A Blockchain-Based Proxy Re-Encryption with Equality Test for Vehicular Communication Systems

Biwen Chen, Debiao He, Neeraj Kumar, Huaqun Wang, Kim-Kwang Raymond Choo

***Abstract*—Vehicular communication systems (VCS) are likely to play an increasingly important role in future smart city design, for example by improving road safety and traffic efficiency. However, there are underpinning security and privacy challenges, which may also result in under-utilization of vehicular data. In this paper, we introduce a new cryptographic primitive, namely: blockchain-based proxy re-encryption with equality test. Specifically, the proposed approach is designed to achieve reliable matching results by leveraging smart contracts, as well as efficient data sharing and privacy-preserving by combining the function of public-key encryption with equality test and proxy re-encryption. We also implement a prototype of the proposed approach and evaluate its performance with those of three other competing approaches. A comparative summary with three other competing approaches demonstrate that the proposed approach achieves all four features (i.e., decentralization, flexibility, non-interaction, and re-encryption). Findings from the prototype evaluation (using hyperledger fabric v1.4.2 as the test blockchain platform, and the fabric-sample repository) also demonstrate the utility of the proposed approach in practice.**

***Keywords*—Vehicular communication system, data sharing, proxy re-encryption, blockchain, public key encryption with equality test.**

The work was supported by National Natural Science Foundation of China (Nos. 61972294, 61932016, 61872192, 61941116), the Major Science Research Project of Jiangsu Provincial Education Department(No. 19KJA310010) and the the Opening Project of Guangdong Key Laboratory of Data Security and Privacy Protection (No.2017B030301004). K.-K. R. Choo was supported in part by the Cloud Technology Endowed Professorship and National Science Foundation CREST(No. HRD-1736209).

B. Chen is with the School of Computer Science, Chongqing University, Chongqing 400044, China and the Guangdong Key Laboratory of Data Security and Privacy Protection, Guangzhou 510632, China
E-mail: macrochen@whu.edu.cn

D. He (*Corresponding author*) is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China and the Guangdong Key Laboratory of Data Security and Privacy Protection, Guangzhou 510632, China
E-mail: hedebiao@163.com

N. Kumar is with the Department of Computer Science and Engineering, Thapar University, Patiala 147001, India and the Department of Computer science and Information Engineering, Asia University, Taizhong 41354, Taiwan
E-mail: neeraj.kumar@thapar.edu

H. Wang is with the Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
E-mail: wanghuaqun@aliyun.com

K.-K. R. Choo is with the Department of Information Systems and Cyber Security and the Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX 78249, USA
Email: raymond.choo@fulbrightmail.org

I. Introduction

The number of vehicles sold and operating on the road increases every year [1], except when there are strict lockdown measures restricting gatherings and social contacts (e.g., due to COVID-19). The significant number of vehicles operating on the road has a number of associated challenges, ranging from environmental pollution to traffic congestion, and so on. For example, the American Automobile Association (AAA) estimated that vehicle crashes cost approximately $300 billion annually [2]. This reinforces the importance of design intelligent solutions technologies to improve road safety and traffic efficiency, for example by leveraging advances in communication technologies (e.g., 5G) and other supporting infrastructure (e.g., intelligent transportation system – ITS, and vehicular communication systems – VCS) [3], [4].

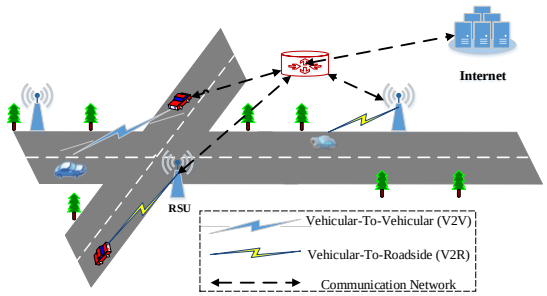


Fig. 1: Communication models in a typical VCS setting

Existing communication models in VCS can be generally categorized into Vehicle-to-Roadside (V2R) and Vehicle-to-Vehicle (V2V) models – see Fig.1. These communication models allow vehicles to exchange information with other supporting infrastructures, such as traffic lights, surrounding vehicles, lane markers, and traffic cameras. There are, however, a number of limitations in existing VCS. For example, the use of encryption in VCS communications complicates data utilization. In other words, how do we ensure data privacy without sacrificing data availability? [5], [6]

In the existing literature, there have been attempts to leverage different cryptographic mechanisms, such as privacy-preserving task matching [7], searchable encryption [8], proxy re-encryption [9] and public key encryption with equality test [10], [11], to address the above limitation. A number of the proposed approaches support both data confidentiality and some ciphertext manipulation functions, such as search, matching, and equality test. However, most of the existing schemes rely on either a honest-but-curious or a honest entity,

which does not (maliciously) deviate from the predefined protocol. However, this assumption may be unrealistic in practical applications since a corrupted (and unsupervised) server may return incorrect results due to random failures or to maximize profit. For example, proxy-based solutions [9], [12] generally depend on trusted centers to transform the ciphertexts. To remove the reliance on centralized trust, a number of existing schemes [13]–[17] use a decentralized blockchain instead of centralized servers.

Directly replacing centralized entities with blockchain may introduce additional problems, such as information leakage. For example, in the approach of Wu *et al.* [15], a malicious adversary may compromise the data user's privacy by launching an inside keyword guessing attack or file-injection attacks [18]. Meanwhile, the approaches of [13], [14] require all data users to share the same secret key. Other schemes may only support the one-user model, which severely limits their application potential. This, therefore, reinforces the importance of designing new approaches to achieve both data privacy and utility for deployment in VCS.

In this paper, we design a new framework, namely: blockchain-based proxy re-encryption with equality test (BPREET) for VCS deployment. As our framework uses smart contracts, no centralized server is required. In addition, this also ensures that the matching process is transparent and reliable. Our framework supports the equality test under ciphertexts in the multi-user model, as well as dynamic authorization. In other words, our framework not only inherits the features inherent of blockchain, but also combines the function of proxy re-encryption (PRE) and PKEET; t achieving creditability of the test result while maintaining privacy-protection.

We will now describe the layout of this paper. We will review the related literature and preliminaries in Sections II and III, respectively. In Section IV, we define the notion of BPREET and present its system model and design goals. Then, we present a detailed construction of BPREET in Section V, prior to presenting its security proof and performance evaluation in Sections VI and VII. Finally, the conclusion is presented in Section VIII.

II. Related work

In this section, we will briefly review the existing literature on relevant cryptographic primitives .

Proxy re-encryption. Proxy re-encryption allows a proxy to convert ciphertexts of the delegator into that of the delegatee. The feature is devoted to improving the flexibility of access control in the data sharing systems [19], [20], since it can reduce the computing and communication overheads between the delegator and the delegatee. However, most proposed schemes require bilinear pairing and hence, such schemes are impractical in terms of performance. To address this problem, Lu *et al.* [21] constructed a pairing-free certificate-based proxy re-encryption scheme, which enjoys the advantages of both certificate-based encryption and proxy re-encryption encryption. In 2018, Manzoor *et al.* [22] designed a decentralized proxy re-encryption scheme without pairing. Thus, designing secure and efficient PRE schemes that support new features remains challenging.

Searchable encryption. Searchable encryption enables searches over encrypted data in a privacy-preserving setting, such as VCS [23]–[25]. Existing approaches can be generally categorized into those based on searchable symmetric-key encryption (SSE) and searchable public-key encryption (SPE). SSE schemes [13], [14] usually have higher efficiency, at the cost of expensive key management and distribution. While SPE schemes [15], [26] avoid expensive key management and distribution costs, they are vulnerable to server-related attacks such as keyword guessing attacks [26] and file-injection attacks [18], and have low efficiency [27].

Public-key encryption with equality test. Public-key encryption with equality test (PKEET) [28] allows an authorized tester to check whether two ciphertexts correspond to the same message without decrypting the messages. In other words, these schemes can be used to achieve controlled equality test or deduplication on ciphertexts. Since the pioneering work of [10] on PKEET, a number of variant schemes have been proposed to support different application requirements [11], [29]–[31]. Wu *et al.* [11], for example, improved the performance of Ma's work [29] by reducing the use of HashToPoint operation. Duong *et al.* [31] proposed the first lattice-based PEKEET scheme in the standard model. The property, equality test under ciphertexts, helps achieve effective data sharing, especially in a multi-user model.

Blockchain-based technologies. Due to the decentralization, immutability and verifiability features of blockchain, a number of blockchain-based solutions to address different challenges in VCS have been proposed in recent literature [32]–[35]. For example, Li *et al.* [32] constructed a privacy-preserving incentive announcement network based on blockchain to motivate users to share traffic information. Singh *et al.* [33] built a trusted, secure environment for the intelligent vehicle, based on blockchain. Recently, Lu *et al.* [35] proposed a blockchain-based privacy-preserving authentication scheme to solve the problem of non-transparency associated with the trusted authorities in the vehicular ad hoc network (VANET). There are, however, a number of limitations associated with blockchain-based approaches [36]–[39]. In other words, there is a need to design new blockchain-based technologies which can be used to achieve enhanced service quality and transparency without sacrificing privacy.

