 CMC, vol.56, no.1, pp.1-17, 2018

Efficient Secure Data Provenance Scheme in Multimedia Outsourcing and Sharing

Zhen Yang[1, 2], Yongfeng Huang[1, 2, *], Xing Li[1, 2] and Wenyu Wang[3]

Abstract: To cope with privacy leakage caused by multimedia outsourcing and sharing, data provenance is used to analyze leaked multimedia and provide reactive accountability. Existing schemes of multimedia provenance are based on watermarking protocols. In an outsourcing scenario, existing schemes face two severe challenges: 1) when data leakage occurs, there exists a probability that data provenance results can be repudiated, in which case data provenance tracking fails; and 2) when outsourced data are shared, data encryption transfer causes key management burden outside the schemes, and privacy leakage threatens users. In this paper, we propose a novel data provenance scheme with an improved LUT-based fingerprinting protocol, which integrates an asymmetric watermarking protocol, robust watermark algorithm and homomorphic encryption and digital signatures to achieve full non-repudiation provenance. We build an in-scheme stream cipher to protect outsourced multimedia data from privacy leakage and complicated key management. Our scheme is also lightweight and easy to deploy. Extensive security and performance analysis compares our scheme with the state of the art. The results show that our scheme has not only better provenance security and data confidentiality but also higher efficiency for multimedia outsourcing, sharing and provenance.

Keywords: Data provenance, asymmetric fingerprint protocol, digital watermarking, multimedia outsourcing.

1 Introduction

In multimedia data distribution, copyright violation disputes become an important problem because of illegal redistribution. To solve this problem, data provenance has been introduced to "construct a disclosure chain of given data" [Wohlgemuth, Echizen, Soneharaand et al. (2010)]. Data provenance identifies "where a piece of data came from and the process by which it arrived in a database" [Buneman, Khanna and Tan (2001)]. In the most common case, multimedia data provenance involves letting the Data Provider embed a distinctive robust digital watermark, called a *fingerprint*, into each authorized users' data. Every distribution's copyright information can be extracted from the fingerprint, which makes illegal redistribution identifiable.

[1] Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China.

[2] Tsinghua National Laboratory for Information Science and Technology, Beijing, 100084, China.

[3] Information Networking Institute, Carnegie Mellon University, Pittsburgh, PA, 15213, USA.

[*] Corresponding Author: Yongfeng Huang. Email: yfhuang@tsinghua.edu.cn.

Recently, data outsourcing services, such as cloud storage, have become more and more popular. However, while data are outsourced, data control is also outsourced. Thus, data leakage threats are becoming more and more serious [Sundareswaran, Squicciarini, Lin et al. (2011)]. When outsourced data leakage occurs, users have to resort to data provenance for accountability.

Among the varieties of approaches of data provenance, watermarking protocols play an important role, especially for multimedia data. A basic problem is the issue of the customer's rights. When the watermark is embedded by the Data Provider, a customer whose watermark has been found on unauthorized copies can claim that he has been framed by a malicious Data Provider who has embedded his identity in an arbitrary object [Bianchi and Piva (2013)]. To cope with this customer's rights problem, an asymmetric watermarking protocol has been proposed whereby only the customer has access to the watermarked data, and thus the Data Provider can later prove that a found copy of data has been shared by the customer. Several such suitable protocols include Buyer-Seller Watermarking Protocols (BSWP) [Memon and Wong (2001); Kuribayashi and Tanaka (2005); Prins, Erkin and Lagendijk (2007)], Zero-Knowledge Watermarking Protocols (ZKWP) [Wohlgemuth, Echizen, Soneharaand et al. (2010); Tharaud, Wohlgemuth, Echizen et al. (2010)], and Oblivious Transfer Watermarking Protocols (OTWP) [Backes, Grimm and Kate (2014)].

Existing watermarking techniques for outsourced data provenance have been developed to address two significant practical issues. The security problem is related to data provenance's non-repudiation. With a robust watermark (usually called a *fingerprint*) embedded into the multimedia data, the data copy can be used to identify the user corresponding with the fingerprint. To ensure data provenance's non-repudiation, the watermarking protocol should protect the watermarked data so that it is only touched by the user that the watermark identifies. Although ZKWP and OTWP [Wohlgemuth, Echizen, Sonehara et al. (2011); Backes (2016)] have been proposed, non-repudiation is still not fully ensured in outsourcing data sharing scenarios.

