This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Privacy-Preserving Smart Contracts for Confidential Transactions Using Dual-Mode Broadcast Encryption

Hongjian Yin, Yan Zhu, *Member, IEEE*, Guanglai Guo, and William Cheng-Chung Chu, *Senior Member, IEEE*

***Abstract*—Blockchain-based smart legal contract, as a legally binding executable contract, has attracted extensive attentions in trade finance. However, since the contract is deployed on open and transparent blockchain network, all transaction data are publicly visible, which brings to privacy disclosure problem. Aiming at this problem, we introduce an improved architecture of smart legal contract with privacy protection, involving contract development, deployment, and execution. In this architecture, the sensitive data of transaction are declared and protected in the form of contract terms. These terms allow the compiler to link predefined cryptographic algorithms into smart contract programs, and then to generate executable contract code. Furthermore, as predefined cryptographic algorithms, we construct a new dual-mode identity-based broadcast encryption (DM-IBBE) scheme to meet specific-purpose or generic-purpose privacy by using selective encryption mode or exclusive encryption mode, respectively. We proved that our DM-IBBE scheme is semantically secure under the decisional Diffie–Hellman assumption. In addition, our experimental results show that the proposed scheme can satisfy the privacy requirements of transaction, and it is practicable and easy-to-develop for introducing privacy preserving mechanisms into smart legal contract languages.**

***Index Terms*—Blockchain smart legal contract (SLC), broadcast encryption (BE), dual-mode, privacy protection, reliability.**

I. Introduction

SMART contracts (SCs) are programs on blockchain that can be automatically executed according to the terms of contracts. It is known as the technical core of the second-generation blockchain. Due to the features of programmability, automatic execution, effective supervision and legalization, blockchain SC can guarantee the security and reliability of transaction data. And it has attracted extensive attentions and been used in many applications, such as finance, digital asset management, energy trading, and so forth [1], [2], [3], [4].

Blockchain is a reliable way to record transactions because all nodes in the network maintain a full copy of the ledger. This ensures that the ledger is always available, even if one node goes offline. However, this also means that SCs deployed on the blockchain are publicly available. This can lead to the leakage of sensitive data in contract transactions, as anyone can view the contract code and the transaction history. Furthermore, it could be used to steal assets, commit fraud, or track the activities of individuals.

This is a major security risk, especially for SCs that handle sensitive data, such as auction contracts. For example, the reserve price of an auction should be kept secret to all bidders before the bid is opened. If the SC for the auction is not privacy preserving, then all bidders will be able to see the reserve price stored on the blockchain. This will give bidders an unfair advantage and could lead to the auction being rigged.

According to Global Smart Contracts Market Research Report 2023,[1] the global SCs market is predicted to reach approximately $1460.3 million by the end of 2029. As SCs become more widely used in fields, such as healthcare, supply chain, and financial services, the issue of privacy leakage in transactions is becoming increasingly serious. Two significant incidents involving vulnerabilities in SC include the DAO hack in 2016, which resulted in the theft of around $50 million worth of digital currency, and the Poly Network hack in August 2021, which led to the theft of over $600 million worth of digital currency. Once privacy disclosure occurs, not only will it result in serious business losses, but also infringe on personal interests. Thus, it is imperative to find ways to ensure private and reliable transactions in blockchain SCs, enable precise execution of transactions in accordance with contract terms, and safeguard the confidentiality of sensitive transaction data.

A. Challenge and Approach

In this article, we will present a practical method for privacy-preserving smart contracts (PPSC). PPSCs are a type of SC that allows users to keep their transaction details private. This is in contrast to traditional SCs, where all of this information is publicly visible on the blockchain. Obviously, encryption is a common method for achieving PPSCs, but there are many

Manuscript received 15 December 2021; revised 27 April 2022, 31 March 2023, and 21 August 2023; accepted 25 October 2023. This work was supported in part by the National Key Technologies Research and Development Programs of China under Grant 2018YFB1402702, and in part by the National Natural Science Foundation of China under Grant 61972032. Associate Editor: Ruizhi Gao. *(Corresponding authors: Yan Zhu; William Cheng-Chung Chu.)*

Hongjian Yin is with the School of Computer and Information Engineering, Henan University, Kaifeng 475004, China (e-mail: yinhongjian@henu.edu.cn).

Yan Zhu and Guanglai Guo are with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China (e-mail: zhuyan@ustb.edu.cn; guanglai_guo@126.com).

William Cheng-Chung Chu is with the Department of Computer Science, Tunghai University, Taichung City 40704, Taiwan (e-mail: cchu@thu.edu.tw).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TR.2023.3328146.

Digital Object Identifier 10.1109/TR.2023.3328146

[1][Online]. Available: https://reports.valuates.com/market-reports/QYRE-Auto-31L1599/global-smart-contracts

0018-9529 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

practical challenges when we consider how to implement PPSCs.

First, some researchers have attempted to introduce encryption schemes into blockchain to achieve a **system-level protection of sensitive data** for all transactions. This method is transparent to users and does not require secondary development by application programmers. However, the following factors deters the implementing of PPSCs.

1) *Key management:* will be complex because it usually require a central authority to generate and manage the keys, which contradicts the decentralization of blockchain.
2) *Key distribution:* distributing decryption keys to all of the nodes in a blockchain network can be difficult and time-consuming.
3) *Key compromise:* If the decryption keys are compromised in a system-level cryptosystem, then all of the data that were encrypted with those keys would be vulnerable.

For these reasons, blockchain networks typically do not use encryption to encrypt the data itself.

Another approach is to directly introduce encryption technology into SCs. This approach targets specific transactions, rather than the entire blockchain system. Hence, we call it **business-level protection technology**. It is a promising approach to protecting sensitive data on the blockchain because there are the following advantages.

1) It can be used to introduce different encryption schemes for various business scenarios.
2) It can solve key distribution and management problems through contract programs.
3) Key compromise only harms one's own business and will not harm other businesses or blockchain systems.

On the other hand, this method has two main disadvantages as follows.

1) It requires application developers to have a deep mathematical background working cryptography.
2) There exists a secondary development process, which is relatively cumbersome.

Fortunately, a new programming method called contract-oriented programming (COP) has been proposed in recent years. Exactly, COP is an effective approach that can be used to convert real-world legal contracts to executable SC programs. In this procedure, smart legal contract (SLC) is the bridge between them. Different from SC, SLC belongs to a legally binding agreement described in natural language, in which some terms could be expressed and implemented by computer-readable code.

In Fig. 1, we show the general procedure of contract conversion. First, the contract described in natural language can be translated into SLC programs written in SLC-language and further converted into SC written in SC language (e.g., solidity, serpent). Finally, the SC is uploaded into blockchain networks in the form of transaction. In the abovementioned conversion procedure, SLC acts as both of programming and development tools.

In paper contracts, confidentiality terms are usually used to stipulate the obligations of relevant parties for protecting the privacy information. For example, in the auction contract

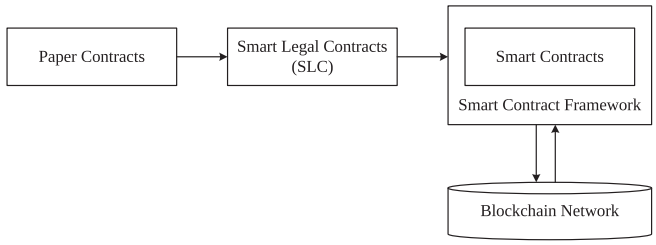

Fig. 1. Procedure of contract conversion.

mentioned previously, the auctioneer should guarantee the confidentiality of reserve price before bid opening, and must bear corresponding legal liabilities for the consequences caused by the price disclosure. In view of this demand, SCL should provide privacy-protection mechanism to ensure the implementation of confidentiality terms during the procedure of contract conversion in Fig. 1.

As far as we are concerned, the most effective way to protect the privacy of sensitive data is encryption. Hence, we hope to find a cryptographic scheme to satisfy different privacy demands in transactions. We roughly classify these privacy demands into two following categories based on the intended audience who can access sensitive data of transactions.

1) *Specific-purpose privacy:* Private data in transactions can be only obtained by a specific user group. It means that only the user who is in the specific group S can get the corresponding private data.
2) *Generic-purpose privacy:* Except some conflicting users, the private data in transactions can be obtained by anyone in this system. It means that all users who are *not* in the conflicting group R can get the corresponding private data.

It is worth noting that, in the abovementioned privacy demands, specific group refer to users who are granted access to sensitive transaction data, while conflicting group refer to users who are explicitly denied access to these data.

Until now, unfortunately, few cryptographic schemes can meet the abovementioned two demands about transaction privacy simultaneously. In order to protect the privacy, including specific-purpose and generic-purpose privacy, a new kind of cryptographic scheme should be proposed to achieve two following encryption modes (called as dual mode).

1) *Selective encryption:* For specific-purpose privacy, the proposed scheme should be able to encrypt the private data under a specific user set S and only the user in S can recover the data from ciphertext.
2) *Exclusive encryption:* For generic-purpose privacy, the proposed scheme should be able to encrypt the data under a specific user set R and all users not in the set R can recover the data from ciphertext.

As a cryptographic primitive, the broadcast encryption (BE) mechanism enables secure communication between a sender and multiple receivers in an open network. Its broadcast structure is well-suited for the blockchain multicast network, facilitating the secure sharing of sensitive data between a broadcaster and a specific group of receivers. However, while BE has been

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

extensively studied and has yielded remarkable results, there are few schemes that can simultaneously meet the two different privacy requirements of transactions. Therefore, our goal is to design a new dual-model BE scheme by combining BE and dual mode encryption. By SLC-language, the SLC combines paper contracts with the dual mode encryption algorithm to generate computer readable contract program code. During the contract conversion, the encryption algorithms are used to ensure the aforementioned privacy demands in SCs. In brief, some of the benefits of using PPSCs are listed as follows.

1) *Financial privacy:* PPSCs can help to protect users' financial privacy. This is important for people who want to keep their spending habits private, or who are concerned about government surveillance.
2) *Compliance with privacy regulations:* PPSCs can help businesses to comply with privacy regulations, such as GDPR. This is important for businesses that collect and process personal data.
3) *Reduced fraud:* PPSCs can help to reduce fraud because it is more difficult for fraudsters to track the flow of funds if the transactions are private.

B. Related Works

BE was first formally introduced by Fiat and Naor [5]. It allows a broadcaster to encrypt a message through a set of receivers. If a receiver belongs to this broadcast set, he can decrypt the ciphertext by his private key; otherwise, the ciphertext cannot be decrypted. BE schemes with different features have also been proposed successively. For example, Delerablée et al. [6] proposed a fully collusion resistant BE with constant-size ciphertexts and user's private keys. Boneh et al. [7] proposed a low overhead BE scheme from multilinear maps, in which the size of ciphertext, private keys, and public keys are all polylogarithmic in the total number of users. Recently, Chen et al. [8] proposed an adaptively secure certificate-based BE and applied their scheme to cloud storage service.

