-1\}$ and gets $R' = \{c'_0, c'_1, \dots, c'_{n-1}\}$. If the NIZK protocol Π_{mem} satisfies completeness and the SoK protocol for the relation \mathcal{R}_{enc} satisfies correctness, both π_{mem} and π can be successfully verified. The signature σ will also pass the verification as a result.

Therefore, if the NIZK protocol used in the CLRS scheme satisfies completeness, and the SoK protocol satisfies correctness, then the CLRS scheme achieves correctness. \square

VI. DAP WITH COLLABORATIVE REGULATION

In this section, we present an efficient DAPCR scheme based on a CLRS scheme and several NIZK protocols.

A. NIZK Protocols in DAPCR

First, we introduce two NIZK protocols Π_{log} and Π_v that are used to construct the DAPCR scheme.

1. $\Pi_{log} = (\mathcal{G}_{log}, \mathcal{P}_{log}, \mathcal{V}_{log})$ represents the NIZK protocol for the relation

$$\mathcal{R}_{log} = \{(A, \alpha, g) : A = g^\alpha \wedge g \in \mathbb{G} \wedge \alpha \in \mathbb{Z}_q^*\}.$$

The protocol can prove the knowledge of a discrete logarithm while ensuring the confidentiality of its actual value.

2. $\Pi_v = (\mathcal{G}_v, \mathcal{P}_v, \mathcal{V}_v)$ represents the NIZK protocol for the relation

$$\mathcal{R}_v = \left\{ \left((tx, c, I_{pub}, h, g, h), (addr_S, addr_R, I_{pri}, v, s, r) \right) : \begin{aligned} tx &\leftarrow \text{TxGen}(addr_S, addr_R, v, s, I_{pub}, I_{pri}) \wedge \\ c &= g^v h^r \wedge g, h \in \mathbb{G} \wedge v, r \in \mathbb{Z}_q^* \end{aligned} \right\}.$$

The protocol can prove that a committed value in c is the payment amount v of a privacy-preserving transaction tx . The details of this protocol are introduced in Appendix A.

B. Construction of DAPCR

An efficient DAPCR scheme consists of four phases: the preparation phase, the transaction phase, the verification phase, and the regulation phase.

1) *Preparation Phase:* The public parameter is published. Entities generate their public and private keys, and new users obtain permission from consensus nodes to join the blockchain.

Setup. Consensus nodes execute

$$\begin{aligned} (g, h, \mathcal{H}) &\leftarrow \text{CLRS.Setup}(1^\lambda), \\ crs &\leftarrow \mathcal{G}_v(1^\lambda, \mathcal{R}_v), \end{aligned}$$

and output $param = (g, h, \mathcal{H}, crs)$ that is implicit input to other algorithms.

Init. \mathcal{F} executes the algorithm CLRS.LKGen to get the public-private key pair (pk_L, sk_L) and publishes their public key $pk_F = pk_L$.

Sinit. \mathcal{S} executes the algorithm CLRS.OKGen to get the public-private key pair (pk_O, sk_O) and publishes their public key $pk_S = pk_O$.

KeyGen. A user computes $(uk, sk) \leftarrow \text{CLRS.UKGen}(pk_O)$ and publishes their public key uk .

Join. A new user \mathcal{U} randomly samples $\beta \in \mathbb{Z}_q^*$, and computes $B = g^\beta$ and $w = \mathcal{H}(pk_O^\beta | uk)$. Then \mathcal{U} sends a request $req = (uk, B, \mu)$, where μ is his payment limit within each trading period, to consensus nodes. To join the blockchain, \mathcal{U} must obtain permission from all nodes. If successful, (uk, B, μ) will be recorded on the chain.

Register. If a new user with public key uk has joined the blockchain, \mathcal{S} reads $req = (uk, B, \mu)$ from the decentralized ledger, and computes $w = \mathcal{H}(B^{sk_O} | uk)$, $tag_\mu = g^\mu h^w$ and $nym = pk_O^{\frac{1}{sk_O}}$. Then, \mathcal{S} adds (uk, μ, nym, tag_μ) to the list L_S . Moreover, \mathcal{S} sends (nym, tag_μ) to \mathcal{F} . Once (nym, tag_μ) is received, \mathcal{F} adds it to the list $L_{\mathcal{F}}$.

2) *Transaction Phase*: Users are allowed to submit two types of transactions: regulated transactions with privacy preservation and public transactions. In a trading period, the total amount paid by a user through regulated transactions cannot exceed their privacy payment limit.

RtxGen. Suppose that a user \mathcal{U} intends to submit n regulated transactions within a trading period. To submit the i -th regulated transaction rtx_i , for $i \in \{0, 1, \dots, n-2\}$, \mathcal{U} randomly samples $z_i, w_i \in \mathbb{Z}_q^*$ and computes

$$\begin{aligned} \text{tx}_i &\leftarrow \text{DAP.TxGen}(\text{addr}_S, \text{addr}_{R,i}, v_i, s, I_{\text{pub},i}, I_{\text{pri},i}), \\ ct_i &= (g^{v_i} h^{z_i}, pk_L^{z_i - w_i}) = (c_i, u_i), \\ \sigma_i &\leftarrow \text{CLRS.Sign}(R_i, ct_i, pk_L, sk), \end{aligned}$$

where ct_i is a ElGamal ciphertext of a pedersen commitment $g^{v_i} h^{w_i}$. Then \mathcal{U} computes

$$\begin{aligned} \pi_i^{\text{log}} &\leftarrow \mathcal{P}_{\text{log}}(u_i, z_i - w_i, pk_L), \\ \pi_i^v &\leftarrow \mathcal{P}_v((\text{tx}_i, c_i, I_{\text{pub},i}, \tilde{h}_i, g, h), (\text{addr}_S, \text{addr}_{R,i}, I_{\text{pri},i}, v_i, s, z_i), crs), \end{aligned}$$

where $\tilde{h}_i = \mathcal{H}(\sigma_i)$, and sets $\text{rtx}_i = (\text{tx}_i, ct_i, \pi_i^v, \pi_i^{\text{log}}, \sigma_i)$.

For the final regulated transaction rtx_{n-1} within a trading period, \mathcal{U} calculates $w_{n-1} = w - \sum_{i=0}^{n-2} w_i$ instead of sampling a random number. All other operations remain unchanged. Therefore, if the total payment amount within the trading period equals \mathcal{U} 's privacy payment limit μ , the equation $\prod_{i=0}^{n-1} g^{v_i} h^{w_i} = \text{tag}_\mu$ holds.

If \mathcal{U} exhausts his payment limit μ but still needs to transact, he can submit a public transaction, which doesn't affect his payment limit. To generate a public transaction tx_{pub} , \mathcal{U} calculates a DAP transaction tx , attaches $(\text{addr}_S, \text{addr}_R, v)$ to it, and generates a proof π_{pub} that confirms addr_S as the sender's address, addr_R as the receiver's address, and v as the payment amount of tx . Finally, \mathcal{U} gets a public transaction $\text{tx}_{\text{pub}} = (\text{tx}, \text{addr}_S, \text{addr}_R, v, \pi_{\text{pub}})$. As the public transactions is not a primary focus of this paper, we will refrain from delving into their specifics.

3) *Verification Phase*: Consensus nodes (or smart contracts) verify the validity of regulated transactions, and only valid transactions are sealed into blocks.

Verify. For a regulated transaction $\text{rtx} = \{\text{tx}, ct, \pi_v, \pi_{\text{log}}, \sigma\}$, one executes the following steps to verify the validity of rtx :

$$\begin{aligned} b_1 &\leftarrow \text{DAP.TxVfy}(\text{tx}, I_{\text{pub}}), \\ b_2 &\leftarrow \text{CLRS.Vfy}(R, ct, \sigma, pk_L), \\ b_3 &\leftarrow \mathcal{V}_v((\text{tx}, c, I_{\text{pub}}, \tilde{h}, g, h), \pi_v, crs), \\ b_4 &\leftarrow \mathcal{V}_{\text{log}}(u, \pi_{\text{log}}, pk_L), \end{aligned}$$

where $\tilde{h} = \mathcal{H}(\sigma)$. If $b_1 \wedge b_2 \wedge b_3 \wedge b_4 = 1$ and all public keys in the ring R are from registered users, rtx is valid and the valid transaction is sealed into a new block.

4) *Regulation Phase*: \mathcal{F} screens out suspicious transactions and submits the report on suspicious transactions to \mathcal{S} . Once the report is received, \mathcal{S} obtains the sender's public key and payment amounts of suspicious transactions.

Extract. For a valid transaction $\text{rtx}_i = (ct_i, \sigma_i, \cdot)$, \mathcal{F} extracts the signer's pseudonym and the amount tag from rtx_i .

$$\begin{aligned} nym_i &\leftarrow \text{CLRS.Ext}(\sigma_i, sk_L), \\ \text{tag}_{v_i} &= c_i u_i^{-\frac{1}{sk_L}} = g^{v_i} h^{w_i}. \end{aligned}$$

Detect. Let $S = \{\text{rtx}_0, \text{rtx}_1, \dots, \text{rtx}_{n-1}\}$ be the set of all n transactions submitted by some user during a trading period. \mathcal{F} can link these transactions according to the user's pseudonym nym .