The second issue is related to the privacy protection of multimedia data. When outsourced multimedia data are shared, the Data Provider needs to transfer data to another user. To protect data privacy, the Data Provider has to use an encryption algorithm, which is not considered in existing multimedia provenance schemes [Wohlgemuth, Echizen, Sonehara et al. (2011); Backes, Grimm and Kate (2016)]. Even without considering the encryption and decryption computation burden, key management is quite complicated: Data encryption usually needs mixed encryption in which symmetric and asymmetric keys are both used, and data provenance needs to use other keys in the scheme as well. Complicated key management schemes always face privacy leakage attacks, which can be caused by key misuse.

In our work, we propose a novel multimedia provenance scheme to cope with the non-repudiation problem, especially in the scenario of outsourcing data sharing. On the basis of LUT-based watermark embedding [Bianchi and Piva (2014)], we present an LUT-based fingerprinting protocol that fuses asymmetric watermarking protocol, robust watermark, homomorphic encryption and digital signature, and achieve non-repudiation for every participant. In consideration of key management, we bring about a stream cipher in our

provenance scheme to implement multimedia encryption for outsourcing, which controls all keys in our scheme and protects multimedia data privacy from leakage. Moreover, we make some reductions to make the protocol efficient with regard to computational cost. Our contribution in this work can be summarized as the following three aspects:

1) We provide a multimedia provenance scheme with an improved LUT-based fingerprinting protocol. The scheme ensures full non-repudiation for every participant in a multimedia outsourcing scenario.
2) We design a stream cipher for multimedia data that is seamless integrated in the provenance scheme to control key management in multimedia outsourcing and sharing and to prevent privacy leakage caused by key abuse.
3) Our scheme has not only stronger security but also greater efficiency than the state of the art, and can be adapted for various watermarking algorithms. The efficiency and usability makes our protocol easy to deploy in reality.

The rest of the paper is organized as follows: Section 2 overviews the related work and Section 3 offers preliminaries. Then, we introduce the problem statement in Section 4, followed by the detailed description of our scheme in Section 5. Section 6 analyze system security. Then, Section 7 gives the performance evaluation. Finally, Section 8 presents the paper's conclusions.

2 Related work

As data provenance has a great influence on accountability for data leakage, many different approaches have been proposed. There are two main directions for building data provenance among these approaches: Reliable logging and asymmetric watermarking protocols. We review the two approaches below.

2.1 Reliable logging schemes

Reliable logging based schemes was proposed for secure provenance in Hasan et al. [Hasan, Sion and Winslett (2009)]. The scheme adopted a hash chain to generate associated logs for data provenance, but only supported record of data writing.

In 2011, data-centric log mechanism was designed to record several normal data operations [Ko, Kirchberg and Lee (2011)]. Later, an automated logging mechanism was proposed to be executed by data itself [Sundareswaran, Squicciarini, Lin et al. (2011)]. In the scheme, data operation can be recorded into logs by a JAR that encloses the data. An extension work adapted an automated logging scheme to the cloud storage environment and improved log generation and storage efficiency [Yang, Wang and Huang (2017)].

However, logs can still be tampered to disable the above provenance schemes. Their alternative solutions involve finding a responsible party for the attack, which is not the leakage provenance.

2.2 Asymmetric watermarking protocols

To resist malicious deletion attacks, robust digital watermark was used to save provenance information [Wohlgemuth, Echizen, Sonehara et al. (2010)]. Thus they proposed a provenance scheme of robust digital watermark tags that are embedded into data.

In the data distribution, we still require asymmetric watermarking protocol in which every participant (distributor or receiver) only holds a data copy embedded into his watermark. Subsequently, Wohlgemuth et al. [Wohlgemuth, Echizen, Sonehara et al. (2011)] put forward an asymmetric watermarking protocol based on commitment of a zero knowledge proof. Although provenance security is ensured, commitment computation yields heavy burden on practical deployment.

