Hindawi
Security and Communication Networks
Volume 2021, Article ID 6637402, 13 pages
https://doi.org/10.1155/2021/6637402

Research Article

Blockchain as a CA: A Provably Secure Signcryption Scheme Leveraging Blockchains

Tzung-Her Chen,[1] Ting-Le Zhu,[1] Fuh-Gwo Jeng,[2] and Chien-Lung Wang[3]

[1]*Department of Computer Science and Information Engineering, National Chiayi University, Chiayi 60004, Taiwan*
[2]*Department of Applied Mathematics, National Chiayi University, Chiayi 60004, Taiwan*
[3]*Information Security Research Center, National Chung Hsing University, Taichung 40227, Taiwan*

Correspondence should be addressed to Chien-Lung Wang; kevinwang@nchu.edu.tw

Received 7 December 2020; Revised 25 January 2021; Accepted 9 March 2021; Published 26 March 2021

Academic Editor: Kaitai Liang

Although encryption and signatures have been two fundamental technologies for cryptosystems, they still receive considerable attention in academia due to the focus on reducing computational costs and communication overhead. In the past decade, applying certificateless signcryption schemes to solve the higher cost of maintaining the certificate chain issued by a certificate authority (CA) has been studied. With the recent increase in the interest in blockchains, signcryption is being revisited as a new possibility. The concepts of a blockchain as a CA and a transaction as a certificate proposed in this paper aim to use a blockchain without CAs or a trusted third party (TTP). The proposed provably secure signcryption scheme implements a designated recipient beforehand such that a sender can cryptographically facilitate the interoperation on the blockchain information with the designated recipient. Thus, the proposed scheme benefits from the following advantages: (1) it removes the high maintenance cost from involving CAs or a TTP, (2) it seamlessly integrates with blockchains, and (3) it provides confidential transactions. This paper also presents the theoretical security analysis and assesses the performance via the simulation results. Upon evaluating the operational cost in real currency based on Ethereum, the experimental results demonstrate that the proposed scheme only requires a small cost as a fee.

1. Introduction

With the rapid development of information and network applications, confidentiality, integrity, and nonrepudiation are the main security concerns. Thus, encryption and digital signatures have become two main fundamental technologies for any well-defined cryptosystem.

In 1997, Zheng presented the first signcryption concept that simultaneously fulfilled the functions of both the digital signature and public key encryption in a logical single step and achieved significantly lower costs than those required by "signature followed by encryption" [1]. Since then, there have been increasingly more signcryption schemes formed using either traditional public key infrastructure- (PKI-) based [1, 2], identity-based [3–7], or certificateless-based [8–17] digital signatures.

In 1984, Shamir presented a novel identity-based cryptosystem (IBC) [18] in which a random string (identity) was set as the participant's public key, and the corresponding private key was generated by a trusted third party (TTP) called the key generation center (KGC). This design eliminated the high cost of maintaining a traditional certificate-based PKI. Later, Boneh and Franklin proposed their identity-based encryption (IBE) [19] that fully exploited Shamir's IBC; this method significantly decreased the computational overhead by eliminating the certificate management problem in the PKI. In 2002, Malone-Lee combined the concepts of the IBC and signcryption to present an identity-based signcryption scheme [5]. Unfortunately, the identity-based signcryption schemes in Ref. [4, 5], to name a few, suffer the key escrow problem.

In 2003, Al-Riyami and Paterson proposed certificateless public key cryptography [20], in which the problems of either the utilization of certificate in PKI or the key escrow in IBC are eliminated. Consequently, Barbosa and Farshim proposed the first certificateless-based signcryption scheme

[8]. A certificateless signcryption scheme consists of four phases: (1) setup—the KGC generates its key pairs; (2) key generation—a participant chooses his key pair under the assistance of the KGC; (3) signcryption; and (4) unsigncryption. Notably, the participant's private key is partially generated by the KGC in the key generation phase and the KGC is assumed to be trusted. Furthermore, the KGC's public key and the signer's public key are involved in the unsigncryption phase and the authentication of the public keys is necessary. It is obvious that the certificateless signcryption schemes in Ref. [13, 16, 17], to name a few, cannot avoid the high maintenance cost of TTP.

Herein, we briefly examine a traditional certificate-based digital signature. Certificates, used to authenticate public keys prior to adoption, are chained as an ordered list containing an entity certificate, a series of intermediate certificates in the middle, and a root certificate at the end of the chain. For example, when Alice intends to use Bob's public key, the public key will be authenticated by verifying Bob's certificate first. Bob's certificate, including his identity information and his public key, is protected by the certificate issuer's signature. Obviously, it is tamper-proof and unforgeable. However, if someone intends to verify the issuer's signature, Alice needs to take the next step to confirm the issuer's public key by verifying the corresponding certificate that is issued by another upper issuer called the intermediate certificate authority (CA). The signatures of the certificates in the certificate chain (see Figure 1) must be verified up to the root CA certificate.

It is clear that maintaining the certificate chain, along with the assumption of a TTP, will increase the overall computational overhead; thus, the use of a blockchain becomes more pertinent. Due to the emergence of Bitcoin [21] in 2009, Ethereum [22] in 2013, and Hyperledger Fabric [23] in 2016, blockchain technologies [21, 22], providing a trusted mechanism without a CA and TTP, have received significant attention and interest in academia and the IT industry.

The fundamental technology of blockchains, such as Bitcoin and Ethereum, has gained increasingly more attention and has begun to be applied to various fields, such as medical data access [24–26], the Internet of Things [27, 28], and privacy preservation [29–31]. However, the high cost of certificate-based public key authentication is still problematic.

A blockchain is a continuously appending list of blocks (see Figure 2) that are linked and secured using cryptographic technologies, and the chain terminates in a genesis block. Each block typically contains an external hash pointer as a link to its previous block and an internal hash of all transactions (Tx). A transaction is a group of data, which includes the messages of "from" and "to" and the signatures. Figure 3 illustrates an instance of a transaction (Tx) in the block number 2752 in an Ethereum test blockchain.

Basically, an elliptic curve digital signature algorithm (ECDSA) [32] is what facilitates the blockchain concept in Bitcoin, Ethereum, and Hyperledger Fabric. The signature fields "r" and "s" in each transaction, as shown in Figure 3, are proven. For a user in a blockchain, his private key is regarded as his identity and security credential. It is worthwhile to note that the private key is generated and maintained by the user himself, not a trusted third party, and it is used to sign outgoing transactions. Instead of interacting with the blockchain, each user can directly interface with his blockchain node to deposit signed transactions and inspect the blockchain.

1.1. Motivation. A blockchain can be viewed as a public decentralized ledger to record all of the transactions in a publicly verifiable and permanent way. The data in (Tx) are tamper-proof and cannot be changed retroactively without altering all subsequent blocks up to the genesis one. The cryptographic advantages of a blockchain are similar to those of a certificate chain such as providing public verification and being tamper-proof. The main difference is that the blockchain is a decentralized trusty mechanism requiring no trusted third party; however, the certificate chain is a centralized trust mechanism with a series of trusted third parties.

In this paper, new cryptographic paradigms called *blockchain as a CA* and *transaction as a certificate* are investigated to implement a blockchain without a CA or TTP. Precisely, users' public keys are extracted and authenticated directly from transactions in blockchains instead of by CAs or a TTP. This combination is named *blockchain as a CA* (BaaCA). BaaCA has the following advantages.

(1) Avoiding the high maintenance cost of CA and TTP: for a block in the blockchain, a transaction with its ECDSA signature can be treated as certificate-like and utilized to extract the ECDSA public key such that both the public key and the transaction itself are authenticated. In this way, the concepts of a blockchain as a CA and a transaction as a certificate become intriguingly analogous to that of a certificate chain in the PKI.

(2) Seamlessly integrating with blockchains: due to the success of Bitcoin and Ethereum along with the promise of blockchain technologies, it is possible and feasible to combine blockchain technologies with signcryption to achieve the goals of confidentiality, integrity, and nonrepudiation simultaneously.

(3) Providing transaction confidentiality: by combining encryption with a digital signature to achieve the goal of "lower computational cost and communication cost," a novel signcryption scheme based on a blockchain that provides more benefits is proposed in this paper.

1.2. Contribution. In view of prior work, this paper proposes a provably secure BaaCA-based signcryption scheme. The main contributions of the proposed scheme are highlighted as follows.

(1) Using *blockchain as a CA* and *transaction as a certificate* to form a new signcryption scheme, it is feasible to eliminate the high maintenance cost from involving CAs or a TTP compared with the related works.

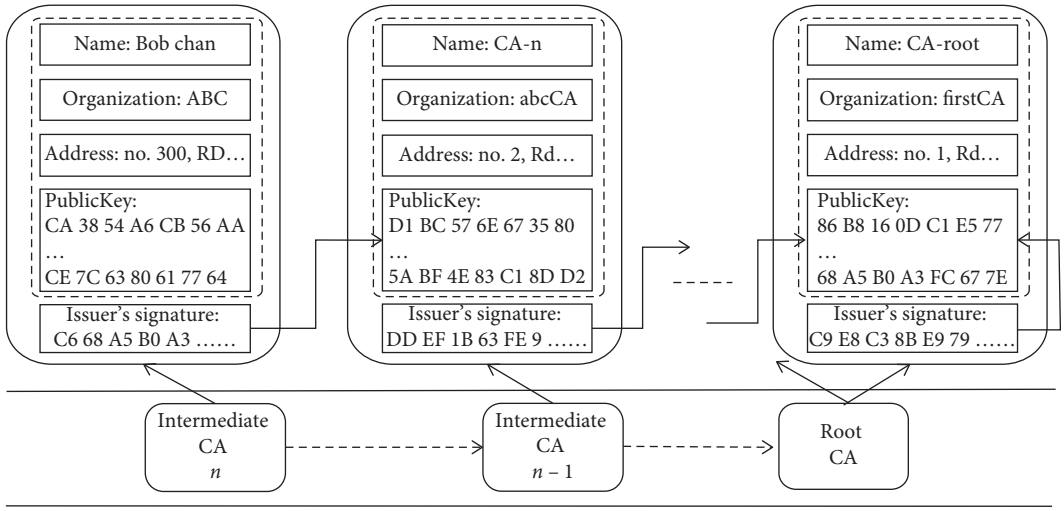

Figure 1: The structure in a certificate chain.

