논문 24-49-01-14 The Journal of Korean Institute of Communications and Information Sciences '24-01 Vol.49 No.01
https://doi.org/10.7840/kics.2024.49.1.113

Blockchain-Based Data Auditing Protocol with Signcryption Scheme in Cloud Storage

Elizabeth Nathania Witanto,◆ Sang-Gon Lee°

ABSTRACT

When users store data in cloud service providers (CSPs), they lack full control over what happens to their data. CSP may unintentionally remove rarely accessed users' data or encounter internal or external issues that compromise data integrity. To guarantee the integrity of users' data stored in cloud storage, an auditing protocol is necessary. However, considering the massive amount of stored data, it is inefficient for the verifier to download all the data in advance. Subsequently, the verifier might not have proper resources and expose the user's data privacy to other parties. Therefore, this paper proposes a blockchain-based data auditing protocol with a signcryption scheme in cloud storage. Smart contracts in a blockchain network are used to solve trust issues between the user, CSP, and the verifier. At the same time, the signcryption scheme is used to offer privacy-preserving during data transmission also, public verifiability, and blockless verification. The evaluation and security analysis show that the proposed protocol offers low computational and communication costs to do auditing tasks while offering desirable properties such as public verifiability, privacy-preserving, blockless verification, and reliable verifier.

Key Words : blockchain, cloud storage, data auditing, digital signature, encryption, signcryption

Ⅰ. Introduction

With various organizations' increasing adoption of cloud computing, ensuring data integrity has emerged as a critical concern. Cloud Service Providers (CSPs) offer users convenient options for storage and processing, but they simultaneously pose a noteworthy threat to the integrity of user data. As the users host their data to the cloud storage, they cannot monitor their data integrity continuously. Furthermore, CSP might face internal or external hazards leading to a breach in the security of stored data.

Data integrity in cloud storage refers to data reliability. Data integrity violations are also termed breaches or tampering with users' data that can cause critical issues to data trust. In fact, tampering with data can go unnoticed and fuel malevolent actions by removing individual entries (i.e., eliminating unwanted traces) or modifying specific chunks of data (i.e., affecting the behavior of data consumers)[1]. Kaspersky Lab discovered a large cyber-attack that stole money from account balances at more than 100 financial institutions globally, with an estimated worth of almost $1 billion[2]. Due to the fact that integrity violations are difficult to detect and highly effective, it make clear how crucial it is to provide data integrity auditing protocol in cloud computing.

In the traditional auditing protocol, the user utilizes a Third-Party Auditor (TPA) to become a verifier that verifies data integrity stored in cloud storage[3]. Because stored data may be enormous, it is impractical and inefficient for the verifier to

◆ First Author : Ciputra University, Surabaya, Indonesia, Department of Information Technology, elizabeth.nathania@ciputra.ac.id, 학생회원
° Corresponding Author : Dongseo University, College of Software Convergence, nok60@dongseo.ac.kr, 정회원
논문번호 : 202306-035-A-RN, Received June 28, 2023; Revised September 22, 2023; Accepted October 2, 2023

download the user's data beforehand during data auditing. Further, the verifier might not have sufficient computation resources. More importantly, there is no assurance of neutral data auditing, and thus endangers users' privacy. Numerous research studies have been carried out to offer auditing protocols for cloud systems. Refs. [4,5] proposed an identity-based signcryption, and Ref. [6] proposed a signcryption scheme based on elliptic curve. Unfortunately, those signcryption schemes did not provide important properties needed in data auditing protocol: public verifiability, privacy-preserving, and blockless verification. Refs. [7-10] delivers a public verifiability property to their works. Nevertheless, it lacks privacy-preserving and blockless verification in the auditing process. The user must forward the original data to the verifier, threatening the user's data privacy. It means the verifier must download the data first, which is counterproductive. The privacy of the user's data is threatened since the user must send the original data to the verifier.

To tackle this difficulty, this paper suggests a protocol for auditing cloud data using blockchain and a signcryption method. Our approach utilizes the reliability and transparency of blockchain technology to empower users to authenticate their data's consistency, which is saved in cloud-based storage. Through the utilization of signcryption, this protocol guarantees not only encryption but also authentication of data prior to transmission into the cloud. This feature serves as an extra level of security for safeguarding sensitive information.

The remainder of the paper is organized as follows. This paper first describes preliminaries in Section Ⅱ. Then, this paper presents the proposed auditing protocol in Section Ⅲ. Discussion and security analysis about the proposed scheme equations' correctness, unforgeability from malicious CSP and verifier, computational and communication cost, and comparative analysis of the proposed scheme will be presented in Section Ⅳ. Finally, Section Ⅴ presents the paper's conclusion.