Backes et al. [Backes, Grimm and Kate (2014); Backes, Grimm and Kate (2016)] proposed an oblivious-transfer-based watermarking protocol to trace the provenance of multimedia data. They proved that this scheme is obviously lightweight with regard to both time and space costs. However, the scheme just has a possible security, which is not secure enough.

3 Preliminaries

3.1 LUT-Based watermark embedding

Given a vector $\boldsymbol{x} = (x_1, x_2, \dots, x_M)$ representing a set of features of the original data and a hidden fingerprint of user U represented as a binary vector $\boldsymbol{b_U} = (b_{U1}, \dots, b_{UL})$, an embedder inserts the fingerprint $\boldsymbol{b_U}$ into host signal data to produce watermarked data $\boldsymbol{y}$. Often $\boldsymbol{b_U}$ is transferred to the same size of $\boldsymbol{x}$ as a watermark signal $\boldsymbol{w}$, so the watermarking can be expressed simply as $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{w}$.

In LUT-Based Watermark Embedding [Celik, Lemma, Katzenbeisser et al. (2008)], the Data Provider generates a long-term Look-Up Table (LUT) $\boldsymbol{E}$ with size T for the master encode, whose entries are i.i.d. random variables following a Gaussian distribution $N(0, \sigma_E^2)$. Then, the Data Provider generates an LUT $\boldsymbol{W_U}$ with size T from $\boldsymbol{b_U}$. With $\boldsymbol{E}$ and $\boldsymbol{W_U}$, the Data Provider can calculate the personalized decode LUT $\boldsymbol{D_U}$ with size T:

$$D_U(t) = W_U(t) - E(t), for\ t = 1, \dots, T \tag{1}$$

Then with a session key, an $M \times R$ set of values t_{ih} is generated featuring integers between 1 and T for $i = 1, \dots, M$ and $h = 1, \dots, R$. Calculated with a perceptual mask vector $\boldsymbol{pm} = (pm_0, \dots, pm_M)$ that ranges within $[0, 1]$, each of the M data features $\boldsymbol{x}$ can be encoded by adding R perceptual masked entries of the encoding LUT $\boldsymbol{E}$ identified by the indexes $t_{i1}, \dots, t_{iR}$, obtaining the encoded feature as follows:

$$c_i = x_i + pm_i \sum_{h=1}^{R} E(t_{ih}) \tag{2}$$

Then, the data consumer can also use the session key to get t_{ih}. Each of the M encoded features c can be decoded by adding R entries of the decoding LUT D_U identified by the indexes $t_{i1}, \dots, t_{iR}$, obtaining the decoded feature as follows:

$$y_i = c_i + pm_i \sum_{h=1}^{R} D_U(t_{ih}) = x_i + pm_i \sum_{h=1}^{R} W_U\ (t_{ih}) \tag{3}$$

3.2 Homomorphic cryptosystem

A homomorphic cryptosystem has an operator $\Phi(\cdot, \cdot)$ in a ciphertext field related to an operator $*$ in a plaintext field such that, for any two plain messages m_1 and m_2, we have: $\Phi([\![m_1]\!], [\![m_2]\!]) = [\![m_1 * m_2]\!]$, where $[\![\cdot]\!]$ denotes the encryption operator. Homomorphic encryption allows a set of operations to be performed by working on encrypted data. In particular, an additively homomorphic cryptosystem usually maps an addition in the plaintext domain to a multiplication in the ciphertext domain, which means that $[\![m_1]\!] *$

$[\![m_2]\!] = [\![m_1 + m_2]\!]$. Then, for a constant integer a, $[\![m]\!]^a = [\![m * a]\!]$. A well-known additively homomorphic asymmetric encryption scheme was proposed by Paillier [Paillier (1999)].

3.3 Robust spread spectrum watermark

Cox et al. [Cox, Kilian, Leighton et al. (1997)] proposed a popular robust watermark with spread spectrum, which shows strong resilience to lossy operations, such as aggressive scale changes, JPEG compression, dithering and data conversion. The watermark vector has a distribution of $N(0, \sigma_w^2)$.

