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

Tianyu Zhaolu, Zhiguo Wan, 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 $1231.2\,\mathrm{ms}$, representing an increase of less than $50\,\%$ compared to ZETH. Additionally, the time overhead for transaction verification is only $1.2\,\mathrm{ms}$.**

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 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, distributed 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 [6]. Chainalysis[1] 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. FATF[2] and APG[3] 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.

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

First, DAPCR is a highly generic solution to achieve the regulation of any DAP scheme. It exhibits compatibility not only with UTXO blockchains but also with account-based blockchains. To realize the universality of DAPCR, we extend the original transaction in the DAP scheme using zk-SNARK.

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 Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China, and also with the Guangxi Key Laboratory of Cryptography and Information Security, Guilin 541004, China (e-mail: wanghuaqun@aliyun.com).

[1]Chainalysis: https://www.chainalysis.com/

[2]Financial Action Task Force: https://www.fatf-gafi.org/

[3]Asia/Pacific Group on Money Laundering: https://apgml.org/

TABLE I: Properties of DAPCR and related works

Scheme	Restricted Regulation	Non-interaction	Amount Aggregation	Universality
DAPCR	✓	✓	✓	✓
[7]	×	✓	×	×
[8]	×	✓	×	×
[9]	✓	✓	×	×
[10]	×	×	×	×
[11]	×	×	×	×

By adding additional regulatable fields to the transaction, the regulator can effectively enforce the regulation, irrespective of the original DAP transaction.

Moreover, we decentralize regulatory power between two entities, similar to the separation of powers in governments, to achieve restrictive regulation. In particular, the regulatory power is divided between two regulators: Filter and Supervisor. 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 accomplish the supervision task.

Furthermore, we design the amount tag to achieve efficient aggregation of transaction amounts. Filter can extract the amount tag from each transaction and aggregate amount tags to determine whether the total amount exceeds the preset upper limit. During this process, no additional privacy is exposed. To clarify the advantages of our scheme, the properties of DAPCR and related works are shown in Table I.

A. Paper Contributions

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

1. We propose the DAPCR scheme with the following advantages: ① Universality ensures compatibility with existing DAP schemes. ② Collaborative regulation prevents abuse of power and privacy breaches. ③ Efficient aggregation of payment amounts via amount tags. To our best knowledge, DAPCR is the first universal collaborative regulation scheme for DAP schemes.
2. 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.
3. We present security definitions of CLRS and DAPCR and provide the security analysis for them.
4. We evaluate the performance of DAPCR on both local devices and the Fabric network. The time cost for transaction generation is about $1231.2\,\mathrm{ms}$ and that of transaction verification is about $12.5\,\mathrm{ms}$. These experimental results indicate the effectiveness of DAPCR.

B. 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 performance analysis for DAPCR in Section VII and the security analysis for DAPCR in Appendix. Section VIII reviews the recent literature relevant to DAP and ring signatures. Section IX concludes our work.

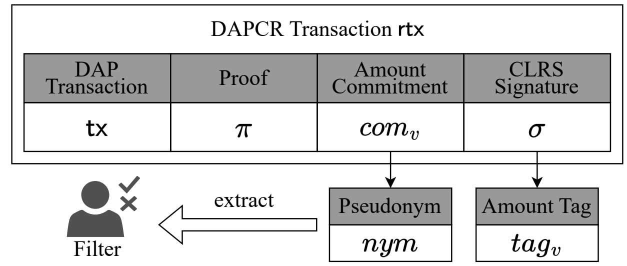


Fig. 1: Transaction Structure.

II. Overview

In this section, we briefly describe the transaction structure, the transaction policy, and the collaborative regulation between Filter and Supervisor in DAPCR.

1) Transaction Structure: As shown in Fig. 1, a regulatable transaction rtx consists of a DAP transaction tx, an amount commitment com_v, and a CLRS signature σ. Additionally, users also generate a proof for that the opening of com_v is the payment amount v of tx, and attach it to the regulatable transaction. Filter can extract the sender's pseudonym nym from the signature σ and the payment amount tag tag_v from the commitment com_v, both of which are the basis for determining whether users comply with the transaction policy. Due to not considering the original DAP transaction tx during the regulatory process, the DAPCR scheme exhibits universality.

2) Transaction Policy: A commonly employed policy in digital payment systems involves assigning each user an upper limit, restricting their total payment amount to not exceed this limit within a given period [12], [13]. We make slight modifications to this policy and apply it to DAPCR:

In DAPCR, each user is allocated a privacy payment limit for each trading period. At the end of a trading period, the total amount of privacy payments must equal the privacy payment limit. If the total amount is lower than the limit, users are required to publish a self-transaction to make up for the difference in the amount. When a user reaches their privacy payment limit but still wishes to engage in transactions, they can publish public transactions. If a user's total payment amount exceeds their payment limit, Filter will mark their transactions during this trading period as suspicious and include these transactions in regulation.

3) Collaborative Regulation: To address the issues of potential power abuse and privacy leakage that may arise from a single entity regulating anonymous transactions, we decentralize the regulatory authority to two entities: Filter ($\mathcal{F}$) and Supervisor ($\mathcal{S}$).

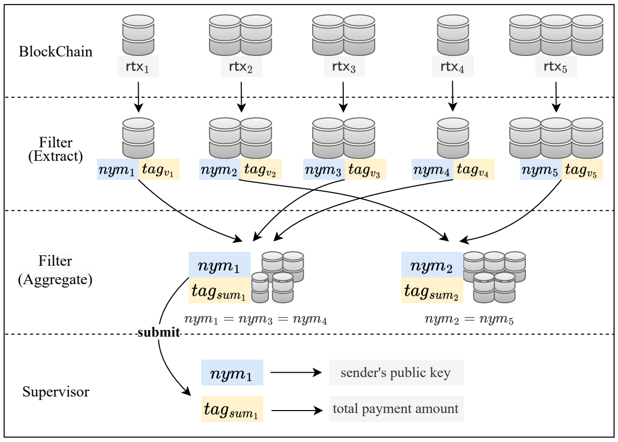


Fig. 2: Collaborative regulation between Filter and Supervisor.

Filter. To screen out suspicious transactions in the ledger, $\mathcal{F}$ utilizes its extracting key to extract the pseudonym nym and the amount tag tag_v from each transaction on the blockchain. The unique correspondence between pseudonyms and users' public keys enables $\mathcal{F}$ to establish links among transactions from the same user. Furthermore, the amount tag tag_v represents the payment amount v of rtx. Multiple amount tags can be aggregated into a total amount tag tag_{sum} for regulatable transactions.

To determine whether the user with the pseudonym nym complies with the transaction policy, Filter performs the following steps:

1. Filter extracts the sender's pseudonym from each transaction on the blockchain, and screens out transactions submitted by the user with the pseudonym nym.
2. For the transactions submitted by nym, Filter extracts amount tags from these transactions and aggregates amount tags into the total amount tag tag_{sum}.
3. Filter determines whether nym's total payment amount exceeds their limit according to tag_{sum}.
4. If exceeded, $\mathcal{F}$ submits the pseudonym nym and the total amount tag tag_{sum} to $\mathcal{S}$.

Note that $\mathcal{F}$ cannot directly obtain the sender's identity and the payment amount from the pseudonym and the amount tag.

Supervisor. Once receiving nym and tag_{sum} from $\mathcal{F}$, $\mathcal{S}$ can recover the user's public key from its pseudonym and obtain the payment amount from the total amount tag using $\mathcal{S}$'s private key. Notably, $\mathcal{S}$ cannot extract pseudonyms or amount tags from transactions. Both regulators must collaborate to accomplish the supervision task, thereby achieving a decentralization of regulatory power between $\mathcal{F}$ and $\mathcal{S}$.

III. Preliminaries

A. Decentralized Anonymous Payments

A DAP scheme such as Zerocash [4], Monero [3] or Zether [5] can be highly simplified into four algorithms as follows.