In 2007, Delerablée [9] and Sakai and Furukawa [10] proposed the concept of identity-based broadcast encryption (IBBE), in which the identity of the receiver is used as a public key to encrypt a message. Subsequently, Zhang et al. [11] constructed an adaptive secure IBBE with composite order group, in which the size of user's private key and ciphertext are constant. Kim et al. [12] proposed an IBBE scheme with outsourced partial decryption, which enables edge computing to use an IBBE for resource-constrained end devices. In the abovementioned IBBE schemes, receivers' identity must be bound to the ciphertext, which will expose the privacy of the receivers.

In order to protect the privacy of broadcast receivers, Barth et al. [13] introduced anonymity into BE. In their scheme, a broadcaster can encrypt messages to multiple receivers without revealing their identities, even to other receivers. After that, Zhang et al. [14] shown an anonymous IBBE scheme with adaptive security. In 2016, He et al. [15] proposed a completely anonymous IBBE scheme, which can against adaptive chosen ciphertext attacks under a standard model. After that, Chen et al. [16] put forward an efficient anonymous IBBE, which can be used into cloud storage access control. In their scheme, the size of public key, user's private key, and ciphertext are all constant.

In 2016, the first receiver revocable IBBE scheme was proposed by Susilo et al. [17]. It allows a third party to revoke any receivers without knowledge of the plaintext. At the same time, Lai et al. [18] proposed an anonymous IBBE with receivers revocation that allows a broadcaster to revoke receivers' identifies from the ciphertext without revealing any information about receivers. Recently, Zhu et al. [19] proposed a novel BE scheme that supported designation and revocation mechanisms, simultaneously. The broadcaster can choose different mechanisms to make the BE scheme efficient according to the size of authorization set.

As described previously, the BE has been extensively studied in terms of different security requirements, such as the anonymity and revocability of broadcast receivers. However, there is almost not a scheme that could meet selective encryption mode (SEM) and exclusive encryption mode (EEM) simultaneously. Although the scheme in [19] can achieve both of designation and revocation mechanisms, but it has not considered the privacy of the broadcast receivers. As far as we know, there is no BE scheme with dual encryption modes that can perfectly meet the privacy demands of transactions in SCs.

C. Contributions

Aiming at confidential transactions on PPSCs, we introduce encryption mechanism into SLCs to prevent sensitive transaction data from disclosing. In view of the current cryptographic schemes, which cannot satisfy the diversified privacy demands, we construct a new dual-mode identity-based broadcast encryption (DM-IBBE) scheme to meet different requirements of transaction privacy. Our contributions can be summarized as follows.

1) We propose an improved architecture of SLC with privacy protection, involving contract development, deployment, and execution. Based on the idea of COP, sensitive data of transaction are declared and protected in a form of contract terms in the conversion procedure from paper contract to SLC-language contract. In auction contract use case, we give some specific contract terms to protect transaction's privacy. Moreover, these terms allow a compiler to link predefined cryptographic algorithms into SC programs, and then to generate executable codes.
2) To meet diversified privacy requirements of different transaction data, we construct a new DM-IBBE scheme as predefined cryptographic algorithms in SC framework. The scheme can protect the transaction's privacy by introducing two different encryption modes: SEM and EEM. Furthermore, the proposed scheme is proved to be semantically secure under the decisional Diffie–Hellman (DDH) assumption. The size of user's private key is short, and user is anonymous in the SEM.

Organization: The rest of this article is organized as follows. In Section II, we introduce some preliminaries. The use case of SLC with privacy protection is presented in Section III. The definition and security model of our DM-IBBE scheme are given

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Contract Model		Grammar
Architecture		*Contracts* ::= *Title* { *Parties+ Assets+ Terms+ Additions+ Signs+* }
Title		*Title* ::= **contract** Cname (: **serial number** Chash)?
Party		*Parties* ::= **party group**? Pname {*field+* }
Asset Declaration		*Assets* ::= **asset** Aname{ **info**{ *field+* } **right**{ *field+* }}
Asset Expression		*AssetExpression* ::= **$** (amount)? (right **of**)? Aname
Asset Operation	Deposit	*Deposits* ::= **deposit** (**value** RelationOperator)? *AssetExpression*
	Withdraw	*Withdraws* ::= **withdraw** *AssetExpression*
	Transfer	*Transfers* ::= **transfer** *AssetExpression* **to** target
Time Expression	ActionEnforcedTimeQuery	*ActionEnforcedTimeQuery* ::= (**all**| **some**|**this**)? party **did** action
	TimePredicate	*TimePredicate* ::= (targetTime)? (**is**|**isn't**) (**before**|**after**) baseTime
	BoundedTimePredicate	*BoundedTimePredicate* ::= (**within**)? boundary (**before**|**after**) baseTime
Terms	General Term	*GeneralTerms* ::= **term** Tname: Pname (**must**|**can**|**cannot**) action(*field+*) (**when** preCondition)? (**while** transactions+)? (**where** postCondition)?
	Breach Term	*BreachTerms* ::= **breach term** Bname (**against** Tname+)? Pname (**must**|**can**) action(*field+*) (**when** preCondition)? (**while** transactions+)? (**where** postCondition)?
	Arbitration Term	*ArbitrationTerms* ::= **arbitration term** : (The statement of any controversy)? administered by **institution** : instName
Addition Information		*Additions* ::= *field* +
Signature		*Signs* ::= **Signature of party** Pname: {printedName: string, signature: string, Date: date}
field		*field* ::= attribute : (constant | type)

Fig. 2. Grammar of SLC language on SPESC.

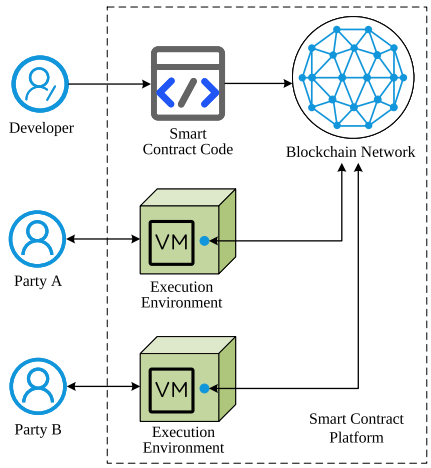

Fig. 3. Blockchain SC system.

in Section IV and the construction of our scheme is presented in Section V. In Section VI and VII, we give the security proof and performance analysis of the proposed scheme, respectively. Finally, Section VIII concludes this article.

II. System Model of PPSCs

A. Blockchain SC System for PPSCs

At present, most blockchain platforms do not provide adequate privacy for SCs, as the data and transactions are visible to anyone who can access the blockchain. To end it, we present a new blockchain model for PPSCs to protect the privacy of the data and transactions involved in their execution. This model is a simple business-level protection technology to support arbitrary privacy-preserving applications for lightweight users.

As shown in Fig. 3, the proposed blockchain SC system is an environment for developer and contracting parties (Party A and Party B) to deploy and execute contract codes on an SC platform. Among them, both contract execution environment and blockchain network constitute SC platform. Generally, the platform interacts with these parties through web interface, and it can restrict the rights and obligations among the parties through the executions of the terms in the contract.

The detail descriptions of the abovementioned entities are as follows.

1) *Developer* is the entity who can convert paper contracts into SC programs according to business rules, compile SC codes, and then deploy these codes to blockchain networks.
2) *Contracting parties* are two or more entities who involve in the conclusion of contracts. Based on contract platform, contracting parties can interactively conclude and execute contracts.
3) *Execution environment* consists of a blockchain node and an execution unit.
   - Blockchain node keeps connecting with blockchain network. This connection makes the node obtain SC codes and current program status from blockchain, and write the results of execution and new program status back into blockchain.
   - Execution unit is an entity used to execute SC codes. It usually adopts virtual environment, such as virtual machine and Docker. In addition, the execution unit contains some auxiliary mechanisms to provide parties' identity authentication, contract signature, incentive mechanism, consensus verification, and other operations.
4) *Blockchain network* provides a powerful underlying distributed communication and storage capability for SCs. It is used to record contract codes, intermediate status, and final results in the form of transactions.

In this system, the developers can introduce cryptographic primitives, such as zero-knowledge proofs, homomorphic encryption, or secret sharing, into PPSCs for enabling computation

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

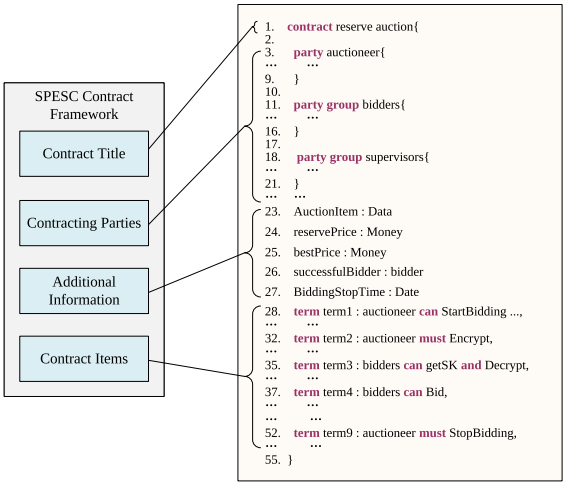

Fig. 4. Framework of SPESC contract.

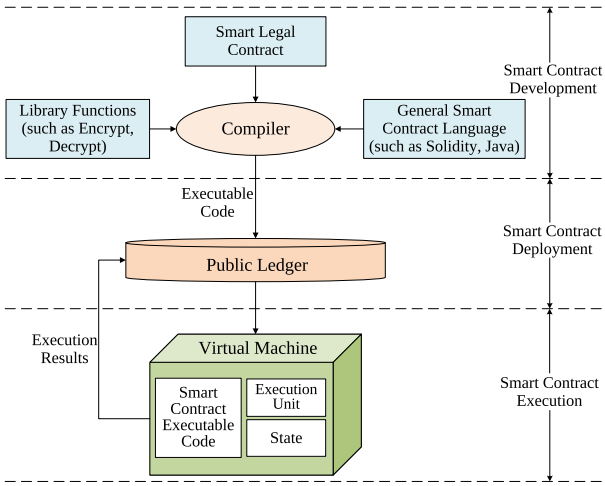

Fig. 5. Flowchart of SC development, deployment, and execution.

over encrypted data without revealing it. Therefore, the PPSCs offer a number of benefits for users and businesses because they can help to protect user privacy, comply with privacy regulations, reduce fraud, and enable new applications.

B. SLC Language for PPSCs

SLC language is a program language that is used to develop SC for meeting the legal requirements. By complying with law, it provides formal specification, including vocabularies and grammar rules, to ensure that SLCs possess legal features similar to paper contract.