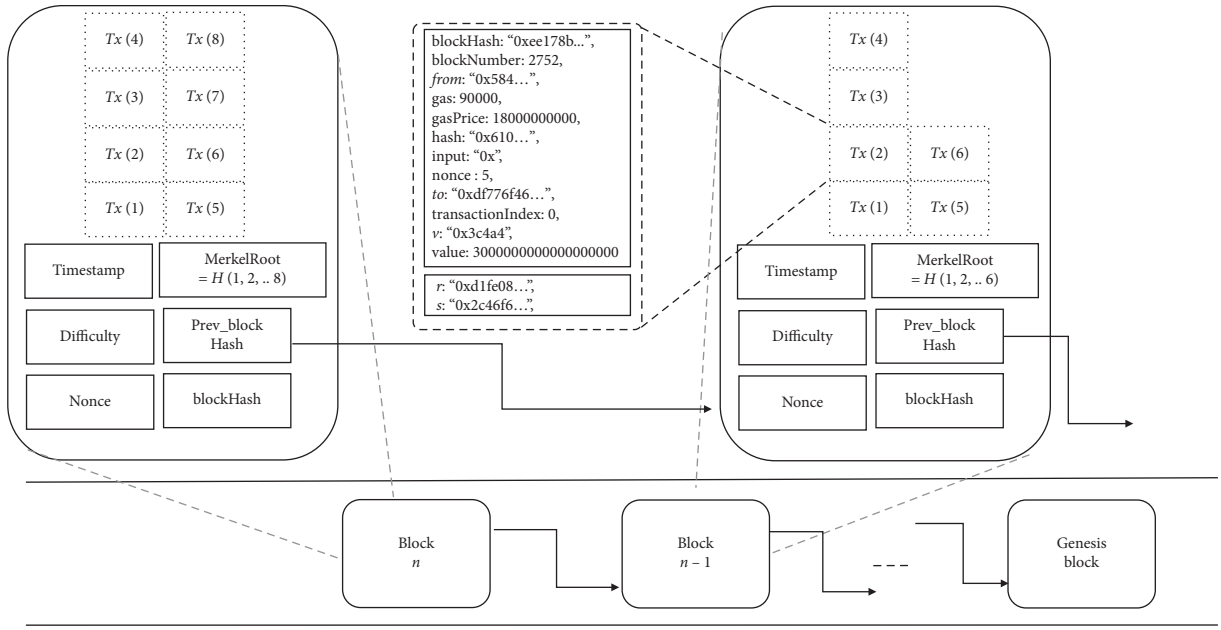

Figure 2: The structure of two blocks with some transactions each in a blockchain.

(2) The theoretical security analysis is demonstrated, and the experimental results show that the proposed scheme can work and interoperate with the Ethereum blockchain. In this way, the proposed scheme does seamlessly comply with a state-of-the-art blockchain.

(3) By combining blockchain technologies with signature and encryption, the proposed scheme achieves the goals of confidentiality, integrity, and nonrepudiation simultaneously. It is worth noting that the proposed scheme presents a new paradigm of providing confidentiality to transactions in the Bitcoin and Ethereum blockchains with a minor impact.

The rest of this paper is organized as follows. The related preliminaries are given in the next section. The proposed scheme with the formal system and security models are illustrated in Section 3. Section 4 presents the formal theoretical security proofs under the random oracle model along with the experimental results. The performance analysis is given in Section 5. Discussion and conclusions are given in Sections 6 and 7, respectively.

2. Preliminaries

2.1. Signature via ECDSA. The ECDSA algorithm [32] includes four phases; they are the setup, key generation, signature generation, and signature verification phases. They are briefly described below.

Setup phase: Let E be the given curve with its default field and equation. Then, base point P of a prime

{
blockHash: "0xee178bef3f572618df66a4c56407c65cb81010d8522cf4132b4d426a807ce1bb",
blockNumber: 2752,
from: "0x584248ba56fb274d4ca3acc98c0913ba096a52db",
gas: 90000,
gasPrice: 18000000000,
hash: "0x6109f152c9920c930004c8d2edeba6e629e2f7eef2ab011a1ffadde75b8bd488",
input: "0x",
nonce: 5,
r: "0xd1fe0847c4243db3db6e11ac1c9c30654a564bea1982b42fed4bbfe3904aa203",
s: "0x2c46f6bae4a08f05a8d3d514d0db627034dae9694952dcf0f28f0f60aa960fc7",
to: "0xdf776f46520fd10e76f41749708ec420c13c9f28",
transactionIndex: 2,
v: "0x3c4a4",
value: 3000000000000000000
}

Figure 3: The structure of a transaction in the Ethereum blockchain.

order on the curve and the multiplicative order n of point P can be determined. A hash function is given as follows: H_1: $\{0,1\}^* \longrightarrow \mathcal{F}_n$.

Key generation phase: The signer performs the following operations.

(1) Choose an integer d, such that $0 < d < n$, as the private key.

(2) Compute $Q = d \times P$ as the corresponding public key. Then, the private key is d and the public key is Q, n.

Signature generation phase: The signer signs message m by doing the following operations.

(1) Compute $e = H_1(m)$.

(2) Choose an integer k such that $0 < k < n$, where k is a cryptographically secure random number.

(3) Compute the curve point $(x_1, y_1) = k \times P$.

(4) Compute $r = x_1 \bmod n \neq 0$ and $s = k^{-1} \times (e + d \times r) \bmod n \neq 0$. Message m and its corresponding signatures (r, s) are sent to the recipient/verifier.

Signature verification phase: The verifier performs the following operations to verify the validation of the signature of message m.

(1) Compute $e = H_1(m)$.

(2) Compute $w = s^{-1} \bmod n$, $u_1 = e \times w \bmod n$, and $u_2 = r \times w \bmod n$.

(3) Compute $X = u_1 P + u_2 Q \neq 0$.

(4) Choose x-coordinate X as v.

(5) Check $v \stackrel{?}{=} r$.

If it holds, the signature of m is successfully authenticated.

2.2. Encryption via ECC. To encrypt or decrypt message m in ECC, the operations [33] are as follows.

Encryption phase:

(1) Choose an integer k such that $0 < k < n$, where k is a cryptographically secure random number.

(2) Suppose message m maps to the curve point M.

(3) Compute the curve point $C_1 = k \times P$.

(4) Compute the curve point $C_2 = M + k \times Q$. Send C_1 and C_2 to the recipient.

Decryption phase:

(1) Compute the curve point $M = C_2 - d \times C_1$.

(2) Decode point M to obtain plaintext m.

3. The Proposed Signcryption Scheme

The proposed BaaCA-based signcryption scheme including its system and security models is illustrated in this section. The notations used in the proposed scheme are described first in Table 1.

The proposed BaaCA-based signcryption scheme includes three different entities: a sender, a recipient, and the blockchain.

(1) Sender: a sender called Alice, with her private/public key pair (d_A, Pub_B), is sending data data_A to a specified recipient. First, the data data_A are split into two parts: data_A^1 and data_A^2. Note that the privacy-sensitive part of data_A is put into data_A^2 and the other remaining part of data_A will be packaged into data_A^1. Second, Alice executes the BaaCA-based signcryption algorithm to generate the encryption key key, which is used to transform data_A^2 into ciphertext c and to create the signature (r_A, s_A) at the same time. Finally, Alice posts the transaction $Tx_A = \{\mathrm{data}_A^1, c, r_A, s_A\}$ to the recipient via the blockchain.

(2) Recipient: a recipient called Bob, with his private/public key pair (d_B, Pub_B), will perform the decryption and

Table 1: Notations of the proposed scheme.

Symbol	Description
k	Global security parameter
$E, F_q, q, G, n, P,$ and F_n^*	Elliptic curve global parameters
$H_i(.), i = 1, 2$	One-way hash function
d_A, Pub_A, Pseudoname_A	Sender Alice's private/public key pairs and pseudo name
d_B, Pub_B, Pseudoname_B	Recipient Bob's private/public key pairs and pseudo name
data_A	Data sent from sender to recipient
data_A^1	Public part of data_A
data_A^2	Sensitive part of data_A
r_i, s_i	ECDSA signature
c	Ciphertext
Tx	Transaction
A	System definition

verification processes when he gets transaction Tx_A to obtain the original data data_A. To decrypt ciphertext c, Bob has to execute the signcryption algorithm first to generate the decryption key key′, which is the same as Alice's encryption key. Consequently, Bob uses key′ to decrypt c to obtain the privacy-sensitive data data_A^2 and then recover the original data data_A. When performing verification, first, Bob uses both the signature (r_A, s_A) and data_A to generate Alice's public key Pub_A'. Therefore, Bob can use Pub_A' to extract Alice's account address and check whether the account address is the same as that in transaction Tx_A. Please note that each account address is transformed from the relative public key in the blockchain. If they are the same, Bob can ensure that data data_A are sent by sender Alice.