III. Preliminaries

A. Notations

A summary of notions used in this paper is presented in Table I.

B. Blockchain and Smart Contract

Blockchain. Blockchain is a distributed database and its security is ensured by the cryptographic algorithms and consensus mechanisms. Existing blockchain networks are generally categorized into permissionless (public) and permissioned (private and Consortium). In permissionless blockchain networks (e.g., Bitcoin and Ethereum), every user creates an address and interacts with the network, while users are not able to

TABLE I: Summary of notations

Notation	Description
λ	security parameter
$(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$	related parameter of bilinear pairing
$P_{i=1,2}$	generators of groups $\mathbb{G}_{i=1,2}$
$(h_0, H_{i=1,2})$	secure hash functions
(pk_i, sk_i)	public/secret key pair of user i
(pk_i^1, pk_i^2)	two parts of user i's public key
$C_{pk_i}(m)$	ciphertext of m under public key pk_i
$x \xleftarrow{R} \mathcal{X}$	choose a random element x from $\mathcal{X}$
rk_{ij}	re-encryption key generated by user i for user j
$RC^i_{pk_j}(m)$	re-encryption ciphertext corresponding to public key pk_j of $C_{pk_i}(m)$
$T_{pk_i \to pk_j}$	authorization trapdoor generated by user i for user j corresponds to $C_{pk_i}(m)$
IC_i	Intermediate ciphertext in the **Test** algorithm
ϵ	negligible probability
$\|\|$	concatenation operation

freely join in permissioned networks (e.g., fabric). In general, a blockchain platform provides the following properties:

- $Transparency$: All transactions recorded and executed computations are transparent in public blockchain networks, or to the authorized users in permissioned blockchain networks.
- $Consensus$: The data stored and computations are verified by all participants in the entire network. A security mechanism (e.g., Proof of Work (PoW) or Practical Byzantine Fault Tolerance (PBFT)) is executed to agree upon its global state.

Smart Contract. Smart contract is a self-enforcing computer program. Once created and deployed to the blockchain platform (e.g. Ethereum and Hyperledger), the contract's code cannot be subsequently modified. In theory, the smart contract can be used to perform any computational task, because it can be programmed in a Turing Complete language.

C. Bilinear Pairing

Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ denote three cyclic groups with the same prime order p. Let P_i be a generator of group $\mathbb{G}_i$, where $i = \{1, 2\}$. A bilinear pairing operation $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ has the following features:

- **Bilinearity**: Given two elements $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ and any two integers $x, y \in \mathbb{Z}_q^*$, the equation $e(xP, yQ) = e(P, Q)^{xy} \in \mathbb{G}_T$ holds.
- **Non-degeneracy**: $e(P_1, P_2) \neq \mathbf{1}$.
- **Computability**: For any two elements $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$, there exists a polynomial complexity algorithm to compute $e(P, Q) \in \mathbb{G}_T$.

D. Hard Problem Assumptions

The security of our scheme is based on the following hard problems.

- **Computational Diffie Hellman (CDH) assumption**. Let $\mathbb{G}_1$ be a cycle addition group and P be a base point generator of prime order q. The CDH problem is $\epsilon-$hard in $\mathbb{G}_1$, if given $(P, aP, bP) \in \mathbb{G}_1$ for random $(a, b) \in \mathbb{Z}_q^*$, any polynomial-time bounded adversary $\mathcal{A}$ computes an element $abP \in \mathbb{G}_1$ with advantage

$$Adv_{\mathcal{A}}^{CDH}(\lambda) = Pr[\mathcal{A}(P, aP, bP) = abP] \leq \epsilon$$

- **Decisional Bilinear Diffie Hellman (DBDH) assumption**. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ be three groups with the same prime order q and e be a map $e(P_1, P_2) \in \mathbb{G}_T$, where $P_i \in \mathbb{G}_i$ and $i = \{1, 2\}$. We say the DBDH problem is $\epsilon-$hard, if given the DBDH tuple $(P_1, aP_1, bP_2, cP_1, e(P_1, P_2)^{abc})$ for random $(a, b, c) \in \mathbb{Z}_q^*$ and the random tuple $(P, aP_1, bP_2, cP_1, Z \xleftarrow{R} \mathbb{G}_T)$, any polynomial-time bounded adversary $\mathcal{A}$ determines $Z \stackrel{?}{=} e(P_1, P_2)^{abc}$ with advantage

$$\begin{aligned} Adv_{\mathcal{A}}^{DBDH}(\lambda) = |&Pr[\mathcal{A}(P_1, aP_1, bP_2, cP_1, e(P_1, P_2)^{abc}) \\ &- Pr[\mathcal{A}(P_1, aP_1, bP_2, cP_1, Z)]| \leq \epsilon \end{aligned}$$

We say that both the CDH assumption and the DBDH assumption hold if the advantage $Adv_{\mathcal{A}}^{CDH}(\lambda)$ and $Adv_{\mathcal{A}}^{DBDH}(\lambda)$ are negligible, that is, ϵ is negligible.

IV. System Overview

In this section, we introduce the system model, the security assumptions and the design goals of our proposed scheme.

A. System Model

We consider the data sharing case in vehicular communication systems, where both the data owner and the authenticated data user upload its encrypted share messages (e.g., warning signals) and interests (e.g., road conditions) to a trustworthy platform such that they can find the right target for each other. In Fig. 2, we outline the system model of our proposed scheme, which consists of four main entities, namely: a trust center, data owners, data users and a blockchain platform with smart contracts (e.g., Ethereum or Hyperledger), respectively.

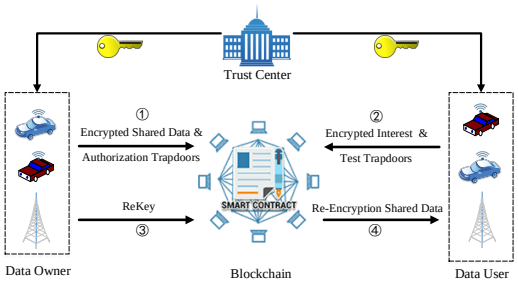


Fig. 2: System model of our proposed scheme

- $Trust\ Center$: is a trusted organization (e.g., government) in charge of generating the system parameters and deploying smart contracts.
- $Data\ Owners$: are the data publishers who publish the shared messages to the blockchain and share the data to the authenticated data users.
- $Data\ Users$: are the data users who receives exchange data from data users based on their interests.
- $Blockchain$: is a public blockchain platform that supports smart contracts and offers ciphertext matching services to users (data owners and data users) and achieves data sharing.

The high-level workflow includes the following four stages: ① If a data owner wishes to protect the privacy of traffic messages while achieving efficient sharing, (s)he first generates the data ciphertexts and the authorization trapdoors for those designated data users. Then, (s)he uploads the data ciphertexts and the authorization trapdoors to the smart contract. ② To find the useful information and protect the privacy of interests, a data user generates the interest ciphertexts and the authorized trapdoors, and submits them to the smart contract. After the ciphertexts are uploaded, the smart contract will honestly check whether the encrypted message and the encrypted interest match up according to the predefined rules. Later, ③ when the smart contract finds a message/interest matching pair, it will notify the data owner to upload a re-encryption key transforming the data ciphertext into a special ciphertext that the data user can correct decrypt. Finally, ④ the data user obtains the specific shared information by decrypting the re-encryption ciphertext.

According to the above workflow, our system consists of nine algorithms: (1)$Setup$. It is used to generate the system parameters and deploy smart contracts. The security of our system requires a trusted setup. (2) $KeyGen$. Users need to register their identity with the system before using it. (3)$Encrypt$. To protect data privacy, the information of both data owners and data users should be submitted in encrypted form to the blockchain platform. The encryption algorithm should be lightweight because it needs to be called frequently. (4) $Self\text{-}Decrypt$. Data owners or data users can recover data with the help of this algorithm. (5) $ReKey$ and $ReEncrypt$. To share the data with an authorized data user, the data owner first generates a re-encryption key by calling the $ReKey$ algorithm, and then with the help of $ReEncrypt$ algorithm, the authorized data user can transmit a ciphertext under the data owner's public key into a ciphertext under his public key. (6) $ReDecrypt$. Receiving a re-encryption ciphertext, the data user can decrypt the ciphertext to obtain the shared data by leveraging his secret key. (7) $Authorization$. To determine whether two ciphertexts corresponding to the same message, the owners of ciphertexts need to provide trapdoors respectively. In addition, the submitted trapdoors should not reveal any useful information. (8) $Test$. Once a ciphertext test transaction is published on the blockchain, the smart contracts will be triggered and automatically executed, returning a credible test result to the publisher of this transaction.

In addition, the blockchain-based proxy re-encryption with equality test scheme not only inherits the features of blockchain, but also combines the function of PRE and PKEET. However, our scheme cannot be obtained by directly combining the PRE scheme and the PKEET, since the new scheme may incur some new security issues, such as information leakage.