Ⅱ. Preliminaries

2.1 Bilinear Pairings

Let G_1 be a cyclic additive group, and G_2 be multiplicative cyclic groups with prime order p, P is the generator of the group G_1. The mapping $e: G_1 \times G_2 \rightarrow G_2$ is a bilinear map with the following properties:

1. **Bilinearity:** $e(aP, bQ) = e(P, Q)^{ab}$, and $e(P + R, Q) = e(P, Q) \cdot e(R, Q)\ \forall P, Q, R \in G_1$, $a, b \in Zp$.
2. **Computability:** There is an efficient algorithm to compute $e(P, Q)\ \forall P, Q \in G_1$.
3. **Non-degeneracy:** There exists $P \in G_1$ such that $e(P, P) \neq 1$.

This paper considers the following problems in the additive group G_1.

- **Discrete Logarithm Problem (DLP):** Given two group elements P and Q, find an integer $n \in Z_p^*$, such that $Q = nP$ whenever such an integer exists.
- **Computational Diffie-Hellman Problem (CDHP):** For $a, b \in Z_p^*$, given P, aP, bP, compute abP.

There are two variants of CDHP:

- **Inverse Computational Diffie-Hellman Problem (Inv-CDHP):** For $a \in Z_p^*$, given P, aP, compute $a^{-1}P$.
- **Square Computational Diffie-Hellman Problem (Squ-CDHP):** For $a \in Z_p^*$, given P, aP, compute a^2P.

2.2 ZSS Signcryption

C. Ma[9] proposed a signcryption scheme that provides public verifiability for data auditing.

- **ComGen**. Given the security parameters k and n. Two cyclic groups $(G_1, +)$ and $(G_2, \cdot)$ of the same prime order $p > 2^k$, a generator P of

G_1, a bilinear map $e : G_1 \times G_1 \rightarrow G_2$, three hash functions $H_1 : \{0, 1\}^* \rightarrow Z_p$, $H_2 : G_1^3 \rightarrow \{0, 1\}^n$, and $H_3 : \{0, 1\}^k$, and an symmetric encryption scheme (E, D). Then, $I = \{k, n, G_1, G_2, P, e, H_1, H_2, H_3, E, D\}$.

- **KeyGen**. Every user picks his private key SK_U from Z_p^* randomly and uniformly. Then, he computes his public key $PK_U = SKU_P$.
- **Signcrypt**. Given a message $m \in \{0, 1\}^*$, the recipient's public key PK_R and the sender's private key SK_S. The sender computes:
  - pick $r \overset{R}{\leftarrow} \{0, 1\}^n$ and compute $u = (H_1(m) + SK_S + r)^{-1} \ mod \ p$.
  - compute $U = uP \in G_1, V = r \oplus H_2(U, PK_R, uPK_R)$ and then $W = E_\kappa \ (m//PK_S)$ where $\kappa = H_3(r)$.

Finally, form the signcryptext $C = (U, V, W)$.

- **Unsigncrypt** by recipient upon receiving (U, V, W).
  - parse C as (U, V, W) and compute $r = V \oplus H_2(U, PK_R, SK_RU)$.
  - compute $m//PK_S = D_\kappa \ (W)$ where $\kappa = H_3(r)$.
  - if $e(U, (H_1(m) + r)P + PK_S) = e(P, P)$, then return the message m; otherwise return $\perp$ means unsigncryption failure.
- **Public Verifiability**. The recipient wants to prove that the sender signcrypted a message m to the trusted third party (TTP). So, the recipient forwards (m, U, r, PK_S) to the TTP. Then, TTP accepts the proof if this equation is valid $e(U, (H_1(m) + r)P + PKS) = e(P, P)$.

ZSS signcryption by [9] gives the public verifiability property. Unfortunately, it lacks three other desirable properties, privacy-preserving, blockless verification, and reliable verifier. The user must send the original message m to the TTP first. As a result, the TTP might obtain user information. Second, the author added the random value r in the ZSS signature generation. As a result, the user must also provide the TTP with the r value. The TTP is harmful because of its ability to construct a symmetric encryption key, κ, using $H_3(r)$. Thus, the TTP might decrypt the original message m, threatening data privacy. Third, it is inefficient for the verifier to download message m for data auditing, especially if message m is a sizable file. Therefore, this paper proposed an improved version of the ZSS signcryption scheme that provides four properties in data auditing, public verifiability, privacy-preserving, blockless verification, and reliable verifier.

Ⅲ. Proposed Scheme

This section explains a blockchain-based data auditing protocol with a ZSS signcryption scheme for cloud storage. This paper adopted the signcryption from [9]. In [9], the author also gives a public verifiability property. However, his work lacks privacy-preserving, blockless verification, and a reliable verifier during auditing because the user needs to deliver the original message m to the verifier. By doing so, users' data privacy would be disclosed to the verifier. Therefore, this paper gives three additional advantages besides public verifiability important for data auditing protocol: privacy-preserving, blockless verification, and

Table 1. List of notations.