Before watermark embedding, the original image img needs to be preprocessed with a DCT transformation and zigzag scan. We use $dctimg$ to represent the preprocessed vector.

The watermark vector is embedded into a subset of $dctimg$, the lowest frequency DCT AC coefficients, to ensure robustness without causing perceptible artifacts.

4 Problem statement

4.1 System model

Fig. 1 shows a multimedia data outsourcing and sharing architecture including three entities as follows.

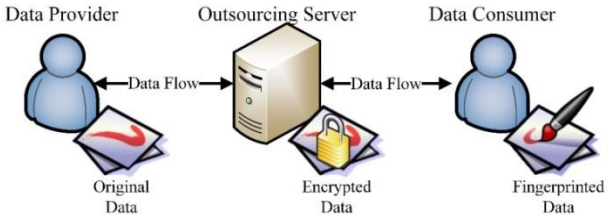

Figure 1: Architecture of data outsourcing and sharing

Data Provider (DP): An entity that outsources its data to an Outsourcing Server can authorize other users as Data Consumers to access that outsourced data. For any given piece of data, there is only one data provider.

Outsourcing Server (OS): An entity that stores data from the Data Provider can transfer data to authorized users (Data Consumers) at their request.

Data Consumer (DC): An entity that hopes to access data on an Outsourcing Server can ask the Data Provider for authorization and ask the Outsourcing Server for data. For any given piece of data, there may be several data consumers.

In the architecture above, the DP hopes that every data copy can be embedded into the fingerprint of a related authorized DC. Once the data leaks, the DP can use a leaked data copy to trace the provenance, ensure accountability and compensate for its loss.

4.2 Security model

We consider a secure scheme to exhibit the following characteristics. *1) No cross-holding:* The DC cannot access the original data, whereas the DP cannot access the data copy that embeds the DC's fingerprint. Moreover, neither the original data nor the DC's data copy

can be accessed by the OS. *2) No framing:* When a user has not leaked data, he cannot be framed by anyone as the leakage provenance.

With these two properties achieved, a data provenance scheme definitely exhibits non-repudiation, which means that no suspect can deny the leakage. As the security definitions given by Wohlgemuth et al. [Wohlgemuth, Echizen, Sonehara et al. (2011); Backes, Grimm and Kate (2016)] both express no denial and no framing, our security model builds these properties based on a more specific trust assumption as below.

The DP first hopes that every shared data copy can be traced for provenance. On the condition that the provenance principle is not violated, the DP may collude with other participants to frame another user.

Based on this trust assumption, our security model ensures strong results, i.e. correctness and non-repudiation of data provenance. In addition, the OS is introduced that enables our security model to be more suitable for a practical three-party system model.

5 The proposed scheme

This section presents our data provenance scheme in detail. We start from an overview of the entire scheme framework. Then, we introduce three underlying protocols to show how non-repudiation is achieved in our scheme design.

In our scheme, we assume that there is a PKI in our framework, which means that each participant (DP/OS/DC) has a pair consisting of a public key and private key. For the purpose of our discussion, we take an image file as an example of multimedia data. Running our outsourcing multimedia provenance protocol includes three phases:

Secure Outsourcing: The DP encrypts the original data with an LUT-based stream cipher and then uploads the ciphertext data to the OS.

Secure Sharing: The DC sends a data request to the DP. After the DP permits authorization of the DC, the DP sends a stream key to the DC for data decryption.

Data Provenance: When the DP finds a leaked copy of its data, it extracts the fingerprint from the leaked copy. With the extracted fingerprint, the DP can identify the leakage data provenance.

5.1 Improved LUT-based fingerprint enabled secure outsourcing

In our architecture, as Fig. 1 shows, multimedia data outsourcing needs both encryption preprocessing for confidentiality and fingerprinting for accountability.

Although classical LUT-based watermarking enables watermark vector embedding between the DP and DC directly, neither transferring the fingerprint into the watermark vector nor encrypting the watermarked data are included. Therefore, we integrate these two parts seamlessly to propose our improved LUT-based fingerprinting for secure outsourcing.