As an instance, specification language of smart contract (SPESC) [20] has attracted extensive attention in academia. SPESC is based on the grammatical structure of a traditional contract and written in a form similar to natural language. As shown in Fig. 2, it not only clearly defines the obligations and rights of parties but includes the basic elements of a binding contract (including offers and acceptance, consideration, mutuality, and intentions). Hence, it can facilitate cooperation between legal experts and programmers for designing and developing SCs.

As shown in Fig. 4, an instance of SLC written in SPESC includes four parts: contract title, contracting parties, additional information, and contract terms.

- *Contract title* is generally a simple reflection about the contract's nature or central purpose and it is the starting sentence in SPESC contract.
- *Contracting parties* may be individual or group, and their attributes and actions are defined in this part.
- *Additional information* is public and available to contracting parties to assist them in their actions.
- *Contract terms* are the principal part of SPESC contract and they determine the rights and obligations of parties to this contract.

To develop PPSCs, encryption technology can be applied to the SPESC-type SC language. This allows PPSCs to operate on encrypted data and transactions without revealing them to anyone, except the parties involved in the contract. In this way, cryptographic functions can be expressed in the form of contract terms. Also, SPESC language is able to define the contract in a similar form to real-world contracts using a natural-language-like grammar. This helps to make PPSCs more understandable and developer friendly.

C. Development, Deployment and Execution of PPSCs

Usually, blockchain SC supports contract-oriented software development. As shown in Fig. 5, there are three entities/tools in the process of blockchain SC developing, deploying, and executing. These entities/tools are compiler, public ledger, and virtual machine, respectively. Their more details are described as follows.

- *Compiler* is a conversion tool that can convert SLC into computer executable program codes by general SC language (such as Java and Solidity). This compilation procedure of contract depends on the conversion rules introduced in [21] and [22]. The compiler can import library functions to simplify the generation of executable codes.
- *Public ledger* is an entity which is used to deploy executable codes of SC (contract codes). At the same time, it can store parties' information, object (asset), and program status. It also provides a decentralized, tamperproof, open, and transparent deployment environment for SC.
- *Virtual machine* is an entity that can be used to execute SC codes. It receives codes and program status from the public ledger and executes them in the execution unit when predetermined conditions of contract are met. Finally, virtual machine uploads execution results and new program states to the public ledger.

The SLC program based on encryption technology is the starting point for designing and developing PPSCs. By importing multifarious library functions, the difficulty of developing SC will be significantly reduced. In particular, cryptographic functions can be deployed and loaded in the form of libraries, and automatically executed in the blockchain SC platform. This

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

allows for efficient and secure execution of cryptographic operations, as the libraries are precompiled and optimized for the specific blockchain platform. In addition, mature commercial platforms, such as Ethereum and Fabric, have been available for deployment and execution of SC [23]. So, we can use the framework of blockchain SC to achieve reliability in processing of business transactions.

III. Use Case of SLC With Privacy Protection

In this section, we will use a reserve auction as an example to illustrate how to create a SLC using the SPESC language to meet the privacy requirements of transactions. A reserve auction is a type of auction in which the auctioneer sets a minimum price before the auction begins. The bidder who bids the closest to the reserve price without going over it will win the auction. The SPESC language can be used to create an SLC that enforces the rules of a reserve auction. The contract can be written to ensure that the reserve price is not disclosed to the bidders before the auction begins, and that the identity of the winning bidder is not disclosed to the public after the auction ends. The contract can also be written to allow supervisors (notaries) to verify the validity of the auction after it has concluded. The supervisors can compare the best price bid with the reserve price to ensure that the auction was conducted fairly.

It should be noted that in order to ensure the fairness of auction, reserve price must be kept confidential to bidders and supervisors before auctioneer opens the bid. To protect the privacy of auction item, the details of auction item are only open to qualified bidders in a commercial auction. Before bidding, the auctioneer shall check all of bidders' qualification and list the unqualified bidders in a blacklist. Thus, in a reserve auction contract, there are two kinds of privacy demands as follows.

1) *Specific-purpose privacy* that means the reserve price is the sensitive information, which can be only obtained by supervisors after the auctioneer opening the bid.
2) *Generic-purpose privacy* that emphasizes the auction item is the sensitive information, which can be only obtained by all bidders who are not in the blacklist.

As shown in Fig. 6, the process of reserve auction includes the following four steps.

1) *Step 1:* Auctioneer starts this auction and sets the end time and reserve price. Then, the auctioneer encrypts reserve price and details of auction item to ensure their confidentiality.
2) *Step 2:* In the bidding procedure, unblacklisted bidders can bid and submit deposits to the capital pool before the bidding deadline.
3) *Step 3:* After the bidding, the auctioneer should open bids to find the successful bidder and his bidding price, i.e., the best price. Then, the auctioneer publishes the best price to all bidders and supervisors.
4) *Step 4:* Supervisors decrypt the reserve price from ciphertext to judge whether it is valid. If so, the auctioneer collects the best price and all of unsuccessful bidders withdraw their deposits; otherwise, all bidders withdraw their deposits form the capital pool.

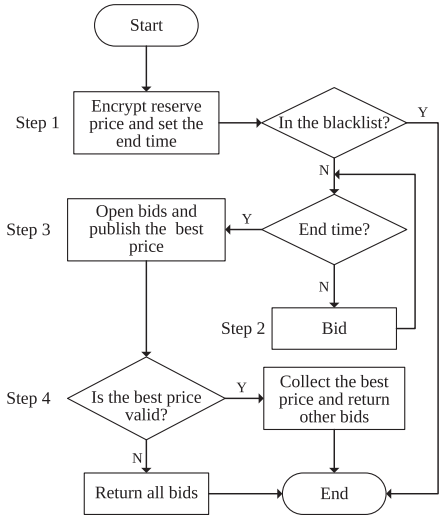

Fig. 6. Process of reserve auction in use case.

According to the abovementioned reserve auction, we use SPESC to write an SLC as Fig. 7. This SLC defines three kinds of contracting parties: auctioneer, bidders, and supervisors. Auctioneer is an individual party who manages the bidding and performs the following actions.

1) **StartBidding**: to start the bidding procedure.
2) **getPK**: to get his public key in this auction.
3) **Encrypt**: to encrypt the reserve price and details of auction item.
4) **OpenTheBid**: to open bids for finding the successful bidder and the best price in this auction.
5) **StopBidding**: to end this auction.

Bidders belong to a party group of users who register as bidders to participate in bidding. These bidders perform the following actions.

1) **Amount**: to record deposits and withdraw invalid bid.
2) **Decrypt**: to decrypt the ciphertext about details of auction item.
3) **Bid**: to bid a price at the auction.
4) **WithdrawOverbidMoney**: to withdraw the deposits.

Supervisors belong to a party group who can supervise the results of bid opening. They perform the following actions.

1) **getSK**: to get his private key.
2) **Decrypt**: to decrypt the ciphertext about reserve price.

To support abovementioned actions, five additional information are introduced into this auction as follows.

1) **AuctionItem** represents the details of auction item.
2) **reservePrice** represents the reserve price.
3) **successfulBidder** represents the successful bidder.
4) **bestPrice** represents the bid-winning price.
5) **BiddingStopTime** represents the auction stop time.

In addition, there are nine terms in this SPESC-based reserve auction contract and we stipulate the rights and obligations of contracting parties by "can" and "must" in terms, respectively. These terms correspond to the four steps of the reserve auction in Fig. 6. More details are described as follows:

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.


```
1.  contract reserve auction{
2.
3.      party auctioneer{
4.          getPK()
5.          Encrypt()
6.          OpenTheBid()
7.          StartBidding(BiddingTime: Date)
8.          StopBidding()
9.      }
10.
11.     party group bidders{
12.         amount : Money
13.         Bid()
14.         WithdrawOverbidMoney()
15.         Decrypt()
16.     }
17.
18.     party group supervisors{
19.         getSK()
20.         Decrypt()
21.     }
22.
23.     AuctionItem : Data
24.     reservePrice : Money
25.     bestPrice : Money
26.     successfulBidder : bidder
27.     BiddingStopTime : Date
28.     term term1 : auctioneer can StartBidding and getPK,
29.         when before auctioneer did StartBidding
30.         while Encrypt reservePrice
31.         where BiddingStopTime = BiddingTime + now.
32.     term term2 : auctioneer must Encrypt,
33.         when after auctioneer did StartBidding and before bidders did Bid,
34.         while Encrypt AuctionItem.
35.     term term3 : bidders can getSK and Decrypt,
36.         when after auctioneer did Encrypt and before bidders did Bid.
37.     term term4 : bidders can Bid,
38.         when after auctioneer did StartBidding and before BiddingStopTime.
39.     term term5 : auctioneer must OpenTheBid,
40.         when after BiddingStopTime and before auctioneer did StopBidding
41.         while publish reservePrice, successfulBidder and bestPrice
42.         where bestPrice = the bidding price of successfulBidder.
43.     term term6 : supervisors can getSK and Decrypt,
44.         when after auctioneer did OpenTheBid.
45.     term term7 : bidders can WithdrawOverbidMoney,
46.         when bestPrice isn't very close to decrypted reservePrice
47.         while withdraw deposit $this bidder : : amount.
48.     term term8 : bidders can WithdrawOverbidMoney,
49.         when bestPrice is very close to decrypted reservePrice and
50.                 this bidder isn't successfulBidder
51.         while withdraw deposit $this bidder : : amount.
52.     term term9 : auctioneer must StopBidding,
53.         when after BiddingStopTime and before auctioneer did StopBidding
54.         while withdraw $bestPrice.
55. }
```

Fig. 7. Use case of auction contract based on SPESC language.

1) **term1-2** correspond to **Step 1** in the process of reserve auction. It means that auctioneer can start the bidding procedure and obtain parties' public key. Then, he can encrypt the reserve price and details of auction item to meet different privacy demands;
2) **term3-4** correspond to **Step 2** in the auction process. It means that unblacklisted bidders can get their private keys and decrypt the auction item before bidding. Then, these bidders can bid on a price after auctioneer starts the bidding procedure and before the auction stop time;
3) **term5** corresponds to **Step 3** in the auction process. It describes that auctioneer is obliged to open the bid for finding the winning bidder, and then publish the reserve price and the best price to supervisors;
4) **term6-9** correspond to **Step 4** in the auction process. By these terms, supervisors can decrypt the ciphertext about reserve price and the auctioneer withdraws the best price and ends this auction.

As mentioned earlier, we have demonstrated the use of SLCs in protecting transaction privacy through a reserve auction contract. However, the SPESC language can also be used to write other real-world contracts, such as house leasing, purchase, and loan contracts, among others. By defining actions for contract parties (e.g., getPK, getSK, Encrypt, Decrypt) and appropriate contract terms, the private data in transactions can only be accessed by the authorized users, thus satisfying different demands for transaction privacy.