- $\mathsf{Setup}(1^\lambda) \rightarrow pp$. This algorithm takes a security parameter λ as input and outputs a public parameter pp, which is an implicit input for other algorithms.
- $\mathsf{AddrGen}(pp) \rightarrow (addr, s)$. This algorithm outputs a user's address $addr$ and secret key s.
- $\mathsf{TxGen}(addr_S, addr_R, v, s, I_{\mathsf{pub}}, I_{\mathsf{pri}}) \rightarrow \mathsf{tx}$. The algorithm takes as input a sender's address $addr_S$, a recipient'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.
- $\mathsf{TxVfy}(\mathsf{tx}, I_{\mathsf{pub}}) \rightarrow 0/1$. 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.

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.

- $\mathcal{G}(1^\lambda, \mathcal{R}) \rightarrow crs$. The 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 the following properties.

1. *Completeness*. For any $(\phi, \varpi) \in \mathcal{R}$,
$$Pr\left[\begin{array}{c} crs \leftarrow \mathcal{G}(1^\lambda, \mathcal{R}); \pi \leftarrow \mathcal{P}(\phi, \varpi, crs): \\ 1 \leftarrow \mathcal{V}(\phi, \varpi, crs) \end{array}\right] = 1.$$
2. *Zero-knowledge* [14]. No information other than the truth of the statement is leaked. For any $(\phi, \varpi) \in \mathcal{R}$, any probabilistic polynomial-time adversary $\mathcal{A}$ and a polynomial-time simulator $\mathcal{S} = (\mathcal{G}_{sim}, \mathcal{P}_{sim})$,
$$Pr\left[\begin{array}{c} (crs, \tau) \leftarrow \mathcal{G}_{sim}(1^\lambda, \mathcal{R}); \pi' \leftarrow \mathcal{P}_{sim}(\phi, crs, \tau): \\ \mathcal{A}(\pi', crs, \tau, \mathcal{R}) = 1 \end{array}\right] -$$
$$Pr\left[\begin{array}{c} crs \leftarrow \mathcal{G}(1^\lambda, \mathcal{R}); \pi \leftarrow \mathcal{P}(\phi, \varpi, crs): \\ \mathcal{A}(\pi, crs, \tau, \mathcal{R}) = 1 \end{array}\right] \leq \mathsf{negl}(\lambda).$$
3. *Knowledge Soundness* [15]. A secure NIZK scheme cannot prove a false statement. For any probabilistic polynomial-time adversary $\mathcal{A}$ and a probabilistic polynomial-time extractor $\mathcal{E}$,
$$Pr\left[\begin{array}{c} crs \leftarrow \mathcal{G}(1^\lambda, \mathcal{R}); \\ (\phi, \pi, \varpi) \leftarrow (\mathcal{A} \| \mathcal{E})(crs, \mathcal{R}): \\ 1 \leftarrow \mathcal{V}(\phi, \pi, crs) \wedge (\phi, \varpi) \notin \mathcal{R} \end{array}\right] \leq \mathsf{negl}(\lambda).$$

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 [16]: 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 protocols [17], also known as SoK, allows a signer to publish signatures on behalf of any NP-relation $\mathcal{R}$. A SoK protocol Π_{SoK} consists of three algorithms as follows.

- $\mathsf{SoK.Setup}(1^\lambda, \mathcal{R}) \rightarrow pp$. The algorithm takes a security parameter λ and an NP-relation $\mathcal{R}$ as input, and outputs a public parameter pp.
- $\mathsf{SoK.Sign}(m, \phi, \varpi, pp) \rightarrow \pi$. The 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.
- $\mathsf{SoK.Vfy}(m, \phi, \pi, pp) \rightarrow 0/1$. The 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 $(\phi, \varpi) \in \mathcal{R}$,
$$Pr\left[\begin{array}{c} pp \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}); \pi \leftarrow \mathsf{Sign}(m, \phi, \varpi, pp): \\ 1 \leftarrow \mathsf{Vfy}(m, \phi, \pi, pp) \end{array}\right] = 1$$
2. *Simulatability*. There exists a probabilistic polynomial-time simulator $\mathsf{Sim} = (\mathsf{Setup}_{sim}, \mathsf{Sign}_{sim})$ such that for any probabilistic polynomial-time adversary $\mathcal{A}$,
$$\begin{aligned} &Pr\left[(pp, \tau) \leftarrow \mathsf{Setup}_{sim}(1^\lambda, R) : \mathcal{A}^{\mathsf{Sim}}(pp) = 1\right] - \\ &Pr\left[pp \leftarrow \mathsf{Setup}(1^\lambda, R) : \mathcal{A}^{\mathsf{Sign}}(pp) = 1\right] \leq \mathsf{negl}(\lambda), \end{aligned}$$
where Sim takes (m, ϕ, ϖ) as input and outputs $\pi \leftarrow \mathsf{Sign}_{sim}(m, \phi, \tau, pp)$ if $(\phi, \varpi) \in \mathcal{R}$ and $\perp$ otherwise. In other words, that interaction with Setup and Sign is indistinguishable from that with Setup_{sim} and Sign_{sim}.
3. *Extractability*. There exists a simulator and an extractor algorithm Ext such that for any probabilistic polynomial-time adversary $\mathcal{A}$,
$$Pr\left[\begin{array}{c} (pp, \tau) \leftarrow \mathsf{Setup}_{sim}(1^\lambda, R); \\ (m, \phi, \pi) \leftarrow \mathcal{A}^{\mathsf{Sim}}(pp); \varpi \leftarrow \mathsf{Ext}(m, \phi, \pi, \tau, pp): \\ (\phi, \varpi) \in \mathcal{R} \vee (m, \phi, \pi) \in L_{sim} \vee \\ 0 \leftarrow \mathsf{Vfy}(m, \phi, \pi, pp) \end{array}\right] = 1,$$
where L_{sim} is a list of queries to Sign_{sim}.

Note that NIZK and SoK protocols in the random oracle model can be efficiently realized by applying the Fiat-Shamir transform [18] to Σ-protocols.

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 3 depicts the system framework of DAPCR, which consists of five entities as follows:

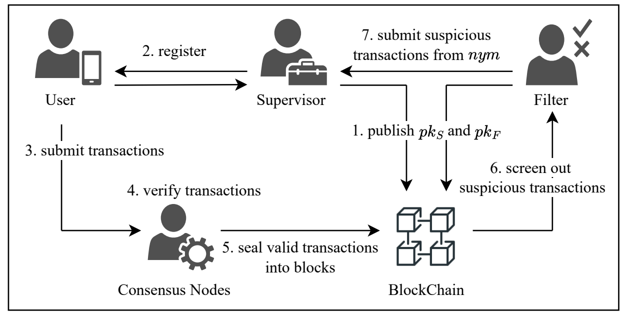


Fig. 3: System framework.

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*. Only with the permission of Supervisor can users join the blockchain and obtain a pseudonym. Supervisor can recover the user's public key from the pseudonym. We assume that users may take any malicious action.
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 $\mathcal{S}$.

The workflow of the system is as follows: ① Filter and Supervisor publish their public key on the blockchain, and a user generates their public-private key pair. ② Users interact with Supervisor to complete registration. ③ A user generates a regulatable transaction and submits it to consensus nodes. ④ Consensus nodes verify the validity of transactions, ⑤ and seal valid transactions into blocks. ⑥ Filter screens out suspicious transactions on the blockchain, ⑦ and submits them to Supervisor who can obtain the sender's address and payment amounts with the assistance of Filter.

2) Formal Definition: A DAPCR scheme is composed by the following probabilistic polynomial-time algorithms.

- $param \leftarrow \mathsf{Setup}(1^\lambda)$. This algorithm takes as input a security parameter λ and outputs a public parameter $param$, which is implicit input to other algorithms.
- $(pk_F, sk_F) \leftarrow \mathsf{FInit}(param)$. $\mathcal{F}$ executes this algorithm to generate their public-private key pair (pk_F, sk_F).
- $(pk_S, sk_S) \leftarrow \mathsf{SInit}(param)$. $\mathcal{S}$ executes this algorithm to generate their public-private key pair (pk_S, sk_S).
- $(uk, sk) \leftarrow \mathsf{KeyGen}(pk_S)$. This algorithm takes pk_S as input and outputs a public-private key pair (uk, sk) for a user.
- $(\mu, tag_\mu, nym) \leftarrow \Sigma_{\mathsf{reg}}(\mathcal{U} : pk_S, uk; \mathcal{S} : pk_S, sk_S)$. $\mathcal{S}$ and a user $\mathcal{U}$ execute the interactive protocol Σ_{reg} for user

registration. This protocol takes uk, pk_S and sk_S as input and outputs the upper limit of total payment amounts μ, a tag tag_μ of μ and the user's pseudonym nym.

