Watermarking-Based Secure Plaintext Image Protocols for Storage, Show, Deletion and Retrieval in the Cloud

Xiaojuan Dong, Weiming Zhang, Mohsin Shah, Bei Wang, and Nenghai Yu

Abstract—In this paper, we propose secure plaintext image storage protocols in the cloud environment for image owners managing and controlling their outsourced images. To solve control and privacy issues, conventional schemes suggest outsourcing images in the encrypted form. However, encrypted images lose their usability and visibility. For example, image owners cannot quickly find their stored images in the cloud. The proposed protocols encourage plaintext image storage in the cloud with copyright protection of images, visible display, lossless retrieval and controlled deletion via techniques of homomorphic encryption and digital watermarking. We allow the image owner to embed and remove the watermark in a privacy-preserving way without compromising the security of the original data and the computed results. Moreover, the image owner can securely detect the cloud's dishonest act of not completely deleting an image as required or leaking an image without permission. Compared with existing works, our work achieves more functions. We prove that the proposed work achieves the controllable management of the outsourced plaintext images without privacy leakage to unauthorized parties and demonstrate the utility and the efficiency of our protocols through experimental evaluation.

Index Terms—Cloud Storage; Homomorphic Encryption; Watermarking; Privacy Preserving; Services Computing; Anti-Collusion Attacks.

1 Introduction

Combining cloud service technologies and networks makes it easy for people to display their images online, specifically on social platforms, such as Facebook and WeChat. According to the statistics of Facebook [1], 300 million photos are uploaded to Facebook per day. Posting data online cloud reduces the storage on local devices. But data placed on social platforms lose their owners' protection. As a result, these data can be abused by others. For instance, Enck [2] investigated the behaviour of 30 popular third-party applications in his 2010 and 2012 studies and found that two-thirds of the applications potentially misused users' sensitive data and that half of the applications reported users' information to remote advertising servers. Given that the owners' uploaded images may contain privacy information, such as an ID card number, image owners delete the uploaded images from social platforms to protect privacy data. Unfortunately, it is difficult for the image owners to completely delete uploaded images since cloud service providers store multiple backups [3] of data over different online or offline servers for fault tolerance. As a result, even though one copy of the to-be-deleted data has been deleted from the current server, the other copies are still on offline/online servers. The lack of trust services impedes the prevalence of clouds as outsourced storage and computing services [4]. Therefore, in the case of the dishonest cloud, determining how to protect the privacy of cloud data and control the deletion of cloud data remains a challenge.

To minimize security and privacy concerns regarding cloud data, common solutions rely on encryption technology, including quantum encryption [5] and [6]. The user's data are usually encrypted before stored in the cloud so that the encrypted cloud data can keep private and secure. Deleting the encrypted cloud data is achieved by destroying the corresponding encryption keys, resulting in the inaccessibility of encrypted cloud data [7]. However, the encrypted data lose convenient usability. Even though the cloud computing services support some ciphertext calculations, such as public auditability [8], similarity search [9], feature extraction [10], and editing images [11], the calculations over ciphertexts are complex and expensive. Furthermore, a large number of calculations still cannot be implemented in the ciphertext domain. Therefore, outsourcing plaintext data is more common than outsourcing ciphertext data in the real world. However, directly outsourcing plaintext to the public cloud inevitably leads to privacy concerns.

To support storing plaintext images in the cloud while protecting the images' copyright, preventing leakage and abuse of images and controlling deletion of images, we exploit the traceability property of digital watermarking to identify the dishonest cloud that leaks its stored images or does not delete images as required by the image owner. The operations of watermarking are performed in the homomorphic encryption domain without leaking the sensitive information.

Our contributions of this paper can be summarized as follows:

1) Secure plaintext image storage (SPIS) protocols in-

- *Xiaojuan Dong, Mohsin Shah and Bei Wang are with CAS Key Laboratory of Electromagnetic Space Information, University of Science and Technology of China, Hefei 230026, China. E-mail: {xjuadong, mohsin, wangbei}@mail.ustc.edu.cn.*
- *Weiming Zhang and Nenghai Yu are with CAS Key Laboratory of Electromagnetic Space Information, University of Science and Technology of China, Hefei 230026, China. E-mail: {zhangwm, ynh}@ustc.edu.cn.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2020.3008957, IEEE Transactions on Services Computing

volving four parties are proposed; these parties are an image owner, a cloud server, a management authority and a judge. SPIS protocols support image display, image deletion and image retrieval, as shown in Fig. 1.

2) The embedding, removal and detection operations on watermarks are performed in the encrypted domain without exposure of any plaintext watermark. The embedding positions of the watermarking are hidden, and no one knows where the watermark is embedded.
3) Assume that the image owner, the cloud server and the management authority are semi-honest. The image owner or the cloud server may conspire with the management authority to obtain the cloud server's or the image owner's plaintext data, such as the original image or the identity watermark. However, the conspiracy is prevented and no clear information is leaked in the communication.
4) We conduct an in-depth security analysis of SPIS protocols. The communication and computation overheads of SPIS protocols are evaluated. We built a simulator in Java to demonstrate the utility of SPIS protocols.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces notations and preliminaries. Section 4 presents the system model, the attack models and the pursuing goals. Section 5 describes the SPIS protocols and four secure computing sub-protocols. In Section 6, the security analysis is described. In Section 7, simulation experiments are performed. We summarize this paper in Section 8.

2 Related Work

The encryption methodology can reduce cloud data security and privacy concerns to a certain degree. To realize fine-grained access control for encrypted cloud data, attribute based encryption (ABE) is an effective mechanism available in the literature [12], [13] and [14]. To delete the ciphertext data stored in the cloud, Tang et al. [15] presented the assured deletion scheme. Before uploading data, the data owner uses a data key to encrypt the data through a symmetric encryption method, and the data owner also uses the access structure to encrypt the data key to control accessibility. The assured data deletion is reached by attribute revocation such that all data keys of the users are useless and none of the users can decrypt the ciphertext. However, symmetric encryption used in ABE schemes does support mathematical operations on encrypted data.

Fortunately, homomorphic encryption supports some manipulation operations on encrypted data without proper decryption [16]. Many applications of homomorphic encryption focus on secure computation over an unsecured channel, cloud computing and cloud storage available in the literature [11], [17], [18], [23], [24], [25], [26] and [27].

Samanthua et al. [17] employed homomorphic encryption and proxy re-encryption, and achieved fine-grained access control over data outsourced to the cloud. This method can be used for documents and numbers but is not fit for images as encrypted images are prone to an extreme loss of their visibility and usability.

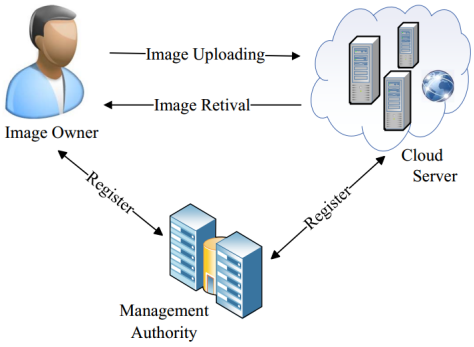

Fig. 1. System model of SPIS protocols

To support the upload of plaintext images, our previous work [19] and [20] proposed plaintext image storage protocols, using homomorphic encryption and watermarking techniques. Homomorphic encryption technology, specifically referring to the Paillier cryptosystem [21], ensures the confidentiality of image content during the upload process; the cloud's watermark is embedded into the uploading encrypted image; the cloud server receives the encrypted and watermarked image, and stores it after decryption. The embedded watermark can track the misbehaviour of the cloud, such as revealing the upload image or not deleting the image as required by the image owner. The protocols in [19] and [20] protect the cloud user's right to be forgotten [22]. The users request the cloud to delete his/her image, but the cloud does not comply. If this image appears again, the authors argue that the cloud does not delete the user' image, and the corresponding cloud violates the user's right to be forgotten. However, the retrieved image in [19] contains double watermarks, which seriously damages the visual quality of the image. Moreover, the protocols in [19] and [20] both assume that the third party is honest and trustworthy. Consequently, the third party can store and know the plaintext watermarks that belong to the image owner and the cloud, and the conspiracy of the third party with the cloud sever or with the image owner is not discussed. Furthermore, the schemes based on the Paillier cryptosystem in [11], [17], [24], [25], [26], and [27] do not consider the dishonest third party. In practice, the third party may be dishonest regarding the interests or disclose the users' data due to invasion. Hence, we adopt our previous work, the restrained Paillier cryptosystem in [28], where the third party is semi-honest and has no knowledge of the privacy data of other participants.

3 Preliminary

In this section, we review the cryptographic primitives and watermarking primitives involved in this paper.

3.1 Cryptographic Primitives

In this section, we introduce typical properties of partially homomorphic cryptosystems and then review the restrained Paillier cryptosystem in [28]. For the sake of brevity, the notations used in the cryptosystems are shown in Table 1.

TABLE 1
Notations in Cryptosystems

Notation	Description
$\|\cdot\|$	Bit length
pk, sk	Partially homomorphic public & private key
$E^{+}_{pk}(\cdot)$	Additive encryption algorithm with public key
$E^{\times}_{pk}(\cdot)$	Multiplicative encryption algorithm with public key
$gcd(x, y)$	Greatest common divisor between x and y

3.1.1 Partially Homomorphic Cryptosystems

Two additive ciphertexts $E^{+}_{pk}(m_1)$ and $E^{+}_{pk}(m_2)$ accord with the following property:

$E^{+}_{pk}(m_1) \times E^{+}_{pk}(m_2) = E^{+}_{pk}(m_1 + m_2)$.

Two multiplicative ciphertexts such as $E^{\times}_{pk}(m_1)$ and $E^{\times}_{pk}(m_2)$ accord with the following property:

$E^{\times}_{pk}(m_1) \times E^{\times}_{pk}(m_2) = E^{\times}_{pk}(m_1 \times m_2)$.

3.1.2 Restrained Paillier Cryptosystem

The public key is $(N, g, h = g^{\theta} \mod N)$ with a generator g of order $\lambda = 2p'q'$ (where $p = 2p' + 1, q = 2q' + 1$ are safe primes). The strong key is λ, and the weak private key $\theta \in [1, N^2/2]$. Such a g can be obtained by selecting a random $a \in \mathbb{Z}^{*}_{N^2}$ but $a \neq 1 \mod N$ and computing $g = -a^{2N} \mod N$. The party i's pair of public and private key is denoted as (pk_i, sk_i). The party j's pair of keys is denoted as (pk_j, sk_j).

Additive Encryption (AddEnc): Given a message $m \in \mathbb{Z}_N$, choose a random $r \in [0, N/4]$ and output the additive ciphertext as $E^{+}_{pk}(m) = \{AC_1, AC_2\}$, where $AC_1 = (h^r \mod N)^N (1 + mN) \mod N^2$; $AC_2 = g^r \mod N$.

Additive Decryption with Weak Private Key (AddDecWkey): An additive ciphertext such as $E^{+}_{pk}(m)$ can be decrypted with private key $sk = \theta$ by calculating: $m = L\{\frac{AC_1}{[(AC_2)^{\theta} \mod N]^N}\}$.

Additive Decryption with Strong Private Key (AddDecSkey): The original message m is calculated as: $m = L[(AC_1)^{\lambda} \mod N^2]\lambda^{-1} \mod N$, since $gcd(\lambda, N) = 1$.

Strong Private Key Splitting (SkeyS): Split the strong private key λ into two partial strong private keys. One, denoted as λ_i, is sent to a party i; the other one, denoted as λ_j, is sent to a party j. λ_i and λ_j meet the following two constraints:

$$\begin{cases} \lambda_i + \lambda_j = 0 \mod \lambda, \\ \lambda_1 + \lambda_j = 1 \mod N. \end{cases}$$

Additive Decryption with Partial Strong Private Key Step One (AddDecPSkey1): This algorithm is run in the party i side. Exploiting the partial strong private key λ_i, the partial decrypted ciphertext DC_1 of $E^{+}_{pk}(m)$ can be calculated as $DC_1 = (AC_1)^{\lambda_i} = g^{\lambda_i \cdot r\theta}(1 + \lambda_i mN) \mod N^2$. The party i forwards $\{E^{+}_{pk}(m), DC_1\}$ to the party j.