B. Security Assumptions

In our system, the trusted center is only used for system initialization and user registration, and the users including data owners and data users. The users are believed to be honest and can keep their secret keys well. Besides, the smart contract is considered as a known and trusted program. The adversary comes from outside the system and tries to extract sensitive information from the system. They have access to all ciphertext data, trapdoors, and re-encryption keys. In addition, we assume that the adversary is a polynomial-time entity and has bound computational power.

C. Design Goals

Our design goals are to enforce the following functional and security requirements:

- **Correctness**. The correctness property for BPREET has two basic requirements. One is that the user can correctly decrypt the ciphertext under his/her public key, and the other is that if two ciphertexts contain the same message, given the legal trapdoors, the **Test** algorithm always outputs 1.
- **Confidentiality**. The confidentiality of both the data owner's messages and the data user's interests should be protected from the adversary. Besides, the equality test for PRE is a new property, which must ensure that the test process of two ciphertexts should do not leak information about the messages.
- **Decentralization**. The decentralization means that the ciphertext test process does not rely on a central crowd-server, avoiding the impractical assumption that the server must be honest or semi-honest. Unlike the black-box model of the centralized scheme, the decentralization property makes the scheme more transparent and more credible.

V. Construction

We construct a concrete BPREET scheme, whose algorithms are described below.

Setup$(1^\lambda) \rightarrow pp$: The algorithm is executed by the trusted center. It takes a security parameter λ as input, and outputs the public parameter pp as follows:

1. selects three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with the bilinear pairing operation $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, all of which have the same prime order q. Let P_i be the generator of the group $\mathbb{G}_i$, where $i = \{1, 2\}$.
2. selects three secure hash functions $h_0 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_1 : \mathbb{G}_1 \rightarrow \{0,1\}^{3\lambda}$ and $H_2 : \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$.
3. publishes the public parameter $pp = \{q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e, h_0, H_1, H_2\}$.
4. deploys the smart contracts (Algorithms 1 and 2).

KeyGen$(pp) \rightarrow (pk_i, sk_i)$: The algorithm is used to generate the public/secret key pairs for data owners and users respectively. It takes the public parameter pp as input, then picks a random number $sk_i = a_i \xleftarrow{R} \mathbb{Z}_q^*$ as the secret key, and computes the corresponding public key $pk_i = \{pk_i^1, pk_i^2\}$, where $pk_i^1 = sk_i P_1 = a_i P_1$ and $pk_i^2 = sk_i P_2 = a_i P_2$.

Encrypt$(pp, pk_i, sk_i, m) \rightarrow C_{pk_i}(m)$: The algorithm is used to generate shared message ciphertexts of the data owner or the interest ciphertexts of the data user. It takes the public parameter pp, the public/secret key pair $(pk_i = \{pk_i^1, pk_i^2\} = \{a_i P_1, a_i P_2\}, sk_i = a_i)$ of user i and a message $m \in \{0,1\}^*$ as inputs, then selects three random numbers $(r_1, r_2, r_3) \xleftarrow{R} \mathbb{Z}_q^*$ and generates the ciphertext $C_{pk_i}(m) = \{C_{i1}, C_{i2}, C_{i3}, (s_m, h_m)\}$:

1. computes $C_{i1} = r_1 r_2 P_2$ and $C_{i2} = r_1 r_2 h_0(m) P_1 + r_3 P_1$.
2. computes $C_{i3} = (m||r_1||r_3) \oplus H_1(r_2 pk_i^1) = (m||r_1||r_3) \oplus H_1(r_2 a_i P_1)$.
3. computes $R = r_2 P_1$ and $s_m = r_2 - h_m sk_i = r_2 - h_m a_i$, where $h_m = H_2(C_{i1}||C_{i2}||R)$

Self-Decrypt$(pp, pk_i, sk_i, C_{pk_i}(m)) \rightarrow m$: The algorithm is used to decrypt the ciphertext generated by the decrypter itself. It takes the public parameter pp, the public/secret key pair $(pk_i = \{pk_i^1, pk_i^2\} = \{a_i P_1, a_i P_2\}, sk_i = a_i)$ and the ciphertext $C_{pk_i}(m) = \{C_{i1}, C_{i2}, C_{i3}, (s_m, h_m)\}$ as inputs, the process is as follows:

1. computes $r_2 = s_m + h_m sk_i = s_m + h_m a_i$.
2. computes $(m||r_1||r_3) = C_{i3} \oplus H_1(r_2 pk_i^1)$.
3. verifies $C_{i1} = r_1 r_2 P_2$ and $C_{i2} = (r_1 r_2 h_0(m) P_1 + r_3 P_1)$.

ReKey$(pp, pk_i, sk_i, C_{pk_i}(m), pk_j) \rightarrow rk_{ij}$: This algorithm is used to generate the proxy re-encryption key. It takes the public parameter pp, the ciphertext $C_{pk_i}(m) = \{C_{i1}, C_{i2}, C_{i3}, (s_m, h_m)\}$ of the user i, the public/secret key pair $(pk_i = \{pk_i^1, pk_i^2\} = \{a_i P_1, a_i P_2\}, sk_i = a_i)$ and the public key $pk_j = \{b_j P_1, b_j P_2\}$ of the data user to be authorized as inputs, and the algorithm generates a re-encryption key rk_{ij}:

1. computes $r_2 = s_m + h_m sk_i = s_m + h_m a_i$.
2. computes $rk_{ij} = H_1(r_2 pk_i^1) \oplus H_1(r_2 pk_j^1) = H_1(r_2 a_i P_1) \oplus H_1(r_2 b_j P_1)$

With the help of rk_{ij} and **ReEncrypt** algorithm, the data user j can transform the ciphertext $C_{pk_i}(m)$ into a re-encryption ciphertext, and then obtain the message m.

ReEncrypt$(pp, rk_{ij}, C_{pk_i}(m)) \rightarrow RC_{pk_j}^i(m)$: It takes the public parameter pp, the re-encryption key rk_{ij} that the user i sends to user j, and the ciphertext $C_{pk_i}(m)) = \{C_{i1}, C_{i2}, C_{i3}, (s_m, h_m)\}$, and the algorithm generates a re-encryption ciphertext $RC_{pk_j}^i(m) = \{C_{j1}, C_{j2}, C_{j3}^i, (s_m, h_m)\}$ under the public key pk_j, where

$$\begin{aligned} C_{j3}^i &= C_{i3} \oplus rk_{ij} \\ &= (m||r_1||r_3) \oplus H_1(r_2 pk_i^1) \oplus H_1(r_2 pk_i^1) \oplus H_1(r_2 pk_j^1) \\ &= (m||r_1||r_3) \oplus H_1(r_2 pk_j^1) \end{aligned}$$

$C_{j1} = C_{i1}$ and $C_{j2} = C_{i2}$.

ReDecrypt$(pp, pk_i, sk_j, RC_{pk_j}^i(m)) \rightarrow m$: The algorithm can be used by the authorized data user to obtain a trusted traffic message published by the data owner. It takes the public parameter pp, the public key $pk_i = \{pk_i^1, pk_i^2\}$ of the data owner, the secret key $sk_j = b_j$ of the authorized data user j and the re-encryption ciphertext $RC_{pk_j}^i = \{C_{j1}, C_{j2}, C_{j3}^i, (s_m, h_m)\}$ as inputs, the algorithm performs as follows:

1. computes $R = r_2 P_1 = s_m P_1 + h_m pk_i^1 = (r_2 - h_m a_i) P_1 + h_m a_i P_1$, and verifies whether the equation $h_m = H_2(C_{j1}||C_{j2}||R)$ holds.
2. computes $r_2 P_2 = s_m P_2 + h_m pk_i^2 = (r_2 - h_m a_i) P_2 + h_m a_i P_2$.
3. computes $(m||r_1||r_3) = C_{j3}^i \oplus H_1(sk_j r_2 P_1) = C_{j3}^i \oplus H_1(r_2 b_j P_1) = (m||r_1||r_3) \oplus H_1(r_2 pk_j^1) \oplus H_1(r_2 pk_j^1)$.
4. verifies $C_{j1} = r_1 r_2 P_2$ and $C_{j2} = (r_1 h_0(m) r_2 P_1 + r_3 P_1)$.

Authorization$(pp, pk_j, sk_i, C_{pk_i}(m)) \rightarrow T_{pk_i \rightarrow pk_j}$: The algorithm outputs an authorization trapdoor $T_{pk_i \rightarrow pk_j}$ corresponding to the ciphertext $C_{pk_i}(m)$ for the user j. It takes the public parameter pp, the public key $pk_j = \{pk_j^1 = b_j P_1, pk_j^2 = b_j P_2\}$ of the user pk_j to be authorized, the secret key $sk_i = a_i$ of the user pk_i and the ciphertext $C_{pk_i}(m) = \{C_{i1}, C_{i2}, C_{i3}, (s_m, h_m)\}$ under pk_i as inputs, the algorithm performs as follows:

1. recovers r_{i1}, r_{i2}, r_{i3} from $C_{pk_i}(m)$ by leveraging the **ReDecrypt** algorithm.
2. computes $T_{pk_i \rightarrow pk_j} = r_{i1} r_{i2} sk_i pk_j^1 - r_{i3} P_1 = (r_{i1} r_{i2} a_i b_j - r_{i3}) P_1$, where b_j denotes the secret key of pk_j.