(3) Blockchain: the blockchain acts as a digital ledger that records all transactions that have ever happened. That is, the blockchain is treated as a CA and the tamper-proof transactions are treated as the *certificates* to extract the related public keys.

The proposed scheme consists of five algorithms. They are system setup, key generation, public-key extraction, signcryption, and unsigncryption. For simplicity, we denote them as Setup, KeyGen, PKExtract, Signcryption, and Unsigncryption, respectively. The details are described below.

(i) Setup: it is a probabilistic algorithm for generating the public parameters *params* of the system using a security parameter k as the input. It can be represented as $(\text{params}) \leftarrow \text{Setup}(k)$.

(ii) KeyGen: it is a probabilistic algorithm that randomly chooses private key d_i as its input to generate its corresponding public key Pub_i and its relative account addressPseudonym_i. It can be represented as $(d_i, \text{Pub}_i, \text{Pseudonym}_i) \leftarrow \text{KeyGen}(d_i)$.

(iii) PKExtract: it is a deterministic algorithm that sets ECDSA signature (r_i, s_i) as its input and returns its relative public key Pub_i. It can be represented as $\text{Pub}_i = \text{PKExtract}(r_i, s_i)$.

(iv) Signcryption: it is a probabilistic algorithm that takes the sender's private key d_A, the recipient's public key Pub_B, and $\text{datadata}_A = \text{data}_A^1 \| \text{data}_A^2$ as inputs and outputs the corresponding signature (r_A, s_A), ciphertext c, and nonsensitive part of data data_A^1. The outputs are denoted as ω. It can be represented as $\omega \leftarrow \text{Signcryption}(\text{data}_A, d_A, \text{Pub}_B)$ $\omega = \{\text{data}_A^1, c, r_A, s_A\}$.

(v) Unsigncryption: it is a deterministic algorithm that takes the sender's public key Pub_A, the recipient's private key d_B, and ω as inputs and outputs either the corresponding data_A if the signature (r_A, s_A) is valid or Reject. It can be represented as $(\text{data}_A/\text{Reject}) \leftarrow \text{Unsigncryption}(\omega, d_B, \text{Pub}_A)$.

3.1. Security Model. There are two security concerns about the security model of the proposed BaaCA-based signcryption scheme in this paper. They are confidentiality and unforgeability.

(1) *Confidentiality*. The property of confidentiality in the proposed BaaCA-based signcryption scheme is essentially provided since it is only the specified recipient that knows data data_A. The following experiment $\text{Exp}_{\mathscr{A}}^{\text{IND-BaaCA-CCA}}(k)$, played with the adversary IND-BaaCA-CCA denoted as $\mathscr{A}$ and a challenger denoted as $\mathscr{C}$, proves the property of confidentiality under chosen ciphertext attacks.

(i) Setup: $\mathscr{C}$ runs this algorithm to obtain public parameters params and then distributes the public parameters to the adversary $\mathscr{A}$.

(ii) Queries (phase 1): $\mathscr{A}$ makes a number of oracle queries to $\mathscr{C}$ and $\mathscr{C}$ would give some information to $\mathscr{A}$. In the experiment, the following queries are allowed:

(a) KeyGen queries: $\mathscr{A}$ sends $d_{\mathscr{A}}$ to $\mathscr{C}$, and then $\mathscr{C}$ runs $\text{KeyGen}(d_{\mathscr{A}}) \longrightarrow (d_{\mathscr{A}}, \text{Pub}_{\mathscr{A}}, \text{Pseudonym}_{\mathscr{A}})$ and returns $(d_{\mathscr{A}}, \text{Pub}_{\mathscr{A}}, \text{Pseudonym}_{\mathscr{A}})$ to $\mathscr{A}$.

(b) Signcryption queries: $\mathscr{A}$ sends $(\text{data}_{\mathscr{A}}, d_{\mathscr{A}}, \text{Pub}_B)$ to $\mathscr{C}$, and then $\mathscr{C}$ runs Signcryption $(\text{data}_{\mathscr{A}}, d_{\mathscr{A}}, \text{Pub}_B)$ $\longrightarrow \omega, \omega = \{\text{data}_{\mathscr{A}}^1, c, r_{\mathscr{A}}, s_{\mathscr{A}}\}$ and returns ω to $\mathscr{A}$.

(c) Unsigncryption queries: $\mathscr{A}$ sends $(\omega, d_B, \text{Pub}_{\mathscr{A}})$ to $\mathscr{C}$, and then $\mathscr{C}$ runs $\text{Signcryption}(\omega, d_B, \text{Pub}_{\mathscr{A}}) \longrightarrow$ $((\text{data}_{\mathscr{A}})/\text{Reject})$ and returns $(\text{data}_{\mathscr{A}}/\text{Rej})$ to $\mathscr{A}$.

(iii) Challenge: after finishing phase 1, adversary $\mathscr{A}$ chooses two data data_{A_0} and data_{A_1} with an arbitrary private key $d^*_{\mathscr{A}}$, which he wishes to challenge. $\mathscr{C}$ randomly chooses a bit $b \in \{0,1\}$ for the two challenged data and then runs signcryption queries to obtain the result ω^*. Finally, $\mathscr{C}$ sends it to $\mathscr{A}$.

(iv) Queries (phase 2): after receiving ω^*, adversary $\mathscr{A}$ asks a number of queries similar to those in phase 1 but $(\omega^*, d^*_B, \text{Pub}^*_{\mathscr{A}})$ will be not sent to $\mathscr{C}$ under unsigncryption queries.

(v) Guess: finally, adversary $\mathscr{A}$ chooses a bit b' from $\{0,1\}$. The adversary$\mathscr{A}$ is said to win this experiment if $b\imath = b$. We can define the advantage of $\mathscr{A}$ in this experiment as$\text{Adv}^{\text{IND-BaaCA-CCA}}_{\mathscr{A}}(k) = |\Pr(b' = b) - (1/2)|$.

(2) *Unforgeability*. In the proposed BaaCA-based signcryption scheme, unforgeability is essential to ensuring that the signature is secure against adaptively chosen message attacks. The following experiment $\text{Exp}^{\text{EU-BaaCA-CMA}}_{\mathscr{A}}(k)$, played with an adversary EU-BaaCA-CMA denoted as $\mathscr{A}$ and a challenger denoted as $\mathscr{C}$, proves the existential property of unforgeability under the chosen message attack.

(i) Setup: it is the same as that in the experiment $\text{Exp}^{\text{IND-BaaCA-CCA}}_{\mathscr{A}}$.

(ii) Queries: it is the same as phase 1 of queries in the experiment $\text{Exp}^{\text{IND-BaaCA-CCA}}_{\mathscr{A}}$.

(iii) Output: the adversary $\mathscr{A}$ is said to win this experiment if $\mathscr{A}$ obtains the signature ω^* together with two arbitrary private keys $d^*_{\mathscr{A}}$ and d^*_B such that

(1) $\text{data}^*_{\mathscr{A}} \leftarrow \text{Unsigncryption}(\omega^*, d^*_B, \text{Pub}^*_{\mathscr{A}})$.

(2) $(\text{data}^*_{\mathscr{A}}, d^*_{\mathscr{A}}, \text{Pub}^*_B)$ is never sent to $\mathscr{C}$ as inputs for the signcryption queries, where $\text{data}^*_{\mathscr{A}}$ are the data corresponding to the forgery.

Definition 1. The proposed BaaCA-based signcryption scheme is said to be secure if there is no adversary $\mathscr{A}$ that wins the experiment $\text{Exp}^{\text{IND-BaaCA-CCA}}_{\mathscr{A}}(k)$ with a non-negligible probability ε.

Definition 2. The proposed BaaCA-based signcryption scheme is said to be secure if there is no adversary $\mathscr{A}$ that wins the experiment $\text{Exp}^{\text{EU-BaaCA-CMA}}_{\mathscr{A}}(k)$ with a non-negligible probability ε. The security of the proposed BaaCA-based signcryption scheme is based on the computational difficulty of some well-known hard problems defined below.

Definition 3. The elliptic curve discrete logarithm problem (ECDLP): if $a \in \mathscr{F}^*_n$ is unknown, compute a by giving P and aP.

Definition 4. The elliptic curve computational Diffie–Hellman problem (ECCDHP): if $a, b \in \mathscr{F}^*_n$ is unknown, compute abP by giving P, aP, and bP.

In this paper, the appropriate $\mathbb{G}$ is determined by blockchains where the ECDLP and ECCDHP are assumed to be computationally difficult.

3.2. The Proposed Scheme. According to the benefits of the *blockchain as a CA* concept, the proposed scheme uses a variant of the ECDSA to produce a new signcryption scheme. Since the ECDSA is the default signature algorithm in Bitcoin, Ethereum, and Hyperledger Fabric, the proposed scheme is seamlessly compliant with the blockchains.

The proposed model consists of two participants of a blockchain: Alice as a *sender* and Bob as a *recipient*. Both of the participants are external actors of the blockchain and have their relative blockchain accounts and some recorded-in-block transactions. Suppose Alice and Bob, denoted as Pseudonym_A and Pseudonym_B, respectively, with the addresses in the fields "from" or "to" in Figure 3 have their private and public key pairs $(d_A, \text{Pub}_A = d_A \times P)$ and $(d_B, \text{Pub}_B = d_B \times P)$ in the blockchain network, respectively.