- $\mathsf{rtx} \leftarrow \mathsf{RtxGen}(addr_S, addr_R, v, s, I_{\mathsf{pub}}, I_{\mathsf{pri}}, R, sk, pk_F)$. The algorithm takes as input a tuple $(addr_S, addr_R, v, s, I_{\mathsf{pub}}, I_{\mathsf{pri}}, R, sk, pk_F)$ and outputs a regulatable transaction rtx, where $R = \{uk_0, uk_1, ..., uk_{n-1}\}$, $uk = uk_j \in R$ and sk is a private key corresponding to uk.
- $0/1 \leftarrow \mathsf{Verify}(\mathsf{rtx}, pk_F, R, I_{\mathsf{pub}})$. This algorithm, which verify the validity of a regulatable transaction rtx, takes rtx, pk_F, R and I_{pub} as input, and outputs 1 if rtx is valid or 0 otherwise.
- $(tag_v, nym) \leftarrow \mathsf{Extract}(\mathsf{rtx}, sk_F)$. 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.
- $(S, \mathsf{susp}/\perp) \leftarrow \mathsf{Detect}(nym, tag_\mu, sk_F, \mathsf{ledger})$. This algorithm takes as input a pseudonym nym, a tag tag_μ of 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 $\perp$.
- $\mathsf{rpt} \leftarrow \mathsf{Report}((S, \mathsf{susp}), nym, pk_F, sk_F)$. 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.
- $(uk, v_{sum})/\perp \leftarrow \mathsf{Recover}(\mathsf{rpt}, pk_F, sk_S)$. 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 serves as the foundational component of DAPCR. To generate a signature, the user needs to specify a linker and an opener. The authorized linker has the ability to extract the user's pseudonym from the signature and establish connections between signatures from the same user using the pseudonym. The authorized opener can retrieve the user's public key from the pseudonym.

The linker is only aware of the linking relationships among the signatures, while the signatures remain anonymous to the linker. On the other hand, the opener, without the assistance of the linker, is unable to retrieve the user's public key from the signature. Thus, CLRS effectively prevents the abuse of identity-tracing capabilities. To clarify the advantages of the proposed scheme, the properties of CLRS and related signatures are shown in Table II.

A CLRS scheme consists of the following algorithms:

- $pp \leftarrow \mathsf{Setup}(1^\lambda)$. The algorithm takes a security parameter λ as input, and generates a public parameter pp which is implicitly input to other algorithms. Note that this algorithm is transparent.

TABLE II: Properties of CLRS and related signatures

Scheme	Controllable Linkability[1]	Restricted traceability[2]	Transparency	Group Manager
CLRS	✓	✓	✓	×
[19]	×	×	✓	×
[20]	✓	×	×	✓
[21]	×	×	✓	×
[22]	×	×	✓	×

[1] Controllable linkability means that only the designated user can link signatures from the same signer.
[2] Restricted traceability means that the opener can only trace the identity of the signer with the assistance of the linker.

- $(pk_L, sk_L) \leftarrow \mathsf{LKGen}(pp)$. The algorithm outputs a public-private key pair (pk_L, sk_L) for a linker.
- $(pk_O, sk_O) \leftarrow \mathsf{OKGen}(pp)$. The algorithm outputs a public-private key pair (pk_O, sk_O) for an opener.
- $(uk, sk) \leftarrow \mathsf{UKGen}(pk_O)$. The algorithm takes an opener's public key pk_O as input and outputs a user's public-private key pair (uk, sk).
- $\sigma/\perp \leftarrow \mathsf{Sign}(R, m, pk_L, sk)$. The algorithm takes a ring $R = \{uk_0, uk_1, ..., uk_{n-1}\}$, 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 $j \in \{0, 1, ..., n-1\}$, the algorithm outputs a signature σ of (R, m) else outputs $\perp$.
- $0/1 \leftarrow \mathsf{Vfy}(R, m, \sigma, pk_L)$. 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.
- $nym \leftarrow \mathsf{Ext}(\sigma, sk_L)$. The algorithm takes a valid signature σ and a linker's private key sk_L as input and outputs the signer's pseudonym nym.
- $\mathsf{link}/\mathsf{unlink} \leftarrow \mathsf{Link}(\sigma_0, \sigma_1, sk_L)$. 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.
- $uk \leftarrow \mathsf{Open}(nym, sk_O)$. 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.

- $\pi \leftarrow \mathsf{Prove}(\sigma_i|_{i=0}^{n-1}, nym, pk_L, sk_L)$. 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.
- $0/1 \leftarrow \mathsf{Judge}(\sigma_i|_{i=0}^{n-1}, nym, \pi, pk_L)$. 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

Anonymity Experiment $\mathsf{exp}^{\mathsf{Ano}}_{\mathcal{A}_i}(\lambda)$:

$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_L, sk_L) \leftarrow \mathsf{LKGen}(pp)$
$(pk_O, sk_O) \leftarrow \mathsf{OKGen}(pp)$
$(R, m, uk_0, uk_1) \leftarrow \mathcal{A}_i(sk_{\mathcal{A}_i}, pk_L, pk_O)$
$b \leftarrow_\$ \{0,1\}$
$\sigma_b \leftarrow \mathsf{Sign}(R, m, pk_L, sk_b)$
$b' \leftarrow \mathcal{A}_i(\sigma_b, sk_{\mathcal{A}_i}, pk_L, pk_O)$
if $b' = b$ **then** output 1
else output 0
end if

Nym-extractability Experiment $\mathsf{exp}^{\mathsf{NExt}}_{\mathcal{A}_i}(\lambda)$:

$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_L, sk_L) \leftarrow \mathsf{LKGen}(pp)$
$(pk_O, sk_O) \leftarrow \mathsf{OKGen}(pp)$
$(R, m, \sigma) \leftarrow \mathcal{A}_i(sk_{\mathcal{A}_i}, pk_L, pk_O)$
$nym \leftarrow \mathsf{Ext}(\sigma, sk_L)$
$\pi \leftarrow \mathsf{Prove}(\sigma, nym, pk_L, sk_L)$
$b_1 \leftarrow \mathsf{Vfy}(R, m, \sigma, pk_L)$
$b_2 \leftarrow \mathsf{Judge}(\sigma, nym, \pi, pk_L)$
if $b_1 = 1 \wedge b_2 = 0$ **then** output 1
else output 0
end if

Unforgeability Experiment I $\mathsf{exp}^{\mathsf{Uf1}}_{\mathcal{A}_i}(\lambda)$:

$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_L, sk_L) \leftarrow \mathsf{LKGen}(pp)$
$(pk_O, sk_O) \leftarrow \mathsf{OKGen}(pp)$
$R \leftarrow \mathcal{C}(pk_O)$
$(m, \sigma) \leftarrow \mathcal{A}(R, sk_{\mathcal{A}_i}, pk_L, pk_O)$
$b \leftarrow \mathsf{Vfy}(R, m, \sigma, pk_L)$
output b

Unforgeability Experiment II $\mathsf{exp}^{\mathsf{Uf2}}_{\mathcal{A}_1}(\lambda)$:

$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_L, sk_L) \leftarrow \mathsf{LKGen}(pp)$
$(pk_O, sk_O) \leftarrow \mathsf{OKGen}(pp)$
$uk \leftarrow \mathcal{C}(pk_O)$
$(R, m, \sigma) \leftarrow \mathcal{A}_1(uk, sk_L, pk_L, pk_O, uk)$
$nym \leftarrow \mathsf{Ext}(\sigma, sk_L)$
$b \leftarrow \mathsf{Vfy}(R, m, \sigma, pk_L)$
if $b = 1 \wedge uk \in R \wedge$ $uk \leftarrow \mathsf{Open}(nym, sk_O)$ **then**
    output 1
else
    output 0
end if

Nym-soundness Experiment $\mathsf{exp}^{\mathsf{NS}}_{\mathcal{A}_i}(\lambda)$:

$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_L, sk_L) \leftarrow \mathsf{LKGen}(pp)$
$(pk_O, sk_O) \leftarrow \mathsf{OKGen}(pp)$
if $i = 1$ **then**
    $sk_{\mathcal{A}_i} = sk_L$
else
    $sk_{\mathcal{A}_i} = sk_O$
end if
$(R, m, \sigma, nym_0, nym_1, \pi_0, \pi_1) \leftarrow \mathcal{A}_i(sk_{\mathcal{A}_i}, pk_L, pk_O)$
$b_1 \leftarrow \mathsf{Vfy}(R, m, \sigma, pk_L)$
$b_2 \leftarrow \mathsf{Judge}(\sigma, nym_0, \pi_0, pk_L)$
$b_3 \leftarrow \mathsf{Judge}(\sigma, nym_1, \pi_1, pk_L)$
if $b_1 = b_2 = b_3 = 1 \wedge nym_0 \neq nym_1 \wedge (nym_i^{sk_O}, \cdot) \in R$ **then**
    output 1
else
    output 0
end if

Fig. 4: Security experiments for CLRS.

that an adversary cannot compromise both the opener and the linker at the same time, because with sk_L and sk_O, an adversary would be able to trace the signer of any valid signature. We believe that this assumption is realistic. In addition, we define that $sk_{\mathcal{A}_1} = sk_L$ and $sk_{\mathcal{A}_2} = sk_O$.

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
$$Pr\left[\begin{array}{c} pp \leftarrow \mathsf{Setup}(1^\lambda); uk \leftarrow \mathsf{UKGen}(sk); \\ (pk_O, sk_O) \leftarrow \mathsf{OKGen}(pp); \\ (pk_L, sk_L) \leftarrow \mathsf{LKGen}(pp); \\ \sigma \leftarrow \mathsf{Sign}(R, m, pk_L, pk_O, sk): \\ \text{If } uk \in R \text{ then } 1 \leftarrow \mathsf{Vfy}(R, m, \sigma, pk_L) \end{array}\right] = 1$$
2. *Unforgeability*. $\mathcal{A}_1$ (or $\mathcal{A}_2$) 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 probabilistic polynomial-time adversary $\mathcal{A}_i$, $Pr[\mathsf{exp}^{\mathsf{Uf1}}_{\mathcal{A}_i}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$ and $Pr[\mathsf{exp}^{\mathsf{Uf2}}_{\mathcal{A}_1}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$
3. *Anonymity*. CLRS satisfies anonymity if for any probabilistic polynomial-time adversary $\mathcal{A}_i$, $|Pr[\mathsf{exp}^{\mathsf{Ano}}_{\mathcal{A}_i}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$.
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 probabilistic polynomial-time adversary $\mathcal{A}_i$, $Pr[\mathsf{exp}^{\mathsf{NExt}}_{\mathcal{A}_i}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.
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 probabilistic polynomial-time adversary $\mathcal{A}_i$, $Pr[\mathsf{exp}^{\mathsf{NS}}_{\mathcal{A}_i}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Security experiments for CLRS are presented in Fig. 4.

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. In addition, we define that $sk_{\mathcal{A}'_1} = sk_F$ and $sk_{\mathcal{A}'_2} = sk_S$.

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 regulatable transaction reveals no information about transaction privacy. For $\mathcal{A}'_1$, they can link regulatable transactions from the same user but cannot obtain any additional information. We say that DAPCR satisfies indistinguishability if, for $i \in \{1,2\}$, any probability polynomial-time adversary $\mathcal{A}'_i$ and a security parameter λ, the advantage of $\mathcal{A}'_i$ in winning the indistinguishability experiment is negligible, i.e. $|Pr[\mathsf{exp}^{\mathsf{Ind}}_{\mathcal{A}'_i}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$.
2. *$\mathcal{F}$-extractability*. For a valid regulatable transaction, $\mathcal{F}$ can extract the sender's pseudonym and the payment amount tag from the transaction. DAPCR satisfies $\mathcal{F}$-extractability if $Pr[\mathsf{exp}^{\mathsf{FExt}}_{\mathcal{A}'_i}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.
3. *$\mathcal{F}$-soundness*. For a valid regulatable transaction, $\mathcal{F}$ cannot extract two different pseudonyms or tags from the transaction. DAPCR satisfies $\mathcal{F}$-soundness if $Pr[\mathsf{exp}^{\mathsf{FS}}_{\mathcal{A}'_1}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.
4. *Balance.* The amount paid by any probabilistic polynomial-time adversaries cannot exceed their balance.
5. *Non-malleability.* No probabilistic polynomial-time adversary possesses the capability to modify the information contained within a regulatable transaction rtx.

Security experiments in DAPCR are presented in Fig. 5.

Indistinguishability Experiment $\mathsf{exp}^{\mathsf{Ind}}_{\mathcal{A}'_i}(\lambda)$:

$param \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_F, sk_F) \leftarrow \mathsf{FInit}(param)$
$(pk_S, sk_S) \leftarrow \mathsf{SInit}(param)$
$(T_0, T_1, R) \leftarrow \mathcal{A}'_i(sk_{\mathcal{A}'_i}, pk_S, pk_F)$
    where $T_j = (addr_{S_j}, addr_{R_j}, v_j, uk_j)$
    and $uk_0, uk_1 \in R$
$\mathsf{rtx}_b \leftarrow \mathcal{C}(T_b, R, sk_b, \cdot)$
$b' \leftarrow \mathcal{A}'_i(\mathsf{rtx}_b, sk_{\mathcal{A}'_i}, pk_S, pk_F)$
if $b = b'$ **then**
    output 1
else
    output 0
end if

$\mathcal{F}$-extractability Experiment $\mathsf{exp}^{\mathsf{FExt}}_{\mathcal{A}'_i}(\lambda)$:

$param \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_F, sk_F) \leftarrow \mathsf{FInit}(param)$
$(pk_S, sk_S) \leftarrow \mathsf{SInit}(param)$
$(R, \mathsf{rtx}, v, r) \leftarrow \mathcal{A}'_i(sk_{\mathcal{A}'_i}, pk_S, pk_F)$
$(nym, tag_v) \leftarrow \mathsf{Extract}(\mathsf{rtx}, sk_F)$
if $1 \leftarrow \mathsf{Verify}(\mathsf{rtx}, pk_F, R, I_{\mathsf{pub}}) \wedge$
    $tag_v = \mathsf{Com}(v, r) \wedge v = \mathsf{amount}(\mathsf{rtx}) \wedge$
    $nym = \mathsf{nym}(uk) \wedge uk \in R$ **then**
    output 1
else
    output 0
end if

$\mathcal{F}$-soundness Experiment $\mathsf{exp}^{\mathsf{FS}}_{\mathcal{A}'_i}(\lambda)$:

$param \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_F, sk_F) \leftarrow \mathsf{FInit}(param)$
$(pk_S, sk_S) \leftarrow \mathsf{SInit}(param)$
$(R, \mathsf{rtx}, T_0, T_1) \leftarrow \mathcal{A}'_i(sk_{\mathcal{A}'_i}, pk_S, pk_F)$
    where $T_j = (nym_j, tag_j, \pi_j^{nym}, \pi_j^{tag})$
if $\forall j, 1 \leftarrow V_{nym}(\mathsf{rtx}, nym_j, \pi_j^{nym}) \vee$
    $\forall j, 1 \leftarrow V_{tag}(\mathsf{rtx}, tag_j, \pi_j^{tag})$
    $1 \leftarrow \mathsf{Verify}(\mathsf{rtx}, pk_F, R, I_{\mathsf{pub}})$ **then**
    output 1
else
    output 0
end if

Fig. 5: Security experiments for DAPCR.

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.