The user first uses binary antipodal modulation to generate the fingerprint message with L-bit fingerprint $\boldsymbol{b}$: $\boldsymbol{fm} = \sigma_w \cdot (2\boldsymbol{b} - 1)$. Then, a $T \times L$ generation matrix $\boldsymbol{G}$ is used to process $\boldsymbol{fm}$ to get $\boldsymbol{W}$, which is subject to $N(0, \sigma_w^2)$: $\boldsymbol{W} = \boldsymbol{G} \cdot \boldsymbol{fm}$.

$\boldsymbol{G}$ is a matrix of linear block code. Note that $\boldsymbol{G}$ can be derived with repetition code, which means that $\boldsymbol{G}$ only has one entry equal to 1 in each row and approximately T/L entries

equal to 1 in each column. Another means of generating $\boldsymbol{G}$ is that all entries of $\boldsymbol{G}$ are subject to $N(0,1/L)$ i.i.d., which is called Gaussian random code.

Thus, the Decode LUT for fingerprinting can be derived as below:

$$\boldsymbol{D} = \boldsymbol{W} - \boldsymbol{E} = \boldsymbol{G} \cdot \boldsymbol{fm} - \boldsymbol{E} = \boldsymbol{G} \cdot \sigma_w \cdot (2\boldsymbol{b} - 1) - \boldsymbol{E} \quad (4)$$

we remove t_{ih} in LUT-based watermark embedding to enhance efficiency:

$$y_i = x_i + \alpha \cdot W_i = x_i + \alpha \cdot (E_i + D_i) \quad (5)$$

Then we use a session key sek to generate the mask vector $\boldsymbol{E_{mask}}$, whose length is that of $\boldsymbol{dctimg}$ minus that of the encode LUT $\boldsymbol{E}$. Thus, we concatenate $\boldsymbol{E}$ and $\boldsymbol{E_{mask}}$ to generate a vector $\boldsymbol{E_{full}}$, which has the same length as the AC coefficients of $\boldsymbol{dctimg}$: $\boldsymbol{E_{full}} = \boldsymbol{E}|\boldsymbol{E_{mask}}$. Then, we embed $\boldsymbol{E_{full}}$ into $\boldsymbol{dctimg}$, which means that $\boldsymbol{E}$ is embedded into the real watermark frequency band and that $\boldsymbol{E_{mask}}$ is embedded into all other frequency bands: $\boldsymbol{enc_{dctimg}} = \boldsymbol{dctimg} + \alpha \cdot \boldsymbol{E_{full}}$, and this $\boldsymbol{enc_{dctimg}}$ is the ciphertext image. Thus, we use sek and the watermark approach to build a stream cipher on a multimedia image. For non-image multimedia data, we can take advantage of the corresponding frequency transformation domain to embed in all frequencies of the data.

To protect the $\boldsymbol{enc_{dctimg}}$ real watermark frequency band from recognition for malicious tampering, the DP chooses $\boldsymbol{E}$ and $\boldsymbol{E_{mask}}$ subject to the same distribution $N(0, \sigma_E^2)$ i.i.d. To ensure that $\boldsymbol{enc_{dctimg}}$ is incomprehensible, σ_E is identified as a large number.

Finally, the DP can outsource $\boldsymbol{enc_{dctimg}}$ to the OS directly.

5.2 Secure sharing protocol

In the data outsourcing scenario, the Data Consumer (DC) shares data from the Data Provider (DP). This process includes three sub-protocols: Fingerprint generation, authorization granting, and data decoding. All messages sent and received are protected with signature and verification.

5.2.1 Fingerprint generation

First, we give a description of our fingerprint structure: DP-ID, DC-ID, $Timestamp$, $Random$, $Signature_{DC}$, as shown in Fig. 2. The first 4 fields can be called fingerprint core.

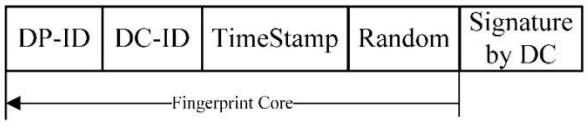

Figure 2: Fingerprint structure

Here DP-ID represents the identity of the Data Provider, and DC-ID represents the identity of the Data Consumer. The $Timestamp$ field expresses the time of fingerprint generation. The $Random$ field is a random number given by the DC. The last field is DC's signature of the fingerprint core. The fingerprint generation sub-protocol is shown in Fig. 3.