Notation	Description
b, d	Security parameter
G_1	Cyclic additive group
G_2	Multiplicative cyclic group
P	Generator of group G1
m	Original message
n	Total number of shards
V	ZSS signature of message m
X	Randomness of encryption key generation
Y	Encryption of message m
SK_S	Sender's secret key
PK_S	Sender's public key
SK_R	Recipient's secret key
PK_R	Recipient's public key
Enc	Symmetric encryption operation
Dec	Symmetric decryption operation

reliable verifier.

This paper picked the ZSS signature because it required less pairing operation than other short signatures, such as the BLS signature[11]. Furthermore, ZSS does not need a particular hash function, i.e., MapToPoint, used in BLS. Users can use a general hash function such as SHA family or MD5[11]. In the proposed protocol, a user will store a file in CSP. First, he will compute a Signcrypted data σ. Then, he will send σ to the CSP. Before CSP stores data in their storage, it will unsigncrypt the σ. This paper provides a list of notations used for the remainder of this chapter in Table 1 and describes the detailed process as follows.

3.1 Setup Phase

- **ParamGen.** Given security parameter b and d. Let G_1 be a cyclic additive group, G_2 be multiplicative cyclic groups with prime order p, and P be the generator of group G_1. The bilinear mapping $e : G_1 \times G_2 \rightarrow G_2$, three hash functions $Hash_1 : \{0, 1\}^* \rightarrow Z_p$, $Hash_2 : G^{3}_{1} \rightarrow \{0, 1\}^d$ and $Hash_3 : \{0, 1\}^d \rightarrow \{0, 1\}^b$, and symmetric encryption scheme (Enc, Dec). Therefore the system parameters are $\{b, d, G_1, G_2, P, e, Hash_1, Hash_2, Hash_3, Enc, Dec\}$.
- **KeyGen.** Sender chooses a random number from Z_p, sets it as his secret key SK_S, and computes his public key $PK_S = SK_SP$. The recipient chooses a random number from Z_p, sets it as his secret key SK_R, and computes his public key $P_KR = SK_RP$.
- **Signcryption.** User as sender S generated a signcryption σ for each message m as follows.
  1. Generates $v = (Hash_1(m) + SK_S)^{-1}$.
  2. Choose $r \xleftarrow{R} \{0, 1\}^d$ and generates $V = vP$, $X = r \oplus Hash_2(V, PK_R, vPK_R)$. Then, generates $k = Hash_3(r)$, a symmetric encryption key. So, $Y = Enc_k(m)$.
  3. Therefore the signcryption $\sigma = (V,X,Y)$ where V is a ZSS signature of message m, X is the randomness of encryption key generation, and Y is the encryption of message m.
- **Unsigncryption.** After receiving $\sigma = (V, X,Y)$ from the sender S, the recipient R start unsigncryption process.
  1. R parse σ to get (V, X,Y).
  2. Computes $r = X \oplus Hash_2(V, PK_R, VSK_R)$ and $k = Hash_3(r)$.
  3. Decrypt Y to get message m. So, $m = Dec_k(Y)$.
  4. If the Equation (1) holds, unsigncryption success; otherwise, R rejects σ from the sender.

$$e(Hash_1(m)P + PK_S, V) = e(P, P) \tag{1}$$

To add randomization to the signature generation process, the author of [9] inserts a random variable r. The consequence is the user must provide the verifier the r. As explained earlier, this presents a concern since r can create a symmetric encryption key using the $Hash_3(r)$ function. In this way, the verifier threatens data privacy by being able to decrypt the original message. Different from [9], by removing the r, the proposed scheme prevents leakage of the encryption key to the verifier. Furthermore, the original message m will not be passed to the verifier during auditing in the proposed scheme. So, it assures data privacy and increases verification efficiency because the verifier does not need to download the original message m beforehand.

3.2 Store File Phase

This phase explains storing the user's data in the CSP. The user and CSP are involved, as shown in Fig. 1. This paper presents details as follows.

1. User divided data F into n shards of m. $F = \{m_1, m_2, m_3, ..., m_n\}$. The user does a signcryption process for each data shard and generates σ_i, where i is the index of each data shard. Therefore the Signcryption $\sigma_i = (V_i, X_i, Y_i)$. Set of signcrypted data is $Sign = \{\sigma_1, \sigma_2, \sigma_3, ..., \sigma_n\}$. Then, the user stores a set of signcryption $Sign$ to the CSP.
2. User stores set of Signature $Sn = \{V_1, V_2, V_3,$

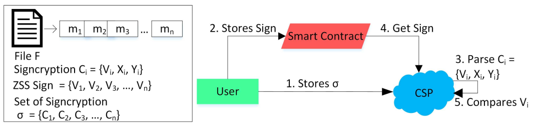
m1 m2 m3 ... mn
File F
Signcryption Ci = {Vi, Xi, Yi}
ZSS Sign = {V1, V2, V3, ..., Vn}
Set of Signcryption
σ = {C1, C2, C3, ..., Cn}
2. Stores Sign
Smart Contract
4. Get Sign
3. Parse Ci = {Vi, Xi, Yi}
User
1. Stores σ
CSP
5. Compares Vi

Fig. 1. Store File Scheme.