1. $\Pi_{mem} = (\mathcal{G}_{mem}, \mathcal{P}_{mem}, \mathcal{V}_{mem})$ represents a NIZK protocol for the relation
$$\mathcal{R}_{mem} = \left\{ \begin{array}{c} ((c_0, c_1, ..., c_{n-1}), (l, r), (g, h)) : \\ \forall i, c_i \in \mathbb{G} \wedge c_l = g^0 h^r \wedge \\ l \in \{0, 1, ..., n-1\} \wedge r \in \mathbb{Z}_q^* \end{array} \right\}.$$
Σ-protocols for the above relation called *one-out-of-many proofs* [23] 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.
2. $\Pi_{enc} = (\mathcal{G}_{enc}, \mathcal{P}_{enc}, \mathcal{V}_{enc})$ represents a NIZK protocol for the relation
$$\mathcal{R}_{enc} = \left\{ \begin{array}{c} ((c, u, pk), (\alpha, \beta), (g, h)) : \\ c = g^\alpha h^\beta \wedge u = pk^\beta \wedge pk, g, h \in \mathbb{G} \wedge \alpha, \beta \in \mathbb{Z}_q^* \end{array} \right\}.$$
The protocol allows one to prove the correctness of an ElGamal ciphertext $ct = (c, u)$ given a specific public key pk.
3. $\Pi_{dec} = (\mathcal{G}_{dec}, \mathcal{P}_{dec}, \mathcal{V}_{dec})$ represents a NIZK protocol for the relation
$$\mathcal{R}_{dec} = \left\{ \begin{array}{c} ((ct_i = (c_i, u_i)|_{i=0}^{n-1}, m, pk), sk, h) : \\ \forall i, c_i = m \cdot u_i^{\frac{1}{sk}} \wedge pk = h^{sk} \wedge \\ m, pk, h, c_i, u_i \in \mathbb{G} \wedge sk \in \mathbb{Z}_q^* \end{array} \right\}$$
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.

A. 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 $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q^*$. 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 $\pi_{enc} \leftarrow \mathcal{P}_{enc}((c, pk, pk_O), (r, sk), (h, g))$. The user outputs $uk = (pk, c, \pi_{enc})$.
One calculates $b \leftarrow \mathcal{V}_{enc}((c, pk, pk_O), \pi_{enc}, (g, h))$ 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 $R = \{uk_0, ..., uk_{n-1}\}$ 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 $R' = \{c'_0, c'_1, ..., c'_{n-1}\}$ 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 $((com, K, pk_L), (sk, k)) \in \mathcal{R}_{enc}$. The formula 1 is a SoK scheme transformed from the interactive protocol Σ_{enc} presented in Appendix A-B. 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.

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

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 $i \in \{0, 1, ..., n-1\}$, 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 $i \in \{0, 1, ..., n-1\}$, 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.

Due to the above NIZK protocols without trusted setup, the proposed CLRS scheme is transparent. Therefore, the CLRS scheme can be used in a trustless networking environment [24].

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

PROOF. For a ring $R = \{uk_0, uk_1, ..., uk_{n-1}\}$, 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 $\sigma \leftarrow \mathsf{CLRS.Sign}(R, m, pk_L, sk)$.

To verify the validity of σ, the verifier first computes $c_i' = c_i/com$ for $i \in \{0, 1, ..., n-1\}$ and gets $R' = \{c_0', c_1', ..., c_{n-1}'\}$. 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. □

VI. DAP with Collaborative Regulation

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

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

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

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.

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

1) Preparation Phase: Consensus nodes execute the initialization algorithm, and Supervisor $\mathcal{S}$ registers users in the DAPCR system.

Setup. Consensus nodes execute

$$(g, h, \mathcal{H}) \leftarrow \mathsf{CLRS.Setup}(1^\lambda),$$
$$crs \leftarrow \mathcal{G}_v(1^\lambda, \mathcal{R}_v),$$

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

FInit. $\mathcal{F}$ 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 $(uk, sk) \leftarrow \mathsf{CLRS.UKGen}(pk_O)$ and publishes their public key uk.

Register. A user $\mathcal{U}$ and Supervisor $\mathcal{S}$ engage in an interactive protocol to register the user.

1. $\mathcal{U}$ randomly samples $\beta \in \mathbb{Z}_q^*$, and computes $B = g^\beta$ and $w = \mathcal{H}(pk_O^\beta|uk)$. Then $\mathcal{U}$ sends (B, uk) to $\mathcal{S}$.
2. Once (B, uk) is received, $\mathcal{S}$ sets the upper limit $\mu \in \mathbb{Z}_q^*$ of total payment volume for $\mathcal{U}$, and computes $w = \mathcal{H}(B^{sk_O}|uk)$, $tag_\mu = g^\mu h^w$ and $nym = pk^{\frac{1}{sk_O}}$. Then, $\mathcal{S}$ adds $(uk, \mu, nym, \mathsf{tag}_\mu)$ to the list $L_\mathcal{S}$, sends μ to $\mathcal{U}$ through a secure channel.

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: regulatable transactions with privacy preservation and public transactions. In a trading period, the total amount paid by a user through regulatable transactions cannot exceed their privacy payment limit.

RtxGen. Suppose that a user $\mathcal{U}$ intends to submit n regulatable transactions within a trading period. To submit the i-th regulatable transaction rtx_i, for $i \in \{0, 1, ..., n-2\}$, $\mathcal{U}$ calculates

$$\mathsf{tx}_i \leftarrow \mathsf{DAP.TxGen}(addr_S, addr_R, v_i, s, I_{\mathsf{pub}}, I_{\mathsf{pri}})$$

where v_i is the payment amount. Subsequently, $\mathcal{U}$ randomly samples $z_i, w_i \in \mathbb{Z}_q^*$ and computes

$$ct_i = (g^{v_i} h^{z_i}, pk_L^{z_i - w_i}) = (c_i, u_i)$$

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

$$\pi_i^{log} \leftarrow \mathcal{P}_{log}(u_i, z_i - w_i, pk_L),$$
$$\pi_i^{v} \leftarrow \mathcal{P}_v((\mathsf{tx}_i, c_i, I_{\mathsf{pub}}, g, h), (addr_S, addr_R, I_{\mathsf{pri}}, v_i, s, z_i), crs).$$

Finally, $\mathcal{U}$ calculates a ring signature

$$\sigma_i \leftarrow \mathsf{CLRS.Sign}(R_i, ct_i, pk_L, sk)$$

and sets $\mathsf{rtx}_i = (\mathsf{tx}_i, ct_i, \pi_i^v, \pi_i^{log}, \sigma_i)$.

For the final regulatable 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 $\prod_{i=0}^{n-1} g^{v_i} h^{w_i} = tag_\mu$ holds.

If $\mathcal{U}$ has already reached their privacy payment limit μ but still needs to make transactions, they can submit public transactions. As public transactions are 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 regulatable transactions, and only valid transactions are sealed into blocks.

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

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

If all validations pass, rtx is valid and the valid transaction is sealed into a new block.

4) Supervision 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 $\mathsf{rtx}_i = (ct_i, \sigma_i, \cdot)$, $\mathcal{F}$ extracts the signer's pseudonym and the amount tag from rtx_i.

$$nym_i \leftarrow \mathsf{CLRS.Ext}(\sigma_i, sk_L),$$
$$tag_{v_i} = c_i u_i^{-\frac{1}{sk_L}} = g^{v_i} h^{w_i}.$$

Detect. Let $S = \{\mathsf{rtx}_0, \mathsf{rtx}_1, ..., \mathsf{rtx}_{n-1}\}$ 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 $tag_{sum} = \prod_{i=0}^{n-1} tag_{v_i}$ where $tag_{v_i}|_{i=0}^{n-1}$ are extracted from all transactions published by the user with pseudonym nym in a period. Considering that $tag_\mu = g^\mu h^w$ and $tag_{sum} = g^{\sum_{i=0}^{n-1} v_i} h^{\sum_{i=0}^{n-1} w_i} = g^{\sum_{i=0}^{n-1} v_i} h^w$, the total payment volume is equal to the upper limit if $tag_\mu = tag_{sum}$. Otherwise, $\mathcal{F}$ flags these transactions in S as suspicious.

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

$$C = \prod_{i=0}^{n-1} c_i = g^{\sum_{i=0}^{n-1} v_i} h^{\sum_{i=0}^{n-1} z_i},$$
$$U = \prod_{i=0}^{n-1} u_i = pk_L^{\sum_{i=0}^{n-1}(z_i - w_i)},$$
$$tag_{sum} = CU^{-\frac{1}{sk_L}}.$$

Report. For a set $S = \{\mathsf{rtx}_0, \mathsf{rtx}_1, ..., \mathsf{rtx}_{n-1}\}$ 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}:

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

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

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

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

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

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

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

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

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

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

PROOF. For any transaction $\mathsf{rtx}_i = (\mathsf{tx}_i, ct_i, \pi_i^v, \pi_i^{log}, \sigma_i)$ 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. □

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 supervision phase.