IV. Definition and Security Model

In this section, we will describe the definition of our DM-IBBE scheme and then we show the correspondence between the SPESC-based reserve auction contract and our DM-IBBE. Finally, we introduce security models about our scheme.

A. The Definition of DM-IBBE

Our DM-IBBE scheme is derived from the IBBE and can be regarded as an improved IBBE scheme with two kinds of authorization mode, especially, it can simultaneously support both SEM and EEM to achieve different authorization requirements.

Similar to the IBBE scheme, our DM-IBBE also includes four algorithms: *Setup*, *Extract*, *Encrypt*, and *Decrypt*. More details will be described as follows.

1) **Setup**$(1^\lambda) \rightarrow (\text{PK}, \text{MSK})$: This algorithm inputs the security parameter 1^λ, outputs the public key PK and the master secret key MSK of this system.
2) **Extract**$(\text{MSK}, \text{ID}_i) \rightarrow (sk_i)$: Inputs the master secret key MSK and identity ID_i, this algorithm outputs private key sk_i for identity ID_i.
3) **Encrypt**(PK, M, S/R, Mode) $\rightarrow$ CT: This algorithm inputs public key PK, message M, specified supervisor set S (for SEM), or blacklisted bidder set R (for EEM) and encryption mode Mode $\in \{\text{SEM}, \text{EEM}\}$, it outputs the ciphertext CT.
4) **Decrypt**(CT, sk_i, Mode) $\rightarrow M$: Inputs the ciphertext CT, Mode $\in \{\text{SEM}, \text{EEM}\}$ and private key sk_i.

- For SEM, any user can recover the massage M form CT if he belongs to S; otherwise, the *Decrypt* algorithm outputs an invalid plaintext.
- For EEM, any user can recover the message M form CT, if he is *not* in R; otherwise, the *Decrypt* algorithm outputs an invalid plaintext.

The correctness of our DM-IBBE scheme requires that any authorized user can successfully decrypt the ciphertext when his

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

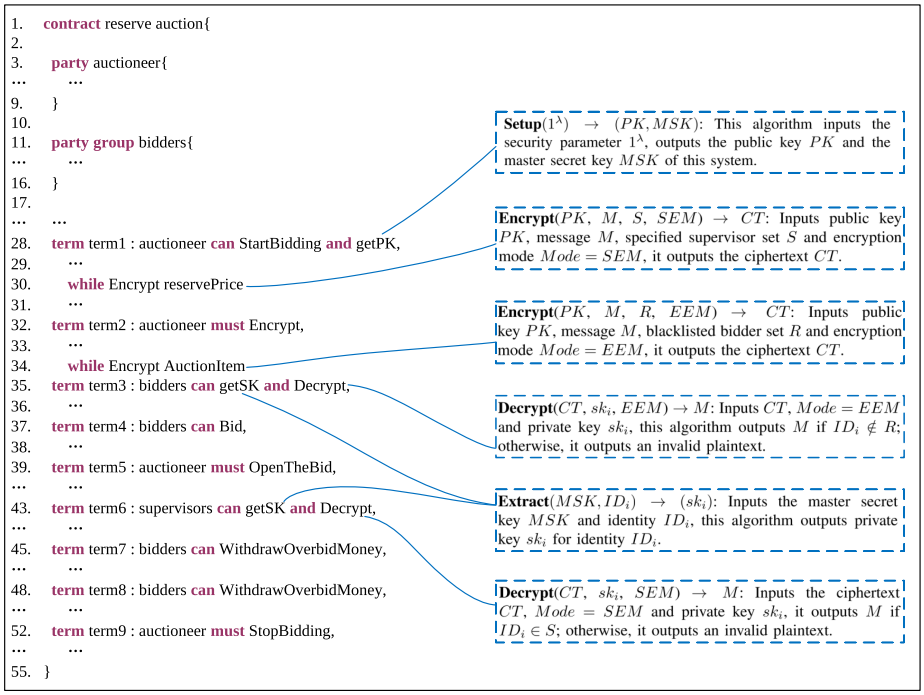

Fig. 8. Correspondence between SPESC-based reserve auction contract and algorithms of our DM-IBBE.

identity belongs to S ($\mathrm{ID}_i \in S$) for SEM or not to R ($\mathrm{ID}_i \notin R$) for EEM. In addition, the definition of our DM-IBBE requires that structures of ciphertext are exactly the same under the different encryption modes.

B. Correspondence Between SPESC Contract and DM-IBBE

Our algorithms in the proposed DM-IBBE are encapsulated in the form of library function, which are placed in a common place called SC platform. In the conversion process from SPESC contract to executable codes, as shown in Fig. 5, the compiler can directly import these predefined library functions to simplify the generation of contract codes. Next, as shown in Fig. 8, we take the SPESC-based reserve auction contract as an example to describe the correspondence between SPESC contract and our algorithms in DM-IBBE scheme.

1) The getPK action in **term1** corresponds to the *Setup* algorithm in our scheme, the auctioneer executes this action to get the public key including supervisors' and bidders' public keys.
2) The Encrypt action in **term1** corresponds our *Encrypt* algorithm under SEM. Auctioneer executes this action to encrypt the reserve price for protecting the specific-purpose privacy.
3) The Encrypt action in **term2** corresponds the *Encrypt* algorithm under EEM. Auctioneer executes this action to encrypt details of auction item for protecting the generic-purpose privacy.
4) The getSK action in **term3** corresponds to our *Extract* algorithm, by which bidders get their private keys.
5) The getSK action in **term6** corresponds to our *Extract* algorithm, by which supervisors get their private keys.
6) The Decrypt action in **term3** corresponds to our *Decrypt* algorithm under mode $=$ Exclusive, by which bidders can recover details of auction item from ciphertext.
7) The Decrypt action in **term6** corresponds to our *Decrypt* algorithm under mode $=$ Selective, by which supervisors can recover the reserve price from ciphertext.

C. Security Models

In our scheme, the SC is deployed on a public ledger, in which there is a large number of curious snoopers. Therefore, the security of our DM-IBBE scheme requires that any PPT adversary cannot learn information about the message from ciphertext without a valid private key, it is called indistinguishability under chosen-plaintext attacks (IND-CPA). Next, we define the IND-CPA security between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ by the following game.

1) *Initialization:* Adversary $\mathcal{A}$ announces the selected receiver set $S^* \subseteq U$ to be challenged, then the excluded receiver set R^* can be denoted as $R^* = U \setminus S^*$, where U is the set of all users.
2) *Setup:* Challenger $\mathcal{C}$ runs the original *Setup* algorithm in the proposed scheme to obtain the public key PK and master secret key MSK. Then, it sends PK to $\mathcal{A}$ and keeps MSK secretly.
3) *Phase 1:* In this step, $\mathcal{A}$ will make a private key query about user $\mathrm{ID}^* \notin S^*$, then challenger runs the *Extract* algorithm and outputs the private key sk^* for $\mathcal{A}$.

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

4) *Challenge:* Adversary submits two equal length messages M_0 and M_1 to challenger. Then, $\mathcal{C}$ chooses encryption mode in {SEM, EEM} and encrypts the message M_ρ, where ρ is randomly picked from $\{0,1\}$. Finally, $\mathcal{C}$ sends the challenge ciphertext CT to adversary.
5) *Phase 2:* Similar to Phase 1, $\mathcal{A}$ makes more private key queries, but it cannot query the private key on $\mathrm{ID}^* \in S^*$.
6) *Guess:* Adversary $\mathcal{A}$ outputs a guess ρ' of ρ and wins the game if $\rho' = \rho$.

Let $\mathrm{Adv}^{\mathcal{A}}_{\text{IND-CPA}}$ denote the advantage of $\mathcal{A}$ in IND-CPA game. We can describe this advantage as follows:

$$\mathrm{Adv}^{\mathcal{A}}_{\text{IND-CPA}} = \Pr[\rho' = \rho] - \frac{1}{2}. \tag{1}$$

Definition 1: A DM-IBBE scheme is IND-CPA secure, if for any PPT adversary $\mathcal{A}$, its advantage $\mathrm{Adv}^{\mathcal{A}}_{\text{IND-CPA}}$ is negligible.

In order to prevent curious snoopers from learning receivers' identity information, the security of our scheme further requires that even if the adversary has a valid key, he cannot learn any identity information about receivers in the SEM, it is called anonymity under chosen-plaintext attacks (ANO-CPA). It should be noted that, we only consider the receivers' anonymity in the SEM rather than EEM. This is because the ciphertext in EEM is generated with an excluded user set instead of a receiver set, and their identities are replaced by hash values.[2] Next, we define the ANO-CPA security between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ by the following game.

1) *Initialization:* Adversary $\mathcal{A}$ announces two selected receiver sets S_0 and S_1 to be challenged.
2) *Setup:* Challenger $\mathcal{C}$ runs the *Setup* algorithm of the proposed scheme to obtain the public key PK and master secret key MSK. Then, challenger sends PK to $\mathcal{A}$ and keeps MSK secretly.
3) *Phase 1:* In this step, $\mathcal{A}$ will make a private key query about user $\mathrm{ID}^* \notin (S_0 \bigcup S_1) \backslash (S_0 \bigcap S_1)$. Then, the challenger runs the *Extract* algorithm and generates the private key sk^* for $\mathcal{A}$.
4) *Challenge:* Adversary $\mathcal{A}$ first submits a message M to challenger. Then challenger $\mathcal{C}$ selects randomly $\eta \in \{0,1\}$. Finally, challenger $\mathcal{C}$ sends the challenge ciphertext CT under S_η to adversary.
5) *Phase 2:* Similar to Phase 1, adversary $\mathcal{A}$ makes more private key queries.
6) *Guess:* Adversary $\mathcal{A}$ outputs the guess η' of η and wins this game if $\eta' = \eta$.

Let $\mathrm{Adv}^{\mathcal{A}}_{\text{ANO-CPA}}$ denote the advantage of $\mathcal{A}$ in ANO-CPA game. We can describe this advantage as follows:

$$\mathrm{Adv}^{\mathcal{A}}_{\text{ANO-CPA}} = \Pr[\eta' = \eta] - \frac{1}{2}. \tag{2}$$

Definition 2: The SEM of our DM-IBBE scheme is ANO-CPA secure, if for any PPT adversary $\mathcal{A}$, its advantage $\mathrm{Adv}^{\mathcal{A}}_{\text{ANO-CPA}}$ is negligible.

[2]The unidirectionality of the cryptographic hash function ensures that the adversary cannot deduce the excluded user's identity from the hash value.

V. Proposed Scheme

In this section, we present the construction about our DM-IBBE. According to the previous definition in section IV-A, our scheme includes the following four algorithms. It should be noted that the number of users in our construction is unlimited except less than a threshold $(m+1)$, i.e., the maximum number of users in our scheme is m. For instance, this maximum set of users can be described in the following as $U = \{\mathrm{ID}_1, \mathrm{ID}_2, \ldots, \mathrm{ID}_m\}$.