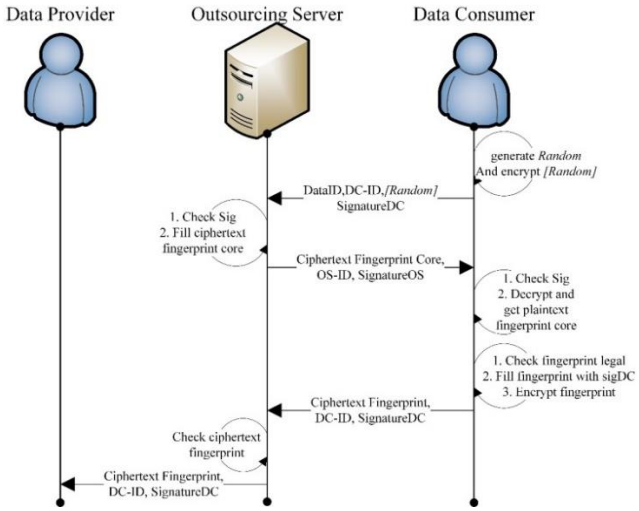

Figure 3: Fingerprint generation process

In the fingerprint generation protocol, the DC first generates a random number and encrypts it with his public key and sends it to the OS.

The OS then finds the DP-ID to whom the request data belongs and records the current time as the timestamp. Next, the OS encrypts the DP-ID, DC-ID, and timestamp with the DC's public key and concatenates these three ciphertext fields and $[\![random]\!]$ to derive the ciphertext fingerprint core. Then these information is sent to the DC.

The DC then checks whether the DC-ID and timestamp in the fingerprint core are correct. When the DC ensures the legality of the fingerprint core, the DC makes the ciphertext full fingerprint with signature and encryption. Then the ciphertext full fingerprint are sent to the OS.

The OS then compares its ciphertext fingerprint core with that from the DC. When comparison shows that both of the two ciphertext fingerprint cores are exactly the same, the OS forwards the message sent by the DC to the DP.

If any verification step of the process fails, the process terminates.

5.2.2 Authorization granting

After fingerprint generation, the Data Provider will decide whether to grant authorization to the DC. We demonstrate the authorization granting process in Fig. 4.

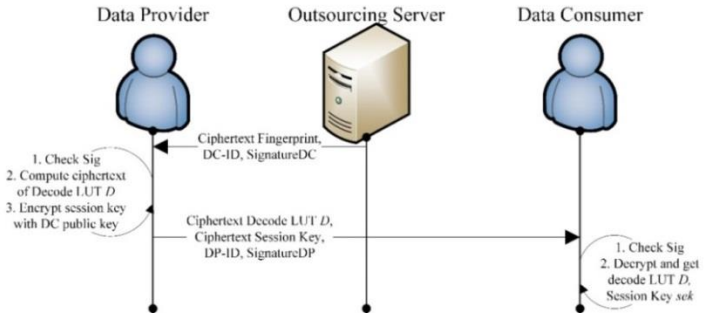

Figure 4: Authorization granting process

When the DP receives a legal ciphertext fingerprint and accepts request, he calculates the ciphertext decode LUT $\boldsymbol{D}$ for DC with the ciphertext fingerprint.

As watermarking and LUT encoding are both linear, the decode LUT D can be generated through additive homomorphic encryption in the ciphertext domain. Here we use $[\![\boldsymbol{b}]\!]$ to express the ciphertext fingerprint that is encrypted by the Paillier homomorphic algorithm. Thus, $[\![\boldsymbol{m}]\!]$ can be derived as below:

$$[\![\boldsymbol{m}]\!] = ([\![\boldsymbol{b}]\!]^2 \cdot [\![-1]\!])^{\sigma_w} \tag{6}$$

Then, the ciphertext watermark can be expressed as

$$[\![W_i]\!] = \prod_{k=1}^{L} [\![m_k]\!]^{G_{ik}}, i = 1, \ldots, T \tag{7}$$

The ciphertext Decode LUT $\boldsymbol{D}$ is then expressed with $\boldsymbol{E}$ and $\boldsymbol{W}$ as

$$[\![D_i]\!] = [\![-E_i]\!] \cdot \prod_{k=1}^{L} [\![m_k]\!]^{G_{ik}}, i = 1, \ldots, T \tag{8}$$

The DP encrypts session key sek with its public key. It sends $[\![\boldsymbol{D}]\!]$, $[\![sek]\!]$ to the DC. Then the DC decrypts the message with its secret key and gets the decode LUT $\boldsymbol{D}$ and session key sek.