Additive Decryption with Partial Strong Private Key Step Two (AddDecPSkey2): This algorithm is run on the party j side. Given $\{E^{+}_{pk}(m), DC_1\}$ and λ_j, the partial decrypted ciphertext DC_2 can be calculated as $DC_2 = (AC_1)^{\lambda_j} = g^{\lambda_j \cdot r\theta}(1 + \lambda_j mN) \mod N^2$. Then, the original message m can be recovered as: $m = L[DC_1 \cdot DC_2]$.

Multiplicative Encryption (MulEnc): Given a message $m \in \mathbb{Z}_N$, choose a random number $r \in [1, N/4]$ and output the multiplicative ciphertext as $E^{\times}_{pk}(m) = \{MC_1, MC_2\}$, where $MC_1 = mg^{r\theta} \mod N$; $MC_2 = g^r \mod N$.

Multiplicative Decryption (MulDec): The multiplicative ciphertext $E^{\times}_{pk}(m)$ can be decrypted with the private key $sk = \theta$ by calculating $m = \frac{MC_1}{(MC_2)^{\theta}} \mod N$.

Multiplicative Ciphertext Refresh (MCR): Refresh $E^{\times}_{pk}(m)$ without decrypting this ciphertext by randomly choosing $r' \in Z_{p'q'}$ and calculate $(E^{\times}_{pk}[m])' = \{(MC_1)', (MC_2)'\}$. $(MC_1)' = mh^{r_{CR}} \mod N$; $(MC_2)' = g^{r_{CR}} \mod N$; $r_{CR} = r + r' \mod 2p'q'$.

A Multiplicative Ciphertext to A Mixed Ciphertext (MultoMix): Gives the multiplicative ciphertext $E^{\times}_{pk}(m) = \{MC_1, C_2\}$, where $MC_1 = mh^r \mod N$; $MC_2 = g^r \mod N$. Randomly choosing a number $r' \in [0, N/4]$ runs the **AddEnc** algorithm and outputs the mixed ciphertext $E^{*}_{pk}(m)$ as $E^{*}_{pk}(m) = \{MixC_1, MixC_2\}$, where $MixC_1 = (h^{r'} \mod N)^N (1 + mh^r N) \mod N^2$; $MixC_2 = g^r \mod N$.

A Mixed Ciphertext to An Additive Ciphertext (MixtoAdd): The **MixtoAdd** algorithm is executed between the party i and the party j as follows.

Step 1 (@j): Given a mixed ciphertext $E^{*}_{pk_{ij}}(m)$, U_j chooses a random number $s \in Z_{2p'q'}$ and computes $t_1 = (MixC_{ij,2} \cdot g^{\theta_j})^s = g^{(r+s)\theta_j} \mod N$ and $t_2 = g^s \mod N$. U_j sends $MixC_{ij,1}, t_1$ and t_2 to party i.

Step 2 (@i): Once t_1 is received, the party i first computes $(t_1)^{\theta_i} = h_{ij}^{(r+s)} \mod N$ and its inverse $[h_{ij}^{(r+s)}]^{-1} \mod N$, which the party i uses to calculate $(MixC_{ij})'$ as

$(MixC_{ij})'$
$= (MixC_{ij})^{[h_{ij}^{(r+s)}]^{-1}} \mod N^2$
$= (h_{ij}^{r'} \mod N)^{[h_{ij}^{(r+s)}]^{-1} N}[1 + m(h_{ij}^{s})^{-1}N] \mod N^2$.

Then, i computes $T_2 = (t_2)^{\theta_i} = g^{s\theta_i} \mod N$, and forwards $(MixC_{ij})'$ and T_2 to party j.

Step 3 (@j): Upon receiving T_2, the party j uses it to compute a middle result $(T_2)' = (T_2)^{\theta_j} = h_{ij}^{s} \mod N$. Next, the additive ciphertext $E^{+}_{pk_{ij}}(m)$ is computed as

$E^{+}_{pk_{ij}}(m)$
$= [(MixC_{ij})']^{(T_2)'}$
$= (h_{ij}^{r'} \, mod \, N)^{(h_{ij}^{r})^{-1} N}(1 + mN) \mod N^2$.

3.2 Watermarking Primitives

Digital watermarking embeds some identification information in the protected digital carrier to prove the ownership of copyright or track the infringement and does not affect the use value of the original carrier. Identification information that is commonly called a watermark can be bit strings, company logos, images, biometrical features, and so on, and a Trojan can also be embedded as a feature [29].

Here, we briefly recall the multiplicative spread spectrum watermarking technique proposed by Cox et al. [31] and then review the method of watermark detection.

3.2.1 Spread Spectrum Watermark

The components of watermark $W = \{w_1, \cdots, w_L\}$ are L coefficients independently chosen from $N(0, 1)$ Gaussian distribution. The watermark W is embedded into the largest L alternating current (AC) coefficients $X = \{x_1, \cdots, x_L\}$ in the discrete cosine transform (DCT) of an original carrier by the

following multiplicative insertion formula: $Y = X(1+\alpha W)$, where Y represents the modified coefficients and α represents the embedding strength factor, which can used to control the visual quality of the carrier.

3.2.2 Watermark Detection

Digital watermark detection can not only detect duplicate digital media such as [30] but can also track the copy behaviour according to the detected watermark. Watermarking detection is accomplished by computing a correlation value denoted as $corr$, which measures the confidence for the presence of a specific watermark $W = \{w_1, \cdots, w_L\}$ in a suspected carrier $Y = \{y_1, \cdots, y_L\}$. The decision regarding the presence of watermark W in Y occurs if $corr > \delta$ holds for a predefined detection threshold δ. Otherwise W is absent. $corr$ is calculated as: $corr = \frac{<Y,W>}{<Y,Y>} = \frac{\sum_{n=1}^{L}(y_i \cdot w_i)}{\sqrt{\sum_{i=1}^{L}(y_i \cdot y_i)}}$.

When L is large enough, it obeys a normal distribution. Assume that Y is independent of W; the threshold δ can be derived as in [32].

4 System Model, Goals and Threat Model

In this section, we formalize the involved four-party system model, define the attack models and design the goals.

4.1 System Model

The proposed protocols involve four parties: an image owner, a cloud server, a management authority and a judge. The main interactions among the image owner, the cloud server and the management authority are shown in Fig. 1.

The image owner (IO) embeds the cloud server's identity watermark into images prior to uploading. When retrieving the IO's images, the IO requires obtaining the original image without any watermark embedded. The IO has his/her unique identity watermark that is considered privacy data and confidential to the other parties.

The cloud server (CS) has a powerful computing capability and enormous storage space and offers the IO the storage service for fee. The CS has its unique identity watermark that is its privacy data and confidential to the other parties.

The management authority (MA) is a semi-trusted party. It is in charge of generating cryptosystem parameters. In addition, the MA preserves and manages the unique identity watermark for each registered party.

The judge (J) executes arbitrations against information leakage and copyright infringement.

4.2 Insider Attacks

Two unique identity watermarks for the IO and the CS are denoted as W_o and W_c. Before uploading to the CS, the image has been watermarked with W_o and W_c. W_o is used to prove the ownership of the image by the IO, and W_c is used to track the improper behaviour of the CS. When retrieving the uploaded image, removal of W_o and W_c is performed to restore the original images. Watermarks W_o and W_c are privacy information and need to be encrypted during calculations.

The IO, the CS, and the MA all are semi-honest parties in that they correctly follow the SPIS protocols, but are curious about other's personal private information, such as private keys, original images and identity watermarks. Hence, we discuss the underlying security issues triggered by IO, the CS, and the MA. These security issues are introduced as follows:

- ***IO's Dishonesty:*** The IO causes three security issues. The first issue is that in the image uploading phase, the IO acquires W_c from the CS and then embeds W_c and W_o concurrently into the image X to forge $X_{W_{oc}}$. The IO leaks the forged image $X_{W_{oc}}$ stealthily, accuses the CS of the leakage behaviour, and then gains compensation from the CS. The first issue has been solved in our previous work [19], where the IO obtains only the encrypted W_c and not the plaintext W_c. The second new issue is that in the image retrieval phase, the IO obtains W_c by not removing W_c from $X_{W_{oc}}$. To solve the second issue, the CS has the task of removing the embedded watermarks from $X_{W_{oc}}$. The third new issue is that the IO hinders the CS from removing the watermark by providing the CS with a fake watermark. Consequently, the IO can obtain the CS's private watermark W_c. To solve the third issue, the CS verifies the watermark authenticity. All solutions will be described later.
- ***CS's Dishonesty:*** The CS induces two security issues. The first issue is that the CS gives the IO a false identity watermark instead of the true W_c so that the CS can escape from watermarking tracing. To solve the first issue, the IO detects the authenticity of W_c before embedding the watermark. The second issue is that in the image retrieval phase, the CS obtains the embedding locations of the watermark. Then, the CS destroys the embedded watermark causing a watermark detection failure. The second issue is settled since the IO hides the locations of the embedded watermark. All solutions will be described later.
- ***MA's Dishonesty:*** For various benefits or as a result of being compromised, the MA may sell out or give out some parties' private keys. To solve this issue, we use the key distribution scheme in our previous work [28].
- ***Conspiracy Problems:*** The IO or the CS conspire with the MA to obtain the CS's or the IO's plaintext data, such as the original image or the identity watermark.

4.3 Outsider Attacks

We introduce an attacker $\mathcal{A}^*$ outside the system. The adversary $\mathcal{A}^*$ is an active adversary who can threat the IO, the CS, and the MA. The attack capabilities of the adversary $\mathcal{A}^*$ are discussed as follows:

1) $\mathcal{A}^*$ may eavesdrop on all communication to obtain the encrypted data.
2) $\mathcal{A}^*$ may threaten the MA to obtain the strong key λ and to open all the additive ciphertexts.
3) $\mathcal{A}^*$ may compromise the IO to guess the plaintext value of all ciphertexts sent from the CS.
4) $\mathcal{A}^*$ may compromise the CS to guess the locations where to embed watermarks of images outsourced

from the IO, and the plaintext value of all ciphertexts sent from the IO.

4.4 Design Goals

Under the assumption that the IO, the CS, and the MA all are semi-honest parties, our ultimate goal is to design well-functioning image protocols that allow storing plaintext images in the cloud in order to support exhibition, lossless retrieval and assurance of the deletion of these plaintext images. The proposed protocols should realize the following goals.

- ***Data Security:*** Data are encrypted during transmission. Only the receiver can decrypt and obtain plaintext data.
- ***Traceability:*** Any copy of an image must be identifiable and traceable to find the illegal distributor. Specially, a watermarking protocol should enable an IO to determine whether a specific CS illegally revealed his/her image.
- ***Fairness:*** The proposed protocols are fair to the IO and the CS. A malicious IO cannot frame an innocent CS. An illegal CS cannot deny its faults.
- ***Conspiracy Problem:*** The IO and the MA may attempt to cheat the CS's identity watermark. The CS and the MA may conspire to obtain the positions of the embedded watermark.
- ***Recovery of Image Quality:*** When the IO retrieves his/her own images, the IO obtains original images without any loss of visual quality.
- ***Right to Be Forgotten:*** If the CS disobeys the user's delete request, the misbehaviour can be found through detecting the CS's unique identity watermark from the image that should have been deleted.

5 Proposed Protocols

5.1 Overview of SPIS Protocols

SPIS protocols, where important notations used are summarized in Table 2, consist of four phases: *system initialization*, *image uploading*, *image retrieval*, and *image tracing*. The initial phase is used for preparing public-private keys and distribution of identity certificates for participant parties.