1) **Setup**$(1^\lambda) \rightarrow$ (**PK**, **MSK**). For a security parameter 1^λ, this algorithm outputs a group $\mathbb{G}$ with prime order p and a generator $g \in \mathbb{G}$. Let cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, then this algorithm randomly chooses $\{a_i \in \mathbb{Z}_p\}_{i=0}^m$ and sets the polynomial $f(x) = \sum_{i=0}^m a_i x^i \mod p$. Finally, it publishes the the public key
$$\mathrm{PK} = \left\langle g, H, \{g^{a_i}\}_{i\in[0,m]} \right\rangle$$
and keeps the master secret key $\mathrm{MSK} = \left\langle \{a_i\}_{i\in[0,m]} \right\rangle$ secretly.
2) **Extract**(**MSK, ID**$_i$) $\rightarrow sk_i$. For any user ID_i, this algorithm first computes $x_i = H(\mathrm{ID}_i)$ and then generates ID_i's private key $sk_i = f(x_i)$ by using MSK.
3) **Encrypt(PK**, M, S/R, **Mode)** $\rightarrow$ **CT**. An encryptor executes this algorithm to encrypt a given sensitive data M under different mode as follows.

Selective encryption mode: In this mode, the encryptor specifies a receiver set $S = \{\mathrm{ID}_{s1}, \mathrm{ID}_{s2}, \cdots, \mathrm{ID}_{sl}\}$ and then encrypts M according to the following steps, where $|S| = l \leq m$.

1) This algorithm randomly chooses $s \in \mathbb{Z}_p^*$.
2) For all $\mathrm{ID}_i \in S$, it computes $x_i = H(\mathrm{ID}_i)$ and $T_i = \prod_{j=0}^m (g^{a_j})^{x_i^j}$, where g^{a_j} can be obtained from the public key.
3) Let $h_S(x)$ be a polynomial of degree l and satisfy the following equations:
$$g^{h_S(x)} = \begin{cases} T_i, & x_i = H(\mathrm{ID}_i) \text{ and } \mathrm{ID}_i \in S \\ g^s, & x_0 = 0. \end{cases} \tag{3}$$
With the help of the abovementioned $(l+1)$ pair of values, (x_0, g^s) and $\{(x_i, T_i)\}_{ID_i \in S}$, this algorithm can define $g^{h_S(x)}$ according to the Lagrange interpolation formula as
$$g^{h_S(x)} = g^{s \cdot L_0(x)} \cdot \prod\nolimits_{\mathrm{ID}_i \in S} T_i^{L_i(x)} \tag{4}$$
where $L_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \mod p$.
4) In this step, the algorithm randomly picks α and $\{\tau_i\}_{i\in[1,l]}$ in $\mathbb{Z}_p^*$, where for all $i \in [1,l]$, we have $\tau_i \neq H(\mathrm{ID}_j)$ and all $j \in [1,m]$. Then, it computes $C_1 = g^\alpha$
$$\{C_{i,1} = \tau_i, C_{i,2} = g^{h_S(\tau_i)\cdot\alpha}\}_{i\in[1,l]}$$
according to the Formula (4).
5) Finally, the algorithm computes $C_0 = g^{\alpha \cdot s} \cdot M$. The ciphertext under SEM can be described as
$$\mathrm{CT} = \left\langle C_0, C_1, \{C_{i,1}, C_{i,2}\}_{i\in[1,l]} \right\rangle.$$

Exclusive encryption mode: Let $R = \{\mathrm{ID}_{r1}, \mathrm{ID}_{r2}, \ldots, \mathrm{ID}_{rk}\}$ be the excluded user set. This algorithm encrypts the details of

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

auction item M according to the following steps, where $|R| = k \leq m$.

1) This algorithm randomly chooses $s \in \mathbb{Z}_p^*$.
2) For all $\mathrm{ID}_i \in U$, it computes $x_i = H(\mathrm{ID}_i)$ and let $T_i = \prod_{j=0}^{m} (g^{a_j})^{x_i^j}$.
3) Let $h_R(x)$ be a polynomial of degree m and satisfy the following equations:

$$g^{h_R(x)} = \begin{cases} T_i, & x_i = H(\mathrm{ID}_i) \text{ and } \mathrm{ID}_i \in U \\ g^s, & x_0 = 0. \end{cases} \tag{5}$$

With the help of the abovementioned $(m+1)$ pair of values, (x_0, g^s) and $\{(x_i, T_i)\}_{\mathrm{ID}_i \in U}$, this algorithm can define $g^{h_R(x)}$ by using the Lagrange interpolation formula as

$$g^{h_R(x)} = g^{s \cdot L_0(x)} \cdot \prod\nolimits_{\mathrm{ID}_i \in U} T_i^{L_i(x)}. \tag{6}$$

4) For $j \in [1,k]$, let $x_{rj} = H(\mathrm{ID}_{rj})$; for $j \in [k+1, m]$, the algorithm randomly picks elements $\delta_j \in \mathbb{Z}_p^*$, s.t.

$$x_{rj} = \begin{cases} H(\mathrm{ID}_{rj}), & j \in [1,k] \\ \delta_j, & j \in [k+1, m]. \end{cases} \tag{7}$$

For the sake of description, we let $\{\delta_j = H(\mathrm{ID}_{rj})\}_{j \in [1,k]}$. Then, the algorithm randomly picks $\alpha \in \mathbb{Z}_p^*$ and computes $C_1 = g^\alpha$. And it computes ciphertext components

$$\{C_{j,1} = \delta_j, C_{j,2} = g^{h_R(\delta_j) \cdot \alpha}\}_{j \in [1,m]}$$

according to the Formula (6).

5) Finally, the algorithm computes $C_0 = g^{\alpha s} \cdot M$. The ciphertext under EEM can be described as

$$\mathrm{CT} = \left\langle C_0, C_1, \{C_{j,1}, C_{j,2}\}_{j \in [1,m]} \right\rangle.$$

Decrypt(CT, sk_{i^*}, **Mode)** $\rightarrow M$. For the ciphertext generated in SEM, a user ID_{i^*} can recover M form CT, if $\mathrm{ID}_{i^*} \in S$; otherwise, this algorithm outputs an invalid plaintext. On the other hand, for ciphertext generated in EEM, a user ID_{i^*} can recover M form CT, if $\mathrm{ID}_{i^*} \notin R$; otherwise, this algorithm outputs an invalid plaintext. The detailed steps of decryption are as follows.

1) The decryptor ID_{i^*} with private key sk_{i^*} computes $x_{i^*} = H(\mathrm{ID}_{i^*})$ and $(C_1)^{sk_{i^*}} = g^{\alpha \cdot f(x_{i^*})}$.
2) For the sake of description, we let $x_{i^*} = C_{0,1}$. Then, ID_{i^*} recovers $g^{\alpha \cdot s}$ by using Lagrange interpolation formula as follows:

$$g^{\alpha \cdot s} = \begin{cases} {C_1}^{sk_{i^*} \cdot L_0} \cdot \prod_{i \in [1,l]} {C_{i,2}}^{L_i}, (\mathrm{ID}_{i^*} \in S) \wedge \text{SEM} \\ {C_1}^{sk_{i^*} \cdot L_0} \cdot \prod_{i \in [1,m]} {C_{i,2}}^{L_i}, (\mathrm{ID}_{i^*} \notin R) \wedge \text{EEM} \end{cases}$$

where $L_i = \prod_{j \neq i} \frac{-C_{j,1}}{C_{j,1} - C_{i,1}}$ and $C_{0,1} = x_{i^*}$.

3) Finally, decryptor ID_{i^*} recovers the sensitive information $M = C_0 / g^{\alpha \cdot s}$.

Validity: Compared with traditional IT platforms, the cryptosystem in blockchain is subjected to diverse security threats due to the openness and decentralization of blockchain. These features allow anyone to write data into the blockchain by contract codes. In particular, attackers can write malicious data into the blockchain to replace or tamper with original decryption results, so as to threaten the validity of the cryptosystem. For example, at the bid opening stage of the auction contract, the auctioneer should decrypt and publish the reserve price to find the successful bidder. At this point, the attacker can submit a wrong decryption result before the auctioneer publishes the reserve price. In order to address this threat, the authentication mechanism is introduced into the SC. Taking the auction contract as an example, the validity of the decryption result is guaranteed by the consensus protocol. This kind of consensus mechanism requires multiple authorized decryptors to endorse (using multisignature) the uploaded data to approve the validity of the decryption result, which makes sure that it is correct and written to the blockchain.

Correctness: For Mode = SEM, any user ID_{i^*} in the specified receiver set, i.e., $\mathrm{ID}_{i^*} \in S$, can use his private key sk_{i^*} to recover the message M from ciphertext as follows:

$$\begin{aligned} M &= \frac{C_0}{{C_1}^{sk_{i^*} \cdot L_0} \cdot \prod_{i \in [1,l]} {C_{i,2}}^{L_i}} \\ &= \frac{M \cdot g^{\alpha \cdot s}}{g^{\alpha \cdot f(x^*) \cdot L_0} \cdot \prod_{i \in [1,l]} g^{h_S(\tau_i) \cdot \alpha \cdot L_i}} \\ &= \frac{M \cdot g^{\alpha \cdot s}}{g^{\alpha \cdot \sum_{i=0}^{l} h_S(\tau_i) \cdot L_i}} = \frac{M \cdot g^{\alpha \cdot s}}{g^{\alpha \cdot s}}. \end{aligned} \tag{8}$$

For Mode = EEM, any user ID_{i^*} not in the excluded user set, i.e., $\mathrm{ID}_{i^*} \notin R$, can use his private key sk_{i^*} to recover the message M from ciphertext as follows:

$$\begin{aligned} M &= \frac{C_0}{{C_1}^{sk_{i^*} \cdot L_0} \cdot \prod_{i \in [1,m]} {C_{i,2}}^{L_i}} \\ &= \frac{M \cdot g^{\alpha \cdot s}}{g^{\alpha \cdot f(x^*) \cdot L_0} \cdot \prod_{i \in [1,m]} g^{h_R(\delta_i) \cdot \alpha \cdot L_i}} \\ &= \frac{M \cdot g^{\alpha \cdot s}}{g^{\alpha \cdot \sum_{i=0}^{m} h_R(\delta_i) \cdot L_i}} = \frac{M \cdot g^{\alpha \cdot s}}{g^{\alpha \cdot s}}. \end{aligned} \tag{9}$$

VI. Security Proof

In this section, we show that our DM-IBBE scheme is semantically secure under chosen-plaintext attacks. In addition, we prove the anonymity of the SEM in our scheme following the Definition 2.