To determine if the transaction behavior of a user with pseudonym nym complies with the transaction rules, \mathcal{F} searches for tag_μ corresponding to nym in $L_{\mathcal{F}}$ and computes $\text{tag}_{\text{sum}} = \prod_{i=0}^{n-1} \text{tag}_{v_i}$ where $\text{tag}_{v_i}|_{i=0}^{n-1}$ are extracted from all transactions published by the user with pseudonym nym in a period. Considering that $\text{tag}_\mu = g^\mu h^w$ and $\text{tag}_{\text{sum}} = g^{\sum_{i=0}^{n-1} v_i} h^{\sum_{i=0}^{n-1} w_i} = g^{\sum_{i=0}^{n-1} v_i} h^w$, the total payment volume is equal to the upper limit if $\text{tag}_\mu = \text{tag}_{\text{sum}}$. Otherwise, \mathcal{F} flags these transactions in S as suspicious.

To increase supervision efficiency, \mathcal{F} can aggregate ciphertexts and then extract tag_{sum} :

$$\begin{aligned} C &= \prod_{i=0}^{n-1} c_i = g^{\sum_{i=0}^{n-1} v_i} h^{\sum_{i=0}^{n-1} z_i}, \\ U &= \prod_{i=0}^{n-1} u_i = pk_L^{\sum_{i=0}^{n-1} (z_i - w_i)}, \\ \text{tag}_{\text{sum}} &= CU^{-\frac{1}{sk_L}}. \end{aligned}$$

Report. For a set $S = \{\text{rtx}_0, \text{rtx}_1, \dots, \text{rtx}_{n-1}\}$ of suspicious transactions, \mathcal{F} proves that these transactions are published by the user with pseudonym nym and tag_{sum} is formed by aggregating the amount tags extracted from these transactions. \mathcal{F} first generates a proof π_{dec} :

$$\pi_{\text{dec}} \leftarrow \mathcal{P}_{\text{dec}}((ct_i|_{i=0}^{n-1}, nym, pk_L), sk_L, h).$$

Then \mathcal{F} generates a proof π_{sum} for that tag_{sum} is extracted from (C, U) :

$$\pi_{\text{sum}} \leftarrow \mathcal{P}_{\text{dec}}((C, U, \text{tag}_{\text{sum}}, pk_L), sk_L, h).$$

Finally, \mathcal{F} sends $\text{rpt} = (S, nym, \pi_{\text{dec}}, \text{tag}_{\text{sum}}, \pi_{\text{sum}})$ to \mathcal{S} .

Recover. Once a report $\text{rpt} = (S, nym, \pi_{\text{dec}}, \text{tag}_{\text{sum}}, \pi_{\text{sum}})$ is received, \mathcal{S} checks whether the transactions in S are submitted by the same user with pseudonym nym :

$$b_1 \leftarrow \mathcal{V}_{\text{dec}}((ct_i|_{i=0}^{n-1}, nym, pk_L), \pi_{\text{dec}}, h).$$

Then \mathcal{S} checks the validity of tag_{sum} :

$$b_2 \leftarrow \mathcal{V}_{\text{dec}}((\prod_{i=0}^{n-1} c_i, \prod_{i=0}^{n-1} u_i, \text{tag}_{\text{sum}}, pk_L), \pi_{\text{sum}}, h).$$

If $b_1 = 0 \vee b_2 = 0$, rpt is invalid else is valid. For a valid report rpt , \mathcal{S} computes $uk \leftarrow \text{CLRS.Open}(nym, sk_O)$ and then requires the user with public key uk to submit v_{sum} and w' satisfies $\text{tag}_{\text{sum}} = g^{v_{\text{sum}}} h^{w'}$.

Theorem 2. *The proposed DAPCR scheme satisfies correctness.*

PROOF. For any transaction $\text{rtx}_i = (\text{tx}_i, ct_i, \pi_i^v, \pi_i^{\text{log}}, \sigma_i)$ generated according to the DAPCR.RtxGen algorithm, a verifier employs the DAPCR.Verify algorithm to ascertain the validity of rtx_i . This verification process involves verifying tx_i , π_i^v , π_i^{log} and σ_i . Only if all of these components pass validation, the transaction rtx_i is deemed valid.

NIZK protocols Π_v and Π_{log} used to construct the DAPCR scheme satisfies completeness. Additionally, both the DAP scheme and the CLRS scheme satisfy correctness. Hence, each of tx_i , π_i^v , π_i^{log} and σ_i can successfully pass the validation, meaning that the transaction rtx_i is validity. \square

VII. SECURITY ANALYSIS

In this section, we analyze the security of our scheme.

A. Security Analysis of CLRS

Theorem 3. *No PPT adversary can break the anonymity of CLRS with non-negligible advantage.*

PROOF. We use $G_{\text{real}}^{\text{Ano}}$ to denote the anonymity experiment $\text{exp}_{\mathcal{A}_i}^{\text{Ano}}(\lambda)$ executed in the real world. To analyze the security of CLRS, we design a simulation $G_{\text{sim}}^{\text{Ano}}$. In $G_{\text{sim}}^{\text{Ano}}$, the challenger \mathcal{C} interacts with \mathcal{A}_1 (or \mathcal{A}_2) as in $G_{\text{real}}^{\text{Ano}}$. The only modification is that \mathcal{C} outputs a signature $\sigma_{\text{sim}} = (\text{com}_{\text{sim}}, K, \pi_{\text{sim}}^{\text{mem}}, \pi_{\text{sim}})$ of (R, m) in which com_{sim} , K , $\pi_{\text{sim}}^{\text{mem}}$ and π_{sim} are independent of uk_0 and uk_1 .

Game $G_{\text{real}}^{\text{Ano}}$. With the purpose of initializing this experiment, \mathcal{C} first computes

$$\begin{aligned} crs_{\text{mem}} &\leftarrow \mathcal{G}_{\text{mem}}(1^\lambda, \mathcal{R}_{\text{mem}}), \\ pp_{\text{enc}} &\leftarrow \text{SoK.Setup}(1^\lambda, \mathcal{R}_{\text{enc}}), \end{aligned}$$

and other public parameters. \mathcal{C} further computes public keys of honest users, and R_1 represents the set of these public keys.

Next, \mathcal{A}_i generates a ring R_2 , a message m and two public keys uk_0, uk_1 satisfies $uk_0, uk_1 \in R_1 \wedge R_2$. \mathcal{A}_i sends (R_2, m, uk_0, uk_1) to \mathcal{C} .

Once (R_2, m, uk_0, uk_1) is received, \mathcal{C} randomly chooses $b \in \{0, 1\}$ and computes

$$\sigma_b \leftarrow \text{CLRS.Sign}(R_2, m, pk_L, sk_b)$$

where $\sigma_b = (\text{com}, K, \pi_{\text{mem}}, \pi) = (g^{sk_b} h^k, pk_L^k, \pi_{\text{mem}}, \pi)$. \mathcal{C} sends σ_b to \mathcal{A}_i .

After receiving σ_b , \mathcal{A}_i attempts to determine which public key was used to compute the signature. \mathcal{A}_1 can query an oracle \mathcal{Q}_{Ext} to extract pseudonyms from signatures. \mathcal{A}_2 can query an oracle $\mathcal{Q}_{\text{Open}}$ to obtain a public key from a given pseudonym. Finally, \mathcal{A}_i makes a guess b' for the value of b , and wins the experiment if and only if $b = b'$.

Game $G_{\text{sim}}^{\text{Ano}}$. During the initialization phase, \mathcal{C} computes the public parameters and public keys of honest users, denoted by R_1 . The only difference is that trapdoors are generated alongside the computation of parameters crs_{mem} and pp_{enc} .

$$\begin{aligned} (crs_{\text{mem}}, \tau_{\text{mem}}) &\leftarrow \mathcal{G}_{\text{sim}}^{\text{mem}}(1^\lambda, \mathcal{R}_{\text{mem}}) \\ (pp_{\text{enc}}, \tau_{\text{enc}}) &\leftarrow \text{SoK.Setup}_{\text{sim}}(1^\lambda, \mathcal{R}_{\text{enc}}) \end{aligned}$$

Similar to the case in $G_{\text{real}}^{\text{Ano}}$, \mathcal{A}_i generates a ring R_2 , a message m and two public keys uk_0, uk_1 satisfies $uk_0, uk_1 \in R_1 \wedge R_2$. \mathcal{A}_i sends (R_2, m, uk_0, uk_1) to \mathcal{C} .