In the image uploading phase, the IO and the CS first link W_c and W_o together. To hide where the watermark is embedded, the IO extends the length of the linked result to the size of X and obtains the combined watermark W_{oc}. The IO embeds W_{oc} in X, and obtains the watermarked image $X_{W_{oc}}$, which is sent to the CS for the watermark removal in the image retrieval phase.

In the image retrieval phase, the IO obtains his/her recovered original image X through the CS removing the embedded watermarked W_{oc}.

The IO finds an image Y that is similar to X. Y may be a copy of $X_{W_{oc}}$ revealed from the CS. The IO exploits watermark detection to find the illegal distributor.

5.2 Concrete Construction

In this subsection, we introduce our SPIS protocols in detail. The SPIS protocols will invoke secure computing sub-protocols, which include the secure combined multiplication (**SCMul**) sub-protocol, the secure strengthened operation (**SStrenOper**) sub-protocol, the secure equivalent verifying (**SEVer**) sub-protocol, and the secure watermark detection protocol (**SWDec**) sub-protocol. Each sub-protocol that will be described in Section 5.3 accomplishes its own independent task without compromising the privacy of data. The function descriptions and formulaic representations of these four sub-protocols are shown in Table 3.

TABLE 2
Summary of Notations in SPIS Protocols

Notation	Description
X	Original image
Y	Illegal copy of X
W, H	Width and height of X
pk_o, θ_o	IO's public key and private key
pk_c, θ_c	CS's public key and private key
pk_{oc}	Joint public key of the IO and the CS
L	Length of each party's unique identity watermark
$\alpha = \{\alpha_1, \cdots, \alpha_L\}$	Strength of embedded watermark
$W_o = \{w_{o,1}, \cdots, w_{o,L}\}$	IO's unique identity watermark
$SW_o = \{sw_{o,1}, \cdots, sw_{o,L}\}$	Strengthened W_o
$W_c = \{w_{c,1}, \cdots, w_{c,L}\}$	CS's unique identity watermark
$SW_c = \{sw_{c,1}, \cdots, sw_{c,L}\}$	Strengthened W_c
$W_{oc} = \{w_{oc,1}, \cdots, w_{oc,W\times H}\}$	Combined watermark containing W_o and W_c
$E^{\times}_{pk_o}(W_o)$	W_o encrypted by pk_o
$E^{\times}_{pk_{oc}}(W_o)$	W_o encrypted by pk_{oc}
$E^{\times}_{pk_{oc}}(SW_o)$	W_o strengthened and encrypted by pk_{oc}
$E^{\times}_{pk_c}(W_c)$	W_c encrypted by pk_c
$E^{\times}_{pk_{oc}}(W_c)$	W_c encrypted by pk_{oc}
$E^{\times}_{pk_{oc}}(SW_c)$	W_c strengthened and encrypted by pk_{oc}
$E^{\times}_{pk_{oc}}(SW_{oc})$	W_{oc} strengthened and encrypted by pk_{oc}

5.2.1 System Initialization

The MA generates the system public key (N, g) and the strong private key λ, according to our previous work [28]. When a party i enrols in the system, i generates his/her own public-private key pair $\{h_i = g^{\theta_i} \mod N, \theta_i\}$, and obtains his/her identity ID_i and the corresponding identity certificate $Cert_i$ from the MA. Before data transmission, communication parties can identify each other through the authentication protocol in [28].

In addition, the party i produces his/her own identification watermark W_i. The length of W_i is set as L. The party i deploys the **MulEnc** algorithm to encrypt W_i and yields $E^{\times}_{pk_i}(W_i)$. Then, $E^{\times}_{pk_i}(W_i)$ is transmitted to the MA. The MA stores $E^{\times}_{pk_i}(W_i)$ in MA's database. The MA cannot modify $E^{\times}_{pk_i}(W_i)$ without the party i's private key. The data related to the party i are stored in the database of the MA, as shown in Table 4.

5.2.2 Image Uploading

Before uploading image, the IO and the CS employ the Diffie-Hellman Key Exchange Agreement [34] to generate

TABLE 3
Functions and Formulas of Four Sub-protocols

Sub-protocol	Function Description	Formulaic Representation
SCMul	Perform multiplication in the ciphertext domain	$X_{W_{oc}} = SCMul(IO, CS, X, E^{\times}_{pk_{oc}}(W_{oc}))$
SStrenOper	Strengthen the encrypted watermark with the embedding strength factor	$E^{\times}_{pk_{oc}}(1+\alpha W_c) = SStrenOper(IO, CS, \alpha, E^{\times}_{pk_{oc}}(W_c))$
SEVer	Verify whether the plaintexts of two different ciphertexts are the same without decryption	$True/False = SEqVer(IO, CS, E^{\times}_{pk_c}(W_c), E^{\times}_{pk_{oc}}(W_c))$
SWDec	Detect Watermark in the ciphertext domain	$corr = SWaterDec(IO, CS, Y, E^{\times}_{pk_{oc}}(W_{oc}))$

TABLE 4
The Party i's Data in the Database of the MA

ID_i	$(g^{\theta_{r_i}} \mod N)^N(1+r_iN) \mod N^2$	$E^{\times}_{pk_i}(W_i)$

their joint public key $pk_{oc} = (N, g, g^{\theta_o\theta_c} \mod N)$. The strong private key λ has been separated into two shares by the MA. One share denoted as λ_o is transmitted to the IO. The other one denoted as λ_c is sent to the CS. Interactions between the IO and the CS of the image uploading phase shown in Fig. 4 are illustrated as follows.

Step 1 (@IO): The IO sends the request to the CS that the IO is going to outsource an image X to the cloud.

Step 2 (@CS): The CS forwards $\{E^{\times}_{pk_c}(W_c), E^{\times}_{pk_{oc}}(W_c)\}$ to the IO.

Step 3 (@IO and CS): In order to check whether the plaintexts of $E^{\times}_{pk_c}(W_c)$ and $E^{\times}_{pk_{\prod_{oc}}}(W_c)$ are the same, the **SEqVer** sub-protocol is run between the IO and the CS. If they are the same, then continue. Otherwise, the IO and CS re-execute Step 2 and Step 3.

Step 4 (@IO and CS): To strengthen the CS's encrypted watermark $E^{\times}_{pk_{oc}}(W_c)$, the **SStrenOper** sub-protocol is executed between the IO and the CS, producing the CS' encrypted and strengthened watermark $E^{\times}_{pk_{oc}}(SW_c)$.

Step 5 (@IO): (1) The IO calculates his/her own encrypted and strengthened watermark $E^{\times}_{pk_{oc}}(SW_o)$, concatenates it with $E^{\times}_{pk_{oc}}(SW_c)$ and appends $E^{\times}_{pk_{oc}}(1)$ until the length of the combined watermark denoted as $E^{\times}_{pk_{oc}}(SW_{oc})$ is $W \times H$. Fig. 2 depicts the process of generating $E^{\times}_{pk_{oc}}(SW_{oc})$.

Fig. 2. Generation of the combined encrypted watermark $E^{\times}_{pk_{oc}}(SW_{oc})$.

(2) The IO selects a key K_o to scramble the $E^{\times}_{pk_{oc}}(SW_{oc})$ and obtains $K_o[E^{\times}_{pk_{oc}}(SW_{oc})] = \{sw_{oc,i}(h_{oc})^{r_{oc,i}} \mod N, g^{r_{oc,i}} \mod N\}(i = 1, \cdots, W \times H)$.

(3) The IO calculates the product of his/her own strengthened watermark as $Mul_o = \prod_{i=1}^{L} sw_{o,i}$ and $InterR_o = \prod_{i=1}^{W\times H} h_o^{r_{oc,i}} \mod N$. Then, $K_o[E^{\times}_{pk_{oc}}(SW_{oc})]$, Mul_o and $InterR_o$ are put forward to the CS.

Step 6 (@CS): (1) The CS computes $Mul_c = \prod_{i=1}^{W\times H} sw_{c,i}$.

(2) The CS checks whether the equation $\frac{\prod_{i=1}^{W\times H} sw_{oc,i}(h_{oc})^{r_{oc,i}}}{(InterR_o)^{\theta_c}} \mod N = Mul_o \times Mul_c \mod N$ holds or not. Only if it holds does the CS proceed. Otherwise, the CS asks the IO for the right data.

(3) The CS executes the **MCR** algorithm to fresh $K_o[E^{\times}_{pk_{oc}}(SW_{oc})]$ using $\{r'_{oc,i}\}(i = 1, 2, \cdots, W \times H)$, and obtains a fresh version. The refreshed result is scrambled by the CS using its scrambling K_c. The scrambled result is marked as $K_c\{K_o[E^{\times}_{pk_{oc}}(LW_{oc})]\} = \{sw_{oc,i}(h_{oc})^{R_{oc,i}} \mod N, g^{R_i} \mod N\}(i = 1, 2, \cdots, W \times H)$, where $R_{oc,i} = r_{oc,i} + r'_{oc,i}$.

(4) The CS computes $InterR_c = \prod_{i=1}^{W\times H} (h_c^{R_{oc,i}}) \mod N$ and then transmits $K_c\{K_o[E^{\times}_{pk_{oc}}(SW_{oc})]\}$, Mul_c and $InterR_c$ to the IO.

(5) $K_c\{K_o[E^{\times}_{pk_{oc}}(SW_{oc})]\}$ is saved in the cloud for image retrieval.

Step 7 (@IO): (1) The IO validates whether SW_o and SW_C are still included in $K_c\{K_o[E^{\times}_{pk_{oc}}(SW_{oc})]\}$ by checking whether the formula $\frac{\prod_{i=1}^{W\times H} sw_{oc,i}(h_{oc})^{R_{oc,i}}}{(InterR_c)^{\theta_o}} \mod N = Mul_o \times Mul_c \mod N$ holds or not.

Only if it holds does the IO proceed. Otherwise, the IO asks the CS to send right data.

Step 8 (@IO and CS): The IO and the CS execute the **SCMul** sub-protocol as $X_{SW_{oc}} = SCMul(IO, CS, X, K_c\{K_o[E^{\times}_{pk_{oc}}(SW_{oc})]\})$, yielding the encrypted and watermarked image $E_{pk_{oc}}(X_{SW_{oc}})$ that is sent to the CS. After decryption of $E_{pk_{oc}}(X_{SW_{oc}})$ is finished, the plaintext $X_{SW_{oc}}$ is stored in cloud.

Step 9 (@CS): The CS returns $K_c\{K_o[E^{\times}_{pk_{oc}}(SW_{oc})]\}$, K_c, and $r'_{oc,i}(i = 1, 2, \cdots, W \times H)$ to the IO.

Step 10(@IO): (1) The IO uses his/her local $K_o[E^{\times}_{pk_{oc}}(SW_{oc})]$, K_c and $r'_{oc,i}(i = 1, 2, \cdots, W \times H)$ to re-execute Step (2) in **Step 6 (@CS)** and acquires $[K_c\{K_o[E^{\times}_{pk_{oc}}(SW_{oc})]\}]_o$.

(2) The IO judges the equivalence between $K_c\{K_o[E^{\times}_{pk_{oc}}(SW_{oc})]\}$ and $[K_c\{K_o[E^{\times}_{pk_{oc}}(SW_{oc})]\}]_o$. If unequal, the IO asks the CS for the right data.

(3) The IO transmits $E^{\times}_{pk_o}(K_o)$ and $E^{\times}_{pk_o}(K_c)$ to the CS.

Step 11(@CS): The CS stores $E^{\times}_{pk_o}(K_o)$ and $E^{\times}_{pk_o}(K_c)$ along with other data for the IO, as shown in Table 5.