Finally, the **Test** algorithm ensures our scheme to support the dynamic authorization, any two users can use it to achieve the equality test.

Test$(pp, C_{pk_i}(m), T_{pk_i \rightarrow pk_j}, C_{pk_j}(m^*), T_{pk_j \rightarrow pk_i}) \rightarrow \{0,1\}$: The algorithm is automatically executed in the blockchain-based computing platform with smart contract when a user sends the transaction. For the convenience of description, we assume that the publisher of this transaction is a data owner, while another entity is data user. The **Test** algorithm takes the public parameter pp, the data ciphertext $C_{pk_i}(m) = \{C_{i1}, C_{i2}, C_{i3}, (s_m, h_m)\}$ with the corresponding trapdoor $T_{pk_i \rightarrow pk_j} = (r_{i1} r_{i2} a_i b_j - r_{i3}) P_1$, the interest ciphertext $C_{pk_j}(m^*) = \{C_{j1}, C_{j2}, C_{j3}, (s_{m^*}, h_{m^*})\}$ with the corresponding trapdoor $T_{pk_j \rightarrow pk_i} = (r_{j1} r_{j2} b_j a_i - r_{j3}) P_1$ as inputs, it performs as follows:

1. computes $IC_i = C_{i2} + T_{pk_i \rightarrow pk_j} = r_{i1} r_{i2} h_0(m) P_1 + r_{i1} r_{i2} a_i b_j P_1 = (h_0(m) + a_i b_j) r_{i1} r_{i2} P_1$.
2. computes $IC_j = C_{j2} + T_{pk_j \rightarrow pk_i} = r_{j1} r_{j2} h_0(m^*) P_1 + r_{j1} r_{j2} a_i b_j P_1 = (h_0(m^*) + a_i b_j) r_{j1} r_{j2} P_1$.
3. if the equation $e(IC_j, C_{i1}) = e(IC_i, C_{j1})$ holds, returns 1, otherwise, returns 0.

A. Data Sharing System

Fig 3 shows the interacting process of data sharing based on BPREET in a top-down fashion. Firstly, after the system is initialized, the first thing the system users (data owners and users) have to do is generate a public/secret key pair by

leveraging the **KeyGen** algorithm. Then, data owners and users encrypt their data by calling the **Encrypt** algorithm, and then upload them to the blockchain, respectively. Meanwhile, to ensure that users can find matching data, data owners and users need to generate authorization trapdoors for their encrypted data using the **Authorization** algorithm, respectively. Then, data owners and users trigger the blockchain to execute a **Test** algorithm to find a matching ciphertext by sending a transaction. When a data owner knows the data that the authorized data user needs, (s)he generates a corresponding re-encryption key, and then uploads it to blockchain. Receiving a re-encryption key, the blockchain executes the **ReEncrypt** algorithm to generate a re-encryption ciphertext and sends it to the corresponding data user. Finally, the data user decrypts the re-encryption ciphertext using his secret key to obtain the sharing data.

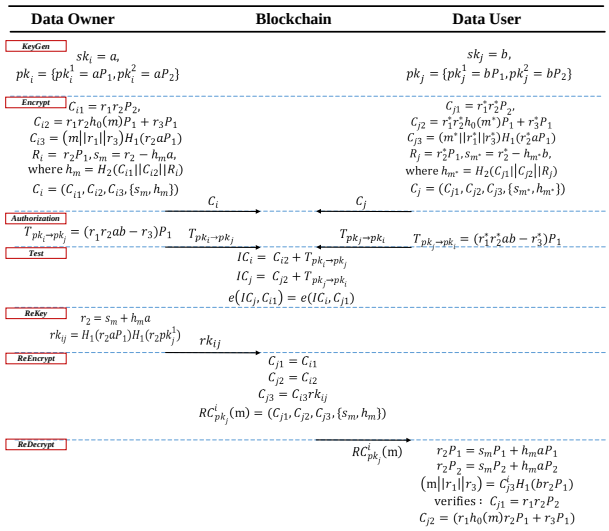


Fig. 3: Data sharing system based on BPREET

B. Contract Design

Algorithms 1 and 2 describe the smart contract design of BPREET. To avoid repetition, we omit the case that the users directly execute the **Test** algorithm of BPREET without inserting the ciphertexts into the blockchain. The smart contract used mainly includes three data structures (**Cipher_Tuple**, **Test_Tuple**, **Role_Info**) and three interfaces (**Insert**, **Update**, **Test**). The details are described as follows:

1) The structure **Cipher_Tuple** defines the storage format of ciphertexts, while structure **Role_Info** is used to store the role information about the user. Finally, the structure **Test_Tuple** contains three key information: the users' role, the ciphertext to be tested and the corresponding authorization trapdoor.
2) The interface **Query** is likely to be used frequently, mainly to find data on the blockchain by index. Then, the interface **Insert** is called to insert the ciphertexts, including the data ciphertexts of data owners and the interest ciphertexts of data users, into the blockchain. In addition, after inserting the data, users can also update their data stored on the blockchain through this interface. Finally, to determine whether two ciphertexts match, a user needs to trigger the **Test** interface.

Algorithm 1 The Structure of Smart Contract

1: **Structure** Cipher_Tuple {
2: C_1, C_2, C_3 **string**
3: s, h **string**}
4: **Structure** Test_Tuple {
5: Role **string** *% The data owner or the data user*
6: C **Cipher_Tuple**
7: Trapdoor **string**}
% Define the role information structure
8: **Structure** Role_Info {
9: ID **string** *% the user's identity*
10: Public_key **string**}

VI. Security Proof

Our construction satisfies the following three design goals.

A. Correctness

It is straightforward to see that a user (data owner or data user) can decrypt the legal ciphertexts under his/her public key, as long as (s)he inputs a correct secret key. Besides, the correctness of the **Test** algorithm is ensured by the following equation:

$$\begin{aligned} e(IC_j, C_{i1}) &= e((h_0(m^*) + a_i b_j) r_{j1} r_{j2} P_1, r_{i1} r_{i2} P_2) \\ &= e((h_0(m^*) + a_i b_j) r_{i1} r_{i2} P_1, r_{j1} r_{j2} P_2) \\ if \quad m^* = m & \\ &= e((h_0(m) + a_i b_j) r_{i1} r_{i2} P_1, r_{j1} r_{j2} P_2) \\ &= e(IC_i, C_{j1}) \end{aligned}$$

The result is credible because the **Test** algorithm is executed as a smart contract. The outputs will be stored as contract states publicly and can be verified by each miner in the blockchain-based platform. Thus, the correctness is ensured as the correct execution of each algorithm.

B. Confidentiality

Confidentiality includes two-level meanings: one means the ciphertexts no leak the information about the corresponding messages, and another means the authorization trapdoors no reveal the information about the corresponding messages. To prove the confidentiality of our proposed scheme, we define

Algorithm 2 The Function of Smart Contract

Query(key **string**):

% D is a map formed as (key, value)

Return D[key] or $\perp$.

Insert(id, $data$ **string**, $counter$, $flag$ **int**):

Retrieve the **Test_Tuple** $t = D[id||counter]$ of the data owner by $(id, counter)$ from the map D.

% flag = 0, update the ciphertext, and flag =1, update the trapdoor.

if $flag$ == 0 **then**

  Convert the **string** $data$ to the **Cipher_Tuple** structure $cdata$.*% data type conversion.*

  Set $t.C = data$.

else

  Set $t.Trapdoor = data$.

end if

Set D[id||counter] = t.

Test(Rid, Wid **string**, Rc, Wc **int**, $T_{pk_i \to pk_j}, T_{pk_j \to pk_i}$**string**):

% Rid and Wid denote the identities(ID) of users and Rc and Wc denote the counters

Retrieve the **Test_Tuple** $t_i = D[Rid||Rc]$ of the data owner pk_i and **Test_Tuple** $t_j = D[Wid||Wc]$ of the data user pk_j from the map D.*% involves data type conversion.*

if $t_i == \perp || t_j == \perp$ **then**

  return nil

end if

Parse $t_i.C$ into $(C_{i1}, C_{i2}, C_{i3}, (s_m, h_m))$

Parse $t_j.C$ into $(C_{j1}, C_{j2}, C_{j3}, (s_{m^*}, h_{m^*}))$.

Computes $IC_i = C_{r2} + T_{pk_i \to pk_j} = r_{r1} r_{r2} h_0(m) P_1 + r_{r1} r_{r2} a_r b_j P_1 = (h_0(m) + a_r b_j) r_{r1} r_{r2} P_1$.