The main design comes from the method that generates the private key from the existing transactions signed by Bob and Alice in the blockchain. Herein, a transaction is regarded as a certificate of Alice or Bob.

Alice is connected with the blockchain, receives Bob's previous signature from a transaction stored in the blockchain, and then extracts Bob's public keyPub_B. The encryption key is obtained by executing $k \times \text{Pub}_B$, where k is a one-time padded random number generated by Alice since Bob can extract this key by performing $d_B \times R$ when he has R, where $R = k \times P$.

The proposed signcryption scheme consists of five phases: the setup phase, key generation phase, public-key extraction phase, signcryption phase, and unsigncryption phase.

(i) Setup phase:

(1) Determine the elliptic curve E in the finite field $\mathscr{F}_q$, where q is a prime number such that all the points on E represent a finite group, e.g., $\mathbb{G}$, with the prime multiplicative order n and a generator P. For example, the ECDSA curve used in Bitcoin and Ethereum is secp256k1, which refers to the curve E: $y^2 = x^3 + 7$ [34] defined over the field $\mathscr{F}^*_n$, while the curves ECC P256 and P384 are adopted by Hyperledger Fabric.

(2) H_1: $\{0,1\}^* \longrightarrow \mathscr{F}_n$ and H_2: $\mathbb{G} \times \{0,1\}^* \longrightarrow \mathscr{F}^*_n$ are two collision resistant hash functions, where $\mathscr{F}_n$ and $\mathbb{G}$ have been described in (1).

(3) The system parameters params: $= \langle \mathbb{G}, n, P, q, E, \mathscr{F}^*_n, \mathscr{F}_q, T \rangle$ are public.

(ii) Key generation phase:

$$\left\lfloor \frac{\text{SHA\_256}(\text{SHA\_256}(0x00 \| \text{RIPEMD\_160}(\text{SHA\_256}(\text{Pub}_B))))}{2^{224}} \right\rfloor, \tag{1}$$

(1) Suppose that Alice has her ECDSA private and public key pairs $(d_A, \text{Pub}_A = d_A \times P)$ and Bob has

his key pairs $(d_B, \mathrm{Pub}_B = d_B \times P)$ for their accounts on the blockchain, respectively.

(2) The ECDSA private and public key pairs are used to specify the address, i.e., $\mathrm{Pseudonym}_A$ and $\mathrm{Pseudonym}_B$. For Bitcoin, Bob's address is computed as Base_58 (0x00||RIPEMD_160 (SHA_256 (Pub_B))||.

where Base_58 represents a large integer as alphanumeric text, RIPEMD_160 and SHA_256 are two well-known cryptographic hash functions, and || denotes concatenation. For Ethereum, the address is

$$\mathrm{B}_{96\ldots255}(\mathrm{SHA}-3(\mathrm{Pub}_B)), \tag{2}$$

where $\mathrm{B}_{96\ldots255}$ is denoted as the right most 160 bits of the SHA-3 hash.

(iii) Public-key extraction phase: because there is no certificate chain in the Bitcoin and Ethereum blockchains, Alice must extract Bob's public key from Bob's transaction $Tx_B = \{\mathrm{data}_B, r_B, s_B\}$, where Tx_B is public verifiable and tamper-proof once stored in the blockchain. With the ECDSA signature r_B, s_B and the transaction data data_B, Alice is able to extract Bob's public key Pub_B by conducting the following operations.

(1) $e = H_1(\mathrm{data}_B)$.

(2) R or $R' \leftarrow r_B = x_1$, where $R = (x_1, y_1)$, and $R' = (x_1, -y_1)$.

(3) $\mathrm{Pub}_B = r_B^{-1} \times (s_B R - eP) \bmod n$. $\mathrm{Pub}_B' = r_B^{-1} \times (s_B R' - eP) \bmod n$.

Actually, Bitcoin and Ethereum transactions require an extra parameter, i.e., the field "v" in Figure 3, to identify which point is correct.

(4) Check if the generated node address from Pub_B via equations (1) or (2) for Bitcoin or Ethereum, respectively, is equal to Bob's account address. If it is, Pub_B is Bob's public key and Bob's transaction Tx_B in the blockchain is also authenticated.

(iv) Signcryption phase: suppose Alice's transaction is $Tx_A = \{\mathrm{data}_A, r_A, s_A\}$, where data_A are composed of two parts: data_A^1 are public and data_A^2 are privacy-sensitive. Alice does the following operations to generate the signature of Tx_A and the ciphertext of data_A^2. Note that the signature in this algorithm is designed to be verified by the designated verifier, i.e., Bob. $k \in_R \{1, \ldots, n-1\}$.

(1) $R_A = k \times P$.

(2) $R_A = (x_1, y_1)$.

(3) $r_A = x_1 \bmod n$. $K = k \times \mathrm{Pub}_B$.key $= H_2(K, \mathrm{Pseudonym}_A, \mathrm{Pseudonym}_B)$.$c = \mathrm{data}_A^2 \oplus \mathrm{key}$.

(4) $e = H_1(\mathrm{data}_A) = H_1(\mathrm{data}_A^1, \mathrm{data}_A^2)$. $s_A = k^{-1} \times (e + d_A \times r_A) \bmod n$.

(5) Return $\omega = \{\mathrm{data}_A^1, c, r_A, s_A\}$.

(5)–(7) are the operations of both key agreement and encryption. Finally, Alice broadcasts the transaction $Tx_A = \omega$ into the blockchain and discards the random number k and the encryption key key.

(v) Unsigncryption phase: upon interacting with the blockchain to obtain $Tx_A = \omega = \{\mathrm{data}_A^1, c, r_A, s_A\}$, Bob verifies the validation of the signature and simultaneously decrypts the ciphertext to obtain data_A^2 by doing the following operations.

(1) Find two points R_A and R_A' of the same value r_A as the x-coordinate. The transaction practically involves an extra parameter, say v in Figure 3, to identify R_A and R_A', say R_A. $K' = d_B \times R_A$.

(2) $\mathrm{key}' = H_2(K', \mathrm{Pseudonym}_A, \mathrm{Pseudonym}_B)$.

(3) $\mathrm{data}_A^2 = c \oplus \mathrm{key}'$.

(4) $e' = H_1(\mathrm{data}_A^1, \mathrm{data}_A^2)$.

(5) $\mathrm{Pub}_A' = r_A^{-1} \times (s_A R_A - e'P) \bmod n$.

(6) Check if the generated node address from Pub_A' by equation (1) or (2) is equal to Alice's account address.

(2)–(4) are both key agreement and decryption operations. If (7) is true, Pub_A' is Alice's public key and this transaction Tx_A is also authenticated. Then, the data data_A are output; otherwise, we get Reject.

3.3. Correctness of Public-Key Extraction, Encryption, and Signature. The correctness of public key extraction in the proposed protocol is derived from the precision of the hash value of the extracted public key $h(\mathrm{Pub}_B')$ to the address $h(\mathrm{Pub}_B)$ indicated in the transaction on the blockchain, where $h(.)$ represents equation (1) or (2). We have $\mathrm{Pub}_B' = r_B^{-1} \times (s_B R_B - eP) = r_B^{-1} \times (k^{-1} \times (e + d_B \times r_B) \times (k \times P) - e \times P) = r_B^{-1} \times (d_B \times r_B \times P) = d_B \times P = \mathrm{Pub}_B$. Thus, the equation $h(\mathrm{Pub}_B') = h(\mathrm{Pub}_B)$ holds.

The correctness of the encryption in the proposed protocol is derived from the situation $\mathrm{key}' = \mathrm{key}$. If it holds, $\mathrm{data}_A^2 = \mathrm{data}_A^2$. In the signcryption phase, Alice's encryption key is $\mathrm{key} = H_2(k \times \mathrm{Pub}_B, \mathrm{Pseudonym}_A, \mathrm{Pseudonym}_B, \mathrm{data}_A^1)$. In addition, in the unsigncryption phase, the decryption key that Bob obtains is $\mathrm{key}' = H_2(d_B \times R_A, \mathrm{Pseudonym}_A, \mathrm{Pseudonym}_B)$. Since $d_B \times R_A = d_B \times k \times P = k \times d_B \times P = k \times \mathrm{Pub}_B$, we have $\mathrm{key} = \mathrm{key}'$, and thus $\mathrm{data}_A^2 = c \oplus \mathrm{key}' = (\mathrm{data}_A^2 \oplus \mathrm{key}) \oplus \mathrm{key}' = \mathrm{data}_A^2$.

The correctness of the signature in the proposed protocol is derived from the same proof as that of correctness of public-key extraction. In this way, the signature r_A, s_A is also authenticated. Thus, the proof is omitted here.

4. Security Analysis

Two theorems are described in detail before the security analytics and proofs of the proposed BaaCA-based signcryption scheme are given.

Theorem 1 (IND-BaaCA-CCA). The proposed BaaCA-based signcryption scheme is indistinguishable against

chosen ciphertext attacks when the ECDDHP problem is hard to resolve.

Proof. Assume there exists an IND-BaaCA-CCA adversary, $\mathscr{A}$, who wins the experiment $\mathrm{Exp}_{\mathscr{A}}^{\mathrm{IND-BaaCA-CCA}}$ with a nonnegligible probability ε based on Definition 1. A challenger $\mathscr{C}$ is designed to take $\mathscr{A}$ as the subprogram to solve the ECCDHP problem based on Definition 4 with a nonnegligible probability. Challenger $\mathscr{C}$ is assigned an instance $\langle \mathbb{G}, n, P, q, E, \mathscr{F}_n^*, \mathscr{F}_q, kP, d_B P\rangle$ and tries to compute the value $k \times d_B P \in \mathbb{G}$.