5.2.3 Data decoding

After authorization, the DC is able to download and decode the outsourced data. Fig. 5 shows the data decoding process.

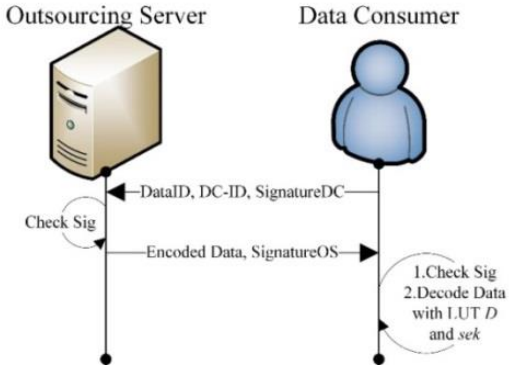

Figure 5: Outsourced data decoding process

The DC first sends download request to the OS. Then he waits for the encoded data from the OS. The DC then decodes the data with the LUT $\boldsymbol{D}$ and session key sek as below: $\boldsymbol{E_{mask}} = generate(sek)$, $\boldsymbol{D_{full}} = \boldsymbol{D}| - \boldsymbol{E_{mask}}$.

$$\boldsymbol{dec_{encdctimg}} = \boldsymbol{dctimg} + \alpha \cdot (\boldsymbol{E_{full}} + \boldsymbol{D_{full}}) = \boldsymbol{dctimg} + \alpha \cdot \boldsymbol{G} \cdot \sigma_w(2\boldsymbol{b} - 1) \quad (9)$$

Here, $\boldsymbol{dec_{encdctimg}}$ is the image DCT vector in which the DC's fingerprint $\boldsymbol{b}$ is embedded. After performing the inverse zigzag scan and inverse DCT tranform on $\boldsymbol{dec_{encdctimg}}$, the DC gets $\boldsymbol{img_{DC}}$, in which its fingerprint $\boldsymbol{b}$ is embedded.

5.3 Multimedia tracing provenance

Data leakage often occurs in the data outsourcing scenario. If the DP outsources and shares its data with our secure outsourcing and sharing, then the DP can trace the provenance of leaked data from the fingerprint in the leaked data copy. We show our multimedia tracing provenance protocol in Fig. 6.

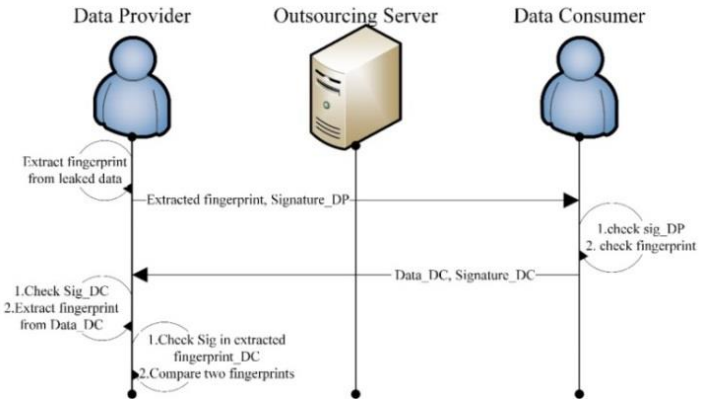

Figure 6: Tracing leakage provenance process

Here, we use $\boldsymbol{img_{leak}}$ to express a leaked data copy. When the DP wants to trace the provenance of $\boldsymbol{img_{leak}}$, he first extracts the watermark from $\boldsymbol{img_{leak}}$:

$$\boldsymbol{W_{leak}} = dct(zigzag(\boldsymbol{img_{leak}})) - \boldsymbol{dctimg})/\alpha \quad (10)$$