TABLE 5
IO's Data in the Cloud in Image Upload Phase

ID_o	$X_{SW_{oc}}$	$K_c\{K_o[E^{\times}_{pk_{oc}}(SW_{oc})]\}$	$E^{\times}_{pk_o}(K_o), E^{\times}_{pk_o}(K_c)$

To clarify the image uploading phase, four important points are explained as follows:

- In (2) of **Step 5 (@IO)**, the IO cannot execute the **SCMul** sub-protocol to embed $K_o[E^{\times}_{pk_{oc}}(SW_{oc})]$ in-

to X and transmit $K_o[E^{\times}_{pk_{oc}}(SW_{oc})]$ to the CS for removing the watermark. It is because the IO may transmit a fake watermark that does not contain SW_o and SW_c to the CS. As a result, SW_o and SW_c cannot be removed in the image retrieval phase. To solve the problem, the CS verifies whether $K_o[E^{\times}_{pk_{oc}}(SW_{oc})]$ contains SW_o and SW_c in (2) **Step 6 (@CS)**.

- After the verification in (2) of **Step 6 (@CS)**, the CS still does not trust the IO. It is because the IO may give the CS a false watermark that contains SW_o and SW_c but does not equal the true watermark element by element. As shown in Fig. 3, the IO first concatenates $E^{\times}_{pk_{oc}}(SW_o)$ with $E^{\times}_{pk_{oc}}(1)$ and finally appends $E^{\times}_{pk_{oc}}(SW_c)$, producing a fake combined encrypted watermark, denoted as $[E^{\times}_{pk_{oc}}(SW_{oc})]'$. The IO conveys $K_o\{[E^{\times}_{pk_{oc}}(SW_{oc})]'\}$ to the CS. As a result, in the image retrieval phase, SW_c cannot be removed. To keep the IO from providing a fake removing watermark, the CS generates the removing watermark before embedding the watermark in (3)-(5) of **Step 6 (@CS)**.
- To prevent the IO from utilizing violent attack to reverse this scrambling key K_c, the CS refreshes the received watermark $K_o[E^{\times}_{pk_{oc}}(SW_{oc})]$ by running the **MCR** algorithm in (3) of Step 6 (@CS).
- K_c and K_o assist the IO to locate the watermark position and detect the watermark in the watermark detection phase.

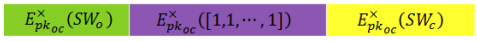

Fig. 3. Generation of the fake combined encrypted watermark watermark $[E^{\times}_{pk_{oc}}(SW_{oc})]'$.

5.2.3 *Image Retrieval*

When the IO requests to take back $X_{SW_{oc}}$ from the CS, the CS first removes the embedded watermark using the stored removing watermark marked as $K_c\{K_o[E^{\times}_{pk_{oc}}(SW_{oc})]\} = \{sw_{oc,i}(h_{oc})^{R_{oc,i}} \mod N, g^{R_{oc,i}} \mod N\}(i = 1, 2, \cdots, W \times H)$. Then, the recovered and encrypted image $E_{pk_{oc}}(X)$ is obtained and sent to the IO. After decrypting $E_{pk_{oc}}(X)$, the IO obtains his/her original image X. This process is illustrated as follows.

Step 1 (@IO): The IO sends a download request to the CS that the IO is going to take back the image X without degradation.

Step 2 (@CS): Receiving the IO's image retrieval request, the CS sends $\{g^{R_{oc,i}}\}(i = 1, 2, \cdots, W \times H)$ to the IO.

Step 3 (@IO and CS): The IO obtains his/her original image X by executing the **SCMul** sub-protocol formalized as: $X = SCMul(CS, IO, X_{SW_{oc}}, K_c\{K_o[E^{\times}_{pk_{oc}}(SW_{oc})^{-1}]\}\}$, where $K_c\{K_o[E^{\times}_{pk_{oc}}(SW_{oc})^{-1}]\} = \{[sw_{oc,i}(h_{oc})^{R_{oc,i}}]^{-1} \mod N, g^{R_{oc,i}} \mod N\}(i = 1, 2, \cdots, W \times H)$.

5.2.4 *Image Tracing*

Once the IO finds an image Y that is similar to X, the IO can take back the identity, scrambling keys, and the corresponding encrypted combined watermark from the CS. Then, the IO decides whether the CS is the illegal divulger if W_c exists in Y after executing the **SWaterDec** sub-protocol.

5.3 Four Security Computing Sub-protocols

In this subsection, we introduce four secure computing sub-protocols involved in SPIS protocols. Each sub-protocol securely implements a specific function in the ciphertext domain.

5.3.1 *Secure Combined Multiplication Sub-protocol (SCMul)*

The multiplicative spread spectrum watermarking technique proposed by Cox et al. [31] cannot be implemented through the multiplication homomorphism in the ciphertext domain. This is because the IO embeds $E^{\times}_{pk_{oc}}(W_{oc})$ into X by computing $E^{\times}_{pk_{oc}}(X_{W_{oc}}) = E^{\times}_{pk_{oc}}(X \cdot W_{oc}) = \{x_i \cdot w_{oc,i} \cdot h_{oc}^{r_i} \mod N, (g^{r_i})^{\theta_o} \mod N\}(i = 1, 2, \cdots, L)$. The computed result is sent to the CS. For decrypting $E^{\times}_{pk_{oc}}(X_{W_{oc}})$, $g^{\theta_o r}$ is sent to the CS. However, the CS can use $(g^r)^{\theta_o}$ to decrypt $E^{\times}_{pk_{oc}}(W_{oc})$.

To prevent the CS from decrypting $E^{\times}_{pk_{oc}}(W_{oc})$, the multiplicative homomorphism is combined with the additive homomorphism to generate the mixed ciphertext $E^{*}_{pk_{oc}}(W_{oc})$. The process of using $E^{*}_{pk_{oc}}(W_{oc})$ to accomplish watermark embedding is described as follows.

Step 1 (@IO and CS): The IO and CS commonly compute $E^{\times}_{pk_{oc}}(W_{oc}) = \{w_{oc,i}(h_{oc})^{r_i} \mod N, g^{r_i} \mod N\}(i = 1, 2, \cdots, L)$, where r_i is a random number.

Step 2 (@CS): The CS chooses $S = \{s_i\}(i = 1, 2, \cdots, W \times H)$ from $[1, \frac{N}{4}]$, and calculates $P = \{p_i | p_i = (h_{oc})^{s_i} \mod N\}$, and $T_1 = \{t_{1,i} | t_{1,i} = g^{\theta_c(r_i + s_i)} \mod N\}(i = 1, 2, \cdots, L)$. P is kept secret. T_1 is transmitted to the IO.

Step 3 (@IO): (1) After T_1 is received, the IO computes $(T_1)^{\theta_o}$ and its inverse $T_2 = [(T_1)^{\theta_o}]^{-1} = \{t_{2,i} | t_{2,i} = [(h_{oc})^{(r_i + s_i)}]^{-1} \mod N\}(i = 1, 2, \cdots, L)$.

(2) $E^{\times}_{pk_{oc}}(X \cdot W_{oc})$ is calculated as:
$\forall i = 1, 2, \cdots, L$
$E^{\times}_{pk_{oc}}(x_i \cdot w_{oc,i})$
$= \{x_i \cdot w_{oc,i}(h_{oc})^{r_i} \mod N, g^{r_i} \mod N\}$.

(3) $E^{*}_{pk_{oc}}(X \cdot W_{oc})$ is calculated using $E^{\times}_{pk_{oc}}(X \cdot W_{oc})$ as:
$\forall i = 1, 2, \cdots, L$
$E^{*}_{pk_{oc}}(x_i \cdot w_{oc,i})$
$= (g^{\theta_o r_{o,i}} \mod N)^N [1 + x_i \cdot w_{oc,i}(h_{oc})^{r_i} N] \mod N^2$.

(4) $E^{*}_{pk_{oc}}(X \cdot W_{oc} \cdot P^{-1})$ is calculated using $E^{*}_{pk_{oc}}(X \cdot W_{oc})$ and T_2 as: $E^{*}_{pk_{oc}}(X \cdot W_{oc} \cdot P^{-1}) = [E^{*}_{pk_{oc}}(X \cdot W_{oc})]^{T_2}$.

(5) The IO uses the **AddDecPSkey1** algorithm to calculate $[E^{*}_{pk_{oc}}(X \cdot W_{oc} \cdot P^{-1})]^{\lambda_o}$. Then, the IO sends $E^{*}_{pk_{oc}}(X \cdot W_{oc} \cdot P^{-1})$ and $[E^{*}_{pk_{oc}}(X \cdot W_{oc} \cdot P^{-1})]^{\lambda_o}$ to the CS.

Step 6 (@CS): (1) After receiving $\{E^{*}_{pk_{oc}}(X \cdot W_{oc} \cdot P^{-1}), [E^{*}_{pk_{oc}}(X \cdot W_{oc} \cdot P^{-1})]^{\lambda_o}\}$, the CS uses the **AddDecPSkey2** algorithm to calculate $X \cdot W_{oc}$ as:
$\forall i = 1, 2, \cdots, L$
$x_i \cdot w_{oc,i}$
$= L(\{[E^{*}_{pk_{oc}}(A_i)]^{\lambda_o} \cdot [E^{*}_{pk_{oc}}(A_i)]^{\lambda_c}\}^{p_i} \mod N^2)$,
where $A_i = x_i \cdot w_{oc,i} \cdot (p_i)^{-1} \mod N$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2020.3008957, IEEE Transactions on Services Computing

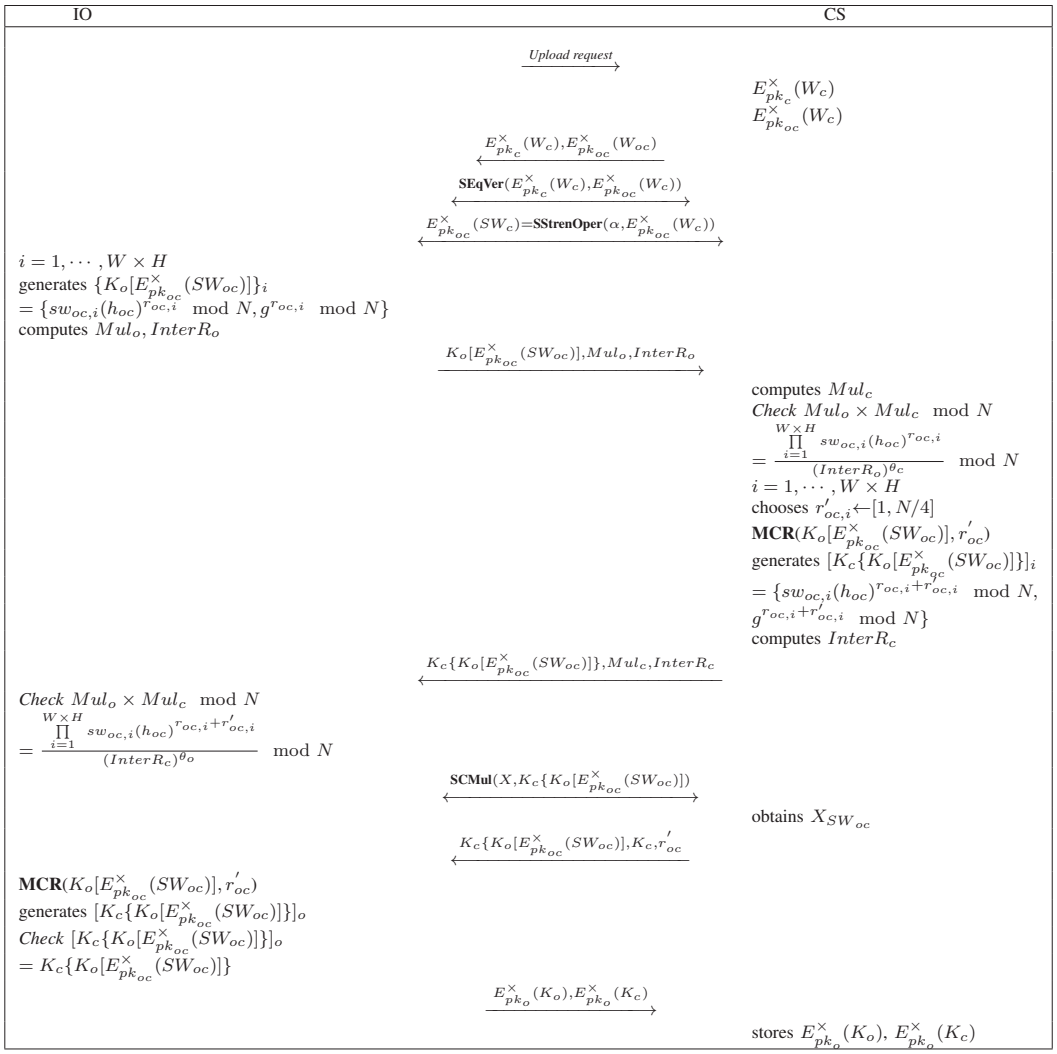

Fig. 4. Interactions between the IO and the CS in the image uploading phase.