Theorem 1: If the DDH assumption holds, then our DM-IBBE scheme, including both SEM and EEM, is semantically secure against chosen plaintext attack.

Proof: If there is a PPT adversary $\mathcal{A}$ that can win the IND-CPA game with a nonneglectable advantage ε, that is, $\mathrm{Adv}^{\mathcal{A}}_{\text{IND-CPA}} \geq \varepsilon$, then we can construct a challenger $\mathcal{C}$ to solve the DDH problem: given elements (g, g^a, g^b) and a random element $Z \in \mathbb{G}$ to output 1 if $Z = g^{ab}$; otherwise output 0.

1) *Initialization:* Adversary $\mathcal{A}$ announces a selected receiver set $S^* = \{\mathrm{ID}^*_{s1}, \mathrm{ID}^*_{s2}, \ldots, \mathrm{ID}^*_{sl}\} \subseteq U$ to be challenged, and the excluded receiver set R^* can be denoted as $R^* = \{\mathrm{ID}^*_{r1}, \mathrm{ID}^*_{r2}, \ldots, \mathrm{ID}^*_{rk}\} = U \setminus S^*$.
2) *Setup:* The challenger randomly picks $\{a_i \in \mathbb{Z}_p\}_{i=0}^m$ and sets the polynomial $f(x) = \sum_{i=0}^{m} a_i x^i \mod p$. Let H be a hash function, $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, then it computes

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

$H(\mathrm{ID}_i) = x_i$ for all $\mathrm{ID}_i \in U$, where $\mathrm{ID}_i \in \{0,1\}^*$ denotes user's identity. Finally, challenger sends the public key PK to $\mathcal{A}$

$$\mathrm{PK} = \left\langle g, H, \{g^{a_i}\}_{i\in[0,m]} \right\rangle.$$

3) *Phase 1:* In this step, adversary $\mathcal{A}$ will query the private key about ID^*. If $\mathrm{ID}^* \notin S^*$, then $\mathcal{C}$ computes $H(\mathrm{ID}^*) = x^*$ and generates the private key $sk^* = f(x^*)$ for adversary $\mathcal{A}$; otherwise, $\mathcal{C}$ returns $\perp$.
4) *Challenge:* Adversary $\mathcal{A}$ first submits two equal length messages M_0 and M_1 to $\mathcal{C}$. Then challenger $\mathcal{C}$ chooses an encryption mode in {SEM, EEM} and encrypts M_ρ, where ρ is a randomly picked in $\{0,1\}$.
For Mode = SEM, the challenger computes ciphertext components $C_0 = M_\rho \cdot Z$ and $C_1 = g^a$, where $\alpha = a$. Let $h_{S^*}(x)$ be a polynomial of degree l, so that the following equations hold:

$$g^{h_{S^*}(x)} = \begin{cases} T_i, & \mathrm{ID}_i \in S^* \\ g^b, & x = 0. \end{cases} \tag{10}$$

In (10), it implies $b = s$. By recalling Lagrange interpolation formula, then $\mathcal{C}$ can recover $g^{h_{S^*}(x)}$ as

$$g^{h_{S^*}(x)} = g^{b\cdot L_0(x)} \cdot \prod\nolimits_{\mathrm{ID}_i\in S^*} T_i^{L_i(x)}. \tag{11}$$

Next, the challenger randomly picks $\{\tau_i\}_{i\in[1,l]}$ in $\mathbb{Z}_p^*$, where $\tau_i \neq H(\mathrm{ID}_j)$ for all $i \in [1,l]$ and $j \in [1,m]$. Finally, it sets $\{C_{i,1} = \tau_i, C_{i,2} = g^{a\cdot h_{S^*}(\tau_i)}\}_{i\in[1,l]}$.
For Mode = EEM, $\mathcal{C}$ computes ciphertext components $C_0 = M_\rho \cdot Z$ and $C_1 = g^a$, where $\alpha = a$. Let $h_{R^*}(x)$ be a polynomial of degree m, so that the following equations hold:

$$g^{h_{R^*}(x)} = \begin{cases} T_i, & \mathrm{ID}_i \in U \\ g^b, & x = 0. \end{cases} \tag{12}$$

In (12), it also implies that $b = s$. Next, challenger $\mathcal{C}$ recalls Lagrange interpolation formula and defines that

$$g^{h_{R^*}(x)} = g^{b\cdot L_0(x)} \cdot \prod\nolimits_{i\in[1,m]} T_i^{L_i(x)}. \tag{13}$$

For $\mathrm{ID}_{ri}^* \in R^*$, the challenger computes $H(\mathrm{ID}_{ri}^*) \to \delta_i$, where $i \in [1,k]$. Then, $\mathcal{C}$ randomly chooses $\{\delta_i \in \mathbb{Z}_p^*\}_{i\in[k+1,m]}$, such that $\delta_i \neq H(\mathrm{ID}_i)$ for any $\mathrm{ID}_i \in U$. Finally, the challenger computes ciphertext components $\{C_{i,1} = \delta_i, C_{i,2} = g^{a\cdot h_{R^*}(\tau_i)}\}_{i\in[1,m]}$.
5) *Phase 2:* Same as the Phase 1, $\mathcal{A}$ makes more private key queries, but it can not query the private key of $\mathrm{ID}^* \in S^*$.
6) *Guess:* Adversary $\mathcal{A}$ outputs a guess $\rho' \in \{0,1\}$ on ρ. If $\rho' = \rho$, challenger $\mathcal{C}$ outputs 1, in which case it indicates that $Z = g^{ab}$; otherwise outputs 0 indicating that Z is a random element in $\mathbb{G}$.

If $Z = g^{ab}$, then the adversary $\mathcal{A}$ gets a valid ciphertext. In this case, the advantage of $\mathcal{A}$ satisfies

$$\mathrm{Adv}_{\text{IND-CPA}}^{\mathcal{A}} = \Pr[\rho' = \rho] - \frac{1}{2} \geq \varepsilon.$$

If Z is a random element in $\mathbb{G}$, denoted as $Z = g^z$, then $\Pr[\rho' = \rho | Z = g^z] = \Pr[\rho' \neq \rho | Z = g^z] = \frac{1}{2}$. Therefore, we have

$$\Pr[\mathcal{C}(g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{C}(g, g^a, g^b, g^z) = 1]$$
$$= \Pr[\rho'{=}\rho | Z = g^{ab}] - \Pr[\rho'{=}\rho | Z = g^z] \geq \frac{1}{2} + \varepsilon - \frac{1}{2} {=} \varepsilon.$$

In fact, the DDH problem is hard on the group $\mathbb{G}$ for any PPT adversary, thus our assumption is not tenable. Therefore, the Theorem 1 is proved. ■

Theorem 2: If the DDH assumption holds, our scheme is anonymous when the encryption mode is SEM.

Proof: If there is a PPT adversary $\mathcal{A}$ that can win the ANO-CPA game with a nonneglectable advantage ε, that is, $\mathrm{Adv}_{\text{ANO-CPA}}^{\mathcal{A}} \geq \varepsilon$, then we can construct a challenger $\mathcal{C}$ to solve the DDH problem: given elements $(g, g^a, g^b) \in \mathbb{G}^3$ and a random element $Z \in \mathbb{G}$ to output 1 if $Z = g^{ab}$; otherwise output 0.

1) *Initialization:* $\mathcal{A}$ first announces two selected receiver sets $S_0 = \{\mathrm{ID}_{01}, \mathrm{ID}_{02}, \ldots, \mathrm{ID}_{0l}\} \subset U$ and $S_1 = \{\mathrm{ID}_{11}, \mathrm{ID}_{12}, \ldots, \mathrm{ID}_{1l}\} \subset U$ to be challenged.
2) *Setup:* The challenger randomly picks $\{a_i \in \mathbb{Z}_p\}_{i=0}^m$ and sets the polynomial $f(x) = \sum_{i=0}^m a_i x^i \mod p$. Let H be a hash function, $H: \{0,1\}^* \to \mathbb{Z}_p^*$, then it computes $H(\mathrm{ID}_i) = x_i$ for all $\mathrm{ID}_i \in U$, where $\mathrm{ID}_i \in \{0,1\}^*$ denotes user's identity. Finally, the challenger sends the public key PK to $\mathcal{A}$

$$\mathrm{PK} = \left\langle g, H, \{g^{a_i}\}_{i\in[0,m]} \right\rangle.$$

3) *Phase 1:* In this step, adversary $\mathcal{A}$ makes a private key query about ID^*. If $\mathrm{ID}^* \notin (S_0 \bigcup S_1) \setminus (S_0 \bigcap S_1)$, then it returns a random element; otherwise, it computes $H(\mathrm{ID}^*) = x^*$ and generates $sk^* = f(x^*)$ for $\mathcal{A}$.
4) *Challenge:* $\mathcal{A}$ submits a message M to $\mathcal{C}$, then the challenger randomly picks $\eta \in \{0,1\}$ and encrypts M under SEM with S_η. The challenger computes $C_0 = M \cdot Z$ and $C_1 = g^a$, where $\alpha = a$. Let $h_{S_\eta}(x)$ be a polynomial of degree l, such that the following equations hold:

$$g^{h_{S_\eta}(x)} = \begin{cases} T_i, & \mathrm{ID}_i \in S_\eta \\ g^b, & x = 0. \end{cases} \tag{14}$$

In the abovementioned equation, it implies that $b = s$. By recalling Lagrange interpolation formula, challenger $\mathcal{C}$ defines $g^{h_{S^*}(x)}$ as follows:

$$g^{h_{S_\eta}(x)} = g^{b\cdot L_0(x)} \cdot \prod\nolimits_{\mathrm{ID}_i\in S_\eta} T_i^{L_i(x)}. \tag{15}$$

Next, $\mathcal{C}$ randomly picks $\{\tau_i\}_{i\in[1,l]}$ in $\mathbb{Z}_p^*$, where for all $i \in [1,l]$, such that $\tau_i \neq H(\mathrm{ID}_j)$ and all $j \in [1,m]$. Finally, it sets $\{C_{i,1} = \tau_i, C_{i,2} = g^{a\cdot h_{S_\eta}(\tau_i)}\}_{i\in[1,l]}$.
5) *Phase 2:* Same as the Phase 1, adversary $\mathcal{A}$ makes more private key queries for the challenger.
6) *Guess:* Adversary $\mathcal{A}$ outputs a guess $\eta' \in \{0,1\}$ on η. If $\eta' = \eta$, challenger $\mathcal{C}$ outputs 1, in which case it indicates that $Z = g^{ab}$; otherwise outputs 0 indicating that Z is a random element in $\mathbb{G}$.