..., V_n} to the smart contract.

3. CSP verify data from user by parsing each σ_i to (V_i, X_i, Y_i).
4. CSP get *Sign* from the smart contract.
5. Then CSP does the unsigncryption process and compares V_i from the smart contract and the user. If equal, store the user's data; otherwise, reject it.

3.3 Data Auditing Phase

In the data auditing phase, three entities are involved: user, CSP, and verifier, as shown in Fig. 2. The user has stored data in the CSP storage. Then, he wants to check his data integrity by publishing an auditing task to the smart contract. Then, other users that joined the blockchain network can be the verifier by applying to the smart contract. The appointed user becomes the verifier for the corresponding auditing task. Then, the verifier generates a challenge variable and sends it to the corresponding CSP. After receiving the challenge from the verifier, CSP generates proof and sends it to the verifier. Next, the verifier will verify whether the proof given by CSP is correct through the validity of an equation. The details are described as follows.

1. When the user wants to verify his stored data in CSP, he publishes an auditing task to the smart contract. This task will be broadcast so anyone who joins the blockchain network can get this notification.

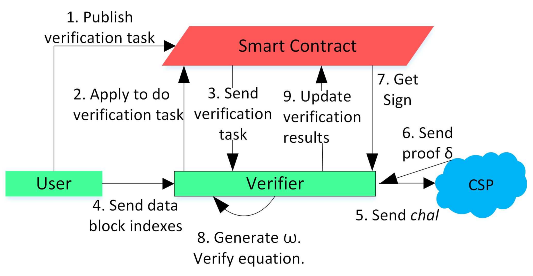
1. Publish verification task
Smart Contract
2. Apply to do verification task
3. Send verification task
9. Update verification results
7. Get Sign
6. Send proof δ
User
Verifier
CSP
4. Send data block indexes
8. Generate ω. Verify equation.
5. Send chal

Fig. 2. Data Auditing Protocol.

2. Other users who want to become verifiers apply to a smart contract for the corresponding auditing task.
3. Smart contract will check the credibility points and current deposit of the applied verifier. It will choose a user with high credibility points and send the file that will be verified to the verifier.
4. Then user generates a set of random numbers $\{i\}_{i \in I}$ for the auditing task, where i is the index of stored challenged data shard. Then, he sends those random numbers and also the set of Signature of the challenged data shards $Sn = \{V_i\}_{i \in I}$ to the verifier.
5. Upon receiving request from user, verifier choose randomly f_i from Z_p and generates a challenge $chal = \{i, f_i\}_{i \in I}$. Then send $chal$ to the corresponding CSP.
6. After receiving $chal$ from verifier, CSP computes $r_i = X_i \oplus Hash_2(V_i, PK_R, V_i SK_R)$. Then he can computes $k_i = Hash_3(r_i)$ to decrypt Y_i. Therefore $m_i = Dec_{ki}(Y_i)$. After that, CSP generates proof δ and sends it to the verifier.

$$\delta = \prod_{i \in I} f_i Hash_1(m_i)P + PK_S \tag{2}$$

7. Verifier get the *Sign* from the smart contract.
8. Then the verifier generates proof ω.

$$\omega = \prod_{i \in I} V_i f_i^{-1} \tag{3}$$

Subsequently, the verifier checks whether Equation (4) holds. If it holds, CSP successfully proves the user's data integrity; otherwise, CSP failed, and the auditing task also failed.

$$e(\delta, \omega) = e(P, P) \tag{4}$$

9. Finally, the verifier reports the auditing results to the smart contract. The smart contract will

broadcast the result to the network and update the credibility points and balance for the corresponding auditing task's participants.

3.4 Credibility Points and Incentive Mechanisms

This paper aims to attract verifiers to participate in auditing tasks by proposing credibility points and incentive mechanisms. This mechanism also could prevent arbitrary behavior by penalizing its actors who do not comply with the rules.

- Users, verifiers, and CSPs must register themselves to the smart contract. To participate in the process, users, verifiers, and CSPs need to stake/ make a deposit as a guarantee. In the case of dishonest/malicious behavior, their deposit will be decreased.
- They will be given initial credibility points. This point shows the credibility of each actor. The higher the points means, the actor more credible and trustable.
- Each actor that does their job faithfully will be given additional points and incentives as a reward; otherwise, they will get penalty points and deduct their deposit.
- In the case of a user, if the user does something malicious, for example, deceiving the verifier/ CSP, the user will get a penalty by decreasing the credibility points and deposit.
- In the case of the verifier, if the verifier does their job faithfully during the auditing task, they will get additional points. Otherwise, their points will be decreased.
- CSPs also will be given credit points that show their credibility. If CSPs fails to prove users' request, their credit points will decrease.
- If their deposit is 0, they cannot participate in any process. They will get suspended for several amounts of time.