At first, challenger $\mathscr{C}$ maintains four lists L_1, L_2, L_S, and L_u, respectively, corresponding to H_1, H_2, signcryption, and unsigncryption query oracles. Then, $\mathscr{C}$ plays the following experiment $\mathrm{Exp}_{\mathscr{A}}^{\mathrm{IND-BaaCA-CCA}}$ with $\mathscr{A}$. In addition, we assume that $\mathscr{A}$ queries oracle H_2 q_2 times and queries KeyGen q_K times.

(1) Setup: $\mathscr{C}$ sends $\mathscr{A}$ params: $= \langle \mathbb{G}, n, P, q, E, \mathscr{F}_n^*, \mathscr{F}_q, H_1, H_2\rangle$, which is an instance for solving the ECCDHP problem. Two hash functions $H_i(.)$, $i = 1, 2$, controlled by $\mathscr{C}$, are regarded as random oracles.

(2) Queries (phase 1): $\mathscr{A}$ makes a number of oracle queries to $\mathscr{C}$ and $\mathscr{C}$ should answer to give $\mathscr{A}$ some information. In the experiment, the following queries are allowed.

(1) Oracle H_1: $(\mathrm{data}_{\mathscr{A}}^1, \mathrm{data}_{\mathscr{A}}^2)$.

Case 1: $((\mathrm{data}_{\mathscr{A}}^1, \mathrm{data}_{\mathscr{A}}^2), e) \in L_1$, return e.

Case 2: $((\mathrm{data}_{\mathscr{A}}^1, \mathrm{data}_{\mathscr{A}}^2), e) \notin L_1$, choose random $e \in \mathscr{F}_n^*$ add $((\mathrm{data}_{\mathscr{A}}^1, \mathrm{data}_{\mathscr{A}}^2), e)$ to L_1, and then return e.

(2) Oracle H_2: $(K, \mathrm{Pseudonym}_{\mathscr{A}}, \mathrm{Pseudonym}_B)$.

Case 1: $(K, \mathrm{Pseudonym}_{\mathscr{A}}, \mathrm{Pseudonym}_B, \mathrm{key}) \in L_2$ and return key.

Case 2: $(K, \mathrm{Pseudonym}_{\mathscr{A}}, \mathrm{Pseudonym}_B, \mathrm{key}) \notin L_2$, choose random key $\in \mathscr{F}_n^*$, add $(K, \mathrm{Pseudonym}_{\mathscr{A}}, \mathrm{Pseudonym}_B, \mathrm{key})$ to L_2, and then return key.

(3) KeyGen queries: $d_{\mathscr{A}}$

Case 1: $(d_{\mathscr{A}}, \mathrm{Pub}_{\mathscr{A}}, \mathrm{Pseudonym}_{\mathscr{A}}) \in L_K$, return $(d_{\mathscr{A}}, \mathrm{Pub}_{\mathscr{A}}, \mathrm{Pseudonym}_{\mathscr{A}})$.

Case 2: $(d_{\mathscr{A}}, \mathrm{Pub}_{\mathscr{A}}, \mathrm{Pseudonym}_{\mathscr{A}}) \notin L_K$, generate public key $\mathrm{Pub}_{\mathscr{A}}$ and account address $\mathrm{Pseudonym}_{\mathscr{A}}$ via equation (1) or (2), and then add $(d_{\mathscr{A}}, \mathrm{Pub}_{\mathscr{A}}, \mathrm{Pseudonym}_{\mathscr{A}})$ to L_K. Finally, return $((d_{\mathscr{A}}, \mathrm{Pub}_{\mathscr{A}}, \mathrm{Pseudonym}_{\mathscr{A}})$.

(4) Signcryption queries: $(\mathrm{data}_{\mathscr{A}}, d_{\mathscr{A}}, \mathrm{Pub}_B)$

(a) Choose random $k \in \mathscr{F}_n^*$.

(b) Compute $R = kP = (x_1, y_1)$.

(c) $\mathrm{data}_{\mathscr{A}}^1 \| \mathrm{data}_{\mathscr{A}}^2 \leftarrow \mathrm{data}_{\mathscr{A}}$.

(d) Compute $r_{\mathscr{A}} = x_1 \bmod n$.

(e) Compute $K = k \cdot \mathrm{Pub}_B$.

(f) Compute $\mathrm{key} = H_2(K, \mathrm{Pseudonym}_{\mathscr{A}}, \mathrm{Pseudonym}_B)$, where $H_2(.)$ is from oracle H_2.

(g) Compute $c = \mathrm{data}_{\mathscr{A}}^2 \oplus \mathrm{key}$.

(h) Compute $e = H_1(\mathrm{data}_{\mathscr{A}}^1, \mathrm{data}_{\mathscr{A}}^2)$, where $H_1(.)$ is from oracle H_1.

(i) Compute $s_{\mathscr{A}} = k^{-1}(e + d_{\mathscr{A}} r_{\mathscr{A}}) \bmod n$, where $d_{\mathscr{A}}$ is taken from the KeyGen queries.

(j) Add $(d_{\mathscr{A}}, \mathrm{Pub}_B, \mathrm{data}_{\mathscr{A}}^1, \mathrm{c}, r_{\mathscr{A}}, s_{\mathscr{A}})$ to L_S.

(k) Return $\omega = \{\mathrm{data}_{\mathscr{A}}^1, c, r_{\mathscr{A}}, s_{\mathscr{A}}\}$.

(5) Unsigncryption queries: $(\omega, d_B, \mathrm{Pub}_{\mathscr{A}})$

(a) Find two points R and R' with the same value r as their x coordinate. The transaction practically involves an extra parameter to identify R and R', say R.

(b) Compute $K = d_B \cdot R$.

(c) Compute $\mathrm{data}_{\mathscr{A}}^2 = c \oplus \mathrm{key}$.

(d) Compute$e = H_1(\mathrm{data}_{\mathscr{A}}^1, \mathrm{data}_{\mathscr{A}}^2)$.

(e) Check whether $\mathrm{Pub}_{\mathscr{A}} \stackrel{?}{=} r^{-1} \times (sR - e'P)$ holds or not.

(f) Return $\mathrm{data}_{\mathscr{A}}$ and add $(\mathrm{data}_{\mathscr{A}}, d_{\mathscr{A}}, \mathrm{Pub}_B)$ to L_u; otherwise, return Reject.

(1) Challenge: after finishing phase 1, $\mathscr{A}$ chooses two datasets$\mathrm{data}_{\mathscr{A}_0}$ and $\mathrm{data}_{\mathscr{A}_1}$ with an arbitrary private key $d_{\mathscr{A}}^*$, which are to be challenged. Here, $d_{\mathscr{A}} = d_A$ with probability $(1/q_K)$. Then, $\mathscr{C}$ randomly chooses a bit $b \in \{0, 1\}$for the two challenged datasets and then runs the signcryption algorithm to obtain a result $\omega^* = \{\mathrm{data}_{\mathscr{A}}^{1*}, c^*, r_{\mathscr{A}}^*, s_{\mathscr{A}}^*\}$. Finally, $\mathscr{C}$ sends ω^* to $\mathscr{A}$.

(2) Queries (phase 2): after receiving ω^*, adversary $\mathscr{A}$ asks a number of queries that are the same as in phase 1 but ω^* is never sent to $\mathscr{C}$ for unsigncryption queries.

(3) Guess: adversary $\mathscr{A}$ produces a bit b' from $\{0, 1\}$. $\mathscr{C}$ ignores the bit that $\mathscr{A}$ produces and randomly chooses K, which is stored in L_2, as the solution for the given instance of ECCDHP problem. If $\mathscr{A}$ does not query oracle H_2, $\mathscr{C}$ will terminate. In contrast, if $\mathscr{A}$ has some advantages to produce the bit correctly, it means that the query to H_2 is required. Therefore, there is enough information in L_2 to help $\mathscr{C}$ obtain the correct K with probability $(1/q_2)$ because $\mathscr{A}$ queries oracle H_2 q_2 times and only one K is right. If so, $K = d_B \times R = d_B \times kP$ is the solution with probability $(\varepsilon/(q_K q_2))$ for the given instance of ECCDHP problem. This is contradictory to our hypothesis at the beginning of the security proof. Thus, this concludes the proof of Theorem 1. □

Theorem 2 (EU-BaaCA-CMA). The proposed BaaCA-based signcryption scheme is existentially unforgeable against an adaptively chosen message attack when the ECDLP problem is hard to resolve.

Proof. Assume there is an EU-BaaCA-CMA adversary, who wins the defined experiment $\mathrm{Exp}_{\mathscr{A}}^{\mathrm{EU-BaaCA-CMA}}$ with a nonnegligible probability ε. We will design a challenger $\mathscr{C}$ to take $\mathscr{A}$ as its subprogram for solving the ECDLP problem with a nonnegligible probability. The instance $\langle \mathbb{G}, n, P, q, E, \mathscr{F}_n^*, \mathscr{F}_q, kP\rangle$ of the ECDLP problem is given to challenger $\mathscr{C}$, and $\mathscr{C}$ will try to compute the value $k \in \mathscr{F}_n^*$.

At first, challenger $\mathscr{C}$ maintains five lists L_1, L_2, L_K, L_S, and L_u, respectively, corresponding to H_1, H_2, KeyGen, signcryption, and unsigncryption oracles. Then, $\mathscr{C}$ plays the following experiment $\text{Exp}_{\mathscr{A}}^{\text{IND-BaaCA-CCA}}$ with $\mathscr{A}$. In addition, we assume that $\mathscr{A}$ queries oracle H_2 q_2 times and queries KeyGen q_K times.