Once (R_2, m, uk_0, uk_1) is received, \mathcal{C} randomly chooses $b \in \{0, 1\}$ and $\text{com}_{\text{sim}} \in \mathbb{G}^*$. Then \mathcal{C} computes

$$\begin{aligned} \pi_{\text{sim}}^{\text{mem}} &\leftarrow \mathcal{P}_{\text{sim}}^{\text{mem}}(R_2, \tau_{\text{mem}}, crs_{\text{mem}}), \\ \pi_{\text{sim}} &\leftarrow \text{SoK.Sign}_{\text{sim}}((\text{com}_{\text{sim}}, K, pk_L), \tau_{\text{enc}}, pp_{\text{enc}}), \end{aligned}$$

and sends $\sigma_{\text{sim}} = \{\text{com}_{\text{sim}}, K, \pi_{\text{sim}}^{\text{mem}}, \pi_{\text{sim}}\}$ to \mathcal{A}_i . Since σ_{sim} is independent of uk_0 and uk_1 , the advantage of \mathcal{A}_i in winning $G_{\text{sim}}^{\text{Ano}}$ is negligible.

Based on the simulatability of a SoK protocol and the zero-knowledge of a NIZK protocol, it can be deduced that the distribution of $(\text{com}, \pi_{\text{mem}}, \pi)$ is equivalent to that of $(\text{com}_{\text{sim}}, \pi_{\text{sim}}^{\text{mem}}, \pi_{\text{sim}})$. While \mathcal{A}_1 can query \mathcal{Q}_{Ext} to extract nym from σ_b and nym' from σ_{sim} , $nym = pk_b^{1/sk_0}$ and nym' are indistinguishable since sk_0 is unknown to \mathcal{A}_1 . Therefore, σ_{sim} in $G_{\text{sim}}^{\text{Ano}}$ and σ_b in $G_{\text{real}}^{\text{Ano}}$ are indistinguishable. Considering the negligible advantage of \mathcal{A}_i in winning $G_{\text{sim}}^{\text{Ano}}$ and the indistinguishability between $G_{\text{real}}^{\text{Ano}}$ and $G_{\text{sim}}^{\text{Ano}}$, the advantage of \mathcal{A}_i in winning $G_{\text{real}}^{\text{Ano}}$ is also negligible. \square

Theorem 4. *No PPT adversary can break the unforgeability of CLRS with non-negligible advantage.*

PROOF. For a discrete logarithm problem $(g, A = g^a)$ where $g \in \mathbb{G}$ and $a \in \mathbb{Z}_q^*$, if \mathcal{A}_i breaks the unforgeability of CLRS, \mathcal{C} can make use of \mathcal{A}_i to solve the discrete logarithm problem. Adversaries can win the experiments in two ways:

Case-1: For an honest ring, i.e. all members in the ring are honest, \mathcal{A}_i generates a valid signature.

To initialize this experiment, \mathcal{C} first executes

$$(crs_{\text{enc}}, \tau_{\text{enc}}) \leftarrow \mathcal{G}_{\text{sim}}^{\text{enc}}(1^\lambda, \mathcal{R}_{\text{enc}}).$$

For $j \in \{0, 1, \dots, l\}$, \mathcal{C} randomly chooses $r_j \in \mathbb{Z}_q^*$ and $pk_j \in \mathbb{G}^*$, and computes

$$\pi'_j \leftarrow \mathcal{P}_{\text{sim}}^{\text{enc}}((h^{r_j} A, pk_j), \tau_{\text{enc}}, crs_{\text{enc}}).$$

Then \mathcal{C} sets $c_j = h^{r_j} A$ and publishes $uk_j = (pk_j, c_j, \pi'_j)$. L_τ represents the set of public keys generated based on the trapdoor. \mathcal{A}_i can query \mathcal{C} for user's secret keys and signatures. If \mathcal{A}_i queries \mathcal{C} for the secret key corresponding to uk , the experiment aborts.

To win the experiment, \mathcal{A}_i forges a signature $\sigma = (\text{com}, K, \pi_{\text{mem}}, \pi)$ of (R, m) where $R \subset L_{\text{UKGen}} \setminus L_{\text{Reveal}}$ and $(\sigma, R, m, pk_L, \cdot) \notin L_{\text{Sign}}$. The probability that $R \subset L_\tau$ is $\frac{1}{\eta_1(\lambda)}$ where $\eta_1(\lambda)$ is a polynomial. By the extractability of a SoK protocol, \mathcal{C} can extract a valid witness $\varpi = (a, k)$ to the statement $\phi = (\text{com}, K)$ from π . Therefore, \mathcal{C} can make use of \mathcal{A}_i to obtain a such that $A = g^a$.

Case-2: There exists an honest user with $uk = (pk, c, \pi_{\text{enc}})$ whose private key sk is unknown to \mathcal{A}_1 . \mathcal{A}_1 calculates a signature σ of (R, m) and a proof π_{dec} for that nym is correctly extracted from σ . If nym is the pseudonym of uk , and σ and π_{dec} are valid, \mathcal{A}_1 successfully breaks the unforgeability of CLRS.

To initialize this experiment, \mathcal{C} first executes

$$(crs_{\text{enc}}, \tau_{\text{enc}}) \leftarrow \mathcal{G}_{\text{sim}}^{\text{enc}}(1^\lambda, \mathcal{R}_{\text{enc}}).$$

Then \mathcal{C} randomly chooses $r \in \mathbb{Z}_q^*$ and computes

$$\pi_{\text{sim}}^{\text{enc}} \leftarrow \mathcal{P}_{\text{sim}}^{\text{enc}}((h^r A, A^{sk_0}), \tau_{\text{enc}}, crs_{\text{enc}}).$$

Finally, \mathcal{C} publishes $uk = (A^{sk_0}, h^r A, \pi_{\text{sim}}^{\text{enc}}) = (pk, c, \pi_{\text{sim}}^{\text{enc}})$.

To win the experiment, \mathcal{A}_1 randomly selects an honest user and forges a signature $\sigma = (\text{com}, K, \pi_{\text{mem}}, \pi)$ of (R, m) where $uk \notin L_{\text{UKGen}} \setminus L_{\text{Reveal}}$ and $(\sigma, R, m, pk_L, uk) \notin L_{\text{Sign}}$.

Assuming \mathcal{A}_1 selects uk with a probability $\frac{1}{\eta_2(\lambda)}$, where $\eta_2(\lambda)$ represents an upper bound on the number of honest users. In the experiment, C aborts if \mathcal{A}_1 queries for the private key sk of the public key uk . \mathcal{A}_1 also generates a proof for that nym is the result of correctly decrypting the ciphertext (com, K) . If nym is a pseudonym of uk , i.e. $nym = g^a$, the knowledge-soundness of NIZK protocols implies that $\phi_{dec} = (com, K, nym, pk_L)$ is a valid statement such that $(\phi_{dec}, sk_L) \in \mathcal{R}_{dec}$. Furthermore, if π is a valid signature of knowledge, the extractability of a SoK protocol ensures that C can extract a valid witness $\varpi_{enc} = (a, k)$ to the statement $\phi_{enc} = (com, K)$ from π . Therefore, C can make use of \mathcal{A}_1 to obtain a such that $A = g^a$.

In conclusion, if \mathcal{A}_i can break the unforgeability of CLRS with a non-negligible probability, C can solve the DL problem using \mathcal{A}_i 's successful attack. \square

Theorem 5. *No PPT adversary can break the nym-extractability of CLRS with non-negligible advantage.*

PROOF. For a valid public key $uk = (pk, c, \pi_{enc})$, the completeness of the protocol Π_{enc} implies that $pk = pk_O^{sk}$ and $c = g^{sk}h^r$. For a valid signature $\sigma = (com, K, \pi_{mem}, \pi)$ of (R, m) , correctness of the protocol Π_{mem} implies that $c' = c/com = g^0h^{r'}$. Thus, $com = g^{sk}h^k$ and $k = r - r'$. The extractability of the SoK protocol implies that a valid witness $\varpi = (sk, k)$ to the statement $\phi = (com, K)$ can be extracted from π . The witness satisfies $uk = (pk_O^{sk}, \cdot) \in R$ and $(com, K) = (g^{sk}h^k, pk_L^k)$.

Since (com, K) can be viewed as an ElGamal encryption under the public key pk_L , correctness of the ElGamal encryption scheme ensures that decrypting (com, K) with the private key sk_L yields $nym = g^{sk}$, which is the output of CLRS.Ext. In addition, based on the completeness of the protocol Π_{dec} , the proof π_{dec} for that nym is correctly extracted from σ will be verified correctly. Therefore, CLRS.Judge takes σ , nym and π_{dec} as input and outputs 1. \square

Theorem 6. *No PPT adversary can break the nym-soundness of CLRS with non-negligible advantage.*

PROOF. As analyzed in the proof of Theorem 5, the completeness of the protocol Π_{enc} implies that $pk = pk_O^{sk}$ and $c = g^{sk}h^r$ if a public key $uk = (pk, c, \pi_{enc})$ is valid. In addition, the correctness of the protocol Π_{mem} implies that $c'_j = g^0h^{r'}$ if $\sigma = (com, K, \pi_{mem}, \pi)$ is valid. Thus, $com = g^{sk}h^k$ and $k = r - r'$. The extractability of the SoK protocol implies that a valid witness $\varpi = (sk, k)$ to the statement $\phi = (com, K)$ can be extracted from π . The witness satisfies $uk = (pk_O^{sk}, \cdot) \in R$ and $(com, K) = (g^{sk}h^k, pk_L^k)$.