Ⅳ. Evaluation and Security Analysis

4.1 Correctness

Below is the correctness proof of the signcryption scheme shown in Equation (1).

$$
\begin{aligned}
& \mathrm{e}(\mathrm{Hash}_1(m_i)P + PK_S, V) \\
&= e(Hash_1(m_i)P + PK_S, (Hash_1(m_i) + SK_S)^{-1}P) \\
&= e((Hash_1(m_i) + SK_S)P,\ (Hash_1(m_i) + SK_S)^{-1}P) \\
&= e(P,\ P)^{(Hash_1(m_i)+SK_S)\,\bullet\,(Hash_1(m_i)+SK_S)^{-1}} \\
&= e(P,\ P)
\end{aligned}
$$

Below is the correctness proof of the data auditing protocol shown in Equation (4).

$$
\begin{aligned}
& \mathrm{e}(\delta,\ \omega) \\
&= e(\Pi_{i\in I}\ f_i Hash_1(m_i)P + PK_S,\ \Pi_{i\in I}\ V_i f_i^{-1}) \\
&= e(\Pi_{i\in I}\ fi(Hash_1(m_i) + SK_S)P,\ \Pi_{i\in I}(f_i(Hash_1(m_i) \\
&\quad + SK_S))^{-1}P) \\
&= e(P,\ P)^{\Pi_{i\in I}\ f_i(Hash_1(m_i)+SK_S)\,\bullet\,\Pi_{i\in I}(f_i(Hash_1(m_i)+SK_S))^{-1}} \\
&= e(P,\ P)
\end{aligned}
$$

4.2 Unforgeability

This paper presents an unforgeability of two cases in the proposed protocol, malicious CSP cannot forge proof δ to deceive the verifier, and malicious verifier cannot forge the auditing results.

- **Malicious CSP.**

Malicious CSP cannot forge proof δ because every time the verifier sends a challenge, a random value f will be given in the $chal$ variable. CSP will generate proof δ based on f from the verifier as shown in Equation (2), and value f is different for each data shard. Even we further assume that malicious CSP forges δ, namely $\delta \neq \delta'$. The auditing process done by the verifier in Equation (4) shows that the verifier must validate δ′ with two other variables, the sender's public key PK_S and proof ω generated by the verifier, which also consists of f value. Therefore, the δ′ cannot make Equation (4) hold. Another case is when a malicious CSP tries to deceive the verifier by replacing challenged data block m_j with another data block mk when the former data block is broken. Accordingly, the proof δ′ becomes

$$\delta' = \prod_{i\in I, i\neq j} f_i Hash_1(m_i)P + f_j Hash_1(m_k)P \tag{5}$$

So, the Equation (4) can be represented as

$$e(\delta', \omega) = e(\delta, \omega) = e(P, P) \qquad (6)$$

Hence, $Hash_1(m_k) = Hash_1(m_j)$. However, $Hash_1(m_k)$ cannot be equal to $Hash_1(m_j)$ due to the anti-collision property of the hash function. Therefore, it is infeasible to make Equation (6) hold, and the proof from CSP cannot pass the auditing process.

• **Malicious Verifier.**

The malicious verifier cannot forge auditing results because the verifier must generate proof ω that requires V, which is a signature generated by the user as shown in Equation (3). The V values are stored in the smart contract, which is very difficult to be tampered with. Even if we further assume that the malicious verifier forges proof ω, to generate variable V, the malicious verifier needs the user's private key SK_S and original message m_i. The proposed scheme provides privacy-preserving and blockless verification properties, meaning the verifier can verify the data without receiving the original message m_i from the user or CSP. So, without the possession of those two variables, SK_S and m_i, it is impossible to generate a forged m'_i and make equation $(Hash1(m'_i) + SK_S)^{-1}P = (Hash_1(m_i) + SK_S)^{-1}P$ holds. Both malicious CSP and verifier cannot calculate the user' s private key from the public key under the **InvCDHP** assumption. Furthermore, in the auditing process, the verifier needs to validate ω with two other variables, the sender's public key PK_S and proof δ generated by CSP as shown in Equation (4). Therefore, it is infeasible to make equation $e(\delta, \omega') = e(P, P)$ holds, where $\omega' \neq \omega$.

4.3 Computational Cost

To analyze the proposed scheme performance, this paper compares the scheme's computational cost with four other signcryption schemes in Table 2. This part will discuss several aspects: signature type, computational cost, signature size, and time complexity. First, signature type. The proposed scheme and [9] use the ZSS signature scheme. While [12] uses the BLS signature and [13], [14] uses the attribute-based signature. The ZSS signature is more efficient because it has fewer pairing operations than the BLS signature used in [12]. In addition, the ZSS signature does not need a particular hash function like in the BLS signature (i.e., MapToPoint). It can use a general hash function like the SHA family or MD5.