Computes $IC_j = C_{j2} + T_{pk_j \to pk_i} = r_{j1} r_{w2} h_0(m^*) P_1 + r_{j1} r_{j2} a_r b_j P_1 = (h_0(m^*) + a_r b_j) r_{j1} r_{j2} P_1$.

if $e(IC_j, C_{r1}) == e(IC_i, C_{j1})$ **then**

  return 1

end if

return 0

two indistinguishable games: ciphertext indistinguishability game and trapdoor privacy game. These games are described by the interaction between the challenger C and the adversary $\mathcal{A}$.

Ciphertext indistinguishability: The game performs as follows:

1. $Setup$: The challenger C first takes a security parameter λ and runs the **Setup** algorithm to generate the public parameter pp. Then it runs the **KeyGen** algorithm to generate two public/secret key pairs $(pk_i, sk_i)(pk_j, sk_j)$ and sends the public information (pp, pk_i, pk_j) to $\mathcal{A}$.
2. $Phase$ 1: The adversary $\mathcal{A}$ can adaptively issue the kinds of queries to the following oracle, which are controlled by the challenger:
   - **Ciphertext Oracle** $\mathcal{O}_c$: Given a message m, the oracle runs the **Encrypt** algorithm and generates the corresponding ciphertext $C_{pk_i}(m)$ under the public key pk_i.
3. $Challenge$: The adversary $\mathcal{A}$ chooses two message (m_0, m_1) to be challenged. C randomly select a bit $b \in \{0, 1\}$, computes the ciphertext $C_{pk_i}(m_b) =$ **Encrypt**(pp, pk_i, sk_i, m_b) and sends $C_{pk_i}(m_b)$ to $\mathcal{A}$.
4. $Phase$ 2: The adversary $\mathcal{A}$ can issue queries to $\mathcal{O}_c$ and $\mathcal{O}_a$ as above, the only restriction is that neither m_0 nor m_1 could be submitted to the oracles.
5. $Guess$: $\mathcal{A}$ outputs a bit $b^{'} \in \{0, 1\}$. It wins the game if and only if $b^{'} = b$

The $\mathcal{A}$'s advantage winning the above game is defined as

$$Adv_{\mathcal{A}}^{IND}(\lambda) = |Pr[b^{'} = b] - \frac{1}{2}|$$

Theorem 1. The BPREET achieves the ciphertext indistinguishability, if the CDH assumption holds and the hash functions are collision-resistant under the random oracle model.

$Proof$. We prove through a series of games and start from the above game. In the following games, each game differs slightly from the previous game, but they all proved to be indistinguishable from each other in the view of the adversary. Finally, in the last game, the ciphertexts are completely independent of the message, that is, the ciphertext does not contain any information about the message.

Game 0. The game is the same the above ciphertext indistinguishability game. The challenger C first generates the public parameter $pp = \{q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e, h_0, H_1, H_2\}$ and two public/secret key pairs $(pk_i = \{aP_1, aP_2\}, sk_i = a)(pk_j = \{bP_1, bP_2\}, sk_j = b)$, and then C publishes the public information (pp, pk_i, pk_j) and keeps the secret keys $(sk_i = a, sk_j = b)$. To response the ciphertext oracle query, C generates the ciphertext $C_{pk_i}(m) = \{C_{i1} = r_1 r_2 P_2, C_{i2} = r_1 r_2 h_0(m) P_1 + r_3 P_1, C_{i3} = (m||r_1||r_3) \oplus H_1(r_2 a_i P_1), (s_m = r_2 - h_m a, h_m = H_2(C_{i1}||C_{i2}||R))\}$ of m, where $r_1, r_2, r_3 \xleftarrow{R} \mathbb{Z}_q^*$. This process simulates the adversary's ability to acquire ciphertexts. To response the trapdoor oracle query, C computes the trapdoor $T_{pk_i \to pk_j} = (r_1 r_2 ab - r_{i3}) P_1$ of $C_{pk_i}(m)$. This process simulates the adversary's ability to acquire trapdoors.

By repeating the above queries, the adversary guesses the challenge ciphertext based on the information (s)he has. The

advantage that (s)he wins in this game is

$$Adv_{\mathcal{A}}^{\textbf{Game}\,0}(\lambda) = Adv_{\mathcal{A}}^{IND}(\lambda)$$

Game 1. The game is the same as the **Game 0** except that instead of using a stand hash function H_1, we model all hash functions as the random oracles. When receiving an input x, the challenger first checks the hash list, if there exists a value y corresponding to x, and then returns y. Otherwise, the challenger randomly chooses a value y and sets $H_1(x) = y$. It is obvious that **Game 1** and the above game are indistinguishable, otherwise, the adversary can break the security of hash function. Therefore, we have the advantage of $\mathcal{A}$ winning this game:

$$Adv_{\mathcal{A}}^{\textbf{Game}\,1}(\lambda) = Adv_{\mathcal{A}}^{\textbf{Game}\,0}(\lambda)$$

Game 2. The game is the same as the **Game 1** except that instead of computing the challenge ciphertext $C_{i3} = (m_b||r_1||r_3) \oplus H_1(r_2 pk_i^1)$ in the encryption phase, the challenger chooses a random string $S \in \{0,1\}^*$ and computes the new challenge ciphertext $C_{i3}^* = (m_b||r_1||r_3) \oplus S$. We know that if the adversary $\mathcal{A}$ can distinguish ciphertext C_{i3} from ciphertext C_{i3}^*, which means $\mathcal{A}$ can determine the relationship between S and $H_1(r_2 pk_i^1) = H_1(r_2 aP_1)$ based on the knowledge (P_1, r_2P_1, aP_1). From the view of the adversary, the tuple (aP_1, r_2P_1, r_2aP_1) constructs a CDH problem since (r_2, a) are unknown and random. Therefore, the adversary cannot distinguish C_{i3} from C_{i3}^*. Finally, we have the advantage of $\mathcal{A}$ winning this game:

$$|Adv_{\mathcal{A}}^{\textbf{Game}\,2}(\lambda) - Adv_{\mathcal{A}}^{\textbf{Game}\,1}(\lambda)| \leq Adv_{\mathcal{A}}^{CDH}(\lambda)$$

Game 3. The game differs from the **Game 2** in two places. Firstly, the challenger chooses a point $C_{i2}^* \xleftarrow{R} \mathbb{G}_1$ instead of computing $C_{i2} = r_1 r_2 h_0(m_b) P_1 + r_3 P_1$. Because the randomness of r_1 and r_3, C_{i2}^* and C_{i2} are indistinguishable in the view of $\mathcal{A}$. Secondly, C randomly chooses a string $C_{i3}^{**} \xleftarrow{R} \{0,1\}^*$ instead of computing $C_{i3}^* = (m_b||r_1||r_3) \oplus S$. Because the randomness of S, C_{i3}^{**} and C_{i3}^* are indistinguishable in the view of $\mathcal{A}$. Therefore, we have the advantage of $\mathcal{A}$ winning this game:

$$Adv_{\mathcal{A}}^{\textbf{Game}\,3}(\lambda) = Adv_{\mathcal{A}}^{\textbf{Game}\,2}(\lambda)$$

However, we find the challenge ciphertext has become completely independent of message m_b, which means the probability that $\mathcal{A}$ wins the above game is equal to $\frac{1}{2}$. Therefore, we have the advantage $Adv_{\mathcal{A}}^{\textbf{Game}\,3}(\lambda) = |\frac{1}{2} - \frac{1}{2}| = 0$ and

$$Adv_{\mathcal{A}}^{IND}(\lambda) \leq Adv_{\mathcal{A}}^{CDH}(\lambda)$$

where the advantage $Adv_{\mathcal{A}}^{CDH}(\lambda)$ is negligible if the CDH assumption holds.

Theorem 2. The BPREET achieves the trapdoor indistinguishability, if the DBDH assumption holds and the hash functions are collision-resistant under the random oracle model.

$Proof$. Compared with the above game, the adversary $\mathcal{A}$ has the additional trapdoor information corresponding to the ciphertext. Therefore, we only need to prove that the trapdoor information is not helpful for the adversary to guess the plaintext, that is, the trapdoor no reveals the information about the message.