If the proof π_{dec} is valid, the soundness of the protocol Π_{dec} ensures that nym is the result of decrypting (com, K) with a private key sk_L . Considering the perfect correctness of the ElGamal encryption scheme, the result of decrypting the ciphertext using a given private key is unique. If this is not the case, \mathcal{A}_i can be employed to construct another adversary capable of compromising the perfect correctness of the ElGamal encryption scheme.

Therefore, the pseudonym extracted from a signature is unique. \square

B. Security Analysis of DAPCR

Theorem 7. *No PPT adversary can break the indistinguishability of the DAPCR scheme with non-negligible advantage.*

PROOF. We use G_{real}^{ind} to denote the experiment $\text{exp}_{\mathcal{A}_i}^{ind}(\lambda)$ executed in the real world. To analyze the security of DAPCR, we design a simulation G_{sim}^{ind} . In G_{sim}^{ind} , the challenger C interacts with \mathcal{A}_i as in G_{real}^{ind} . The only modification is that C outputs a challenge transaction $\text{rtx}_{sim} = \{\text{tx}_{sim}, ct_{sim}, \pi_{sim}^v, \pi_{sim}^{log}, \sigma\}$ where tx_{sim} , ct_{sim} , π_{sim}^v and π_{sim}^{log} are independent of T_b .

Game G_{real}^{ind} . C first calculates

$$\begin{aligned} param &\leftarrow \text{DAPCR.Setup}(1^\lambda), \\ (pk_F, sk_F) &\leftarrow \text{DAPCR.FInit}(param), \\ (pk_S, sk_S) &\leftarrow \text{DAPCR.SInit}(param), \end{aligned}$$

publishes $(param, pk_F, pk_S)$ and sends sk_F to \mathcal{A}_1 (or sk_S to \mathcal{A}_2). C also generates public keys of honest users, and R_1 represents the set of these public keys.

Next, \mathcal{A}_i generates a ring R_2 and two tuples T_0 and T_1 where $T_j = (addr_{S_j}, addr_{R_j}, v_j, uk_j)$ and $uk_0, uk_1 \in R_1 \wedge R_2$. \mathcal{A}_i sends (T_0, T_1, R) to C .

After receiving (T_0, T_1, R) , C randomly chooses $b \in \{0, 1\}$ and computes

$$\text{rtx}_b \leftarrow \text{RtxGen}(addr_{S_b}, addr_{R_b}, v_b, R, sk_b, \cdot)$$

where sk_b is the private key associated with uk_b . C sends $\text{rtx}_b = (\text{tx}, ct, \pi_v, \pi_{log}, \sigma)$ to \mathcal{A}_i .

Once rtx_b is received, \mathcal{A}_i makes a guess b' for the value of b , and wins the experiment if and only if $b = b'$.

Game G_{mid}^{ind} . The experiment in G_{mid}^{ind} is similar to that in G_{real}^{ind} , but the only modification is that C replaces tx with tx_{sim} . To initialize the experiments, C makes use of simulators to generate trapdoors alongside the computation of common reference strings. After receiving (T_0, T_1, R) , C generates a transaction tx_{sim} that is independent of T_0 and T_1 , and makes use of a trapdoor to calculate a proof π_{mid}^v . C sends $\text{rtx}_{mid} = (\text{tx}_{sim}, ct, \pi_{mid}^v, \pi_{log}, \sigma)$ to \mathcal{A}_i .

If the DAP scheme used to construct DAPCR is secure, tx and tx_{sim} are indistinguishable. Then zero-knowledge of NIZK protocols implies that the distribution of π_v is identical to that of π_{mid}^v . Thus, the absolute value of the difference between the advantage of \mathcal{A}_i in winning G_{mid}^{ind} and that in winning G_{real}^{ind} is negligible.

Game G_{sim}^{ind} . G_{sim}^{ind} is similar to G_{mid}^{ind} , but the only modification is that C replaces ct with ct_{sim} . To initialize the experiments, C makes use of simulators to generate trapdoors alongside the computation of common reference strings. After receiving (T_0, T_1, R) , C randomly samples $ct_{sim} = (c_{sim}, u_{sim}) \in \mathbb{G}^2$ that is independent of T_0 and T_1 , and makes use of a trapdoor to calculate proofs π_{sim}^v and π_{sim}^{log} . C sends $\text{rtx}_{sim} = (\text{tx}_{sim}, ct_{sim}, \pi_{sim}^v, \pi_{sim}^{log}, \sigma)$ to \mathcal{A}_i .

For \mathcal{A}_2 , the indistinguishability of the ElGamal encryption ensures that the distribution of ct_{sim} is identical to that of ct . The zero-knowledge of NIZK protocols implies that the distribution of $(\pi_{sim}^v, \pi_{sim}^{log})$ is identical to that of (π_{mid}^v, π_{log}) .

For \mathcal{A}_1 , the amount tag can be extracted from a transaction. In particular, \mathcal{A}_1 calculates $tag_{sim} = c_{sim}u_{sim}^{-1/sk_L}$ and

$tag_v = cu^{-1/sk_L} = g^v h^{w_i}$, both of which have the same distribution. Similar to the case of \mathcal{A}'_2 , the distribution of $(ct_{sim}, \pi_{sim}^v, \pi_{sim}^{log})$ is identical to that of $(ct, \pi_{mid}^v, \pi_{log})$.

Thus, the absolute value of the difference between the advantage of \mathcal{A}'_i in winning G_{sim}^{ind} and that in winning G_{mid}^{ind} is negligible. Since $tx_{sim}, ct_{sim}, \pi_{sim}^v$ and π_{sim}^{log} are independent of uk_0 and uk_1 , the advantage of \mathcal{A}'_i in winning G_{sim}^{ind} is equal to the advantage of \mathcal{A}'_i in breaking the anonymity of CLRS.

Therefore, the advantage of \mathcal{A}'_i in winning G_{real}^{ind} is negligible. The proposed DAPCR scheme satisfies indistinguishability if the DAP scheme, the NIZK protocols, the ElGamal encryption and the CLRS scheme making up it are secure. \square

Theorem 8. *No PPT adversary can break the \mathcal{F} -extractability of the DAPCR scheme with non-negligible advantage.*

PROOF. For a valid transaction $rtx = (tx, ct, \pi_v, \pi_{log}, \sigma)$ where $ct = (c, u)$, the completeness of the protocol Π_v implies that $c = g^v h^z$ where v is the payment amount of tx , and the completeness of the protocol Π_{log} ensures that $u = pk_L^r$ where $r \in \mathbb{Z}_q^*$. Since $ct = (g^v h^z, pk_L^r)$ can be viewed as a well-formed ciphertext of a pedersen commitment $g^v h^{z-r}$, the correctness of the encryption scheme implies that decrypting ct with the private key $sk_F = sk_O$ yields $tag_v = g^v h^{z-r}$.

Since the CLRS scheme satisfies nym-extractability, \mathcal{F} can extract the signer's pseudonym from σ . Thus, the pseudonym nym and the amount tag tag_v can be extracted from rtx by \mathcal{F} . \square

Theorem 9. *No PPT adversary can break the \mathcal{F} -soundness of the DAPCR scheme with non-negligible advantage.*

PROOF. For a valid transaction $rtx = (tx, ct, \pi_v, \pi_{log}, \sigma)$ where $ct = (c, u)$, the completeness of the protocol Π_v implies that $c = g^v h^z$ where v is the payment amount of tx , and the completeness of the protocol Π_{log} ensures that $u = pk_L^r$ where $r \in \mathbb{Z}_q^*$. Since $ct = (g^v h^z, pk_L^r)$ can be viewed as a well-formed ciphertext of a pedersen commitment $g^v h^{z-r}$, the perfect correctness of the encryption scheme implies that the result of decrypting the ciphertext ct using a given private key $sk_F = sk_O$ is unique. If this is not the case, \mathcal{A}'_i can be employed to construct another adversary capable of compromising the perfect correctness of the encryption scheme.

Since the proposed CLRS scheme satisfies nym-soundness, \mathcal{F} cannot two different valid pseudonyms from σ . Thus, the pseudonym and the amount tag extracted from rtx are unique. \square

The proposed DAPCR scheme, which is based on DAP, adds additional regulated fields to privacy-preserving transactions. If an adversary generates a regulated transaction $rtx = (tx, ct, \pi_v, \pi_{log}, \sigma)$ with a payment exceeding their balance, they also gets a transaction tx breaking the balance of DAP. If an adversary modifies the data stored within a regulated transaction rtx , they also obtains a modified transaction tx breaking the non-malleability of DAP, or forges a proof breaking the security of zk-SNARK. Thus, the proposed DAPCR scheme satisfies balance and non-malleability if the DAP scheme and the zk-SNARK scheme making up it is secure.