I-Preparation Phase

In the preparation phase, consensus nodes execute the initialization algorithm. Filter, Supervisor and users generates their public-private key pair. Supervisor and each user engage in an interactive protocol to carry out user registration.

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 $pp = (g, h, \mathcal{H}) \leftarrow \mathsf{CLRS.Setup}(1^\lambda)$
  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 $(pk_L, sk_L) \leftarrow \mathsf{CLRS.LKGen}(pp)$
  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 $(pk_O, sk_O) \leftarrow \mathsf{CLRS.OKGen}(pp)$
  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 $\mathsf{CLRS.UKGen}$ to generate their public-private key pair: $(uk, sk) \leftarrow \mathsf{CLRS.UKGen}(pk_O)$.
- The user publishes their public key uk.

User $\Leftrightarrow$ Supervisor:

DAPCR.Σ_{reg}:
- 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 (B, uk) to Supervisor
- Supervisor:
  1. receive (B, uk) from the user
  2. set the upper limit $\mu \in \mathbb{Z}_q^*$ of total payment volume for the user
  3. compute $w = \mathcal{H}(B^{sk_O} | uk)$, $tag_\mu = g^\mu h^w$ and $nym = pk^{1/sk_O}$
  5. add $(uk, \mu, nym, \mathsf{tag}_\mu)$ to the list L_S
  6. send μ to the user through a secure channel
- Supervisor also sends (nym, tag_μ) to Filter. Once (nym, tag_μ) is received, Filter adds it to the list $L_\mathcal{F}$.

II-Transaction Phase

In the transaction phase, users generates regulatable 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_i
  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_i
- Outputs: regulatable transaction rtx_i
- A user executes the following steps to generate a regulatable transaction:
  1. compute $\mathsf{tx}_i \leftarrow \mathsf{DAP.TxGen}(addr_S, addr_R, v_i, s, I_{\mathsf{pub}}, I_{\mathsf{pri}})$
  2. randomly sample $z_i, w_i \in \mathbb{Z}_q^*$
  3. compute $ct_i = (g^{v_i} h^{z_i}, pk_L^{z_i - w_i})$
  4. compute $\pi_i^{log} \leftarrow \mathcal{P}_{log}(u_i, z_i - w_i, pk_L)$
  5. compute $\pi_i^v \leftarrow \mathcal{P}_v((\mathsf{tx}_i, c_i, I_{\mathsf{pub}}, g, h), (addr_S, addr_R, I_{\mathsf{pri}}, v_i, s, z_i), crs)$
  6. calculate a ring signature $\sigma_i \leftarrow \mathsf{CLRS.Sign}(R_i, ct_i, pk_L, sk)$
  7. set $\mathsf{rtx}_i = (\mathsf{tx}_i, ct_i, \pi_i^v, \pi_i^{log}, \sigma_i)$
- The user submits the regulatable transaction rtx_i.

III-Verification Phase

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

Consensus nodes:

DAPCR.Verify:
- Inputs:
  1. regulatable 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 $b_1 \leftarrow \mathsf{DAP.TxVfy}(\mathsf{tx}, I_{\mathsf{pub}})$
  2. compute $b_2 \leftarrow \mathsf{CLRS.Vfy}(R, ct, \sigma, pk_L)$
  3. compute $b_3 \leftarrow \mathcal{V}_v((\mathsf{tx}, c, I_{\mathsf{pub}}, g, h), \pi_v, crs)$
  4. compute $b_4 \leftarrow \mathcal{V}_{log}(u, \pi_{log}, pk_L)$
  5. if $b_1 \cdot b_2 \cdot b_3 \cdot b_4 = 1$ output 1 else output 0
- Consensus nodes seal valid transactions into blocks.

IV-Supervision Phase

In the supervision 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. regulatable 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 $nym_i \leftarrow \mathsf{CLRS.Ext}(\sigma_i, sk_L)$
  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 $S = \{\mathsf{rtx}_0, \mathsf{rtx}_1, ..., \mathsf{rtx}_{n-1}\}$ of transactions submitted by nym
  2. $\mathsf{susp}/\perp$
- 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 $S = \{\mathsf{rtx}_0, \mathsf{rtx}_1, ..., \mathsf{rtx}_{n-1}\}$
  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 $\mathsf{rpt} = (S, nym, \pi_{dec}, tag_{sum}, \pi_{sum})$ to $\mathcal{S}$
- Filter sends rpt to Supervisor.

Supervisor:

$\mathsf{DAPCR.Recover}$:

- 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 $b_1 = b_2 = 1$, the report is valid else is invalid
- If the report is valid, $\mathcal{S}$ computes $uk \leftarrow \mathsf{CLRS.Open}(nym, sk_O)$ and then requires the user with public key uk to submit v_{sum} and w' satisfies $tag_{sum} = g^{v_{sum}} h^{w'}$.

VII. 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 [25] 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 [16], along with the SNARK-friendly elliptic curve BabayJubjub [26] and the hash algorithm MIMC [27] to implement the DAPCR scheme. Moreover, the experimental implementation makes use of the programming language Golang[4] and the zero-knowledge proof tool Snarkjs[5].

We utilize the SNARK-friendly elliptic curve Baby Jubjub, which is a special twisted Edwards curve with parameters $a = 168700$ and $d = 168696$, to construct CLRS and DAPCR. The twisted Edwards curve can be described by the equation $ax^2 + y^2 = 1 + dx^2y^2$. On local devices, the computation cost of the addition and multiplication operations for Baby Jubjub is $0.003\,\mathrm{ms}$ and $0.244\,\mathrm{ms}$, respectively.

In Experiment-I, we evaluate the computational and communication overhead of CLRS, which is shown in Fig. 6a. First, the time cost for user key generation is $0.7\,\mathrm{ms}$, 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 $13.3\,\mathrm{ms}$ and $6.9\,\mathrm{ms}$, 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 $13.9\,\mathrm{KB}$. Hence, we consider the CLRS scheme to be efficient.

In Experiment-II, we evaluate the computational overhead of transaction generation in DAPCR and compared it with that in ZETH, which is shown in Fig. 6b. The time cost of transaction generation is linear to the number of transactions. It takes about $1231.2\,\mathrm{ms}$ to generate one transaction and less than $250\,\mathrm{s}$ to generate 200 transactions in DAPCR. In Fig. 6b, *Additional Overhead* represents the additional overhead introduced by DAPCR during the transaction generation phase to achieve restricted regulation, as compared to ZETH. It is evident that the additional overhead is less than 50% of the computational overhead of transaction generation in ZETH. 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. 6c presents the time cost of transaction verification in a consensus node, which is linear to the number of transactions. It takes about only $12.5\,\mathrm{ms}$ to verify one transaction and about $2.5\,\mathrm{s}$ 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 Fabric[6], where the time interval between a user submitting a transaction and receiving a response indicating successful verification is about $2.0\,\mathrm{s}$. 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. 6d. 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 $2.5\,\mathrm{s}$. 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. 6d. 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 supervision phase.

VIII. Related Work

In this section, a non-exhaustive review of related works is presented.