In the **SCMul** sub-protocol, the IO has no ability to decrypt $E^{\times}_{pk_{oc}}(W_{oc})$ because the IO only can access $t_1 = g^{(r+s)\theta_c}$ but not $g^{\theta_c r}$ in **Step 3 (@IO)**. The CS also cannot decrypt $E^{\times}_{pk_{oc}}(W_{oc})$ as the **SCMul** sub-protocol prevents the calculation of $g^{\theta_o r}$.

The process of the **SCMul** sub-protocol is formulated as $X_{W_{oc}} = SCMul(IO, CS, X, E^{\times}_{pk_{oc}}(W_{oc}))$.

5.3.2 *Secure Strengthened Operation Protocol (SStrenOper)*

Before the watermark is embedded into the image, the watermark is strengthened with an embedding strength factor α to control the image's quantity after embedding. For instance, in the Cox et al. scheme [31], the embedding data are $1+\alpha W$, not singly W. Here, W represents the original watermark. The IO cannot use $E^{\times}_{pk_{oc}}(W_c)$ to compute $E^{\times}_{pk_{oc}}(1+\alpha W_c)$ through the property of the multiplication homomorphism since the strengthened operations contain the additive operation.

The CS can calculate $E^{\times}_{pk_{oc}}(1+\alpha W_c)$ and send it to the IO. The IO takes the responsibility to verify the correctness of $E^{\times}_{pk_{oc}}(1+\alpha W_c)$ to prevent the CS from offering false data. The length of W_c is L. The process of generating $E^{\times}_{pk_{oc}}(1+\alpha W_c)$ is described as follows.

Step 1 (@IO): The IO gives the CS the embedding strength factor $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_L)$.

Step 2 (@CS): (1) Receiving α, the CS calculates $1+\alpha W_c$.

(2) The CS chooses $r = \{r_i\}$ from $[1, \frac{N}{4}]$ and $r' = \{r'_i\}$ from $[1, \frac{N}{4}](i = 1, 2, \cdots, L)$ and runs the **MulEnc** algorithm to compute $E^{\times}_{pk_{oc}}(W_c)$ and $E^{\times}_{pk_{oc}}(1+\alpha W_c)$.

(3) The CS puts forward $E^{\times}_{pk_{oc}}(W_c)$ and $E^{\times}_{pk_{oc}}(1+\alpha W_c)$ to the IO.

Step 3 (@IO): (1) After receiving $E^{\times}_{pk_{oc}}(W_c)$ and $E^{\times}_{pk_{oc}}(1+\alpha W_c)$, the IO computes $E^{*}_{pk_{oc}}(\alpha W_c)$ and $E^{*}_{pk_{oc}}(1+\alpha W_c)$ by running the **AddEnc** algorithm.

(2) The IO sends $E^{*}_{pk_{oc}}(\alpha W_c)$ and $E^{*}_{pk_{oc}}(1+\alpha W_c)$ to the CS.

Step 4 (@CS): (1) The CS calculates $INC = [(h_{oc})^r]^{-1} \mod N$ and $INC' = [(h_{oc})^{r'}]^{-1} \mod N$.

(2) Receiving $E^*_{pk_{oc}}(\alpha W_c)$ and $E^*_{pk_{oc}}(1+\alpha W_c)$, the CS computes $[E^*_{pk_{oc}}(\alpha W_c)]^{INC}$, $[E^*_{pk_{oc}}(1+\alpha W_c)]^{INC'}$ and the product of elements of them as:
$\prod_{i=1}^{L}[E^*_{pk_{oc}}(\alpha W_c)]^{INC}=\{M_{1,1},M_{1,2}\}$;
$\prod_{i=1}^{L}[E^*_{pk_{oc}}(1+\alpha W_c)]^{INC'}=\{M_{2,1},M_{2,1}\}$.

(3) The CS forwards the IO $M_{1,1}$, $M_{1,2}$, $M_{2,1}$ and $M_{2,2}$.

Step 5 (@IO): (1) After receiving $\{M_{1,1},M_{1,2},M_{2,1},M_{2,2}\}$, the IO uses the **AddDecWkey** algorithm to calculate the decrypted results as:
$F_1=L\{\frac{M_{1,1}}{[(M_{1,2})^{\theta_o} \mod N]^N \mod N^2}\}$;
$F_2=L\{\frac{M_{2,1}}{[(M_{2,2})^{\theta_o} \mod N]^N \mod N^2}\}$.

(2) The IO checks whether the equation $F_1+\sum_{i=1}^{L}1=F_2$ holds or not. If it holds, the IO is sure that $F_1=\sum_{i=1}^{L}(\alpha_i w_{c,i})$, $F_2=\sum_{i=1}^{L}(1+\alpha_i w_{c,i})$ and that $E^{\times}_{pk_{oc}}(1+\alpha M_c)$ sent from the CS is correct. If not, the IO asks the CS for the right data.

The process of the **SStrenOper** sub-protocol is formulated as $E^{\times}_{pk_{oc}}(1+\alpha W_c)=SStrenOper(IO,CS,\alpha,E^{\times}_{pk_{oc}}(W_c))$.

To clarify the **SStrenOper** sub-protocol, an important point is explained. In (3) of Step 4 (@CS), the CS sends the product of elements of the $[E^*_{pk_{oc}}(\alpha W_c)]^{INC}$ and $[E^*_{pk_{oc}}(1+\alpha W_c)]^{INC'}$ instead of the single elements to the IO. This is to prevent collusion between the IO and the MA. Consequently, the IO obtains the strong key λ to decrypt $[E^*_{pk_{oc}}(\alpha W_c)]^{INC}$ and $[E^*_{pk_{oc}}(1+\alpha W_c)]^{INC'}$ and obtains W_c.

5.3.3 *Secure Equivalent Verifying Protocol (SEqVer)*

The encrypted watermark $E^{\times}_{pk_c}(W_c)=\{w_{c,i}(h_c)^{r_{c,i}} \mod N, g^{r_{c,i}} \mod N\}$ and $E^{\times}_{pk_{oc}}(W_c)=\{w_{oc,i}(h_{oc})^{r_{oc,i}} \mod N, g^{r_{oc,i}} \mod N\}(i=1,2,\cdots,L)$ belong to the CS under pk_c and pk_{oc}, respectively. The IO verifies whether the plaintexts of $E^{\times}_{pk_c}(W_c)$ and $E^{\times}_{pk_{oc}}(W_c)$ are the same without decryption. The validation process using the property of the multiplicative homomorphism is described as follows.

Step 1 (@CS): The CS sends $E^{\times}_{pk_c}(W_c)$ and $E^{\times}_{pk_{oc}}(W_c)$ to the IO.

Step 2 (@IO): (1) After $E_{pk_c}(W_c)$ and $E_{pk_{oc}}(W_c)$ are received, the IO selects $r_{o,i}(i=1,2,\cdots,L)$ randomly from $[1,\frac{N}{4}]$ to compute a middle result M_1 as: $M_1=\frac{g^{\theta_o(r_{oc}+r_o)}}{g^{r_c}} \mod N$.

(2) The IO chooses a scrambling key K_o to scramble M_1, obtaining $K_o(M_1)$ and sending it to the CS.

Step 3 (@CS): (1) Upon receiving $K_o(M_1)$, the CS computes $[K_o(M_1)]^{\theta_c}=K_o({M_1}^{\theta_c})$.

(2) $K_o(M_1^{\theta_c})$ is returned to the IO.

Step 4 (@IO): (1) After receiving $K_o(M_1^{\theta_c})$, the IO uses K_o to inversely scramble $K_o(M_1^{\theta_c})$ and recover $M_1^{\theta_c}$ as: $M_1^{\theta_c}=\frac{h_{oc}^{(r_{oc}+r_o)}}{h_c^{r_c}} \mod N$.

(2) The IO calculates $(h_{oc})^{r_o}$ to verify whether the equation $\frac{W_{oc}\cdot h_{oc}^{r_{oc}}}{W_c\cdot h_c^{r_c}}\times\frac{(h_{oc})^{r_o}}{M_1^{\theta_c}} \mod N=1$ holds or not. If it holds, then the IO believes plaintexts $E_{pk_c}(w_c)$ and $E_{pk_{oc}}(W_c)$ are the same and sends K_o and $r_{o,i}(i=1,2,\cdots,L)$ to the CS. Otherwise, the IO can ask the CS for the correct data.

Step 5 (@CS): (1) The CS uses the received K_o to re-scramble $[K_o(M_1)]^{\theta_c}$ and obtains M_1^{θ}.

(2) The CS uses the received r_o to calculate $(h_{oc})^{r_o}$ and then checks whether the equation $M_1^{\theta_c} \mod N=\frac{(h_{oc})^{r_{oc}}+(h_{oc})^{r_o}}{(h_c)^{r_c}} \mod N$ holds or not. If it holds, the CS is sure that the IO is honest. Otherwise, the CS can accuse the IO of dishonesty in cheating W_{oc}.

To clarify the **SEqVer** sub-protocol, two important points are explained.

- Step 5 (@CS) is used to stop the IO from sending the CS fake $(M_1)'$ as: $(M_1)'=g^{\theta_o(r_{oc}+r_o)} \mod N$. The CS returns $[(M_1)']^{\theta_c}$ to the IO. The IO can obtain $h_{oc}^{r_{oc}}=\frac{[(M_1)']^{\theta_c}}{h_{oc}^{r_o}} \mod N$ that can be used to decrypt $E^{\times}_{pk_{oc}}(W_c)$.
- In Step 2 (@IO), the K_o cannot be guessed with an exhaustive search method in the presence of r_o.
- K_o can prevent the CS from providing fake data that can pass **Step 4 (@IO)** when the two plaintexts involved are not the same. For instance, $E^{\times}_{pk_c}(E)=\{e_i g^{\theta_c r_{e,i}} \mod N, g^{r_{e,i}} \mod N\}$ and $E^{\times}_{pk_{oc}}(F)=\{f_i(h_{oc})^{r_{f,i}} \mod N, g^{r_{f,i}} \mod N\}(i=1,2,\cdots,L)$, where $e_i\neq f_i$ are used to test whether the plaintexts contained are consistent. With the help of K_o, the CS can only create $\frac{e_i}{f_i}\times K_o[(\frac{g^{\theta_o(r_{e,i}+r_{o,i})}}{g^{r_{f,i}}})^{\theta_c}]$ instead of the true data $K_o[\frac{e_i}{f_i}\times(\frac{g^{\theta_o(r_{e,i}+r_{o,i})}}{g^{r_{f,i}}})^{\theta_c}]$ that can pass **Step 4 (@IO)**.

The process of the **SEqVer** sub-protocol is formulated as $True/False=SEqVer(IO,CS,E^{\times}_{pk_{oc}}(W_c),E^{\times}_{pk_{oc}}(W_{oc}))$.