C. Attacks

In this section, we take the example of DAPCR constructed based on ZETH or Zether to analyze how the scheme resists the following malicious attacks.

1) *Sybil attack*: The DAPCR scheme is designed for consortium blockchains, making it effective against certain malicious attacks targeting public blockchains, such as Sybil attacks. Consortium chains have strict controls on user admissions. Each user in the system undergoes identity authentication. If any user is found engaging in malicious behavior, they will be removed from the blockchain.

2) *One-time account attack*: One-time account attack means that an adversary controls multiple one-time accounts for transactions, making it difficult for Filter to link transactions from this adversary and thus evade regulation. Since the DAPCR scheme is deployed on a consortium blockchain, a new account needs permission from all consensus nodes to join the consortium chain, and the user's account address and real-world identity are transparent to consensus nodes. Therefore, a user cannot simultaneously control multiple accounts and conduct one-time account attacks.

3) *Double-spending attack*: Double-spending attack means that an adversary spends the same amount multiple times. In the DAPCR scheme based on ZETH, it is necessary to publish the corresponding serial number to spend the balance in a coin, and the same serial number cannot be used twice. If consensus nodes detect that a serial number has already been revealed, the transaction will be rejected. In the Zether-based scheme, each user is limited to publishing a single transaction in each epoch. This transaction contains proof that the nonce is generated based on the user's private key and the epoch number. The associated smart contract accumulates these nonces and, crucially, rejects any transaction bearing a duplicate nonce. In simple terms, if the anonymous payment mechanism that constitutes DAPCR can resist double-spending attacks, then the DAPCR scheme itself can resist such attacks.

4) *Over-spending attack*: Over-spending attack means that an adversary spends more than they possess. In the DAPCR scheme based on ZETH or Zether, a user should attach a proof to his transaction, which can show that the amount spent does not exceed the balance in his coin or account. The security of zero-knowledge proof protocols ensures that the probability of any adversary forging a valid proof for over-spending is negligible. Therefore, if the anonymous payment mechanism that constitutes DAPCR can resist over-spending attacks, then the DAPCR scheme itself can resist such attacks.

5) *Malleability attack*: Malleability attack means that an adversary modifies the data stored within a transaction. In the ZETH-based scheme, the honest user generates a transaction $rtx = (tx, ct, \pi_v, \pi_{log}, \sigma)$. The payment data is stored within tx , which can be considered as a transaction in ZETH, and the regulatory data is stored within the regulated field $reg = (ct, \sigma)$. Due to the non-malleability of ZETH, any adversary cannot modify the payment data in tx . To modify the regulatory data within reg , the adversary needs to forge a proof π'_v to replace π_v . The security of zk-SNARK ensures that the probability of any adversary forging a valid proof is negligible.

VIII. PERFORMANCE EVALUATION

In this section, we present a performance evaluation of the DAPCR scheme proposed in Section VI.

We design four experiments to assess the performance of DAPCR, all of which are executed on a local device with an 8-core Intel(R) Core(TM) i7-10700 @ 2.90GHz CPU, 8 GB RAM and Ubuntu 20.04 LTS OS. DAPCR is constructed based on ZETH [26] which is an adaptive version of Zerocash designed for deployment on public or consortium blockchains with smart contracts. In addition, we utilize the zk-SNARK algorithm Groth16 [11], along with the SNARK-friendly elliptic curve BabayJubjub [27] and the MiMC hash algorithm to implement the DAPCR scheme. Moreover, the experimental implementation makes use of the programming language Golang⁴ and the zero-knowledge proof tool Snarkjs⁵.

In Experiment-I, we evaluate the computational and communication overhead of CLRS, which is shown in Fig. 5a. First, the time cost for user key generation is 0.7ms, independent of the ring-size. Then, we evaluate the time overhead for signature generation and verification under various ring-sizes. It is evident that the time costs of signature generation and verification are linearly related to the ring-size. When the ring-size is 16 which is similar to the ring-size employed in Monero, the time costs for signature generation and verification are 13.3ms and 6.9ms, respectively. Additionally, we analyze the signature-length, which exhibits a logarithmic relationship with the ring-size. When the ring-size is set to 16, the signature-length is 2.1 KB. Hence, we consider the CLRS scheme to be efficient.

In Experiment-II, we evaluate the time cost of transaction generation in DAPCR, which is shown in Fig. 5b. The time cost of transaction generation is linear to the number of transactions. It takes about 1231.2ms to generate one transaction and less than 250s to generate 200 transactions in DAPCR. We also compare the time cost of transaction generation in DAPCR and that in XLN22 [28], which is also a DAP scheme with regulation based on Zerocash. In XLN22, the average time required to generate a transaction is 11.52s, while in DAPCR, it is only 1.23s. Compared to Zerocash, XLN22 introduces an additional computational cost of 10.70s and a communication cost of over 75 KB for a transaction, while DAPCR only introduces an additional computational cost of 0.41s and a communication cost of 2.1 KB. Hence, we consider DAPCR to be efficient in the transaction phase.

In Experiment-II, we also evaluate the computational overhead of transaction verification in DAPCR and compared it with that in ZETH. Fig. 5c presents the time cost of transaction verification in a consensus node, which is linear to the number of transactions. It takes about only 12.5ms to verify one transaction and about 2.5s to verify 200 transactions in DAPCR. Although the time cost of transaction verification in DAPCR is approximately three times higher compared to ZETH, it still remains at a relatively low level. Furthermore,

we deploy the DAPCR scheme in Fabric⁶, where the time interval between a user submitting a transaction and receiving a response indicating successful verification is about 2.0s. Hence, we consider DAPCR to be efficient in the verification phase.

In Experiment-III, we evaluate the time cost of Filter in screening out suspicious transactions, which is shown in Fig. 5d. We use *Extract Nym* to denote the time cost of extracting pseudonyms from transactions, which is linear to the number of transactions. If there are 10,000 transactions in the DAPCR system, *Extract Nym* is less than 2.5s. Assuming that transactions from Alice account for 2% (5%, 10%) of all transactions, the time cost of obtaining the total amount tag of Alice is shown in Fig. 5d. Compared to *Extract Nym*, the time cost of extracting the total amount tag is negligible. The result of Experiment-III indicates that DAPCR is effective in the regulation phase.

IX. RELATED WORK

A. DAP with Regulation

Wang et al. [6] propose a decentralized anonymous payment scheme with supervision (DAPS) based on zk-SNARK and the elliptic curve cryptography. A transaction in DAPS contains a ciphertext encrypted with the public key of the regulator. The regulator can decrypt ciphertexts in transactions with the private key to obtain the privacy. Faced with numerous transactions, it is inefficient for the regulator needs to decrypt the ciphertext in each transaction. Lin et al. [7] present a secure and efficient decentralized conditional anonymous payment system (DCAP) based on signatures of knowledge. A transaction in DCAP contains anonymous addresses of the sender and the recipient. The regulator can trace the long-term address for an anonymous address, but the payment amount of each transaction is public in DCAP. Wang et al. [6] and Lin et al. [7] neglect to restrict the regulatory power, which may be abused.

Some solutions recognize the importance of restricted regulation, but still have some shortcomings. Garman et al. [8] design a DAP scheme based on Zerocash that forces users to comply with specific policies and grants regulators the power of coin tracing and user tracing. [8] restricts the power of the regulator who is asked to provide an accountable record of the power being used. However, the regulator can still obtain privacy from a transaction. PRCash [29] is a new blockchain currency with privacy preservation and regulation, in which the sender's identity is also encrypted with the regulator's public key and included in the transaction. UTT [30] is a decentralized e-cash system with accountable privacy. In UTT each user needs to get *budget coins*, which are used to limit the total sum of payments, from the auditor per month.

zkLedger [10] and miniLedger [9] are two decentralized payment systems that achieve privacy preservation and verifiability auditing. These solutions realize rich auditing functions, but require auditors to interact with users. However, users may

⁴<https://go.dev/>

⁵<https://github.com/iden3/snarkjs>

⁶We deploy a Fabric (v2.4.9) network on a local device, which consists of two peer nodes and one order node.