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
COMPARISON BETWEEN EXISTING BE SCHEMES AND OURS

Scheme	Dual mode	Anon-ymity	Security model	Hardness assumption	Without pairing
[8]	×	×	Standard	q-ABDHE, 1-BDHI	×
[18]	×	√	Oracle	BDH	×
[19]	√	×	Standard	BDHE	×
[24]	×	×	Standard	ADDH, DDH	×
Ours	√	√ *	Standard	DDH	√

* Anonymity holds in SEM

If $Z = g^{ab}$, then the adversary $\mathcal{A}$ gets a valid ciphertext. In this case, the advantage of $\mathcal{A}$ satisfies

$$\mathrm{Adv}^{\mathcal{A}}_{\text{ANO-CPA}} = \Pr[\eta' = \eta] - \frac{1}{2} \geq \varepsilon.$$

If Z is a random element in $\mathbb{G}$, denoted as $Z = g^z$, then $\Pr[\eta' = \eta] = \Pr[\eta' \neq \eta] = \frac{1}{2}$. Therefore, we have

$$\Pr[\mathcal{C}(g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{C}(g, g^a, g^b, g^z) = 1]$$
$$= \Pr[\eta'{=}\eta | Z = g^{ab}] - \Pr[\eta'{=}\eta | Z{=}g^z] \geq \frac{1}{2} + \varepsilon - \frac{1}{2} = \varepsilon.$$

In fact, the DDH problem is hard on the group $\mathbb{G}$ for any PPT adversary, thus our assumption is not tenable. Therefore, the Theorem 2 is proved. ■

VII. PERFORMANCE ANALYSIS

As shown in Table I, we make a comparison between four existing BE schemes and ours in aspects of dual encryption modes, anonymity, security model, hardness assumption, and pairing friendly. Especially, the inclusion of dual encryption modes indicates that the schemes satisfy both SEM and EEM. The hardness assumption refers to the reduced mathematical difficulty problem in the security proof, and the pairing friendliness indicates whether the schemes involve bilinear pairing operations during the running process.

It is easy to see that the scheme in [19] and ours can provide two kinds of different encryption mode in a broadcast system to meet various privacy requests. In addition, in order to prevent the receiver's identity from revealing in ciphertext, the receiver's anonymity is considered in [18] and ours. Furthermore, other schemes except for ours need bilinear pairing operation, which is considered as a high expensive operation. Therefore, the programming implementation of our scheme is less difficult than others. In terms of security, [8] is reduced to the truncated decisional q-augmented bilinear Diffie–Hellman exponent (q-ABDHE) and decisional 1-bilinear Diffie–Hellman inversion (1-BDHI) assumptions. The scheme in [18] is proven secure under the bilinear Diffie–Hellman (BDH) assumption in a random oracle model, which is less conceiving than the standard model. The security of [19] is reduced to the decisional bilinear Diffie–Hellman exponent assumption (BDHE) assumption. The security of [24] depends on both of the augmented decisional Diffie–Hellman (ADDH) and DDH assumptions. Our scheme is proved secure in standard model under the DDH assumption.

Among these assumptions, their relations can be summarized with appropriate group settings as DDH $\geq$ BDH = 1-BDHI [25], [26], DDH $\geq$ q-ABDHE [27], and DDH $\geq$ ADDH [28]. In the abovementioned relations, A $\geq$ B means the assumption A is weaker than B. In other words, the assumption B can be solved in polynomial time with polynomially many queries to the oracle in order to solve the assumption A. Obviously, DDH is the weakest assumption in the Table I. Indeed, a weaker assumption is considered preferable to a stronger one in the construction of cryptosystems, this is because the time cost of breaking a weaker assumption is much greater than that for a stronger [29]. It is well-known that the security level of a cryptosystems depends on several factors, such as security parameters, hardness assumptions, and execution environment. Therefore, our scheme has an advantage of strong security compared with other schemes considering that DM-IBBE is constructed under the weakest assumption DDH.

In Table II, we turn attention to the theoretical comparison of computation and storage overheads between these BE schemes and ours. Let m denote the number of all users in this system, l and k denote the number of specified and excluded receivers, respectively. Let $E_{\mathbb{G}}$, $M_{\mathbb{G}}$, and $D_{\mathbb{G}}$ denote exponentiation, multiplication, and division operations in $\mathbb{G}$. $E_{\mathbb{G}_T}$, $M_{\mathbb{G}_T}$, and $D_{\mathbb{G}_T}$ denote exponentiation, multiplication, and division operations in $\mathbb{G}_T$. B is the bilinear pairing operation. In addition, we define $l_{\mathbb{G}}$, $l_{\mathbb{G}_T}$, and $l_{\mathbb{Z}}$ as the lengths of elements in $\mathbb{G}$, $\mathbb{G}_T$, and $\mathbb{Z}$, respectively. We neglect operations in $\mathbb{Z}$ and hash operation, because they are much more efficient than operations in $\mathbb{G}$ and $\mathbb{G}_T$.

In computation overheads, as shown in the Table II, costs of encryption in [8], [18], [19], [24] are linear with the number of specified receivers, however, that of ours in SEM is related to the number of both specified receivers and all users, that is, our encryption algorithm includes $\mathcal{O}(lm)$ exponentiation operations in $\mathbb{G}$. On the other hand, encryption costs of [19] is linear with the number of excluded receivers in EEM, that of ours needs $\mathcal{O}(m^2)$ exponentiation operations in $\mathbb{G}$. In decryption, the time costs of all schemes in SEM are linear with the number of specified receivers. While the decryption overheads of [19] are linear with the number of excluded receivers k, that of ours are linear with the total number of users m. As a result, [19] incurs lower computational costs for exponentiation operations in $\mathbb{G}$ due to the fact that $m > k$. However, our scheme does not involve pairing operation in neither encryption nor decryption.

In terms of storage overheads, as shown in the Table II, the size of the public key remains constant in [18], while in all other schemes, including ours, it is dependent on the total number of users m. The ciphertext size remains constant in both [19] and [24]. However, the size of the ciphertext in [8], [18], and our SEM scheme is correlated with the number of specified receivers, while in our EEM scheme, it is related to the total number of users. It is worth noting that the ciphertext sizes in our EEM scheme are larger than those in SEM because the EEM ciphertext requires additional information for excluded receivers.

In order to evaluate the performance of BE schemes in [8], [18], [19], [24] and ours, we implement them based on the Java

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE II
COMPARISON OF THE COMPUTATION AND STORAGE OVERHEADS

Scheme	Computation costs				Storage Costs		
	Encrypt		Decrypt		Public Key	Ciphertext	
	SEM	EEM	SEM	EEM		SEM	EEM
[8]	$(2l+2)E_{\mathbb{G}}+2lB$ $+(3l+1)E_{\mathbb{G}_T}$	—	$(l+2)E_{\mathbb{G}}$ $+3E_{\mathbb{G}_T}+B$	—	$(3m+4)l_{\mathbb{G}}$ $+l_{\mathbb{G}_T}$	$(l+3)l_{\mathbb{G}}$ $+2l_{\mathbb{Z}}$	—
[18]	$(2l+3)E_{\mathbb{G}}+2lE_{\mathbb{G}_T}$ $+(l-1)M_{\mathbb{G}}+2lB$	—	$(2l-1)E_{\mathbb{G}}+2D_{\mathbb{G}}$ $+(2l-2)M_{\mathbb{G}}+2B$	—	$2l_{\mathbb{G}}$	$(2l+3)l_{\mathbb{G}}$	—
[19]	$2E_{\mathbb{G}}+lM_{\mathbb{G}}$ $+1E_{\mathbb{G}_T}+1B$	$2E_{\mathbb{G}}+1D_{\mathbb{G}}+1B+$ $1E_{\mathbb{G}_T}+(k-1)M_{\mathbb{G}}$	$(l-1)M_{\mathbb{G}}+2B$ $+1D_{\mathbb{G}_T}$	$kM_{\mathbb{G}}+1D_{\mathbb{G}}$ $+2B+1D_{\mathbb{G}_T}$	$(3m+1)l_{\mathbb{G}}$	$2l_{\mathbb{G}}$	$2l_{\mathbb{G}}$
[24]	$(l+5)E_{\mathbb{G}}+1E_{\mathbb{G}_T}$ $+(l+2)M_{\mathbb{G}}$	—	$2lE_{\mathbb{G}}+(2l-2)M_{\mathbb{G}}+$ $3B+1M_{\mathbb{G}_T}+2D_{\mathbb{G}_T}$	—	$(3m+14)l_{\mathbb{G}}$ $+l_{\mathbb{G}_T}$	$4l_{\mathbb{G}}$	—
Ours	$(lm+2l+2)E_{\mathbb{G}}$ $+(lm+1)M_{\mathbb{G}}$	$(m^2+2m+2)E_{\mathbb{G}}$ $+(m^2+1)M_{\mathbb{G}}$	$(l+2)E_{\mathbb{G}}$ $+lM_{\mathbb{G}}+1D_{\mathbb{G}}$	$(m+2)E_{\mathbb{G}}+$ $mM_{\mathbb{G}}+1D_{\mathbb{G}}$	$(m+2)l_{\mathbb{G}}$	$(l+2)l_{\mathbb{G}}$ $+1l_{\mathbb{Z}}$	$(m+2)l_{\mathbb{G}}$ $+ml_{\mathbb{Z}}$

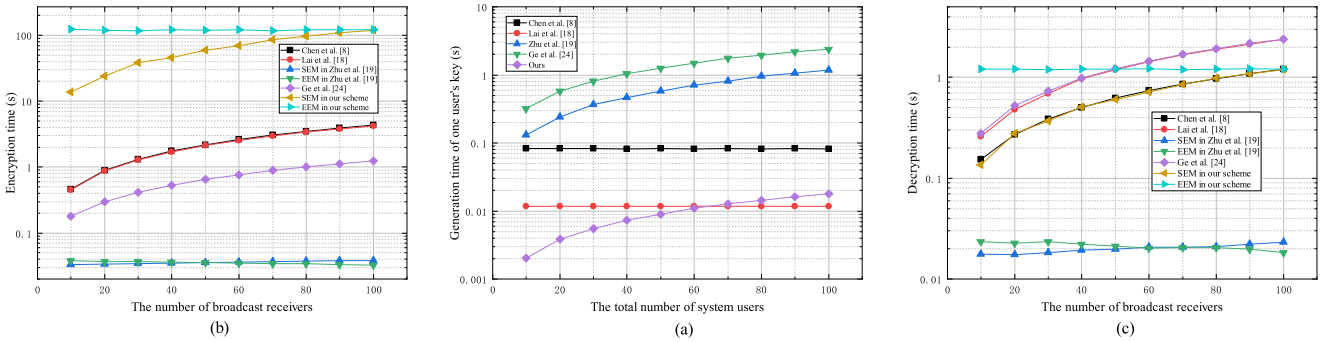

Fig. 9. Time overhead comparison between existing four BE schemes and ours. (a) Extraction time comparison. (b) Encryption time comparison. (c) Decryption time comparison.

pairing-based cryptography library (JPBC) and IntelliJ IDEA 2020.3.3. Our experiments are executed on a 64-bit Windows 10 computer with an Intel(R) Core(TM) i5-4590S CPU @3.00 GHz and 8 GB of RAM. In these experiments, we chose Type-A pairings on JPBC with a 160-bit group order, where the pairing are constructed on the curve $y^2 = x^3 + x$ over the field $\mathbb{F}_q$ for some prime $q = 3 \mod 4$. The order r is a prime factor of $q + 1$.