The second is computational cost. Table 2 also shows costs for the sender and recipient that compute signcryption and unsigncryption, respectively. The pairing and exponential operations are considered highcost operations. Unfortunately, the attribute-based signature required more exponential operations than other schemes. In the [13], the sender needs to do nine exponential operations, and in [14], the sender requires eight exponential operations. Unlike the two previous schemes, the BLS-based signature in [12] required the sender to do fourteen multiplication and two pairing operations.

However, in the proposed scheme, the computational cost for the sender is $1Exp + 1Inv + 1Mul + 1Add + 3Hash$, and for the recipient is $1Add + 2Hash + P$. The sender needs to do fewer exponentiation and multiplication operations than other schemes. The recipient also has fewer operations when doing an unsigncryption scheme. Furthermore, in the unsigncryption scheme, the bilinear pairing of $e(P, P)$ can be precomputed. So, there is only one pairing operation. In addition, even though the proposed scheme has more hash

Table 2. Computational Cost and Time Complexity Comparison.

Ref	S/R	Signature Type	Signature Size (bits)	Computational Cost	Time Complexity
[12]	S	BLS	160	$n(14Mul + 2P)$	$O(n)$
	R			n/a	n/a
[13]	S	Attribute-based	n/a	$n(9Exp)$	$O(n)$
	R			P	$O(1)$
[14]	S	Attribute-based	n/a	$n(8Exp)$	$O(n)$
	R			$n(3Exp + 3P)$	$O(n)$
[9]	S	ZSS	260	$n(1Exp + 1Inv + 1Mul + 1Add + 3Hash)$	$O(n)$
	R			$n(1Add + 2Hash + P)$	$O(n)$
Proposed Scheme	S	ZSS	260	$n(1Exp + 1Inv + 1Mul + 1Add + 3Hash)$	$O(n)$
	R			$n(1Add + 2Hash + P)$	$O(n)$

S = Sender, R = Recipient, n = number of message, Add = addition, Hash = hash function, Mul = multiplication, Inv = Inverse, Exp = exponentiation, P = Bilinear Pairings, n/ a = not available.

operations, the cost hash function is negligible compared to the other operations. Nevertheless, the computational cost is the same as the based reference, which is [9].

Another variable to measure the proposed scheme performance is to analyze the signature length and time complexity. Regarding signature length, the BLS signature size is approximately 160 bits[15], while ZSS is around 260 bits[9]. The bigger the signature size, the longer the time needed to complete signcryption or unsigncryption. In terms of time complexity, the proposed scheme and four other schemes have linear time complexity ($O(n)$), except the recipient part in [13] that has constant time complexity ($O(1)$). The former indicates that the time in this instance, relies on the input n. The greater the value of n, the longer the required time. The latter shows that the required time is always constant and does not depend on the input variable.

The discussion above shows that[12] has a shorter signature size but a high computational cost. The proposed scheme and [9] have lower computational costs than [12], but this paper has a more extended signature size. However, with those conditions, the proposed scheme offers more advantages than other schemes. The trade-off is this paper can achieve desirable properties in the data integrity auditing process, public verifiability, privacy-preserving, blockless verification, and reliable verifier. More importantly, the proposed scheme can accomplish three more properties with the exact cost as the primary reference [9].

4.4 Communication Cost

This section presents the communication cost comparison between the proposed scheme and the primary reference[9]. Communication cost is the amount of data transfer involved in a protocol. For the proposed scheme, this paper measures the communication cost of the data auditing scheme presented in Fig. 2. Bits are used as the measurement unit. Three entities that interacted in the proposed protocol are the user, CSP, and verifier. In Table 3, there are five interactions among entities in the proposed scheme: send

Table 3. Communication cost comparison.

	User		**CSP**		**Verifier**	
	Our	[9]	Our	[9]	Our	[9]
Send signcryption	$n \times 772$	$n \times 788$	-	-	-	-
Send challenged data shards	$c \times 276$	-	-	-	-	-
Send proof δ	-	-	$c \times 1440$	-	-	-
Send *chal*	-	-	-	-	$c \times 176$	-
Send verification results	-	-	-	-	128	-
Forward (m_i, U, r, PK_s)	-	$c \times (\lvert m_i \rvert + 532)$	-	-	-	-
Return m_i	-	-	-	$c \times \lvert m_i \rvert$	-	-
Total	$(n \times 772) + (c \times 276)$	$(n \times 788) + (c \times (\lvert m_i \rvert + 532))$	$c \times 1440$	$c \times \lvert m_i \rvert$	$(c \times 176) + 128$	-

Note : the measurement unit is in bits, n = number of data shards, c = number of challenged data shards.

signcryption, send challenged data shards, send proof δ , send *chal*, and send verification results. All of those messages are unicast messages.