Given the trapdoor $T_{pk_i \to pk_j} = (r_{i1} r_{i2} a_i b_j P_1 - r_{i3} P_1)$ corresponding to the ciphertext $C_{pk_i}(m) = \{C_{i1}, C_{i2}, C_{i3}, (s_m, h_m)\}$, the adversary can obtain the intermediate ciphertext $IC_i = C_{i2} + T_{pk_i \to pk_j} = (h_0(m) + a_i b_j) r_{i1} r_{i2} P_1$. Due to the randomness of r_{i2}, IC_i and a random point $S \xleftarrow{R} \mathbb{G}_1$ are indistinguishable. Besides, we assume the adversary choses a guess message m^*, and then the adversary can compute $h_0(m^*) r_{i1} r_{i2} P_2$ from the ciphertext C_{i1}. According to the tuple $(P_1, a_i P_1, r_{i1} r_{i2} P_2, b_j P_1, h_0(m^*) r_{i1} r_{i2} P_2)$, the adversary cannot determine whether the tuple is an DBDH tuple $(P_1, a_i P_1, h_0(m^*) r_{i1} r_{i2} P_2, b_j P_1, e(a_i b_j P_1, h_0(m^*) r_{i1} r_{i2} P_2))$ or a random tuple $(P_1, a_i P_1, h_0(m^*) r_{i1} r_{i2} P_2, b_j P_1, Z \in \mathbb{G}_T)$ if the DBDH problem is hard. Therefore, the adversary cannot obtain information about the plaintext through keyword guessing attack, even if (s)he obtains the corresponding trapdoor information.

C. Decentralization

In our scheme, instead of using a central server, the **Test** algorithm is implemented as a smart contract and the test results are public and verifiable. We don't have to assume, as most existing solutions do, that there is a semi-honest server which honestly executes our scheme. Meanwhile, the consensus mechanism of blockchain ensures that each test operation is correctly executed. In practical applications, if the executor of the smart contract returns the error result for some reasons, the malicious operation will be detected by other miners and the executor will get nothing in return. Therefore, the BPREET supports decentralization, due to the decentralization of blockchain technology.

VII. Evaluation

We compare the features of our proposed scheme (BPREET) and three other schemes (i.e., PRES [40], APTM [7], and BPTM [15]), in terms of utility, computational complexity, and communication cost. Then, we implement a prototype of BPREET and evaluate its performance.

A. Comparative Summary

In our comparative summary, we focus on four key features, namely: *Decentralization*, *Flexibility*, *Non-interaction* and *Re-encryption*.

1) *Decentralization*: A decentralized scheme is one that does not require a central server (or the proxy); thus, avoiding the associated management cost and server-related vulnerabilities.
2) *Flexibility*: A flexible scheme is one that achieves flexible authorization and supports the single-single model (S/S), the single-multi model (S/M), and the multi-multi model (M/M).
3) *Non-interaction*: This feature implies that no additional round of communication is required between the data owners and data users.

4) *Re-encryption*: This feature is primarily for the delegation process, which implies the scheme's ability to share data.

Table II summarizes the four key features of BPREET, PRES, APTM, and BPTM. One can observe that BPREET supports flexible authorization (unlike PRES, APTM, and BPTM), decentralization (unlike PRES and APTM), and re-encryption as well as avoiding the need for data owner-user interaction.

Tables III and IV summarize the computation and communication costs for all the four schemes, respectively. Let E_1, E_2, E_T denote the scalar multiplication operations (or called group exponentiation) in groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, respectively. Let e denotes the bilinear pairing operation on $e(\mathbb{G}_1, \mathbb{G}_2) \rightarrow \mathbb{G}_T$ and $H_{\mathbb{G}_1}, H_{\mathbb{G}_2}$, denotes two hash-to-point operations on $H_{\mathbb{G}_1} : \{0,1\}^* \rightarrow \mathbb{G}_1, H_{\mathbb{G}_2} : \{0,1\}^* \rightarrow \mathbb{G}_2$ respectively. Let **xor** denotes a xor operation, and its computational complexity is usually negligible.

TABLE II: A comparative summary of four key features

Scheme	Decentralization	Flexibility	Non-interaction	Re-encryption
PRES [40]	×	×	✓	✓
APTM [7]	×	×	✓	×
BPTM [15]	✓	×	×	×
BPREET	✓	✓	✓	✓

Encryption: In this phase, the data owner (or the data user) generates the data ciphertext. The **Encrypt** algorithm will be frequently invoked, so the efficiency of it will directly affect the user experience. Compared with the other schemes (PRES, APTM, BPTM), the computational complexity of BPREET is lower because it avoids the time-consuming bilinear pairing operation. However, we also observe that the communication cost of ciphertexts in BPREET is $|\mathbb{G}_1| + |\mathbb{G}_2| + 2|\mathbb{Z}_q|$, which slightly higher than that of other schemes.

Trapdoor: In the trapdoor generation, the data owner (or the data user) generates the authorization trapdoor for equality test. The complexity of these four schemes (PRES, APTM, BPTM, BPREET) are $E_1 + H_{\mathbb{G}_1}$,$E_1 + H_{\mathbb{G}} + (9E_1 + 3E_T + 3e_{12})$, e and $2E_1$, respectively. Among the four schemes, the computational complexity of BPREET is very close to that of PRES, which has the lowest complexity. In addition, the communication cost for both PRES and BPREET is the group element size $|\mathbb{G}_1|$, while that of BPTM is only λ, the value of which usually is 128 bits.

Other algorithms: Compared with another proxy re-encryption scheme (PRES), BPREET is more efficient in terms of generating re-encryption ciphertext and decryption since the **Re-Encrypt** algorithm only takes an **xor** operation and the decryption algorithms (**Self-Decrypt, ReDecrypt**) need E_1 or $3E_1 + 2E_2$.

B. Prototype Evaluation

To evaluate the performance of BPREET, we implement a prototype of our proposed approach and perform our experiments on a Windows 7 machine (64bit OS, i5-4210U CPU @1.70GHz 2.40GHz, 4GB RAM) and a Ubuntu 16.04 machine (64bit OS,i5-6500 CPU @ 3.20GHz×4, 8GB RAM, gcc version 5.4.0). The elliptic curve in our experiments is an asymmetric elliptic curve called BN256 (parameter: $y^2 = x^3 + 2$, based field:$36X^4 + 36X^3 + 24X^2 + 6X + 1$ and $X = -4080000000000001$) with AES-128 security level, where $|\mathbb{G}_1| = 512$ bits, $|\mathbb{G}_2| = 1024$ bits and $|\mathbb{G}_T| = 3072$ bits.

We also implement the related main cryptographic operations using the popular C/C++ cryptography library (MIRACL 7.0)[1], and the associated time costs are shown in Table V where $e > E_T > E_2 > H_{\mathbb{G}_2} > E_1 > H_{\mathbb{G}_1}$. In Fig. 4, we vary the number of messages between 10 and 100 to evaluate the time cost of all phases (except for the **Test** phase), where the messages are randomly selected. The performance of encryption algorithm of BPREET is the best among all the schemes, and the time cost to encrypt 10 messages is at most 503.3 ms. One can observe that PRES, APTM and BPTM need 2639.9 ms, 1378.1 ms, 1866.2 ms, respectively. Although BPREET's communication cost is slightly higher than those in APTM and BPTM, it is far less than that in PRES. Moreover, the computation complexity of all the schemes increases linearly with the number of messages. A comparative summary of communication costs is depicted in Fig.5. Clearly, BPREET does not perform as well in terms of communication cost. We consider this can be seen as a tradeoff for efficiency.

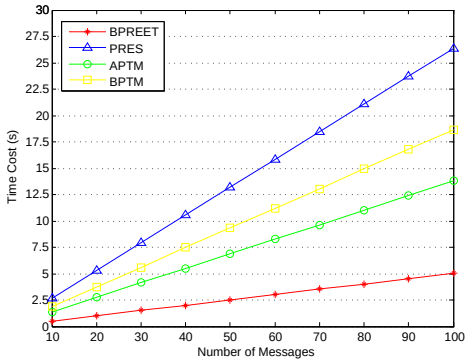


Fig. 4: Time cost of encryption algorithm

After the ciphertexts are uploaded, one of the main jobs on the data owner/user is to generate the authorization trapdoor. Therefore, the performance of the authorization phase is crucial to the whole scheme. Fig. 6 shows the time cost of the authorization algorithm with respect to the number of messages, in comparison with PRES, APTM, and BPTM. The time cost in the BPREET is more efficient than the APTM and BPTM. To generate 100 the authorization trapdoors, BPREET only takes 1.9 s while the APTM and the BPTM need 59.3s and 11.662s respectively. Fig. 7 shows a comparative summary of communication costs. Since the trapdoor in BPTM is a pointer, its communication cost is the

[1]https://www.miracl.com/

TABLE III: Computation complexity: A comparative summary

Scheme	Encryption	Re-Encryption	Trapdoor	Test	Decryption
PRES [40]	$3E_1 + 2e + H_{\mathbb{G}_1}$	E_1	$E_1 + H_{\mathbb{G}_1}$	e	$e + E_T$
APTM [7]	$2E_1 + e + H_{\mathbb{G}_1}$	-	$E_1 + H_{\mathbb{G}} + (9E_1 + 3E_T + 3e_{12})^*$	$e + (8E_1 + 5E_T + 5e_{12} + H_{\mathbb{G}})^*$	-
BPTM [15]	$2E_1 + +E_T + e + H_{\mathbb{G}_1}$	-	e	$-$	e
BPREET	$3E_1 + 1E_2$	**xor**	$3E_1$	e	E_1 (or $3E_1 + 2E_2$)**

* APTM prevents outside malicious users from requesting the query services by utilizing a group signature.