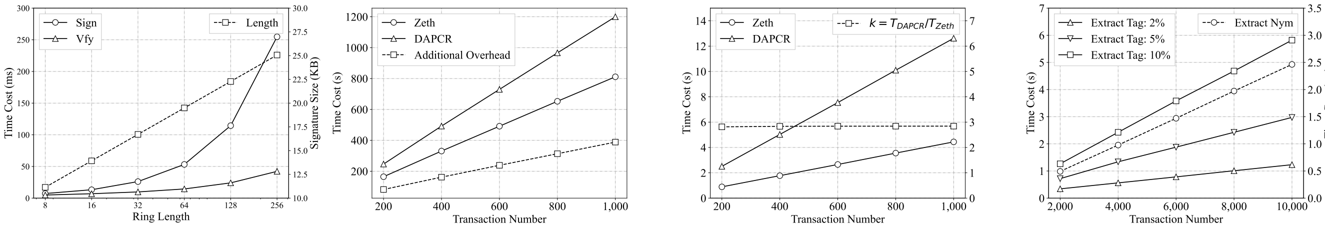
A. DAP with Regulation

To address the privacy concerns of traditional decentralized payment system, several decentralized anonymous payment

[4] https://go.dev/

[5] https://github.com/iden3/snarkjs

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



(a) Experiment-I (b) Experiment-II (c) Experiment-III (d) Experiment-IV

Fig. 6: Performance experiments for DAPCR and CLRS

(DAP) schemes have been proposed, such as Zerocash, Zether, SofitMix [28] and BlockMaze [29]. Privacy preservation enables users to engage in transactions without disclosing their identities or even the amounts involved, which also makes it possible to conduct illicit activities using blockchain [30]. To tackle this issue and combat potential criminal behavior, researchers have proposed various solutions in recent years.

Wang et al. [7] 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. [8] 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. [7] and Lin et al. [8] 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. [9] 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. [9] 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 [31] 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 [32] 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 [11] and miniLedger [10] 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 not always be online, and a malicious user may ignore the auditor's queries, which leads to delays in the audit.

Platypus [13] and PEReDi [33] 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

Ring signatures were first proposed by Rivest et al. [34] Classical ring signatures allow any member in a ring to generate a signature for a message on behalf of all members in the ring and provide unconditional anonymity, meaning any user cannot determine which user in the ring has signed the signature. However, the strong privacy preservation also poses potential risks that some users may sign malicious messages.

To address this issue, Bootle et al. [19] proposed the concept of accountable ring signatures. Like classical ring signatures, the signature remains completely anonymous to ordinary users. However, when a user signs a message, he or she selects an authorized user, who can open the signature and revoke its anonymity, allowing for the identification of the signer. This approach carries serious privacy risks that are exacerbated if the authorized user is hacked.

Our scheme is based on accountable ring signatures and decentralizes the ability to revoke anonymity to the linker and opener. The linker can extract the label of the signer from the signature. Since the public key and label of the signer correspond uniquely, the linker can link the signatures from the same signer without revealing the public key of the signer. The linker can then send the label to the opener, who can recover the public key of the signer from the label.

In group signatures, there exists a similar type of scheme known as group signature with controllable linkability [20]. This type of scheme allows an entity holding the linkability key to link signatures from the same signer without revealing their real identity. However, group signatures rely on trusted centers and group administrators, and the security of the system is severely affected once they are hacked. Our scheme is based on the idea of ring signatures and does not require administrators to manage the ring members, nor does it have trusted settings. Without any single point of failure, our scheme is also suitable for distributed scenarios, such as blockchains.

Another approach to achieving accountability is through traceable ring signatures [21] and linkable ring signatures [22]. Linkable ring signatures allow publicly linking signatures from the same signer without leaking the signer's real identity. In contrast, traceable ring signatures enable any user to publicly trace the identity of the signer, provided that the same signer signed two signatures.

IX. Conclusion

In this paper, we propose a decentralized anonymous payment scheme with collaborative regulation, which achieves universality, collaborative regulation and efficient aggregation of transaction amounts. To achieve efficient regulation, users are required to register with Supervisor before publishing transactions. Therefore, compared to public blockchains, the DAPCR scheme is more suitable for consortium blockchains. Furthermore, the regulation in DAPCR relies on the functioning of Filter at the end of each trading period, which is a potential target for malicious attackers. In future work, there are several potential directions for improvement. First, we can focus on optimizing user registration. Secondly, reducing the workload of Filter can lead to improved performance and efficiency. Finally, exploring and expanding the application scenarios of the CLRS signature can unlock new possibilities and benefits.

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 $((c, u, pk), (\alpha, \beta)) \in \mathcal{R}_{enc}$.

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 $ct_i = (c_i, u_i)$, $i \in \{0, 1, ..., n-1\}$, 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 $i \in \{0, 1, ..., n-1\}$, 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
$$(\prod_{i=0}^{n-1} \frac{c_i}{m})^y \stackrel{?}{=} (\prod_{i=0}^{n-1} u_i)^e \prod_{i=0}^{n-1} B_i$$
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[7] 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. $g, h \in \mathbb{G}$
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 $\mathsf{DAP.TxGen}(\cdot)$ with inputs $(addr_S, addr_R, v, s, I_{\mathsf{pub}}, I_{\mathsf{pri}})$.
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], [36] 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
Security Analysis

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

[7]Zk-SNARK has lower communication overhead and computational overhead for proof verification compared to zk-STARK. This makes zk-SNARK more suitable for our purpose, as it minimizes the utilization of valuable on-chain resources.

A. Security Analysis of CLRS

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

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

Game $\mathsf{G}^{\mathsf{Ano}}_{\mathsf{real}}$. With the purpose of initializing this experiment, C first computes

$$crs_{mem} \leftarrow \mathcal{G}_{mem}(1^\lambda, \mathcal{R}_{mem}),$$
$$pp_{enc} \leftarrow \mathsf{SoK.Setup}(1^\lambda, \mathcal{R}_{enc}),$$

and other public parameters. 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 C.

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

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

where $\sigma_b = (com, K, \pi_{mem}, \pi) = (g^{sk_b}h^k, pk_L^k, \pi_{mem}, \pi)$. 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 $\mathcal{Q}_{\mathsf{Ext}}$ to extract pseudonyms from signatures. $\mathcal{A}_2$ can query an oracle $\mathcal{Q}_{\mathsf{Open}}$ 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 $\mathsf{G}^{\mathsf{Ano}}_{\mathsf{sim}}$. During the initialization phase, 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}.

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

Similar to the case in $\mathsf{G}^{\mathsf{Ano}}_{\mathsf{real}}$, $\mathcal{A}_i$ 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 C.

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

$$\pi^{mem}_{sim} \leftarrow \mathcal{P}^{mem}_{sim}(R'_2, \tau_{mem}, crs_{mem}),$$
$$\pi_{sim} \leftarrow \mathsf{SoK.Sign}_{sim}((com_{sim}, K, pk_L), \tau_{enc}, pp_{enc}),$$

and sends $\sigma_{sim} = \{com_{sim}, K, \pi^{mem}_{sim}, \pi_{sim}\}$ to $\mathcal{A}_i$. Since σ_{sim} is independent of uk_0 and uk_1, the advantage of $\mathcal{A}_i$ in winning $\mathsf{G}^{\mathsf{Ano}}_{\mathsf{sim}}$ 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 (com, π_{mem}, π) is equivalent to that of $(com_{sim}, \pi^{mem}_{sim}, \pi_{sim})$. While $\mathcal{A}_1$ can query $\mathcal{Q}_{\mathsf{Ext}}$ to extracted nym from σ_b and nym' from σ_{sim}, $nym = pk_b^{1/sk_O}$ and nym' are indistinguishable since sk_O is unknown to $\mathcal{A}_1$. Therefore, σ_{sim} in $\mathsf{G}^{\mathsf{Ano}}_{\mathsf{sim}}$ and σ_b in $\mathsf{G}^{\mathsf{Ano}}_{\mathsf{real}}$ are indistinguishable. Considering the negligible advantage of $\mathcal{A}_i$ in winning $\mathsf{G}^{\mathsf{Ano}}_{\mathsf{sim}}$ and the indistinguishability between $\mathsf{G}^{\mathsf{Ano}}_{\mathsf{real}}$ and $\mathsf{G}^{\mathsf{Ano}}_{\mathsf{sim}}$, the advantage of $\mathcal{A}_i$ in winning $\mathsf{G}^{\mathsf{Ano}}_{\mathsf{real}}$ is also negligible. □

Theorem 4. *No probabilistic polynomial-time 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, 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.

With the purpose of initializing this experiment, C first executes

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

Then C randomly chooses $r_i \in \mathbb{Z}_q^*$ and $pk_i \in \mathbb{G}^*$, and computes

$$\pi^{enc}_{sim,i} \leftarrow \mathcal{P}^{enc}_{sim}((h^{r_i}A, pk_i, pk_O), \tau_{enc}, crs_{enc})$$

for $i \in \{0, 1, ..., n-1\}$. Finally, C sets $c_i = h^{r_i}A$ and publishes $uk_i = (pk_i, c_i, \pi^{enc}_{sim,i})$.