In the proposed scheme's auditing process, file F is divided into n shards of m. To give a practical example in the analysis, this paper denotes the size of file F the same as that is used in [16], which is 40M. Then, file F is divided into 10,000 data shards, so each size of m_i is 4kb = 32,000 bits, where i is the index of data shards. When the user asks a verifier to do the auditing process, he randomly sends several challenged data shards; this paper denotes it as c. Assuming c is half of n, c is 5,000 data shards. Therefore, according to Table 3, the total communication cost for all entities in the proposed scheme is 2,147kb. While in [9] scheme, entities have three interactions: send signcryption, forward parameters (m_i, U, r, PKs), and return m_i. All of those messages are unicast messages. This paper denotes values n and c, the same as in the previous example. Therefore, the total communication cost for all entities in [9] is 41,417kb.

The results show that the total communication cost in the proposed scheme is lower than[9]. This paper provides blockless verification that allows the auditing process to happen without accessing the original message. In contrast, the communication cost in [9] is higher because the user and CSP must send an original message shard m_i. This scheme's communication cost depends on the size of m_i. For the example, the size of m_i is 32,000 bits. As a result, in this case, the total communication cost of

[9] is almost twenty times higher than ours. Nevertheless, the proposed scheme offers a cheaper communication cost than [9].

4.5 Comparative Analysis

In this section, this paper presents an analysis of the compatibility of the proposed scheme with the three desirable properties. Also, this paper compares the proposed scheme with four other works related to data auditing using the signcryption schemes in Table 4 as follows.

• **Public Verifiability.**

The proposed scheme provides the public verifiability property shown in Equation (4), where this paper presented a use case in which a verifier can verify data stored in CSP. So the user has the option to choose someone other than CSP to carry out the data integrity checking process. The comparison with four other works in Table 4 shows that only [9] fulfilled this property. Unfortunately, only the authorized receiver may verify the data in [12-14] since their protocols did not allow other parties to verify saved data.

• **Privacy-preserving.**

The proposed scheme can ensure that the verifier cannot know the users' data during auditing. As shown in Equation (2), CSP sends proof δ to the verifier that is generated not from the user's original data. In addition, the verifier only computes the message's signature V_i during Equation (3) creation, not the original data. Therefore, the proposed scheme can prevent data leaks to the verifier during auditing. The other works that provide this property are [12-14]. While protocol in [9] requires the user to submit to the verifier the original message that has to be verified. Hence it does not support this property. This would expose the user's private information to third parties.

Table 4. Comparison of Desirable Properties.

Ref	Public Verifiability	Privacy-Preserving	Blockless Verification	Reliable Verifier
[9]	√	×	×	×
[12]	×	√	×	×
[13]	×	√	×	√
[14]	×	√	√	√
Proposed Scheme	√	√	√	√

• **Blockless Verification.**

The proposed scheme supports blockless verification. In blockless verification, the verifier is not required to download all the challenged data blocks. Equation (4) demonstrates that while the verifier performs an auditing process, no original data is downloaded from CSP. It increases efficiency rather than downloading the data beforehand. Table 4 shows that only[14] supports this property. While three other works[9,12,13] did not provide this property since the verifier needs the original message in order to complete the auditing task, the availability of this property was not provided by them.

• **Reliable Verifier.**

The proposed scheme and two other works [13,14] provide a reliable verifier using blockchain technology. Blockchain enables transparency between users, CSPs, and verifiers through its decentralized nature. Furthermore, the proposed protocol assures a credible verifier by enforcing credibility points and incentive mechanisms for the honest verifier. However, two other works [9,12] do not support this attribute since they rely on TPA, which is not the best solution in actual circumstances.

Ⅴ. Conclusion

This paper proposed a blockchain-based data auditing protocol in a cloud storage with an improved ZSS signcryption scheme that provides desirable properties such as public verifiability, privacy-preserving, blockless verification, and reliable verifier. A use case for signcryption implementation in data auditing is also presented in this paper that aims to ensure data integrity, confidentiality, and non-repudiation. Ultimately, this paper also presented security analysis by demonstrating the validity of the proposed scheme equation, its unforgeability in the presence of malicious CSP and verifiers, examining the computational and communication cost, and finally, by presenting comparative studies analysis with four other works. The comparative studies show that only the proposed scheme can fulfill the four desirable data auditing protocol properties. The proposed protocol is also equipped with credibility points and incentive mechanisms to attract the participation of verifiers. Furthermore, it could also prevent arbitrary behavior of participants by introducing penalty points. In addition, the proposed system can also accomplish three more attributes than the primary reference with the exact computational cost, according to the examination of computational cost and time complexity. Subsequently, this paper's communication cost analysis shows that the proposed auditing protocol can achieve lower communication costs than the primary reference. Those analyses implied that this paper offers more advantages in data auditing protocol through blockchain and improved ZSS signcryption.