(1) Setup: $\mathscr{C}$ sends params: $= \langle \mathbb{G}, n, P, q, E, \mathscr{F}_n^*, \mathscr{F}_q, H_1, H_2 \rangle$ to $\mathscr{A}$, where $\langle \mathbb{G}, n, P, q, E, \mathscr{F}_n^*, \mathscr{F}_q \rangle$is an instance of the ECDLP problem. The hash functions $H_i(.), i = 1, 2$, controlled by $\mathscr{C}$, are regarded as random oracles.

(2) Queries (phase 1): $\mathscr{A}$ makes a number of oracle queries to $\mathscr{C}$ and $\mathscr{C}$ should answer to give $\mathscr{A}$ some information. In the experiment, the following queries are allowed:

(1) Oracle H_1: $(\text{data}_{\mathscr{A}}^1, \text{data}_{\mathscr{A}}^2)$

Case 1: $((\text{data}_{\mathscr{A}}^1, \text{data}_{\mathscr{A}}^2), e) \in L_1$ and return e.

Case 2: $((\text{data}_{\mathscr{A}}^1, \text{data}_{\mathscr{A}}^2), e) \notin L_1$, choose random $e \in \mathscr{F}_n^*$, add $((\text{data}_{\mathscr{A}}^1, \text{data}_{\mathscr{A}}^2), e)$ to L_1, and then return e.

(2) Oracle H_2: $(K, \text{Pseudonym}_{\mathscr{A}}, \text{Pseudonym}_B)$

Case 1: $(K, \text{Pseudonym}_{\mathscr{A}}, \text{Pseudonym}_B, \text{key}) \in L_2$ and return key.

Case 2: $(K, \text{Pseudonym}_{\mathscr{A}}, \text{Pseudonym}_B, \text{key}) \notin L_2$, choose random $\text{key} \in \mathscr{F}_n^*$, add $(K, \text{Pseudonym}_{\mathscr{A}}, \text{Pseudonym}_B, \text{key})$ to L_2, and then return key.

(3) KeyGen queries: $d_{\mathscr{A}}$

Case 1: $(d_{\mathscr{A}}, \text{Pub}_{\mathscr{A}}, \text{Pseudonym}_{\mathscr{A}}) \in L_K$ and return $(d_{\mathscr{A}}, \text{Pub}_{\mathscr{A}}, \text{Pseudonym}_{\mathscr{A}})$.

Case 2: $(d_{\mathscr{A}}, \text{Pub}_{\mathscr{A}}, \text{Pseudonym}_{\mathscr{A}}) \notin L_K$, generate the public key $\text{Pub}_{\mathscr{A}}$ and the account address $\text{Pseudonym}_{\mathscr{A}}$ via equation (1) or (2), add $(d_{\mathscr{A}}, \text{Pub}_{\mathscr{A}}, \text{Pseudonym}_{\mathscr{A}})$ to L_K, and then return $(d_{\mathscr{A}}, \text{Pub}_{\mathscr{A}}, \text{Pseudonym}_{\mathscr{A}})$.

(4) Signcryption queries: $(\text{data}_{\mathscr{A}}, d_{\mathscr{A}}, \text{Pub}_B)$

(a) Choose random $k \in \mathscr{F}_n^*$.

(b) Compute $R = kP = (x_1, y_1)$.

(c) $\text{data}_{\mathscr{A}}^1 \| \text{data}_{\mathscr{A}}^2 \leftarrow \text{data}_{\mathscr{A}}$.

(d) Compute $r_{\mathscr{A}} = x_1 \bmod n$.

(e) Compute $K = k \cdot \text{Pub}_B$.

(f) Compute $\text{key} = H_2(K, \text{Pseudonym}_{\mathscr{A}}, \text{Pseudonym}_B)$, where $H_2(.)$ is from H_2 oracle.

(g) Compute $c = \text{data}_{\mathscr{A}}^2 \oplus \text{key}$.

(h) Compute $e = H_1(\text{data}_{\mathscr{A}}^1, \text{data}_{\mathscr{A}}^2)$ where $H_1(.)$ is from oracle H_1.

(i) Compute $s_{\mathscr{A}} = k^{-1}(e + d_{\mathscr{A}} r_{\mathscr{A}}) \bmod n$ where $d_{\mathscr{A}}$ is taken from the KeyGen queries.

(j) Add $(d_{\mathscr{A}}, \text{Pub}_B, \text{data}_{\mathscr{A}}^1, c, r_{\mathscr{A}}, s_{\mathscr{A}})$ to L_S.

(k) Return $\omega = \{\text{data}_{\mathscr{A}}^1, c, r_{\mathscr{A}}, s_{\mathscr{A}}\}$.

(5) Unsigncryption queries: $(\omega, d_B, \text{Pub}_{\mathscr{A}})$

(a) Find two points R and R' with the same value r as their x coordinate. The transaction practically involves an extra parameter to identify R and R', say R.

(b) Compute $K = d_B \cdot R$.

(c) Compute $\text{data}_{\mathscr{A}}^2 = c \oplus \text{key}$.

(d) Compute $e = H_1(\text{data}_{\mathscr{A}}^1, \text{data}_{\mathscr{A}}^2)$.

(e) Check whether $\text{Pub}_{\mathscr{A}} \stackrel{?}{=} r^{-1} \times (sR - e'P)$ holds or not.

(f) Return $\text{data}_{\mathscr{A}}$ and add $(\text{data}_{\mathscr{A}}, d_{\mathscr{A}}, \text{Pub}_B)$ to L_u; otherwise, return Reject.

(i) Output: after finishing phase 1, $\mathscr{A}$ produces the signature $\omega^* = \{\text{data}_{\mathscr{A}}^{1^*}, c^*, r_{\mathscr{A}}^*, s_{\mathscr{A}}^*\}$ along with the verifier $(d_B^*, \text{Pub}_B^*, \text{Pseudonym}_B^*)$ with probability ε based on Definition 2. If $\{d_{\mathscr{A}}^*, d_B^*\} \neq \{d_A, d_B\}$, $\mathscr{C}$will terminate the experiment. However, if $\{d_{\mathscr{A}}^*, d_B^*\} = \{d_A, d_B\}$ with probability $\left(1/\binom{q_K}{2}\right) = (2/q_K(q_K - 1))$, $\mathscr{C}$ picks the correct K from L_2 with probability $(1/q_2)$ to recover $\text{data}_A^{2^*}$. Finally, $\mathscr{C}$ can compute $k^* = (s^*)^{-1}(H_1(\text{data}_A^{1^*}, \text{data}_A^{2^*}) + d_A^* r^*)$ with the probability $(2\varepsilon/q_2 q_K(q_K - 1))$ to solve the ECDLP problem. This is contradictory to our hypothesis at the beginning of the security proof. Thus, this concludes the proof of Theorem 2.

Considering the strength of the keys, which is one of the primary factors of cryptographic algorithms, the security analysis of the key strength is further illustrated. There are three state-of-the-art algorithms for hard problems including integer factorization (such as RSA [35]), the discrete logarithm (such as DSA [35]), and the elliptic curve discrete logarithm (such as ECDSA). It is well known that the ECDSA achieves not only the same level of security with a smaller key size but also higher computational efficiency than those of the RSA and DSA. For example, ECC-256 (resp. ECC-224) provides comparable security to RSA-3072 (resp. RSA-2048) [36]. By employing the ECDSA over the standard elliptic curve "secp256k1," which Ethereum and Bitcoin blockchains have adopted, the proposed scheme also provides the same level of security. □

5. Performance Analysis

In this section, the performance of the proposed BaaCA-based signcryption scheme is analyzed by comparing the computational costs with those of other related works and the experimental results of the operational costs based on the Ethereum blockchain.

5.1. Computational Cost of Signcryption. The computational costs of the proposed scheme mainly come from two phases: the signcryption and unsigncryption phases. The operations executed in each phase are depicted in Table 2. In the signcryption phase, Alice will execute two elliptic curve scalar multiplication operations, one map-to-point hash function, and one modular inversion. In the unsigncryption phase, Bob will execute two elliptic curve scalar multiplication operations, one map-to-point hash function, and one modular inversion. It is noted that the proposed scheme does not adopt any pairing operation.

Table 2: Comparisons between related signcryption schemes and the proposed scheme.

Schemes	PKI-/ID-/ certificateless-based	Hardness assumption	High maintenance cost of CAs or a TTP	Seamless compliance with blockchains	Computational cost		
					Signcryption	Unsigncryption	Total cost
Malone-Lee [5]	ID-based	BDH	Yes (TTP)	No	T_P	$4T_P$	435.0 t_m
Karati et al. [4]	ID-based	BDH	Yes (TTP)	No	$3T_E + T_P$	$2T_E + 2T_P + T_i$	377.6 t_m
Zhou et al. [13]	Certificateless-based	BDH	Yes (TTP)	No	$8T_E + 4T_P$	$2T_E + 8T_P$	1254.0 t_m
Karati et al. [16]	Certificateless-based	BDH	Yes (TTP)	No	$3T_E$	$2T_E + 2T_P + 2T_i$	302.2 t_m
Rastegari et al. [17]	Certificateless-based	BDH	Yes (TTP)	No	$4T_E + 2T_P$	$2T_E + 8T_P$	996.0 t_m
ECDSA + ECC	PKI-based	ECDLP	Yes (CA)	No	$(T_M + T_i) + (2T_M + T_H)$	$(2T_M + T_i) + (T_M + T_H)$	243.2 t_m
The proposed scheme	Blockchain as a CA	ECDLP	No	Yes	$2T_M + T_i + T_H$	$3T_M + T_i + T_H$	214.2 t_m

The proposed scheme is compared with several existing related works. The computational cost of primitive time-consuming cryptographic operations is adopted from [4, 37] and is summarized as follows. $T_P \approx 87t_m$, $T_M \approx 29t_m$, $T_H \approx 23t_m$, $T_E \approx 21t_m$, and $T_i \approx 11.6t_m$ are the times required to execute a bilinear pairing operation, an elliptic curve scalar point multiplication, a map-to-point hash function, an exponentiation, and a modular inversion, respectively, where t_m is the time required to execute a scalar multiplication in $\mathcal{F}_n^*$.