** E,$3E$ represent the computation complexity of **Self-Decrypt** algorithm and **ReDecrypt** algorithm respectively. (exclude the verification process).

TABLE IV: Communication costs: A comparative summary

Scheme	Ciphertext	Trapdoor	Re-key
PRES [40]	$3\|\mathbb{G}_1\| + 2\|\mathbb{G}_T\| + \|\lambda\|$	$\|\mathbb{G}_1\|$	$\|\mathbb{Z}_q\|$
APTM [7]	$\|\mathbb{G}_1\| + \|\lambda\|$	$\|\mathbb{G}_1\| + 3\|\mathbb{G}_1\| + 6\|\mathbb{Z}_q\|$	$-$
BPTM [15]	$\|\mathbb{G}_1\| + 3\|\lambda\|$	$\|\lambda\|$	$-$
BPREET	$\|\mathbb{G}_1\| + \|\mathbb{G}_2\| + 2 * \|\mathbb{Z}_q\|$	$\|\mathbb{G}_1\|$	$\|\lambda\|$

TABLE V: Time cost of some cryptographic operations

Operation	E_1	E_2	E_T	e	$H_{\mathbb{G}_1}$	$H_{\mathbb{G}_2}$
Time cost[1](ms)	9.56	21.65	48.81	116.62	2.07	14.32

[1] The average time required to execute one operation (repeat 10000 times).

lowest. The communication cost of generating a trapdoor in BPREET is equal to that in PRES, but significantly lower than that in APTM even though the APTM utilizes the aggregation method for optimization. After finding the matching messages, the data user needs to execute the decryption algorithm to obtain accurate information. Fig. 8 shows the time cost to decrypt the data ciphertext under different schemes (except for the APTM). The **Self-Decryption** algorithm only takes 0.95 s to decrypt 100 ciphertexts while the **ReDecryption** algorithm needs about 7 s. The decryption algorithms in BPREET far outperform the PRES and the BPTM. Moreover, the time cost will become more efficient than PRES and BPTM with the increasing number of ciphertexts.

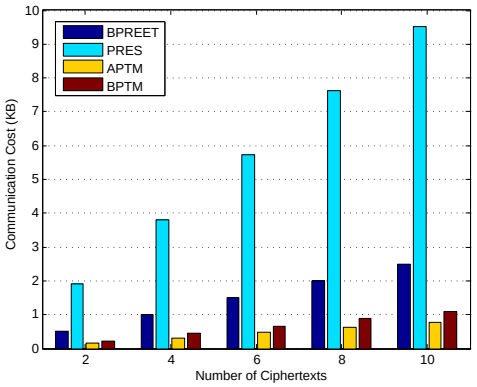


Fig. 5: Communication costs for ciphertexts

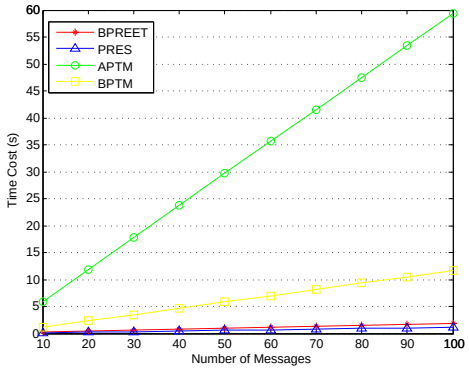


Fig. 6: Time cost of generating the authorization trapdoor

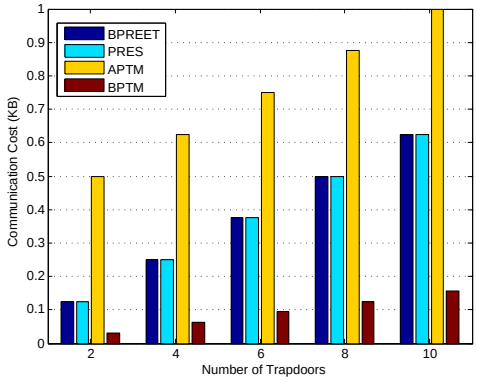


Fig. 7: Communication costs for trapdoors

To evaluate the performance of the smart contract, we use the popular hyperledger fabric (v1.4.2) [2] as the test blockchain platform and the fabric-sample repository [3] is used to test

[2]https://www.hyperledger.org/

[3]https://github.com/hyperledger/fabric-samples/tree/release-1.4/chaincode-docker-devmode

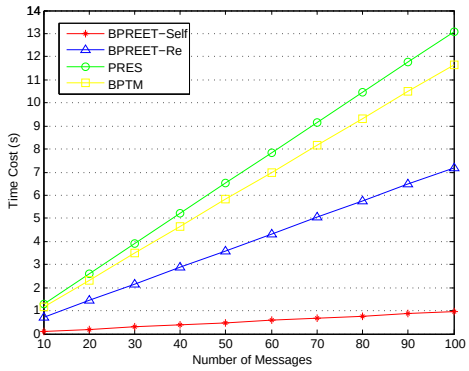


Fig. 8: Time cost of decryption

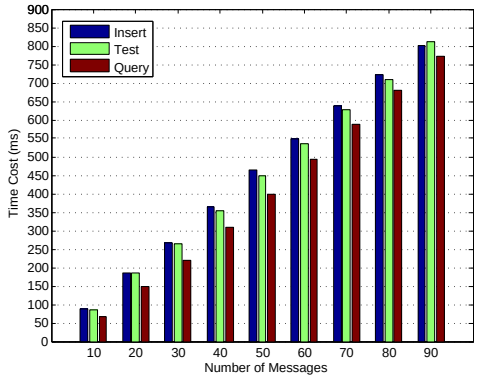


Fig. 9: Time cost of every algorithm in smart contract

the smart contract. Fig. 9 shows that the time cost takes an endorser peer to execute every algorithm deployed on the smart contract in a locally simulated network. Specifically, the total time the **Insert** algorithm takes to upload 10 tuple ciphertexts is approximately 87.3 ms, and the **Test** algorithm takes 86.6 ms to complete 10 message-interest ciphertext pairs. Besides, the **Query** algorithm requires 66 ms to query 10 records stored on the blockchain. Obviously, the time cost of every algorithm grows with an increase in the number of messages. Unlike the BPTM, which uses the public blockchain (Ethereum), the BPREET is more suitable for deployment in the consortium blockchain with lower cost and higher efficiency.

VIII. Conclusion

In this work, we focused on the security and utility issues in VCS and constructed a blockchain-based proxy re-encryption scheme supporting the equality test (BPREET). The BPREET scheme combines the advantages of both the PRE and the PKEET, and includes ciphertext conversion and ciphertext equality test under different public keys. By leveraging blockchain, we avoid the need to rely on a central server to manipulate test jobs. A comparative summary with three other existing solutions demonstrated the potential of BPREET in practice, although one future agenda is to deploy a prototype of BPREET in a real-world setting.

References

[1] J. VOELCKER, "1.2 billion vehicles on world's roads now, 2 billion by 2035: Report." https://www.greencarreports.com/news/1093560_1-2-billion-vehicles-on-worlds-roads-now-2-billion-by-2035-report/, 2014.

[2] A. A. Association, "Crashes vs. congestion–what's the cost ot society?." http://www.camsys.com/pubs/2011_AAA_CrashvCongUpd.pdf/, 2014.

[3] N. Kumar, N. Chilamkurti, and J. H. Park, "Alca: agent learning-based clustering algorithm in vehicular ad hoc networks," *Personal & Ubiquitous Computing*, vol. 17, no. 8, pp. 1683–1692, 2013.

[4] N. Kumar, R. Iqbal, S. Misra, and J. J. P. C. Rodrigues, "Bayesian coalition game for contention-aware reliable data forwarding in vehicular mobile cloud," *Future Generation Computer Systems*, vol. 48, no. jul., pp. 60–72, 2015.

[5] N. Kumar, S. Misra, J. J. P. C. Rodrigues, and M. S. Obaidat, "Coalition games for spatio-temporal big data in internet of vehicles environment: A comparative analysis," *IEEE Internet of Things Journal*, vol. 2, no. 4, pp. 310–320, 2015.

[6] R. S. Bali and N. Kumar, *Secure clustering for efficient data dissemination in vehicular cyber-physical systems*, vol. 56. Elsevier Science Publishers B. V., 2016.

[7] J. Shu, X. Liu, X. Jia, K. Yang, and R. H. Deng, "Anonymous privacy-preserving task matching in crowdsourcing," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3068–3078, 2018.

[8] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, "Efficient encrypted keyword search for multi-user data sharing," in *European symposium on research in computer security*, pp. 173–195, Springer, 2016.

[9] J. Shu, X. Jia, K. Yang, and H. Wang, "Privacy-preserving task recommendation services for crowdsourcing," *IEEE Transactions on Services Computing*, 2018.

[10] S. Ma, Q. Huang, M. Zhang, and B. Yang, "Efficient public key encryption with equality test supporting flexible authorization," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 458–470, 2015.