References

[1] E. Gaetani, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Blockchain-based database to ensure data integrity in cloud computing environments," *University of Southampton Institutional Repository,* 2017.

[2] Fran Howarth, "*Sabotage: The latest threat to the financial/banking industry,*" (Aug. 2016), [Online]. Available: https://ibm.co/3LND6tE (visited on 08/30/2022).

[3] N. Garg and S. Bawa, "Comparative analysis of cloud data integrity auditing protocols," *J. Netw. and Comput. Appl.*, vol. 66, pp. 17-32, 2016.

[4] B. Libert and J.-J. Quisquater, "A new identity based signcryption scheme from pairings," in *Proc. 2003 IEEE Inf. Theory Wkshp.* (*Cat. No. 03EX674*), pp. 155-158, 2003.

[5] X. Boyen, "Multipurpose identity-based signcryption: A swiss ary knife for identity-based cryptology, crypto' 03, lncs 2729," in *Annual Int. Cryptology Conf.,* pp. 383-399, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[6] W.-J. Cui, Z.-J. Jia, M.-S. Hu, L.-P. Wang, et al., "A new signcryption scheme based on elliptic curves," in *Int. Conf. Secur. and Privacy in New Comput. Environ.*, pp. 538-544, Springer, 2019.

[7] F. Bao and R. H. Deng, "A signcryption scheme with signature directly verifiable by public key," in *Int. Wkshp. Public Key Cryptography,* pp. 55-59, Springer, 1998.

[8] S. S. Chow, S.-M. Yiu, L. C. Hui, and K. Chow, "Efficient forward and provably secure id-based signcryption scheme with public verifiability and public ciphertext authenticity," in *Int. Conf. Inf. Secur. and Cryptology,* pp. 352-369, Springer, 2003.

[9] C. Ma, "Efficient short signcryption scheme with public verifiability," in *Information Security and Cryptology,* vol. 4318, Springer Berlin Heidelberg, pp. 118-129, 2006. (https://doi.org/10.1007/11937807_10) [Online]. Available: http://link.springer.com/10.1007/11937807_10 (visited on 07/18/2022).

[10] M. Toorani and A. Beheshti, "A directly public verifiable signcryption scheme based on elliptic curves," in *2009 IEEE Symp. Comput. and Commun.*, pp. 713716, 2009.

[11] F. Zhang, R. Safavi-Naini, and W. Susilo, "An efficient signature scheme from bilinear pairings and its applications," in *Public Key Cryptography PKC 2004,* vol. 2947, Springer Berlin Heidelberg, pp. 277-290, 2004. (https://doi.org/10.1007/978-3-540-24632-9_20). [Online]. Available: http://link.springer.com/10.1007/978-3-54024632-9_20 (visited on 07/18/2022).

[12] A. Alamer, "An efficient group signcryption scheme supporting batch verification for securing transmitted data in the internet of things," *J. Ambient Intell. and Humanized*

Comput., pp. 1-18, 2020.

[13] N. Eltayieb, R. Elhabob, A. Hassan, and F. Li, "A blockchain-based attribute-based signcryption scheme to secure data sharing in the cloud," *J. Syst. Architecture*, vol. 102, p. 101 653, 2020.

[14] X. Yang, T. Li, W. Xi, A. Chen, and C. Wang, "A blockchain-assisted verifiable outsourced attribute-based signcryption scheme for ehrs sharing in the cloud," *IEEE Access*, vol. 8, pp. 170 713-170 731, 2020.

[15] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Int. Conf. The Theory and Application of Cryptology and Inf. Secur.*, pp. 514-532, Springer, 2001.

[16] C. Zhang, Y. Xu, Y. Hu, J. Wu, J. Ren, and Y. Zhang, "A blockchain-based multi-cloud storage data auditing scheme to locate faults," *IEEE Trans. Cloud Computing*, 2021.

Elizabeth Nathania Witanto

Aug. 2015 : Bachelor's degree, Petra Christian University, Surabaya, Indonesia

Aug. 2020 : M.Sc. degree, Dongseo University, Busan, South Korea

Aug. 2023 : Ph.D. degree, Dongseo University, Busan, South Korea

Sep. 2023~Current : Lecturer, Ciputra University, Surabaya, Indonesia

<Research Interest> blockchain and cloud computing

[ORCID:0000-0003-1085-4272]

Sang-Gon Lee

1986 : BEng. degree, Kyungpook National University

1988 : MEng. degree, Kyungpook National University

1993 : Ph.D. degree, Kyungpook National University

Aug. 2003~July 2004 : Visiting Scholar at QUT, Australia

July 2012~Jun. 2013 : Visiting Scholar at the University of Alabama at Huntsville, USA

1997~Current : Professor, Dongseo University, Busan, South Korea

<Research Interests> Information security, network security, wireless mesh/sensor networks, and the future Internet.

[ORCID:0000-0002-6678-0500]