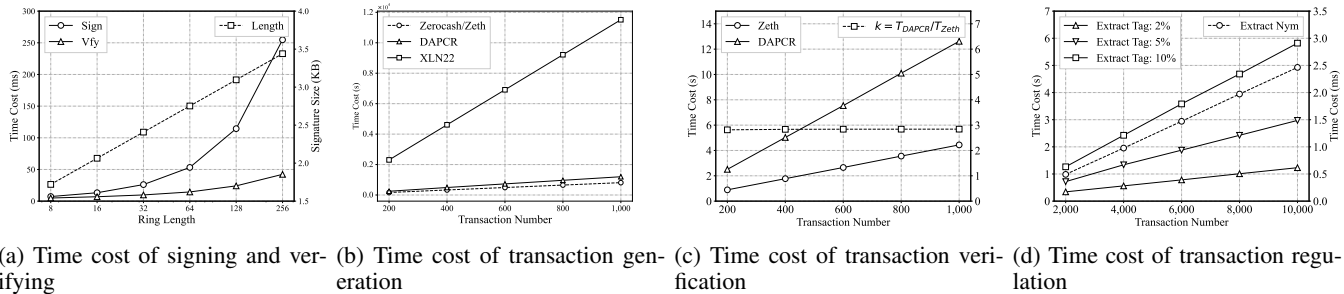


Fig. 5: Performance experiments for DAPCR and CLRS

not always be online, and a malicious user may ignore the auditor’s queries, which leads to delays in the audit.

Platypus [31] and PEReDi [32] are two central bank digital currencies with privacy preservation and regulation. In Platypus each user needs to encrypt his privacy with the regulator’s public key and the ciphertext is included in the transaction. The regulator can decrypt the ciphertext for transaction privacy. In PEReDi several authorities form a committee to revoke privacy or trace transactions from some user. The committee revokes the privacy of a transaction by decrypting the ciphertext saved in the ledger. It is inefficient to decrypt the ciphertext of each transaction for its privacy when auditing numerous transactions.

B. Ring Signature with Privacy Preservation and Regulation

The ring signature [33] allows a user in the ring to sign a message on behalf of all members with unconditional anonymity, which poses a potential risk of signing malicious messages. To address this issue, the linkable ring signature has been proposed. In linkable ring signatures [15], anyone can link signatures signed by the same user without opening these signatures. In the ring signature with escrowed linkability [16], only a trusted authority can link signatures, which maintains anonymity for regular users. In the traceable ring signature [17], [18], also known as the revocable-iff-linked ring signature, if a user generate two or more signatures with the same tag, the anonymity of these signatures can be revoked.

There are some schemes that attempt to achieve the balance between anonymity and accountability. The accountable ring signature [19] and the revocable ring signature [20] allow the designated user or trusted authority to revoke the anonymity of signatures. Additionally, in the revocable and linkable ring signature [21], anyone can link signatures signed by the same user, and the trusted authority can revoke the anonymity of signatures. However, if the designated user or trusted authority is compromised, it can lead to a severe security crisis.

In the ring signature with user-controlled linkability [23] and the claimable ring signature [24], a signer can link their signatures and provide a publicly verifiable linking proof. These schemes are not suitable for anonymous payment mechanisms because only the signer can link their signatures. In group signatures with controlled linkability [22], only entities authorized by a trusted center can link signatures. As an

inherent characteristic of group signatures, the trusted center can revoke the anonymity of the signatures. In addition, a new user needs to engage in an interactive protocol with the issuer to join the group signature system.

Compared to group/traceable signatures [34], users in CLRS generate public and private keys locally, avoiding centralized key generation and distribution. Furthermore, in group/traceable signatures, the central authority can unconditionally revoke the anonymity of a signature, whereas in the CLRS scheme, the opener must cooperate with the linker to determine the identity of the signer, preventing the abuse of power and privacy leakage.

X. CONCLUSION

In this paper, we propose a decentralized anonymous payment scheme with collaborative regulation, aiming to achieve universality, collaborative regulation, and efficient aggregation of transaction amounts. The regulation in our scheme relies on the functioning of *Filter* at the end of each trading period, which is a potential target for malicious attackers. However, transaction validation is managed by consensus nodes. Therefore, even if *Filter* is compromised, validated transactions can still be added to the decentralized ledger. In future work, we plan to optimize the regulatory process to eliminate single points of failure and modify the DAPCR scheme to achieve UC security.

REFERENCES

- [1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *Decentralized Business Review*, p. 21260, 2008.
- [2] G. Wood *et al.*, “Ethereum: A secure decentralised generalised transaction ledger,” *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.
- [3] S. Noether, A. Mackenzie, *et al.*, “Ring confidential transactions,” *Ledger*, vol. 1, pp. 1–18, 2016.
- [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*, pp. 459–474, IEEE, 2014.
- [5] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, “Zether: Towards privacy in a smart contract world,” in *International Conference on Financial Cryptography and Data Security*, pp. 423–443, Springer, 2020.
- [6] Z. Wang, Q. Pei, X. Liui, L. Ma, H. Li, and S. Yu, “Daps: A decentralized anonymous payment scheme with supervision,” in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 537–550, Springer, 2019.
- [7] C. Lin, D. He, X. Huang, M. K. Khan, and K.-K. R. Choo, “Dcap: A secure and efficient decentralized conditional anonymous payment system based on blockchain,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2440–2452, 2020.

- [8] C. Garman, M. Green, and I. Miers, “Accountable privacy for decentralized anonymous payments,” in *International conference on financial cryptography and data security*, pp. 81–98, Springer, 2016.
- [9] P. Chatzigiannis and F. Baldimtsi, “Miniledger: compact-sized anonymous and auditable distributed payments,” in *European Symposium on Research in Computer Security*, pp. 407–429, Springer, 2021.
- [10] N. Narula, W. Vasquez, and M. Virza, “{zkLedger}: {Privacy-Preserving} auditing for distributed ledgers,” in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pp. 65–80, 2018.
- [11] J. Groth, “On the size of pairing-based non-interactive arguments,” in *Annual international conference on the theory and applications of cryptographic techniques*, pp. 305–326, Springer, 2016.
- [12] M. Chase and A. Lysyanskaya, “On signatures of knowledge,” in *Advances in Cryptology-CRYPTO 2006: 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006. Proceedings 26*, pp. 78–96, Springer, 2006.
- [13] A. Fiat and A. Shamir, “How to prove yourself: Practical solutions to identification and signature problems,” in *Conference on the theory and application of cryptographic techniques*, pp. 186–194, Springer, 1986.
- [14] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al., “Hyperledger fabric: a distributed operating system for permissioned blockchains,” in *Proceedings of the thirteenth EuroSys conference*, pp. 1–15, 2018.
- [15] J. Bootle, K. Elkhiyaoui, J. Hesse, and Y. Manevich, “Duality: Logarithmic-verifier linkable ring signatures through preprocessing,” in *European Symposium on Research in Computer Security*, pp. 427–446, Springer, 2022.
- [16] S. S. Chow, W. Susilo, and T. H. Yuen, “Escrowed linkability of ring signatures and its applications,” in *Progress in Cryptology-VIETCRYPT 2006: First International Conference on Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006. Revised Selected Papers*, pp. 175–192, Springer, 2006.
- [17] A. Scafuro and B. Zhang, “One-time traceable ring signatures,” in *Computer Security-ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part II 26*, pp. 481–500, Springer, 2021.
- [18] H. Feng, J. Liu, D. Li, Y.-N. Li, and Q. Wu, “Traceable ring signatures: general framework and post-quantum security,” *Designs, Codes and Cryptography*, vol. 89, pp. 1111–1145, 2021.
- [19] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, J. Groth, and C. Petit, “Short accountable ring signatures based on ddh,” in *European Symposium on Research in Computer Security*, pp. 243–265, Springer, 2015.
- [20] J. Wen, L. Bai, Z. Yang, H. Zhang, H. Wang, and D. He, “Larrs: Lattice-based revocable ring signature and its application for vanets,” *IEEE Transactions on Vehicular Technology*, 2023.
- [21] X. Zhang, J. K. Liu, R. Steinfeld, V. Kuchta, and J. Yu, “Revocable and linkable ring signature,” in *Information Security and Cryptology: 15th International Conference, Inscrypt 2019, Nanjing, China, December 6–8, 2019, Revised Selected Papers 15*, pp. 3–27, Springer, 2020.
- [22] J. Y. Hwang, L. Chen, H. S. Cho, and D. Nyang, “Short dynamic group signature scheme supporting controllable linkability,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1109–1124, 2015.
- [23] D. Fiore, L. Garms, D. Kolonelos, C. Soriente, and I. Tucker, “Ring signatures with user-controlled linkability,” in *European Symposium on Research in Computer Security*, pp. 405–426, Springer, 2022.
- [24] S. Park and A. Sealton, “It wasn’t me! repudiability and claimability of ring signatures,” in *Advances in Cryptology-CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III 39*, pp. 159–190, Springer, 2019.
- [25] 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*, pp. 253–280, Springer, 2015.
- [26] A. Rondelet and M. Zajac, “Zeth: On integrating zerocash on ethereum,” *arXiv preprint arXiv:1904.00905*, 2019.
- [27] B. WhiteHat, J. Baylina, and M. Bellés, “Baby jubjub elliptic curve,” *Ethereum Improvement Proposal, EIP-2494*, vol. 29, 2020.
- [28] L. Xue, D. Liu, J. Ni, X. Lin, and X. S. Shen, “Enabling regulatory compliance and enforcement in decentralized anonymous payment,” *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 2, pp. 931–943, 2022.
- [29] K. Wüst, K. Kostiaainen, V. Čapkun, and S. Čapkun, “Prcash: fast, private and regulated transactions for digital currencies,” in *International Conference on Financial Cryptography and Data Security*, pp. 158–178, Springer, 2019.
- [30] A. Tomescu, A. Bhat, B. Applebaum, I. Abraham, G. Gueta, B. Pinkas, and A. Yanai, “Utt: Decentralized ecash with accountable privacy,” *Cryptology ePrint Archive*, 2022.
- [31] K. Wüst, K. Kostiaainen, N. Delius, and S. Capkun, “Platypus: a central bank digital currency with unlinkable transactions and privacy-preserving regulation,” in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2947–2960, 2022.
- [32] A. Kiayias, M. Kohlweiss, and A. Sarencheh, “Peredi: Privacy-enhanced, regulated and distributed central bank digital currencies,” in *The 29th ACM Conference on Computer and Communications Security*, 2022.
- [33] T. H. Yuen, M. F. Esgin, J. K. Liu, M. H. Au, and Z. Ding, “Dualring: generic construction of ring signatures with efficient instantiations,” in *Annual International Cryptology Conference*, pp. 251–281, Springer, 2021.
- [34] E. Fujisaki and K. Suzuki, “Traceable ring signature,” in *International Workshop on Public Key Cryptography*, pp. 181–200, Springer, 2007.
- [35] S. Bowe, A. Gabizon, and I. Miers, “Scalable multi-party computation for zk-snark parameters in the random beacon model,” *Cryptology ePrint Archive*, 2017.