In Fig. 9, we further compare the time overheads of our scheme with that of [8], [18], [19], [24]. In our experiments, we let the total number of users $m = 100$, the number of broadcast receivers represents the number of specified receivers (k) in the SEM and the number of excluded receivers (l) in the EEM, respectively. Considering the *Setup* algorithm executed one time for generating system parameters, we only compare the time costs in *Extract*, *Encrypt*, and *Decrypt* between the abovementioned schemes and ours.

From Fig. 9(a), it is obvious that our scheme takes almost less time to generate a private key for one user than other schemes. When the number of users is more than 70, the extraction time of user private key in our scheme will be greater than [18]. However, compared with the other three schemes it still has obvious advantages. As shown in Fig. 9(b), our encryption time is higher than other schemes, which is consistent with the results of theoretical analysis in Table II. The reason is that our encryption algorithms in SEM and EEM need $\mathcal{O}(lm)$ and $\mathcal{O}(m^2)$ exponentiation operations in $\mathbb{G}$, respectively. As shown in Fig. 9(c), our decryption time in SEM is approximate to that of [8] but lower than that of [18] and [24]. While the decryption time in our EEM scheme is longer than that in [8] and [19], it is due to the fact that our decryption time is dependent on the total number of users m, whereas in the other two schemes, decryption time increases as the number of receivers k increases. However, our decryption algorithm performs better than that of [18] and [24] when the number of receivers is over 50.

Furthermore, to verify the efficiency of the DM-IBBE, we apply this scheme into four different real-world contracts and deploy these contracts on a blockchain system. Specifically, we introduce the algorithms constituting the DM-IBBE into contract programs, encapsulating them as predefined functions within SPESC-style housing-leasing, loan, purchase, and auction contracts.[3] Our experiments are conducted within a dedicated subchain denominated as ContractChain, which is developed on a Bitcoin architecture. Note that the cryptographic functions in our DM-IBBE are developed leveraging the JPBC library. In our experimental setup, we set the total count of users as 10, the number of specified receivers and that of excluded receivers is both 5. As shown in Table III, we compare the four aforementioned contracts in terms of the number of contract parties, the number of contract terms, the execution time of contract terms, and the storage cost of contract execution. In these contract instances, the number of parties and terms are ranged from 2 to 3 and from 8 to 10, respectively. It is easy to

[3]The introduction of these contracts and their codes can be found in https://www.smartlegalcontract.cn/mediawiki/index.php/Examples.

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE III
EXECUTION COMPARISON OF DIFFERENT CONTRACTS

Contract	Number of parties	Number of terms	Time cost	Storage cost
House-leasing contract	2	10	2760 ms	417968 Byte
Loan contract	3	8	3218 ms	435016 Byte
Purchase contract	2	10	2903 ms	425160 Byte
Auction contract	3	9	2558 ms	415632 Byte

see that all of contracts can be executed in a very short time, and the storage costs during the contract execution does not exceed 550 bits. Therefore, the experimental results shown that the proposed scheme has a stable cost of time and storage for different contracts.

VIII. CONCLUSION

In this article, we introduced encryption mechanism into SLCs to protect both specific-purpose and generic-purpose privacy on PPSCs. Based on the idea of COP, the sensitive data are protected in the form of contract terms. In order to respond to the implementation of protecting different privacy, we constructed a new DM-IBBE scheme with two kinds of encryption modes. And the proposed scheme can satisfy privacy demands of transactions by evaluating a use case of auction contract. In the future, the security of the DM-IBBE scheme will be further improved, such as exploring its security under other cryptographic hardness assumptions and establishing the security model under different chosen-ciphertext attacks. By undertaking these endeavors, a more comprehensive security assessment of the PPSCs could be constructed to ensure their business application secure in a wider range of contract transactions.

REFERENCES

[1] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F. Wang, "Blockchain-enabled smart contracts: Architecture, applications, and future trends," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 49, no. 11, pp. 2266–2277, Nov. 2019.

[2] D. Li, W. E. Wong, S. Pan, L. Koh, and M. Chau, "Design principles and best practices of central bank digital currency," *Int. J. Perform. Eng.*, vol. 17, no. 5, pp. 411–421, 2021.

[3] T. M. Hewa, M. Ylianttila, and M. Liyanage, "Survey on blockchain based smart contracts: Applications, opportunities and challenges," *J. Netw. Comput. Appl.*, vol. 177, 2021, Art. no. 102857.

[4] R. P. Priya, A. Tiwari, A. Pandey, and S. Krishna, "Identifying video tampering using watermarked blockchain," *Int. J. Perform. Eng.*, vol. 17, no. 8, pp. 722–732, 2021.

[5] A. Fiat and M. Naor, "Broadcast encryption," in *Proc. Adv. Cryptology - CRYPTO*, 1993, pp. 480–491.

[6] C. Delerablée, P. Paillier, and D. Pointcheval, "Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys," in *Proc. Pairing-Based Cryptography - Pairing*, 2007, pp. 39–59.

[7] D. Boneh, B. Waters, and M. Zhandry, "Low overhead broadcast encryption from multilinear maps," in *Proc. Adv. Cryptology - CRYPTO*, 2014, pp. 206–223.

[8] L. Chen, J. Li, Y. Lu, and Y. Zhang, "Adaptively secure certificate-based broadcast encryption and its application to cloud storage service," *Inf. Sci.*, vol. 538, pp. 273–289, 2020.

[9] C. Delerablée, "Identity-based broadcast encryption with constant size ciphertexts and private keys," in *Proc. Adv. Cryptology - ASIACRYPT*, 2007, pp. 200–215.

[10] R. Sakai and J. Furukawa, "Identity-based broadcast encryption," *IACR Cryptol. ePrint Arch.*, vol. 2007, p. 217, 2007.

[11] L. Zhang, Y. Hu, and Q. Wu, "Adaptively secure identity-based broadcast encryption with constant size private keys and ciphertexts from the subgroups," *Math. Comput. Model.*, vol. 55, no. 1–2, pp. 12–18, 2012.

[12] J. Kim, S. Camtepe, W. Susilo, S. Nepal, and J. Baek, "Identity-based broadcast encryption with outsourced partial decryption for hybrid security models in edge computing," in *Proc. Asia Conf. Comput. Commun. Secur.*, 2019, pp. 55–66.

[13] A. Barth, D. Boneh, and B. Waters, "Privacy in encrypted content distribution using private broadcast encryption," in *Proc. Financial Cryptography Data Secur. - FC*, 2006, pp. 52–64.

[14] L. Zhang, Q. Wu, and Y. Mu, "Anonymous identity-based broadcast encryption with adaptive security," in *Proc. Cyberspace Saf. Secur. - CSS*, 2013, pp. 258–271.

[15] K. He, J. Weng, J. Liu, J. K. Liu, W. Liu, and R. H. Deng, "Anonymous identity-based broadcast encryption with chosen-ciphertext security," in *Proc. Asia Conf. Comput. Commun. Secur.*, 2016, pp. 247–255.

[16] L. Chen, J. Li, and Y. Zhang, "Adaptively secure anonymous identity-based broadcast encryption for data access control in cloud storage service," *KSII Trans. Internet Inf. Syst.*, vol. 13, no. 3, pp. 1523–1545, 2019.

[17] W. Susilo, R. Chen, F. Guo, G. Yang, Y. Mu, and Y. Chow, "Recipient revocable identity-based broadcast encryption: How to revoke some recipients in IBBE without knowledge of the plaintext," in *Proc. Asia Conf. Comput. Commun. Secur.*, 2016, pp. 201–210.

[18] J. Lai, Y. Mu, F. Guo, W. Susilo, and R. Chen, "Anonymous identity-based broadcast encryption with revocation for file sharing," in *Proc. Inf. Secur. Privacy - ACISP*, 2016, vol. 9723, pp. 223–239.

[19] Y. Zhu, R. Yu, E. Chen, and D. Huang, "An efficient broadcast encryption supporting designation and revocation mechanisms," *Chin. J. Electron.*, vol. 28, no. 3, pp. 5–16, 2019.

[20] X. He, B. Qin, Y. Zhu, X. Chen, and Y. Liu, "SPESC: A specification language for smart contracts," in *Proc. Comput. Softw. Appl. Conf.*, 2018, pp. 132–137.

[21] Y. Zhu, W. Song, D. Wang, D. Ma, and C. C. Chu, "TA-SPESC: Toward asset-driven smart contract language supporting ownership transaction and rule-based generation on blockchain," *IEEE Trans. Reliab.*, vol. 70, no. 3, pp. 1255–1270, Sep. 2021.

[22] E. Chen et al., "SPESC-translator: Towards automatically smart legal contract conversion for blockchain-based auction services," *IEEE Trans. Serv. Comput.*, vol. 15, no. 5, pp. 3061–3076, Sep./Oct. 2022.

[23] Z. Zheng et al., "An overview on smart contracts: Challenges, advances and platforms," *Future Gener. Comput. Syst.*, vol. 105, pp. 475–491, 2020.

[24] A. Ge and P. Wei, "Identity-based broadcast encryption with efficient revocation," in *Proc. Public-Key Cryptography - PKC*, 2019, pp. 405–435.

[25] J. C. Choon and J. H. Cheon, "An identity-based signature from gap Diffie-Hellman groups," in *Proc. Public-Key Cryptography – PKC*, 2003, pp. 18–30.

[26] L. Chen and Z. Cheng, "Security proof of sakai-kasahara's identity-based encryption scheme," in *Proc. IMA Cryptography Coding - IMACC*, 2005, pp. 442–459.

[27] J. H. Cheon, "Security analysis of the strong Diffie-Hellman problem," in *Proc. Adv. Cryptology - EUROCRYPT*, 2006, pp. 1–11.

[28] Y. Watanabe, K. Emura, and J. H. Seo, "New revocable IBE in prime-order groups: Adaptively secure, decryption key exposure resistant, and with short public parameters," in *Proc. Topics Cryptology - CT-RSA*, 2017, pp. 432–449.

[29] F. Guo, W. Susilo, and Y. Mu, *Introduction to Security Reduction*. Berlin, Germany: Springer, 2019.

Authorized licensed use limited to: University of North Texas. Downloaded on March 02,2024 at 23:24:55 UTC from IEEE Xplore. Restrictions apply.