5.3.4 *Secure Watermark Detection Sub-protocol (SWaterDec)*

The IO decides if an image Y that is similar to X contains CS's identical watermark W_c. The IO takes back $K_c\{K_o[E^{\times}_{pk_{oc}}(SW_{oc})]\}$, which is in the encrypted form. For brevity, $K_c\{K_o\{[E^{\times}_{pk_{oc}}(SW_{oc})]\}$ is denoted as $E^{\times}_{pk_{oc}}(W_{oc})=\{w_{oc,i}(h_{oc})^{r_i} \mod N, g^{r_i} \mod N\}(i=1,2,\cdots,L)$.

The **SWaterDec** sub-protocol that is used to realize the watermarking detection in the ciphertext domain is described as follows.

Step 1 (@IO): (1) The IO uses Y and $E^{\times}_{pk_{oc}}(W_{oc})$ to calculate $E^*_{pk_{oc}}(Y\cdot W_{oc})$.

(2) $E^*_{pk_{oc}}(Y\cdot W_{oc})$ is sent to the CS.

Step 2 (@CS): (1)Receiving $E^*_{pk_{oc}}(Y\cdot W_{oc})$, the CS calculates the inverse element $INC=(g^{\theta_c r})^{-1} \mod N$, $[E^*_{pk_{oc}}(Y\cdot W_{oc})]^{INC}$ and the product of elements of it as :
$\prod_{i=1}^{L}[E^*_{pk_{oc}}(Y\cdot W_{oc})]^{INC}=\{M_1,M_2\}$.

(2) $(M_2)^{\lambda_c}$ are obtained by running the **AddDecPSkey1** algorithm.

(4) The CS launch M_1, M_2 and $(M_2)^{\lambda_c}$ to the IO.

Step 3 (@IO): (1) Using the received M_2, the IO runs the **AddDecPSkey2** algorithm to compute the correlation value $corr_1$ as:

$$corr_1 = \frac{L[(M_2)^{\lambda_o} \cdot (M_2)^{\lambda_c}]}{\sqrt{\sum_{i=1}^{L}(y_i)^2}} = \frac{\sum_{i=1}^{L}(y_i \times w_{oc,i})}{\sum_{i=1}^{L}(y_i)^2}.$$

Using the **AddDecWkey** algorithm, the IO computes the correlation value $corr_2$ as:

$$corr_2 = \frac{L\{(M_2) \backslash \{[(M_1)^{\theta_o}]^N \bmod N^2\}\}}{\sqrt{\sum_{i=1}^{L}(y_i)^2}} = \frac{\sum_{i=1}^{L}(y_i \cdot w_{oc,i})}{\sum_{i=1}^{L}(y_i)^2}.$$

(3) The IO compares $corr_1$ with $corr_2$. If not equal, the IO asks the CS for right data. As a result, repeat **Step 1 (@IO)** to **Step 3 (@IO)**. Otherwise, the final correlation value $corr = corr_1 = corr_2$ is obtained. The IO compares $corr$ with a detection threshold δ. If $corr \geq \delta$ holds, then the IO makes sure that Y contains W_{oc}.

To clarify the **SWaterDec** sub-protocol, an important point is explained. In (4) of Step 2 (@CS), the CS does not launch the single element of $[E^*_{pk_{oc}}(Y \cdot W_{oc})]^{INC}$ element by element. This is to prevent collusion between the IO and the MA, where the IO obtains the strong key λ to decrypt $[E^*_{pk_{oc}}(Y \cdot W_{oc})]^{INC}$and obtains W_{oc}.

The process of the **SWaterDec** sub-protocol is formulated as $corr = SWaterDec(IO, CS, Y, E^{\times}_{pk_{oc}}(W_{oc}))$.

6 Security Analysis

Limited by the pages, we gave strict security analysis on the restrained Pallier in our previous work [28]. In this section, we analyze the security of these four secure computing sub-protocols, that is the **SCMul** sub-protocol, the **SStrenOper** sub-protocol, the **SEVer** sub-protocol, and the **SWDec** sub-protocol, before elaborating the security of SPIS protocols.

6.1 Security of Computing Sub-Protocols

The general case definitions of the security model can be referred to in [35]. Based on the security model, we prove the security of our computing sub-protocols. We consider the presence of semi-honest adversaries for achieving an ideal functionality. We use the scenario involving three parties, the IO, the CS and the MA. The IO can collude with the MA to obtain the strong key λ. The CS can also collude with the MA to obtain λ. For the sake of simplicity, the strengthened watermarks SW_{oc} are simplified to W_{oc}.

We construct two simulators $Sim = (Sim_{IO}, Sim_{CS})$ against two kinds of adversaries $\mathcal{A} = (\mathcal{A}_{IO}, \mathcal{A}_{CS})$ that corrupt the IO and the CS, respectively.

Theorem 1: The **SCMul** sub-protocol described in Section 5.3.1 can securely compute multiplication of a plaintext and a ciphertext in the present of semi-honest adversaries $\mathcal{A} = (\mathcal{A}_{IO}, \mathcal{A}_{CS})$.

Proof. We construct two simulators $Sim = (Sim_{IO}, Sim_{CS})$.

Sim_{CS} simulates $\mathcal{A}_{CS}$ as follows: First, it generates encryption $E^{\times}_{pk_{oc}}(\widehat{W_{oc}})$ on randomly chosen $\widehat{W_{oc}}$, randomly chooses $s_i \in [1, \frac{N}{4}]$, calculates T_1. Sim_{CS} and sends $E^{\times}_{pk_{oc}}(\widehat{W_{oc}})$ and T_1 to $\mathcal{A}_{CS}$. If $\mathcal{A}_{CS}$ replies with $\perp$, then Sim_{CS} returns $\perp$.

The view of $\mathcal{A}_{CS}$ consists of the multiplicative encrypted data it creates. The views of $\mathcal{A}_{CS}$ on the real and the ideal executions consist of $E^{\times}_{pk_{oc}}(\widehat{W_{oc}})$ and T_1. In the real world, this is guaranteed by the semantic security of the **MulEnc** algorithm that is proven in [28], even though the MA is dishonest. The views of $\mathcal{A}_{CS}$ in the real and the ideal executions are indistinguishable.

Sim_{IO} simulates $\mathcal{A}_{IO}$ as follows: First, it calculates $T_2 = \{t_{2,i} | t_{2,i} = (h_{oc})^{r_i} \bmod N\}$ using the randomly chosen number $r_i \in [1, \frac{N}{4}]$. Second, it generates fictitious encryption $E^{\times}_{pk_{oc}}(\widehat{XW})$ by running the **MulEnc** algorithm on randomly chosen $\widehat{XW}$ and then calculates $E^*_{pk_{oc}}(\widehat{XW})$, $[E^*_{pk_{oc}}(\widehat{XW})]^{T_2}$ and $\{[E^*_{pk_{oc}}(\widehat{XW})]^{T_2}\}^{\lambda_o}$ using the **AddDecPSkey1** algorithm. Sim_{IO} sends $[E^*_{pk_{oc}}(\widehat{XW})]^{T_2}$ and $\{[E^*_{pk_{oc}}(\widehat{XW})]^{T_2}\}^{\lambda_o}$ to $\mathcal{A}_{IO}$. If $\mathcal{A}_{IO}$ replies with $\perp$, then Sim_{IO} returns $\perp$.

The view of $\mathcal{A}_{IO}$ consists of the encrypted data it creates. In both the real and the ideal executions, $\mathcal{A}_{IO}$ receives the outputs $[E^*_{pk_{oc}}(\widehat{XW})]^{T_2}$ and $\{[E^*_{pk_{oc}}(\widehat{XW})]^{T_2}\}^{\lambda_o}$. In the real world, this is guaranteed by the semantic security of the **AddEnc** algorithm and the **MultoMix** algorithm that are proven in [28], even though the MA is dishonest. The views of $\mathcal{A}_{IO}$ in the real and the ideal executions are indistinguishable. □

The security proofs of the **SStrenOper**, **SEqVer**, and **SWaterDec** sub-protocols are similar to that of the **SCMul** sub-protocol under the semi-honest adversaries $\mathcal{A} = (\mathcal{A}_{IO}, \mathcal{A}_{CS})$. All the calculations are performed over ciphertext domains that are secure due to the semantic security of the restrained Paillier cryptosystem [28].

6.2 Security of SPIS Protocols

In this section, we will illustrate that SPIS protocols are secure under an active adversary $\mathcal{A}^*$ defined in Section 4.3. For the sake of simplicity, the strengthened watermarks SW_{oc} are simplified to W_{oc} in this subsection.

6.2.1 Security of Image outsourcing phase

If $\mathcal{A}^*$ eavesdrops on the transmission between the challenge IO and the CS, the original encrypted data, such as $E^{\times}_{pk_c}(W_c)$, $E^{\times}_{pk_{oc}}(W_c)$ and $E^{\times}_{pk_{oc}}(W_{oc})$, and the final results $E^{\times}_{pk_{oc}}(X_{W_{oc}})$ and $E^*_{pk_{oc}}(X_{W_{oc}})$ may be obtained by $\mathcal{A}^*$. Moreover, middle ciphertext results acquired by running the **SCMul**, **SStrenOper**, and **SEqVer** sub-protocols transmitted between the IO and the CS may also be obtained by $\mathcal{A}^*$.

First, $\mathcal{A}^*$ will not be able to decrypt the multiplicative ciphertexts without the challenge parties' private keys due to the semantic security of the **MulEnc** algorithm. Second, suppose $\mathcal{A}^*$ has compromised the IO or the CS to obtain the partial strong key λ_o or λ_c. However, $\mathcal{A}^*$ is unable to recover the strong private key λ. Even if $\mathcal{A}^*$ has compromised both the IO and the CS at the same time or compromised the MA and obtains λ, $\mathcal{A}^*$ still cannot decrypt the encrypted and watermarked image given the property of the mixed ciphertexts that λ cannot decrypt. Third, $\mathcal{A}^*$ may compromise the CS to obtain the locations of the embedding watermark. However, each location of the IO's image is pretended to be embedded with the watermark so that $\mathcal{A}^*$ cannot distinguish specific embedding locations. $\mathcal{A}^*$ may compromise the IO to provide the CS a fake watermark such that CS cannot remove the CS's watermark W_c, and $\mathcal{A}^*$

can obtain an image watermarked with W_c after retrieving the IO's uploaded image. This issue is addressed as the CS actively joins the construction of the embedding watermark to prevent the IO from faking the watermark.

6.2.2 Security of Image Retrieval phase

The security of the image retrieval phase is based on that of the **SCMul** sub-protocol.

6.2.3 Security of Image Tracing Phase

The security of the image tracing is based on the security of the **SWaterDec** sub-protocol, which employs both the semantic security of the **MulEnc** algorithm and the characteristic of the **SCMul** sub-protocol.

6.3 Analysis of the Security Requirements

In this subsection, we show that the proposed protocol can achieve the design goals described in Section 4.4.

Data Security: The transmitted data are either multiplicative ciphertexts or mixed ciphertexts. These ciphertexts can only be decrypted by the receiver, which is guaranteed by the **MulEnc** algorithm.

Traceability: Tracing the dishonest party is realized by detecting the dishonest party's identity watermark, which is achieved by the **SWaterDec** sub-protocol.

Fairness: No one can access the others' plaintext watermarks to fake the watermarked images with the others' watermark.

Conspiracy Problem: The MA cannot obtain each party's private key and cannot decrypt their own encrypted watermark. Even if the IO or the CS conspire with the MA and the MA gives the IO or the CS the strong key λ, the encrypted watermarks still fail to be decrypted.

Recovery of Image Quality: The CS removes the embedded watermark and returns the original image to the IO .

Right to Be Forgotten: The IO demands the CS to delete his/her image, but the CS does not perform deletion. If the IO meets the image that is asked to be deleted, the **SWaterDec** sub-protocol is performed to detect the CS's identity watermark in front of the **J**. If the CS's identity watermark is detected, the **J** convicts the CS of breaking the right to be forgotten.