APPENDIX A

NIZK PROTOCOLS FOR CLRS AND DAPCR

A. Σ -protocol for the relation \mathcal{R}_{enc}

For a ElGamal ciphertext $ct = (c, u) = (g^\alpha h^\beta, pk^\beta)$, a prover interacts with a verifier to prove $((c, u), (\alpha, \beta)) \in \mathcal{R}_{enc}$.

1. The prover randomly chooses $x_1, x_2 \in \mathbb{Z}_q^*$, computes $c' = g^{x_1} h^{x_2}$ and $u' = pk^{x_2}$ and sends (c', u') to the verifier.
2. Once receiving (c', u') , the verifier randomly chooses $e \in \mathbb{Z}_q^*$ and sends it to the prover.
3. Once receiving e , the prover computes $y_1 = x_1 + e \cdot \alpha$, $y_2 = x_2 + e \cdot \beta$ and sends them to the verifier.
4. Finally, the verifier determines whether equations $pk^{y_2} \stackrel{?}{=} u' u^e$ and $g^{y_1} h^{y_2} \stackrel{?}{=} c' c^e$ hold. If both of the equations hold, the proof is valid else is invalid.

B. Σ -protocol for the relation \mathcal{R}_{dec}

For several ciphertexts $ct_i = (c_i, u_i)$, $i \in \{0, 1, \dots, n-1\}$, a message m and a public key $pk = h^{sk}$, a prover interacts with a verifier to prove $((ct_i)_{i=0}^{n-1}, m, pk), sk \in \mathcal{R}_{dec}$.

1. The prover randomly chooses $x \in \mathbb{Z}_q^*$, computes $A = h^x$ and $B_i = (c_i/m)^x$ for $i \in \{0, 1, \dots, n-1\}$, and sends $(A, B_i)_{i=0}^{n-1}$ to the verifier.
2. Once receiving $(A, B_i)_{i=0}^{n-1}$, the verifier randomly chooses $e \in \mathbb{Z}_q^*$ and sends it to the prover.
3. Once receiving e , the prover computes $y = x + e \cdot sk$ and sends it to the verifier.
4. Finally, the verifier determines whether the equations $h^y \stackrel{?}{=} pk^e A$ and

$$\left(\prod_{i=0}^{n-1} \frac{c_i}{m}\right)^y \stackrel{?}{=} \left(\prod_{i=0}^{n-1} u_i\right)^e \prod_{i=0}^{n-1} B_i$$

hold. If both of the equations hold, the proof is valid else is invalid.

The Fiat-Shamir transform can convert the above Σ -protocol into a NIZK protocol or a SoK protocol for the same relation.

C. NIZK protocol for the relation \mathcal{R}_v

To ensure the general applicability of our scheme, we adopt the zk-SNARK protocol to generate a proof for the relation \mathcal{R}_v . Before calculating proofs, an arithmetic circuit C needs to be constructed based on the relation \mathcal{R}_v .

Public inputs to C are as follows.

1. privacy-preserving transaction tx
2. pedersen commitment c
3. additional public inputs I_{pub} used to calculate tx
4. hash \tilde{h} of a signature σ
4. $g, h \in \mathbb{G}$
5. public parameter pp_{tx} in the DAP scheme

Private inputs to C are as follows.

1. sender's address $addr_S$ and receiver's address $addr_R$
2. transaction amount v
3. sender's private key s
4. additional private inputs I_{pri} used to calculate tx
5. random number $r \in \mathbb{Z}_q^*$

C imposes the following constraints on public inputs and private inputs.

1. tx is generated by the algorithm $\text{DAP.TxGen}(\cdot)$ with inputs $(addr_S, addr_R, v, s, I_{\text{pub}}, I_{\text{pri}})$.
2. c is a pedersen commitment of (v, r) , i.e. $c = g^v h^r$.

Based on the arithmetic circuit C , a trusted center can execute the algorithm \mathcal{G}_v to generate crs_v . However, if the process of parameter generation is compromised, an adversary can generate forged proofs. To address these issues, Bowe et al. [35] proposed a secure multiparty computation protocol among n nodes, which ensures that no one can generate forged proofs if at least one node is honest. Thus, we can deploy multiple nodes to execute the MPC protocol for initializing the DAPCR system.

APPENDIX B

DISCUSSION OF DEPLOYING DAPCR IN DIFFERENT MODELS

A regulated transaction $\text{rtx} = (\text{tx}, ct, \pi_v, \pi_{\log}, \sigma)$ consists of three parts: a DAP transaction tx, a regulated additional field $\text{reg} = \{ct, \sigma\}$, and the zero-knowledge proof $\pi = \{\pi_v, \pi_{\log}\}$. Whether the DAPCR scheme is deployed in the UTXO model or the account model, the algorithm for generating regulated fields is the same. The difference lies in the zero-knowledge proof component.

1) *Account Model*: In the account model, a user's balance is stored in their account. A user publishes a transaction to deduct a certain amount from his account and add an equivalent amount to the recipient's account. To transform the DAP transaction tx into a regulated one, the regulated field $\text{reg} = \{ct, \sigma\}$ should be attached to tx. The sender also needs to provide a proof $\pi = \{\pi_v, \pi_{\log}\}$ to show the consistency of regulatory data within reg and payment data within tx.

2) *UTXO Model*: In the UTXO model, a user's balance is stored in unspent transaction outputs (UTXOs). Users need to combine or split UTXOs to make payments of arbitrary amounts. Taking Zerocash in the UTXO model as an example, a transaction takes two old coins c_1^{old} and c_2^{old} as inputs and

outputs two new coins c_1^{new} and c_2^{new} . To realize effective regulation, we restrict transactions to be one-to-one. In other words, coins c_1^{old} , c_2^{old} and c_1^{new} must belong to the same address, with c_1^{new} considered as change and its value can be 0. The actual payment amount in the transaction is the value of c_2^{new} . This restriction can be realized using zk-SNARKs: the sender generates an additional proof when constructing the transaction to show that c_1^{old} , c_2^{old} and c_1^{new} belong to the same address. Hence, the DAP scheme in the UTXO model can also be represented using algorithms in Section III-A. Specifically, the transaction generation algorithm is $\text{TxGen}(addr_S, addr_R, v, s, I_{\text{pub}}, I_{\text{pri}}) \rightarrow \text{tx}$, where $addr_S$ is the address to which coins c_1^{old} , c_2^{old} , and c_1^{new} belong, $addr_R$ is the address to which c_2^{new} belongs, v represents the value of c_2^{new} , and $(c_1^{\text{old}}, c_2^{\text{old}}, c_1^{\text{new}}, c_2^{\text{new}})$ are included in the transaction tx. To transform the DAP transaction tx in the UTXO model into a regulated one, the regulated fields $\text{reg} = \{ct, \sigma\}$ should be attached to tx. The zero-knowledge proof π' should not only prove the consistency of regulatory data within reg and payment data within tx but also show that coins c_1^{old} , c_2^{old} , and c_1^{new} belong to the same address. This is the distinction between deploying DAPCR in the UTXO model and deploying it in the account model.

APPENDIX C

ALGORITHMS OF DAPCR SCHEME

For convenience, we summarize the main algorithms of DAPCR as follows.

An Efficient Construction of DAPCR

The workflow of DAPCR is divided into four phases: the preparation phase, the transaction phase, the verification phase, and the regulation phase.