To win the experiment, $\mathcal{A}_i$ forges a signature $\sigma = (com, K, \pi_{mem}, \pi)$ of (R, m). By the extractability of a SoK protocol, C can extract a valid witness $\varpi = (r_i, a)$ for the statement $\phi = (com, K, pk_L)$ from π. Therefore, C can make use of $\mathcal{A}_i$ to obtain a such that $A = g^a$.

Case-2: There exists an honest user with $uk = (pk, c, \pi_{enc})$ 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.

With the purpose of initializing this experiment, C first executes

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

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

$$\pi^{enc}_{sim} \leftarrow \mathcal{P}^{enc}_{sim}((h^rA, A^{sk_O}, pk_O), \tau_{enc}, crs_{enc}).$$

Finally, C publishes $uk = (A^{sk_O}, h^rA, \pi^{enc}_{sim}) = (pk, c, \pi^{enc}_{sim})$. For public-private key pairs of other honest users, C randomly chooses the private key, calculates the public key and keep the private key locally.

To win the experiment, $\mathcal{A}_1$ randomly selects a user from honest users and forges a signature $\sigma = (com, K, \pi_{mem}, \pi)$. Assuming $\mathcal{A}_1$ selects uk with a probability $\frac{1}{\eta(\lambda)}$, where $\eta(\lambda)$ 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 $\varpi_{enc} = (a, k)$ for the statement $\phi_{enc} = (com, K, pk_L)$ 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. □

Theorem 5. *No probabilistic polynomial-time 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 $\mathit{\Pi}_{enc}$ 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 $\mathit{\Pi}_{mem}$ implies that $c' = c/com = g^0 h^{r'}$. Thus, $com = g^{sk}h^k$ and $k = r - r'$. The extractability of the SoK protocol implies that a valid witness $\varpi = (sk, k)$ for the statement $\phi = (com, K, pk_L)$ 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 $\mathsf{CLRS.Ext}$. In addition, based on the completeness of the protocol $\mathit{\Pi}_{dec}$, the proof π_{dec} for that nym is correctly extracted from σ will be verified correctly. Therefore, $\mathsf{CLRS.Judge}$ takes σ, nym and π_{dec} as input and outputs 1. □

Theorem 6. *No probabilistic polynomial-time 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 $\mathit{\Pi}_{enc}$ 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 $\mathit{\Pi}_{mem}$ implies that $c'_j = g^0 h^{r'}$ 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 $\varpi = (sk, k)$ for the statement $\phi = (com, K, pk_L)$ 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 $\mathit{\Pi}_{dec}$ 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. □

B. Security Analysis of DAPCR

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

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

Game $\mathsf{G}^{\mathsf{ind}}_{\mathsf{real}}$. C first calculates

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

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 $tuple_j = (addr_{S_j}, addr_{R_j}, v_j, uk_j)$ 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

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

where sk_b is the private key associated with uk_b. C sends $\mathsf{rtx}_b = (\mathsf{tx}, ct, \pi_v, \pi_{log}, \sigma)$ 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 $\mathsf{G}^{\mathsf{ind}}_{\mathsf{mid}}$. The experiment in $\mathsf{G}^{\mathsf{ind}}_{\mathsf{mid}}$ is similar to that in $\mathsf{G}^{\mathsf{ind}}_{\mathsf{real}}$, but the only modification is that $\mathcal{C}$ 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^v_{mid}. C sends $\mathsf{rtx}_{mid} = (\mathsf{tx}_{sim}, ct, \pi^v_{mid}, \pi_{log}, \sigma)$ 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^v_{mid}. Thus, the absolute value of the difference between the advantage of $\mathcal{A}'_i$ in winning $\mathsf{G}^{\mathsf{ind}}_{\mathsf{mid}}$ and that in winning $\mathsf{G}^{\mathsf{ind}}_{\mathsf{real}}$ is negligible.

Game $\mathsf{G}^{\mathsf{ind}}_{\mathsf{sim}}$. $\mathsf{G}^{\mathsf{ind}}_{\mathsf{sim}}$ is similar to $\mathsf{G}^{\mathsf{ind}}_{\mathsf{mid}}$, but the only modification is that $\mathcal{C}$ 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^v_{sim} and π^{log}_{sim}. C sends $\mathsf{rtx}_{sim} = (\mathsf{tx}_{sim}, ct_{sim}, \pi^v_{sim}, \pi^{log}_{sim}, \sigma)$ 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 $(\pi^v_{sim}, \pi^{log}_{sim})$ is identical to that of (π^v_{mid}, 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 $(ct_{sim}, \pi^v_{sim}, \pi^{log}_{sim})$ is identical to that of $(ct, \pi^v_{mid}, \pi_{log})$.

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

Therefore, the advantage of $\mathcal{A}'_i$ in winning $\mathsf{G}^{\mathsf{ind}}_{\mathsf{real}}$ 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. □

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

PROOF. For a valid transaction $\mathsf{rtx} = (\mathsf{tx}, ct, \pi_v, \pi_{log}, \sigma)$ 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 $\mathcal{F}$. □

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

PROOF. For a valid transaction $\mathsf{rtx} = (\mathsf{tx}, ct, \pi_v, \pi_{log}, \sigma)$ 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. □

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

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] S. Allen, S. Čapkun, I. Eyal, G. Fanti, B. A. Ford, J. Grimmelmann, A. Juels, K. Kostiainen, S. Meiklejohn, A. Miller, *et al.*, "Design choices for central bank digital currency: Policy and technical considerations," tech. rep., National Bureau of Economic Research, 2020.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] K. Wüst, K. Kostiainen, 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.

[14] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct nizks without pcps," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 626–645, Springer, 2013.

[15] J. Groth and M. Maller, "Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks," in *Annual International Cryptology Conference*, pp. 581–612, Springer, 2017.

[16] 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.

[17] 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.

[18] 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.

[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. 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.

[21] E. Fujisaki and K. Suzuki, "Traceable ring signature," in *International Workshop on Public Key Cryptography*, pp. 181–200, Springer, 2007.

[22] J. K. Liu and D. S. Wong, "Linkable ring signatures: Security models and new schemes," in *Computational Science and Its Applications–ICCSA 2005: International Conference, Singapore, May 9-12, 2005, Proceedings, Part II 5*, pp. 614–623, Springer, 2005.

[23] 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.

[24] G. Liu, Z. Yan, D. Wang, H. Wang, and T. Li, "Deptvm: Decentralized pseudonym and trust value management for integrated networks," *IEEE Transactions on Dependable and Secure Computing*, 2023.

[25] A. Rondelet and M. Zajac, "Zeth: On integrating zerocash on ethereum," *arXiv preprint arXiv:1904.00905*, 2019.

[26] B. WhiteHat, J. Baylina, and M. Bellés, "Baby jubjub elliptic curve," *Ethereum Improvement Proposal, EIP-2494*, vol. 29, 2020.

[27] M. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen, "Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 191–219, Springer, 2016.

[28] H. Xie, S. Fei, Z. Yan, and Y. Xiao, "Sofitmix: a secure offchain-supported bitcoin-compatible mixing protocol," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[29] Z. Guan, Z. Wan, Y. Yang, Y. Zhou, and B. Huang, "Blockmaze: An efficient privacy-preserving account-model blockchain based on zk-snarks," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[30] L. Peng, W. Feng, Z. Yan, Y. Li, X. Zhou, and S. Shimizu, "Privacy preservation in permissionless blockchain: A survey," *Digital Communications and Networks*, vol. 7, no. 3, pp. 295–307, 2021.

[31] K. Wüst, K. Kostiainen, 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.

[32] 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.

[33] 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.

[34] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*, pp. 552–565, Springer, 2001.

[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.

[36] S. Bowe, A. Gabizon, and M. D. Green, "A multi-party protocol for constructing the public parameters of the pinocchio zk-snark," in *International Conference on Financial Cryptography and Data Security*, pp. 64–77, Springer, 2018.