7 Performance Analysis

In this section, we first give the functionality comparisons among our scheme and several related schemes and then analyse the performance of the proposed protocols in terms of theoretical analysis and experimental results over both the computational overhead and the communication cost.

7.1 Functionality Comparison

We give the functionality comparison of our scheme with the relevant scheme in [19] and [20]. As shown in Table 6, our scheme is the only scheme that can satisfy all of the following properties: image display, image lossless retrieval, protection of private data in the case of semi-honest third party, resistance to collusion with the third party, and hiding the positions of embedded watermark.

7.2 Theoretical Analysis

We assume that one regular exponentiation operation with an exponent of length $|N|$ requires $1.5|N|$ multiplications [36] (i.e., a length of r is $|N|$ and that computing g^r occupies $1.5|N|$ multiplications). Since the exponentiation operation introduces significantly higher cost than the addition and multiplication operations, we ignore the fixed numbers of addition and multiplication operations in our analysis. For the restrained-Paillier cryptosystem [28], the bit length of random is chosen as $|r| = \frac{|N|}{4}$, and the bit length of the private key is set as $|\theta| = \frac{|N^2|}{2} \approx |N|$. The additive ciphertext $E^{+}_{pk}(x)$ needs $3|N|$ to transmit. The multiplicative ciphertext $E^{\times}_{pk}(x)$ needs $2|N|$ to transmit.

7.2.1 Theoretical Costs of Secure Computing Sub-protocols

The computational (Compu.) costs of the **SCMul**, **SStrenOper**, **SWaterDec** and **SEqVer** sub-protocols are $O(|N|)$ multiplications. In order to compare the computational cost of participants, the specific computational consumption is presented in Table 7. The length of involved vectors is denoted L, $L \ll N$, and the length of the scrambling keys involved are denoted as $|N|$.

From Table 7, we observe that both the computational costs of the **SStrenOper** and **SWaterDec** sub-protocols are higher than those of the **SCMul** and **SEqVer** sub-protocols, as the **SStrenOper** and **SWaterDec** sub-protocols employ high consumption algorithms, namely, the **AddEnc**, **AddDecPSkey1** and **AddDecPSkey2** algorithms. The **SEqVer** sub-protocol consumes the least computational overhead. This is because the **SEqVer** sub-protocol only exploits the lightweight **MulEnc** algorithm.

As shown in Table 7, the calculation costs of the **SStrenOper** and **SWaterDec** sub-protocols consist of the L term and constant term, such as $(4.5L + 15)|N|$. This is because all elements of an involved vector participate in the operation after summation or multiplication instead of each single element, such as **Step 4 (@CS)** of the **SStrenOper** sub-protocol and **Step 2 (@CS)** of the **SWaterDec** sub-protocol.

Communication (Commu.) costs of the four sub-protocols are $O(|N|)$ bits. The last row of Table 7 presents the concrete communication overheads.

7.2.2 Theoretical Costs of SPIS Protocols

In the SPIS protocols, the length of the mentioned watermark vectors is denoted as L_W, ,$L_W \ll N$, the size of relative images is denoted as L_I, $L_I \ll N$, the length of the scrambling keys and weighting factors of the watermark involved are $|N|$. The CS's watermark $E^{\times}_{pk_c}(W_c)$ and $E^{\times}_{pk_{oc}}(W_c)$ under different public keys have been ready, and the IO has pre-computed $E^{\times}_{pk_o}(W_o)$, $E^{\times}_{pk_{oc}}(W_o)$ and $E^{\times}_{pk_{oc}}(1, 1, \cdots, 1)$. The computational cost of these ready-made ciphertext data is not accumulated. Furthermore, the system initialization phase is considered to run only once, and thus the overhead of such a phase is little and can be ignored. The computational costs of each party in each phase are $O(|N|)$. To clarify the differences of the consumption, Table 8 shows the concrete computational costs

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2020.3008957, IEEE Transactions on Services Computing

TABLE 6
Function Comparison with Other Protocols

	Image display	Image lossless retrieval	Semi-honest third party	Hiding embedding positions of watermark	Cryptosystem
Dong [19]	√	×	×	×	Paillier [21]
Liu [20]	√	√	×	×	Paillier [21]
Proposed protocols	√	√	√	√	restrained Paillier [28]

TABLE 7
Costs of the SCMul, SWOper and SWaterDec Sub-protocols

	SCMul	SStrenOper	SWaterDec	SEqVer
IO	$7.875L\|N\|$	$(9L+3)\|N\|$	$(2.25L+6)\|N\|$	$3.375L\|N\|$
CS	$6.75L\|N\|$	$9L\|N\|$	$(3L+3)\|N\|$	$1.875L\|N\|$
Compu.	$14.625L\|N\|$	$(18L+3)\|N\|$	$(5.25L+9)\|N\|$	$5.25L\|N\|$
Commu.	$5L\|N\|$	$(10.5L+6)\|N\|$	$(3L+5)\|N\|$	$(6.25L+1)\|N\|$

TABLE 8
Costs of SPIS Protocols in Theory

	Image Outsourcing	Image Retrieval	Image Tracing
IO	$(13.5L_W+10.5L_I+18)\|N\|$	$9L_I\|N\|$	$(4.5L_W+9)\|N\|$
CS	$(12L_W+12L_I+9)\|N\|$	$9L_I\|N\|$	$(3L_W+3)\|N\|$
Compu.	$(25.5L_W+22.5L_I+27)\|N\|$	$18L_I\|N\|$	$(7.5L_W+12)\|N\|$
Commu.	$(17L_W+9L_I+10)\|N\|$	$6L_I\|N\|$	$(3L_W+9)\|N\|$

in SPIS protocols, including the image outsourcing phase, the image retrieval phase and the image tracing phase. Communication costs in SPIS protocols are $O(|N|)$ bits. The last row of Table 8 presents the specific communication overheads, and B represents 8 bits.

As shown in Table 8, the computation cost and communication cost of the image upload phase are the highest since outsourcing an image dispatches the **SCMul**, **SStrenOper**, and **SEqVer** sub-protocols. The image retrieval phase only uses the **SCMul** sub-protocol. The image tracing phase only uses the **SWaterDec** sub-protocol. The overheads of the image tracing phase in Table 8 are greater than those of the **SWaterDec** sub-protocol because the image tracing phase not only executes the **SWaterDec** sub-protocol but also performs the retrieval and decryption scrambling keys.

7.3 Experimental Analysis

In this section, we evaluate the time consumption and communication cost of four secure computing sub-protocols and SPIS protocols. We perform the experiments using a personal computer powered by an Intel(R) Core(TM) i5-4490 @3.30 GHz processor, 8 GB of RAM memory and a Windows 7 professional operating system. The experimental results are evaluated as an average over 1000 times using a custom simulator built in Java.

7.3.1 Experimental Costs of Secure Computing Sub-protocols

Secure computing sub-protocols involve the IO and the CS. For the sake of brevity, we use the party i to replace the IO and the party j to replace the CS. Any positive integer x is involved in the three sub-protocols, and x is limited in the range of $[0, R]$, where $Len(R) < \frac{Len(N)}{4}$.

The computation and communication cost of the **SCMul** sub-protocol increases only as $|N|$ increases, as shown in Table 9. The performance in the **SStrenOper**, **SEqVer**, and **SWaterDec** is shown in Fig. 5(a) - Fig. 5(f).

As seen from Fig. 5(a) - Fig. 5(f), we find that both computational and communication overheads of sub-protocols not only increase with the length of N but also with the length of the involved vector L. The **SEqVer** sub-protocol only relies on the **MulEnc** algorithm. Thus, all the overheads of the **SEqVer** sub-protocol are much lower than those of the **SStrenOper** sub-protocol and **SWaterDec** sub-protocol.

7.3.2 Experimental Costs of SPIS Protocols

In this implementation, we show the results of applying 2D DCT on non-overlapping 8×8 blocks of the greyscale Lena image of size 256×256. We randomly choose 1024 DCT blocks, and then we take the DCT coefficients from the third position to the tenth position of each chosen block in the classical Zig-Zag order to make up the carrier $X = \{x_1, x_2, \cdots, x_{8192}\}$. We choose a 1024-bit length of N and achieve 80-bit security levels [36]. The watermark embedding strengths α for uploading an image twice are set to 0.06. and 0.6. The length of W_{IO} and W_{CS} is 2048, one-fourth of the length of X. The scaling factor Q is 2^{10} for quantizing X, W_{IO}, W_{CS}, and α into integers.

The computational cost and communication overheads are recorded in Table 10, where the time consumption is at the millisecond level.

TABLE 10
Costs of SPIS Protocols in Practice

	Image Outsourcing	Image Retrieval	Image Tracing
IO	44123.0 ms	15268.0 ms	2929.0 ms
CS	53887.0 ms	44158.0 ms	3249.0 ms
Commu.	6384334.0 B	3127173.0 B	391191.0 B

The visual quality metric to measure the quality of the greyscale image after embedding the watermark is the peak signal-to-noise ratio (PSNR) since PSNR is more suitable for greyscale images [37]. The experimental results of the Lena image are shown in Fig. 6(a)-Fig. 6 (c).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2020.3008957, IEEE Transactions on Services Computing

TABLE 9
Performance of the SCMul Sub-protocol with Different $|N|$

Sub-protocol \ $\|N\|$		512	768	1024	1280	1536	1792	2048
SCMul	Party i	5.363 ms	15.01 ms	33.941 ms	67.175 ms	118.771 ms	195.344 ms	282.641 ms
	Party j	4.158 ms	11.814 ms	26.649 ms	52.497 ms	92.991 ms	152.886 ms	221.537 ms
	Commu.	318.258 B	476.986 B	636.975 B	797.002 B	957.0420 B	1118.264 B	1277.239 B

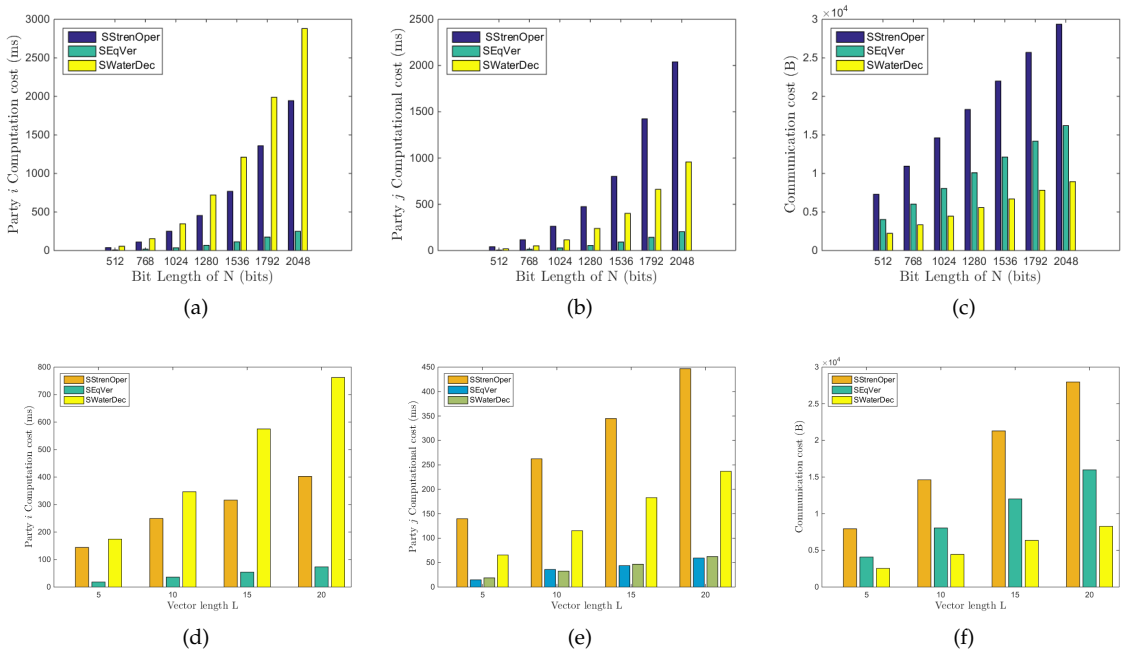

Fig. 5. (a) Run time on party i with different bit length of N at $L = 10$. (b) Run time on party j with different bit length of N at $L = 10$. (c) Communication costs with different bit length of N at $L = 10$. (d) Run time on Party i with different vector length L at $|N| = 1024$. (e) Run time on party j with vector length L at $|N| = 1024$. (f) Communication costs with different vector length L at $|N| = 1024$.