[11] L. Wu, Y. Zhang, K.-K. R. Choo, and D. He, "Efficient and secure identity-based encryption scheme with equality test in cloud computing," *Future Generation Computer Systems*, vol. 73, pp. 22–31, 2017.

[12] J. Shu and X. Jia, "Secure task recommendation in crowdsourcing," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2016.

[13] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 792–800, IEEE, 2018.

[14] C. Cai, J. Weng, X. Yuan, and C. Wang, "Enabling reliable keyword search in encrypted decentralized storage with fairness," *IEEE Transactions on Dependable and Secure Computing*, 2018.

[15] Y. Wu, S. Tang, B. Zhao, and Z. Peng, "Bptm: Blockchain-based privacy-preserving task matching in crowdsourcing," *IEEE Access*, vol. 7, pp. 45605–45617, 2019.

[16] C. Lin, D. He, X. Huang, K.-K. R. Choo, and A. V. Vasilakos, "Bsein: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0," *Journal of Network and Computer Applications*, vol. 116, pp. 42–52, 2018.

[17] C. Lin, D. He, N. Kumar, X. Huang, P. Vijaykumar, and K.-K. R. Choo, "Homechain: A blockchain-based secure mutual authentication system for smart homes," *IEEE Internet of Things Journal*, 2019. DOI:10.1109/JIOT.2019.2944400.

[18] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *The 25^{th} USENIX Security Symposium (USENIX Security 16)*, pp. 707–720, 2016.

[19] C. Zuo, J. Shao, J. K. Liu, G. Wei, and Y. Ling, "Fine-grained two-factor protection mechanism for data sharing in cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 186–196, 2017.

[20] S. Myers and A. Shull, "Practical revocation and key rotation," in *Cryptographers' Track at the RSA Conference*, pp. 157–178, Springer, 2018.

[21] Y. Lu and J. Li, "A pairing-free certificate-based proxy re-encryption scheme for secure data sharing in public clouds," *Future Generation Computer Systems*, vol. 62, pp. 140–147, 2016.

[22] A. Manzoor, M. Liyanage, A. Braeken, S. S. Kanhere, and M. Ylianttila, "Blockchain based proxy re-encryption scheme for secure iot data sharing," *arXiv preprint arXiv:1811.02276*, 2018.

[23] M. A. Ferrag and A. Ahmim, "Esspr: an efficient secure routing scheme based on searchable encryption with vehicle proxy re-encryption for vehicular peer-to-peer social network," *Telecommunication Systems*, vol. 66, no. 3, pp. 481–503, 2017.

[24] Z. Yang, H. Yu, J. Tang, and H. Liu, "Toward keyword extraction in constrained information retrieval in vehicle social network," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4285–4294, 2019.

[25] G. S. Poh, J. Chin, W. Yau, K. R. Choo, and M. S. Mohamad, "Searchable symmetric encryption: Designs and challenges," *ACM Comput. Surv.*, vol. 50, no. 3, pp. 40:1–40:37, 2017.

[26] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Information Sciences*, vol. 403, pp. 1–14, 2017.

[27] P. Xu, S. He, W. Wang, W. Susilo, and H. Jin, "Lightweight searchable public-key encryption for cloud-assisted wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3712–3723, 2017.

[28] G. Yang, C. H. Tan, Q. Huang, and D. S. Wong, "Probabilistic public key encryption with equality test," in *Cryptographers' Track at the RSA Conference*, pp. 119–131, Springer, 2010.

[29] S. Ma, "Identity-based encryption with outsourced equality test in cloud computing," *Information Sciences*, vol. 328, pp. 389–402, 2016.

[30] Y. Ling, S. Ma, Q. Huang, X. Li, and Y. Ling, "Group public key encryption with equality test against offline message recovery attack," *Information Sciences*, 2019.

[31] D. H. Duong, K. Fukushima, S. Kiyomoto, P. S. Roy, and W. Susilo, "A lattice-based public key encryption with equality test in standard model," in *Australasian Conference on Information Security and Privacy*, pp. 138–155, Springer, 2019.

[32] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang, "Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2204–2220, 2018.

[33] M. Singh and S. Kim, "Branch based blockchain technology in intelligent vehicle," *Computer Networks*, vol. 145, pp. 219–231, 2018.

[34] R. Chaudhary, A. Jindal, G. S. Aujla, S. Aggarwal, N. Kumar, and K.-K. R. Choo, "Best: Blockchain-based secure energy trading in sdn-enabled intelligent transportation system," *Computers & Security*, vol. 85, pp. 288–299, 2019.

[35] Z. Lu, Q. Wang, G. Qu, H. Zhang, and Z. Liu, "A blockchain-based privacy-preserving authentication scheme for vanets," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2792–2801, 2019.

[36] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar, "A survey on privacy protection in blockchain system," *Journal of Network and Computer Applications*, vol. 126, pp. 45–58, 2019.

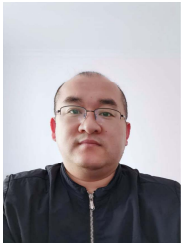
[37] S. Aggarwal, R. Chaudhary, G. S. Aujla, N. Kumar, K.-K. R. Choo, and A. Y. Zomaya, "Blockchain for smart communities: Applications, challenges and opportunities," *Journal of Network and Computer Applications*, 2019.

[38] A. Jindal, G. S. Aujla, and N. Kumar, "Survivor: A blockchain based edge-as-a-service framework for secure energy trading in sdn-enabled vehicle-to-grid environment," *Computer Networks*, vol. 153, pp. 36–48, 2019.

[39] A. Miglani, N. Kumar, V. Chamola, and S. Zeadally, "Blockchain for internet of energy management: Review, solutions, and challenges," *Computer Communications*, 2020.

[40] J. Shao, Z. Cao, X. Liang, and H. Lin, "Proxy re-encryption with keyword search," *Information Sciences*, vol. 180, no. 13, pp. 2576–2587, 2010.

Biwen Chen received his M.S. degree from Hubei University of Technology of China in 2016. He is currently pursing his Ph.D. degree at Computer School, Wuhan University, Wuhan, China. His main research interests include cryptography and information security, in particular, cryptographic protocols.



Debiao He received his Ph.D. degree in applied mathematics from School of Mathematics and Statistics, Wuhan University, Wuhan, China in 2009. He is currently a professor of the School of Cyber Science and Engineering, Wuhan University, Wuhan, China. His main research interests include cryptography and information security, in particular, cryptographic protocols. He has published over 100 research papers in refereed international journals and conferences, such as IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Security and Forensic, and Usenix Security Symposium. He is the recipient of the 2018 IEEE Sysems Journal Best Paper Award and the 2019 IET Information Security Best Paper Award. His work has been cited more than 7000 times at Google Scholar. He is in the Editorial Board of several international journals, such as Journal of Information Security and Applications, Frontiers of Computer Science, and Human-centric Computing & Information Sciences.

Neeraj Kumar received his Ph.D. in CSE from Shri Mata Vaishno Devi University, Katra, India. He is now an Associate Professor in the Department of Computer Science and Engineering, Thapar University, Patiala, Punjab (India). He is a member of IEEE. His research is focused on mobile computing, parallel/distributed computing, multi-agent systems, service oriented computing, routing and security issues in mobile ad hoc, sensor and mesh networks. He has more than 100 technical research papers in leading journals such as-IEEE TII, IEEE TIE, IEEE TDSC, IEEE ITS, IEEE TWPS, IEEE SJ,IEEE ComMag, IEEE WCMag, IEEE NetMag and conferences. His research is supported from DST, TCS and UGC. He has guided many students leading to M.E. and Ph.D.

Huaqun Wang received the BS degree in mathematics education from the Shandong Normal University and the MS degree in applied mathematics from the East China Normal University, both in China, in 1997 and 2000, respectively. He received the Ph.D. degree in Cryptography from Nanjing University of Posts and Telecommunications in 2006. He is currently a Professor of Nanjing University of Posts and Telecommunications, China. His research interests include applied cryptography, network security, and cloud computing security.

Kim-Kwang Raymond Choo (SM'15) received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). In 2016, he was named the Cybersecurity Educator of the Year - APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn), and in 2015 he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He is the recipient of the 2019 IEEE Technical Committee on Scalable Computing (TCSC) Award for Excellence in Scalable Computing (Middle Career Researcher), 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, Outstanding Associate Editor of 2018 for IEEE Access, British Computer Society's 2019 Wilkes Award Runner-up, 2019 EURASIP Journal on Wireless Communications and Networking (JWCN) Best Paper Award, Korea Information Processing Society's Journal of Information Processing Systems (JIPS) Survey Paper Award (Gold) 2019, IEEE Blockchain 2019 Outstanding Paper Award, International Conference on Information Security and Cryptology (Inscrypt 2019) Best Student Paper Award, IEEE TrustCom 2018 Best Paper Award, ESORICS 2015 Best Research Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award in 2008.