The total computational cost required in the proposed scheme is $(5T_M + 2T_i + 2T_H) = 214.2\ t_m$. Table 2 shows that the computational cost of the proposed BaaCA-based signcryption scheme is nearly 49% that of Malone-Lee [5], 57% that of Karati et al. [4], 17% that of Zhou et al. [13], 71% that of Karati et al. [16], 22% that of Rastegari et al. [17], and 88% that of ECDSA + ECC.

Figure 4 shows the computational overhead during signcryption and unsigncryption among the proposed scheme and the related works.

5.2. Experimental Results Based on the Ethereum Blockchain. To prove the computational cost of the BaaCA-based signcryption scheme compared with the ECDSA in the original transaction on the Ethereum blockchain, we implement the proposed scheme using the Python programming language and the Ethereum Ropsten Testnet environments [38]. All experiments are carried out with the following settings:

(1) CPU: Intel Core i5-4440 CPU @ 3.1 GHz (quad core).

(2) Physical memory: 16 GB DDR3 1600 MHz.

(3) OS: Windows 7.

Finally, the proposed scheme is executed 500 times and its average is taken as our experimental result. Figure 5 shows the computational costs of the signcryption and unsigncryption phases. We observe that the cost of the unsigncryption phase is higher than that of the signcryption phase. This is a reasonable result since unsigncryption executes one more elliptic curve scalar point multiplication operation, as shown in Table 2. Moreover, the interval among the test data size is set as 5 kB, and the ratio of the sensitive part to the public part is 1 : 4.

In addition to the computational cost, the proposed scheme is also evaluated in terms of its operational cost in real currency. In the proposed scheme, one transaction only is broadcasted into the blockchain in the signcryption phase. The operational cost is estimated by calculating the total amount of gas and then converting it into the real currency. We use the Ethereum Ropsten Testnet environment [38] since it is able to automatically calculate the amount of gas of the proposed scheme. After that, the amount of gas is converted into USD according to the CoinGecko conversion table [39]. At the time of inversion, the rate was 1 gas = 0.000000001 ETH = 1.65×10^{-7} USD.

Figure 6 shows the operational costs for different amounts of data. It is observed that the cost increases when the amount of data increases. It is reasonable since more data will lead to greater amounts of gas. However, it seems useless for the result since the cost remains very low. Even if the amount of data increases to 30 kB, the cost in the real currency is still less than 0.35 USD. That means that the proposed scheme is a suitable solution to protect transaction privacy and it only requires a small cost as a fee.

6. Discussion

Since the security model has been formally analyzed in Section 3, the main advantages of the proposed scheme compared with related works will be given in this section. The comparisons between the previous related works and the proposed method are given in Table 2. It is obvious that the superiority of the proposed scheme is demonstrated by achieving the following advantages:

(1) Removing the high maintenance cost of involving CAs or a TTP: without having a CA or TTP to

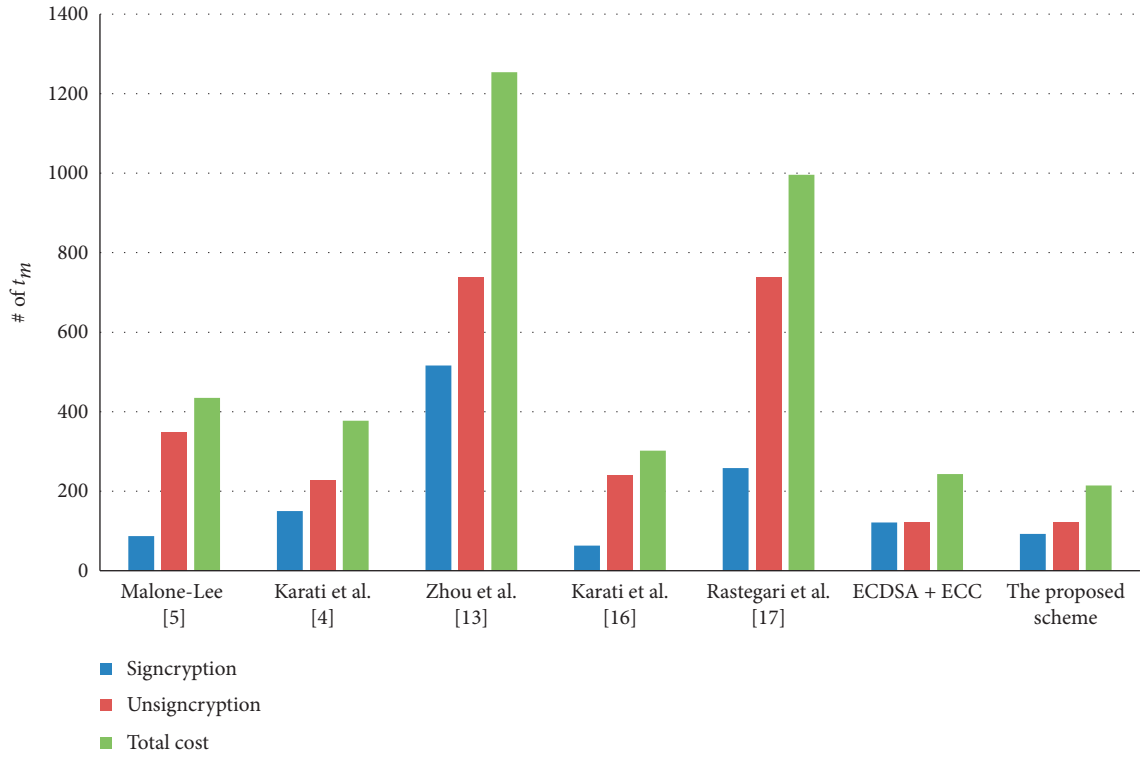

Figure 4: Performance comparisons among the proposed and the related signcryption schemes in terms of signcryption, unsigncryption, and total costs.

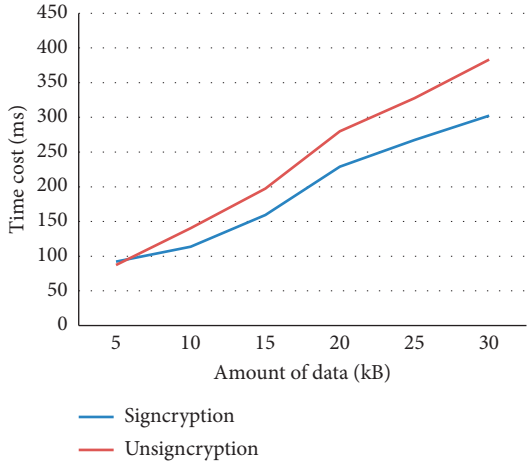

Figure 5: Computational costs in the proposed scheme with different amounts of data.

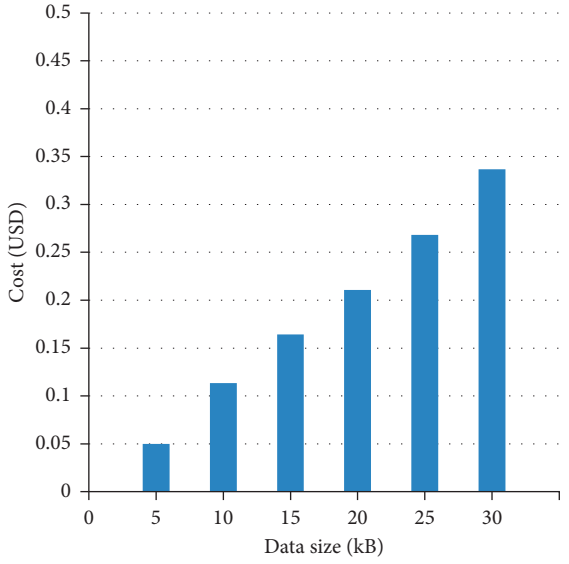

Figure 6: The cost of the proposed scheme in USD.

register a user in the proposed scheme, key pairs are generated by users themselves. It is clear that with the concepts of *blockchain as a CA* and *transaction as a certificate*, users' public keys are extracted and authenticated directly from transactions in blockchains instead of by CAs or a TTP. Since blockchains such as Bitcoin and Ethereum have no mechanism to publish users' public keys, the public key extraction operation in the proposed scheme can achieve the goal of the settlement and the authentication of public keys from transactions directly.

(2) Seamless compliance with blockchains: it is not necessary to maintain a KGC. In the proposed scheme, the signcryption scheme can exploit the ECDSA private/public keys without any modification of blockchains. Furthermore, the encryption key used for confidentiality is deduced under the Diffie–Hellman key exchange [40] such that both the prover and the verifier can obtain an identical secret key. In the signcryption phase, Alice's transaction is

$Tx_A = \{\text{data}_A, r_A, s_A\}$, where data_A are composed of public data_A^1 and privacy-sensitive data_A^2. The field "input" in Figure 3, which could be an arbitrary message, can be treated as a privacy-sensitive message while the rest of the fields are treated as public information. Precisely, the "input" field is the ciphertext instead (not the plaintext) in the proposed BaaCA-based scheme. This implies that Alice performs the regular operations of generating a common transaction of blockchains except for the additional encryption-related operations. In this way, the implementation of the proposed scheme affects the "input" field, and so the impact must be minimized.