Fig. 6. (a) Watermarked image at $\alpha = 0.06$. (b) Watermarked image embedded at $\alpha = 0.6$. (c) Retrieved image.

Fig. 6(a) shows that in the image uploading phase, the PSNR of the watermarked image embedded with W_{oc} is 55.63 dB when α is 0.06. The PSNR of the watermarked image embedded with W_{oc} is 34.43 dB, as shown in Fig. 6(b), when α is 0.6. As shown in Fig. 6(c), the PSNR of the retrieved image achieves infinity. This proves that the IO can recover his/her original image in the image retrieval phase.

In the image tracing phase, when α is 0.06, the correlation $corr_o$ between the watermark W_c and its carrier $X_{W_{oc}}$ is 18.1126 and the correlation $corr_c$ between W_c and $X_{W_{oc}}$ is 18.8784. Both $corr_o$ and $corr_c$ are higher than the detection threshold δ that is mentioned in Section 3.2.2. Thus, the traceability of the proposed protocol can be guaranteed and effective.

8 Conclusion

In this paper, we propose secure plaintext image storage (SPIS) protocols integrated with multiple functions. SPIS protocols achieve plaintext image storage, display and deletion in the cloud environment with privacy-preserving content. The image owner can control the quality of the uploaded images and trace the cloud's misbehaviour such

as leaking images and violating the right to be forgotten. The cloud can store the plaintext image with reduction in the storage space. Considering a semi-honest third party, any conspiracy between a party and the third party is blocked. Our security analysis and the experimental results have shown that the SPIS protocols are sufficiently secure and efficient for a real-world deployment. As future work, we will study how to securely outsource the plaintext 3D model, which will allow us to refine the framework to address more complex and larger computations.

ACKNOWLEDGMENTS

This work was supported in part by the Natural Science Foundation of China under Grant U1636201 and 61572452, by Anhui Initiative in Quantum Information Technologies under Grant AHY150400.

REFERENCES

[1] Zephoria. *The Top 20 Valuable Facebook Statistics-Updated September 2018*. [Online]. Available: https://zephoria.com/top-15-valuable-facebook-statistics/.

[2] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: An information flow tracking system for real-time privacy monitoring on smartphones," *ACM Trans. Comput. Syst.*, vol. 32, no. 2, pp. 1-29, Jun. 2014.

[3] A. Rahumed, H. C. H. Chen, Y. Tang, P. P. C. Lee, and J. C. S. Lui,"A secure cloud backup system with assured deletion and version control," in *Proc. 40th Int. Conf. Parallel Process. Workshops*, 2011, pp. 160-167.

[4] X. Li, H. Ma, W. Yao, and X. Gui, "Data-driven and feedback-enhanced trust computing pattern for large-scale multi-cloud collaborative services," *IEEE Trans. Serv. Comput.*, vol. 11, no. 4, pp. 671-684, Jul./Aug. 2018.

[5] A. A. A. El-Latif, B. Abd-El-Atty, E. M. Abou-Nassar, and S. E. Venegas-Andraca, "Controlled alternate quantum walks based privacy preserving healthcare images in Internet of Things," *Optics & Laser Technology*, vol. 124, 105942, 2020.

[6] A. A. A. El-Latif, B. Abd-El-Atty, M. S. Hossain, M. A. Rahman, A. Alamri, and B. B. Gupta, "Efficient quantum information hiding for remote medical image sharing," *IEEE Access*, vol. 6, pp. 21075-21083, 2018.

[7] S. Swanson and M. Wei, "SAFE: Fast, verifiable sanitization for SSDs," *San Diego, CA: University of California-San Diego*, 2010.

[8] Q. Wang, C. Wang, K. Ren, W Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847-859, May. 2011.

[9] L. Yingying, J. Ma, M. Yinbin, Y. Wang, X. Liu, and K. K. R. Choo, "Similarity search for encrypted images in secure cloud computing," *IEEE Trans. Cloud Comput.*, 2020.

[10] L. Jiang, C. Xu, X. Wang, B. Luo, and H. Wang, "Secure outsourcing sift: efficient and privacy-preserving image feature extraction in the encrypted domain," *IEEE Trans. Depend. Sec. Comput.*, vol. 17, no. 1, pp. 179-193, Jan./Feb. 2020.

[11] M. Mohanty, M. R. Asghar, and G. Russello, "$2DCrypt$: Image scaling and cropping in encrypted domains," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2542-2555, Nov. 2016.

[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Computer and Comm. Security*, 2006, pp. 89-98.

[13] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with nonmonotonic access structures," in *Proc. 14th ACM Conf. Computer and Comm. Security*, 2007, pp. 195-203.

[14] J. Han, W. Susilo, Y. Mu, J. Zhou, and M. H. A. Au, "Improving privacy and security in decentralized ciphertext-policy attribute-based encryption," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 665-678, Mar. 2015.

[15] Y. Tang, P. P. C. Lee , J. C. S. Lui, and R. Perlman, "Secure overlay cloud storage with access control and assured deletion," *IEEE Trans. Depend. Sec. Comput.*, vol. 9, no. 6, pp. 903-916, Nov./Dec. 2012.

[16] M. Alloghani, M. M. Alani, D. Al-Jumeily, T. Baker, and A. J. Aljaaf, "A systematic review on the status and progress of homomorphic encryption technologies," *Journal of Inf. Security and Appl.*, vol. 48, 102362, 2019.

[17] B. K. Samanthula, G. Howser, Y. Elmehdwi, and S. Madria, "An efficient and secure data sharing framework using homomorphic encryption in the cloud," In *Proc. 1st Int. Workshop on Cloud Intelligence*, 2012, pp. 1-8.

[18] L. Li, A. A. A. El-Latif, and X. Niu, "Elliptic curve elgamal based homomorphic image encryption scheme for sharing secret images," *Signal Process.*, vol. 92, no. 4, pp. 1069-1078, 2012.

[19] X. Dong, W. Zhang, X. Hu, and K. Liu, "A cloud-user watermarking protocol protecting the Right to Be Forgotten for the outsourced plain images," *Int. Journal Digit. Crime Forensics*, vol. 10, no. 4, pp. 118-139, 2018.

[20] K. Liu, W. Zhang, and X. Dong, "A cloud-user protocol based on ciphertext watermarking technology," *Security & Commun. Networks*, pp. 1-14, 2017.

[21] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn. Adv. Cryptol. (EUROCRYPT)*, Prague, Czech Republic, May 1999, pp. 223-238.

[22] *Right to Be Forgotten*. Available: https://en.wikipedia.org/wiki/Right_to_be_forgotten#cite_note-16.

[23] Q. Wang, J. Huang, Y. Chen, C. Wang, F. Xiao, and X. Luo, "$PROST$: Privacy-preserving and truthful online double auction for spectrum allocation," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 374-386, Feb. 2019.

[24] X. Liu, B. Qin, R. H. Deng, and Y. Li, "An efficient privacy-preserving outsourced computation over public data," *IEEE Trans. Serv. Comput.*, vol. 10, no. 5, pp. 756-770, 2015.

[25] X. Liu, R. H. Deng, K. K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Trans Inf. Forensics Security*, vol. 11, no. 11, pp. 2401-2414, Nov. 2016.

[26] X. Liu, R. H. Deng, W. Ding, R. Lu, and B. Qin, "Privacy-preserving outsourced calculation on floating point numbers," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2513-2527, Nov. 2016.

[27] X. Liu, R. Choo, R. Deng, R. Lu, and J. Weng, "Efficient and privacy-preserving outsourced calculation of rational numbers," *IEEE Trans. Depend. Sec. Comput.*, vol. 15, no. 1, Jan./Feb. 2018.

[28] X. Dong, W. Zhang, M. Shah, B. Wang, and N. Yu, "A restrained paillier cryptosystem and its applications for access control of common secret," Available: http://arxiv.org/abs/1912.09034.

[29] S. Al Sharif, F. Iqbal, T. Baker, and A. Khattack, "White-hat hacking framework for promoting security awareness," in *Proc. 8th IFIP Int. Conf. on New Technologies, Mobility and Security*, 2016.

[30] S. Al Sharif, M. Al Ali, N. Al Reqabi, F. Iqbal, T. Baker, and A. Marrington, "Magec: An image searching tool for detecting forged images in forensic investigation," in *Proc. 8th IFIP Int. Conf. on New Technologies, Mobility and Security*, 2016.

[31] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Trans. Image Process.*, vol. 6, no. 12, pp. 1673-1687, Dec. 1997.

[32] M. Barni, F. Bartolini, V. Cappellini, and A. Piva, "A DCT-domain system for robust image watermarking," *Signal process.*, vol. 66, no. 3, pp. 357-372, 1998.

[33] H. W. Lim, S. Tople, P. Saxena, and E. Chang, "Faster secure arithmetic computation using switchable homomorphic encryption," *IACR Cryptol. ePrint Arch. Tech. Rep. 2014/539*, Jul. 2014.

[34] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644-654, Nov. 1976.

[35] S. Kamara, P. Mohassel, and M. Raykova, "Outsourcing multi-party computation," *IACR Cryptol. ePrint Arch. Tech. Rep. 2011/272*, Oct. 2011.

[36] E. Barker, W. Barker, W. Burr,, W. Polk, and M. Smid, "Recommendation for key management part 1: General (revision 3)," *NIST special publication*, vol. 800, no. 57, pp. 1-147, 2012.

[37] X. Yan, S. Wang, A. A. A. El-Latif, X. Niu, and Z. Wei, "A new assessment measure of shadow image quality based on error diffusion techniques," *Journal of Inf. Hiding and Multimedia Signal Process.*, vol. 4, no. 2, pp. 118-126, 2013.

Xiaojuan Dong received her B.S. degree from Anhui University in 2016. She is currently pursuing the Ph.D. degree in information security with the University of Science and Technology of China. Her research interests include applications of digital watermark, information and cryptography.

Weiming Zhang received his M.S. degree and Ph.D. degree in 2002 and 2005, respectively, from the Zhengzhou Information Science and Technology Institute, P.R. China. Currently, he is a professor with the School of Information Science and Technology, University of Science and Technology of China. His research interests include information hiding and multimedia security.

Mohsin Shah received the B.Sc degree in telecommunication engineering from the University of Engineering and Technology Peshawar, Pakistan in 2007, M.Sc degree in telecommunication engineering from the University of Engineering and Technology Taxila, Pakistan in 2012 and the Ph.D. degree in information and communication engineering from the University of Science and Technology of China Hefei, China in 2019. He has been a faculty member with the Department of Telecommunication, Hazara University Mansehra Pakistan since 2009, where he is currently an assistant professor. His research interests include information hiding, multimedia security, secure signal processing and image forgery detection.

Bei Wang received her B.S. degree from Hefei University of Technology in 2013. She is currently working toward the Ph.D. degree at University of Science and Technology of China. Her research interests mainly include fast computation of elliptic curve cryptography.

Nenghai Yu received his B.S. degree in 1987 from Nanjing University of Posts and Telecommunications, M.E. degree in 1992 from Tsinghua University and Ph.D. degree in 2004 from the University of Science and Technology of China, where he is currently a professor. His research interests include multimedia security, multimedia information retrieval, video processing and information hiding.