I-Preparation Phase

In the preparation phase, the public parameter is published. Entities generate their public and private keys, and new users obtain permission from consensus nodes to join the blockchain.

Consensus nodes:

DAPCR.Setup:

- Inputs: security parameter λ
- Outputs: public parameter $param$
- Consensus nodes execute the following steps to generate the public parameter:
 1. compute $(g, h, \mathcal{H}) \leftarrow \text{CLRS.Setup}(1^\lambda)$
 2. compute $crs \leftarrow \mathcal{G}_v(1^\lambda, \mathcal{R}_v)$
- Consensus nodes publish $param = (g, h, \mathcal{H}, crs)$.

Filter:

DAPCR.FInit:

- Inputs: public parameter $param$
- Outputs: Filter's public-private key pair (pk_F, sk_F)
- Filter executes the following steps to generate their public-private key pair:
 1. compute $(pk_L, sk_L) \leftarrow \text{CLRS.LKGen}(pp)$
 2. set $sk_F = sk_L$ and $pk_F = pk_L$
- Filter publishes their public key pk_F .

Supervisor:

DAPCR.SInit:

- Inputs: public parameter $param$
- Outputs: Supervisor's public-private key pair (pk_S, sk_S)
- Supervisor executes the following steps to generate their public-private key pair:
 1. compute $(pk_O, sk_O) \leftarrow \text{CLRS.OKGen}(pp)$
 2. set $sk_S = sk_O$ and $pk_S = pk_O$

- Supervisor publishes their public key pk_O .

User:

DAPCR.KeyGen:

- Inputs: Supervisor's public key pk_S
- Outputs: user's public-private key pair (uk, sk)
- A user executes the algorithm CLRS.UKGen to generate their public-private key pair: $(uk, sk) \leftarrow \text{CLRS.UKGen}(pk_O)$.
- The user publishes their public key uk .

User \rightarrow Consensus nodes:

DAPCR.Join:

- User:
 1. randomly sample $\beta \in \mathbb{Z}_q^*$ and compute $B = g^\beta$
 2. compute $w = \mathcal{H}(pk_O^\beta | uk)$
 3. send req = (uk, B, μ) to consensus nodes
- To join the blockchain, the user must obtain permission from all nodes. If successful, req will be recorded on the blockchain.

Supervisor:

DAPCR.Register:

- Supervisor:
 1. read req = (uk, B, μ) from the decentralized ledger
 2. compute $w = \mathcal{H}(B^{sk_O} | uk)$, $tag_\mu = g^\mu h^w$ and $nym = pk^{1/sk_O}$
 3. add (uk, μ, nym, tag_μ) to the list L_S
- Supervisor also sends (nym, tag_μ) to Filter. Once (nym, tag_μ) is received, Filter adds it to the list L_F .

II-Transaction Phase

In the transaction phase, users generates regulated transactions to transfer their assets.

User:

DAPCR.RtxGen:

- Inputs:
 1. sender's address $addr_S$
 2. receiver's address $addr_R$
 3. payment amount v
 4. sender's secret key s
 5. additional public inputs I_{pub} and private inputs I_{pri}
 6. Filter's public key pk_F
 7. user's private key sk
 8. ring R
- Outputs: regulated transaction rtx
- A user executes the following steps to generate a regulated transaction:
 1. compute $tx \leftarrow \text{DAP.TxGen}(addr_S, addr_R, v, s, I_{pub}, I_{pri})$
 2. sample $z, w \in \mathbb{Z}_q^*$ and compute $ct = (g^v h^z, pk_L^{z-w})$
 3. compute $\sigma \leftarrow \text{CLRS.Sign}(R, ct, pk_L, sk)$ and $\tilde{h} = \mathcal{H}(\sigma)$
 4. compute $\pi_v \leftarrow \mathcal{P}_v((tx, c, I_{pub}, \tilde{h}, g, h), (addr_S, addr_R, I_{pri}, v, s, z), crs)$
 5. compute $\pi_{log} \leftarrow \mathcal{P}_{log}(u, z - w, pk_L)$
 6. set $rtx = (tx, ct, \pi_v, \pi_{log}, \sigma)$
- The user submits the regulated transaction rtx .

III-Verification Phase

In the verification phase, consensus nodes (or smart contracts) verify the validity of regulated transactions, and only valid transactions are sealed into blocks.

Consensus nodes:

DAPCR.Verify:

- Inputs:
 1. regulated transaction rtx
 2. Filter's public key pk_F
 3. ring R
 4. additional public inputs I_{pub}
- Outputs: 0/1
- Consensus nodes execute the following steps to verify a transaction:
 1. compute $\tilde{h}_i = \mathcal{H}(\sigma_i)$
 2. compute $b_1 \leftarrow \text{DAP.TxVfy}(tx, I_{pub})$
 3. compute $b_2 \leftarrow \text{CLRS.Vfy}(R, ct, \sigma, pk_L)$
 4. compute $b_3 \leftarrow \mathcal{V}_v((tx, c, I_{pub}, \tilde{h}, g, h), \pi_v, crs)$
 5. compute $b_4 \leftarrow \mathcal{V}_{log}(u, \pi_{log}, pk_L)$
 6. if $b_1 \wedge b_2 \wedge b_3 \wedge b_4 = 1$ and all public keys in the ring R are from registered users output 1 else output 0

- Consensus nodes seal valid transactions into blocks.

IV-Regulation Phase

In the regulation phase, Filter screens out suspicious transactions and submits the report on suspicious transactions to Supervisor. Once the report is received, Supervisor obtains the sender's public key and payment amounts of suspicious transactions.

Filter:

DAPCR.Extract:

- Inputs:
 1. regulated transaction rtx_i
 2. Filter's private key sk_F
- Outputs:
 1. pseudonym nym_i
 2. amount tag tag_{v_i}
- Filter extracts the pseudonym and the amount tag from a transaction:
 1. compute $nym_i \leftarrow \text{CLRS.Ext}(\sigma_i, sk_L)$
 2. compute $tag_{v_i} = c_i u_i^{-1/sk_L} = g^{v_i} h^{w_i}$

DAPCR.Detect:

- Inputs:
 1. pseudonym nym and its corresponding upper limit tag tag_μ
 2. Filter's private key $sk_F = sk_L$
 3. ledger which denoted all transactions sealed in blocks during a trading period
- Outputs:
 1. a set $S = \{rtx_0, rtx_1, \dots, rtx_{n-1}\}$ of transactions submitted by nym
 2. susp/ \perp
- Filter executes the following steps to determine if the transaction behavior of a user with pseudonym nym complies with the transaction rules:
 1. extract pseudonyms from all transactions in ledger
 2. link transactions submitted by nym and get a set $S = \{rtx_0, rtx_1, \dots, rtx_{n-1}\}$
 2. extract amount tags $tag_{v_i} |_{i=0}^{n-1}$ from all transactions in S
 3. compute $tag_{sum} = \prod_{i=0}^{n-1} tag_{v_i}$
- If $tag_\mu = tag_{sum}$, Filter outputs (S, \perp) else outputs $(S, susp)$.

DAPCR.Report:

- Inputs:
 1. $(S, susp)$ where S is a set of transactions
 2. pseudonym nym
 3. Filter's public-private key pair (pk_F, sk_F)
- Outputs: report rpt of suspicious transactions
- Filter executes the following steps to generate a report:
 1. compute $\pi_{dec} \leftarrow \mathcal{P}_{dec}((ct_i |_{i=0}^{n-1}, nym, pk_L), sk_L, h)$
 2. compute $\pi_{sum} \leftarrow \mathcal{P}_{dec}((\prod_{i=0}^{n-1} c_i, \prod_{i=0}^{n-1} u_i, tag_{sum}, pk_L), sk_L, h)$
 3. set rpt = $(S, nym, \pi_{dec}, tag_{sum}, \pi_{sum})$ to \mathcal{S}
- Filter sends rpt to Supervisor.

Supervisor:

DAPCR.Recover:

- Inputs:
 1. report rpt of suspicious transactions
 2. Filter's public key pk_F
 3. Supervisor's private key sk_S
- Outputs:
 1. sender's public key uk
 2. sender's total payment amount v_{sum}
- Supervisor executes the following steps to verify the report:
 1. compute $b_1 \leftarrow \mathcal{V}_{dec}((ct_i |_{i=0}^{n-1}, nym, pk_L), \pi_{dec}, h)$
 2. compute $b_2 \leftarrow \mathcal{V}_{dec}((\prod_{i=0}^{n-1} c_i, \prod_{i=0}^{n-1} u_i, tag_{sum}, pk_L), \pi_{sum}, h)$
 3. if $b_1 \wedge b_2 = 1$, the report is valid else is invalid
- If the report is valid, \mathcal{S} computes $uk \leftarrow \text{CLRS.Open}(nym, sk_O)$ and then requires the user with public key uk to submit v_{sum} and w' satisfies $tag_{sum} = g^{v_{sum}} h^{w'}$.