Furthermore, the revisited applicability of the proposed scheme achieves the following advantages:

(1) Preserving transaction confidentiality: the proposed scheme lowers the payload for only encrypting the privacy-sensitive part such that the other part of a transaction may remain as plaintext. Thus, the scheme is compliant with the design of blockchains and protects transaction confidentiality at the same time.

(2) Reducing computational and communication costs: as we know, a signcryption scheme is proven to achieve lower computational and communication costs [1]. Thus, the proposed blockchain-compliant signcryption scheme benefits from the advantages of confidentiality, integrity, and nonrepudiation simultaneously.

(3) Designated recipient: the proposed signcryption scheme sets a designated recipient. When trying to verify signature (r_A, s_A), a recipient must check whether Pub_A' is equal to $r_A^{-1} \times (s_A R_A - e'P)$. Without knowing e', the recipient cannot accomplish this task. Via Theorem 1, attackers have no feasible way to deduce the decryption key to compute data_A^2 and then obtain $e' = H_1(\text{data}_A^1, \text{data}_A^2)$. Thus, the verification of (r_A, s_A) must be performed by the designated verifier. Alternatively, if $e' = H_1(\text{data}_A^1, c)$ in the signcryption phase, every participant involved in the blockchain can be the verifier.

7. Conclusions

As mentioned above, the cryptographic advantages of a blockchain are similar to those of a certificate chain such as providing public verification and being tamper-proof. The main difference is that the blockchain is a decentralized trusty mechanism requiring no trusted third party; however, the certificate chain is a centralized trust mechanism with a series of trusted third parties. Because of the success of Bitcoin and Ethereum along with the promising blockchain technologies, it is possible and feasible to combine blockchain technologies with signcryption. Thus, the concept of blockchain as a CA proposed in this paper aims to skillfully leverage the blockchain and achieves the following advantages: (1) it removes the need to involve CAs or a TTP, (2) it seamlessly complies with blockchains, and (3) it preserves transaction privacy.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This study was supported in part by the Ministry of Science and Technology of Taiwan under grant nos. MOST 107-2221-E-415-001-MY3.

References

[1] Y. L. Zheng, "Digital signcryption or how to achieve cost (signature & encryption) << cost(signature) plus cost (encryption)," in *Proceedings of the 17th Annual International Cryptology Conference*, pp. 165–179, Santa Barbara, CA, USA, August 1997.

[2] J. Baek, R. Steinfeld, and Y. Zheng, "Formal proofs for the security of signcryption," *Journal of Cryptology*, vol. 20, no. 2, pp. 203–235, 2007.

[3] F. Li and T. Takagi, "Secure identity-based signcryption in the standard model," *Mathematical and Computer Modelling*, vol. 57, no. 11-12, pp. 2685–2694, 2013.

[4] A. Karati, S. H. Islam, G. P. Biswas, M. Z. A. Bhuiyan, P. Vijayakumar, and M. Karuppiah, "Provably secure identity-based signcryption scheme for crowdsourced industrial Internet of Things environments," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2904–2914, 2017.

[5] J. Malone-Lee, "Identity-based signcryption," IACR Cryptol. ePrint Arch., 2002.

[6] G. Wei, J. Shao, Y. Xiang, P. Zhu, and R. Lu, "Obtain confidentiality or/and authenticity in big data by id-based generalized signcryption," *Information Sciences*, vol. 318, pp. 111–122, 2015.

[7] Y. Yu, B. Yang, Y. Sun, and S.-l. Zhu, "Identity based signcryption scheme without random oracles," *Computer Standards & Interfaces*, vol. 31, no. 1, pp. 56–62, 2009.

[8] M. Barbosa and P. Farshim, "Certificateless signcryption," in *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security*, pp. 369–372, Tokyo, Japan, March 2008.

[9] F. Li, J. Hong, and A. A. Omala, "Efficient certificateless access control for industrial Internet of Things," *Future Generation Computer Systems*, vol. 76, pp. 285–292, 2017.

[10] F. Li, M. Shirase, and T. Takagi, "Certificateless hybrid signcryption," *Mathematical and Computer Modelling*, vol. 57, no. 3-4, pp. 324–343, 2013.

[11] Z. Liu, Y. Hu, X. Zhang, and H. Ma, "Certificateless signcryption scheme in the standard model," *Information Sciences*, vol. 180, no. 3, pp. 452–464, 2010.

[12] A. Yin and H. Liang, "On security of a certificateless hybrid signcryption scheme," *Wireless Personal Communications*, vol. 85, no. 4, pp. 1727–1739, 2015.

[13] C. Zhou, G. Gao, and Z. Cui, "Certificateless signcryption in the standard model," *Wireless Personal Communications*, vol. 92, no. 2, pp. 495–513, 2017.

[14] C. Zhou, W. Zhou, and X. Dong, "Provable certificateless generalized signcryption scheme," *Designs, Codes and Cryptography*, vol. 71, no. 2, pp. 331–346, 2014.

[15] M. Luo and Y. Wan, "An enhanced certificateless signcryption in the standard model," *Wireless Personal Communications*, vol. 98, no. 3, pp. 2693–2709, 2018.

[16] A. Karati, C.-I. Fan, and R.-H. Hsu, "Provably secure and generalized signcryption with public verifiability for secure data transmission between resource-constrained IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10431–10440, 2019.

[17] P. Rastegari, W. Susilo, and M. Dakhlalian, "Efficient certificateless signcryption in the standard model: revisiting Luo and Wan's scheme from wireless personal communications," *The Computer Journal*, vol. 62, no. 8, pp. 1178–1193, 2019.

[18] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Workshop on the Theory and Application of Cryptographic Techniques*Springer, Berlin, Heidelberg, 1984.

[19] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing,," in *Proceedings of the Annual International Cryptology Conference*, pp. 213–229, Santa Barbara, CA, USA, August 2001.

[20] S. S. Al-Riyami and K. G. Paterso, "Certificateless public key cryptography," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 452–473, Springer, Berlin, Heidelberg, 2003.

[21] S. Nakamoto, *Bitcoin: A Peer-To-Peer Electronic Cash System*, 2008.

[22] V. Buterin, *Ethereum White Paper*, 2013.

[23] C. Cachin, "Architecture of the Hyperledger blockchain fabric," in *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016.

[24] Q. Xia, E. B. Sifah, K. O. Asamoah et al., "MeDShare: trust-less medical data sharing among cloud service providers via blockchain," *IEEE Access*, vol. 5, pp. 14757–14767, 2017.

[25] J. Sun, X. Yao, S. Wang, and Y. Wu, "Blockchain-based secure storage and access scheme for electronic medical records in IPFS," *IEEE Access*, vol. 8, pp. 59389–59401, 2020.

[26] N. Garg, M. Wazid, A. K. Das et al., "BAKMP-IoMT: design of blockchain enabled authenticated key management protocol for Internet of medical Things deployment," *IEEE Access*, vol. 8, pp. 95956–95977, 2020.

[27] A. Gauhar, N. Ahmad, Y. Cao et al., "xDBAuth: blockchain based cross domain authentication and authorization framework for Internet of Things," *IEEE Access*, vol. 8, pp. 58800–58816, 2020.

[28] M. Wazid, A. K. Das, S. Shetty, and M. Jo, "A tutorial and future research for building a blockchain-based secure communication scheme for Internet of intelligent Things," *IEEE Access*, vol. 8, pp. 88700–88716, 2020.

[29] X. Li, Y. Mei, J. Gong, F. Xiang, and Z. Sun, "A blockchain privacy protection scheme based on ring signature," *IEEE Access*, vol. 8, pp. 76765–76772, 2020.

[30] B. Ernest and J. Shiguang, "Privacy enhancement scheme (PES) in a blockchain-edge computing environment," *IEEE Access*, vol. 8, pp. 25863–25876, 2020.

[31] Q. Wang, T. Ji, Y. Guo, L. Yu, X. Chen, and P. Li, "TrafficChain: a blockchain-based secure and privacy-preserving traffic map," *IEEE Access*, vol. 8, pp. 60598–60612, 2020.

[32] Wiki, "Elliptic curve digital signature algorithm," https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm.

[33] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, p. 203, 1987.

[34] D. R. L. Brown, "SEC 2: recommended elliptic curve domain parameters," *Standards for Efficient Cryptography Group (SECG)*, http://www.secg.org/sec2-v2.pdf, 2010.

[35] B. Schneier, *Applied Cryptography*, Wiley, New York, NY, USA, 2nd edition, 1996.

[36] T. M. Fernández-Caramès and P. Fraga-Lamas, "Towards post-quantum blockchain: a review on blockchain cryptography resistant to quantum computing attacks," *IEEE Access*, vol. 8, pp. 21091–21116, 2020.

[37] A. Karati, S. Hafizul Islam, and G. P. Biswas, "A pairing-free and provably secure certificateless signature scheme," *Information Sciences*, vol. 450, pp. 378–391, 2018.

[38] TESTNET Ropsten (ETH) Blockchain Explorer, https://ropsten.etherscan.io/.

[39] CoinGecko:360° Market Overview of Coins & Cryptocurrency, https://www.coingecko.com.

[40] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.