Then, the DP extracts the binary fingerprint vector from $\boldsymbol{W_{leak}}$ with matched filter (MF) decoder:

$$\boldsymbol{b_{leak}} = sgn(\boldsymbol{G^T} \cdot \boldsymbol{W_{leak}}) \quad (11)$$

or pseudo-inverse (PI) decoder:

$$\boldsymbol{b_{leak}} = sgn((\boldsymbol{G^T G})^{-1} \boldsymbol{G^T} \cdot \boldsymbol{W_{leak}}) \quad (12)$$

The DP first checks whether the signature in $\boldsymbol{b_{leak}}$ is correct. If the result is no, the leaked data copy may be damaged and cannot be used to trace the provenance.

Otherwise, the DP extracts the DC-ID from $\boldsymbol{b_{leak}}$ and sends $\boldsymbol{b_{leak}}$ with $signature_{DP}$ to the DC. If the DC verifies that the message signature or received fingerprint signature is incorrect, the DC declines to answer this message. Otherwise, the DC sends its $\boldsymbol{img_{DC}}$ and its signature to the DP. If the DP checks that the $signature_{DC}$ is incorrect with $\boldsymbol{img_{DC}}$,

DP can conclude that the DC-ID in $\boldsymbol{b_{leak}}$ is the leakage provenance.

If the DP verifies that $signature_{DC}$ is correct, the DP then extracts $\boldsymbol{b_{DC}}$ from $\boldsymbol{img_{DC}}$ and checks whether the signature in $\boldsymbol{b_{DC}}$ is correct. If the signature in $\boldsymbol{b_{DC}}$ fails the check, the DC-ID in $\boldsymbol{b_{leak}}$ is also determined to be the provenance of the leaked copy.

If the signature in $\boldsymbol{b_{DC}}$ is correct, the DP compares the fingerprint cores of $\boldsymbol{b_{leak}}$ and $\boldsymbol{b_{DC}}$. If these two are the same, then the DC-ID in $\boldsymbol{b_{leak}}$ is the provenance of the leakage. Otherwise, the DC-ID in $\boldsymbol{b_{leak}}$ is innocent, and its secret key must have been leaked.

6 Security analysis

6.1 Multimedia encryption security

Encoded data is the ciphertext of the stream cipher, which satisfies the security needs of users in the data outsourcing scenario.

Encoded data generation includes the DCT transform, zigzag scan, and $\boldsymbol{E_{full}}$ LUT embedding. The first 2 steps require no parameters, whereas the last step needs the stream key sek to generate. Our sek generation cycle is $10^{19.2M}$, which is much bigger than global data scale 3.52×10^{23} bits from IDC statistics. And sek is transmitted with Paillier encrypted, so content of sek has semantic security against chosen-plaintext attacks (CPA) [Paillier (1999)]. Thus, encoded data is secure.

6.2 No cross-holding

1) The DC cannot hold the original data $\boldsymbol{img}$. He cannot obtain the original data $\boldsymbol{img}$ from the encoded data $\boldsymbol{enc_{dctimg}}$, because he cannot guess out or decrypt the correct content of sek.

2) The DP cannot hold the data copy $\boldsymbol{img_{DC}}$ that is embedded with the DC's fingerprint. From fingerprinting process, we know that $\boldsymbol{b}$ is protected by Paillier cryptosystem, Thus the DP cannot derive $\boldsymbol{img_{DC}}$.

3) The OS cannot hold both $\boldsymbol{img}$ and $\boldsymbol{img_{DC}}$. The OS cannot obtain a correct content of sek, so he cannot hold $\boldsymbol{img}$. The OS also cannot break Paillier cryptosystem to get $\boldsymbol{b}$, so he cannot hold $\boldsymbol{img_{DC}}$.

6.3 No framing

1) The DP cannot frame the DC. Because DC's fingerprint $\boldsymbol{b}$ includes a signature field, DP cannot forge $\boldsymbol{b}$. Thus, the DP cannot frame a DC.

2) A DC cannot frame another DC. Also because of the DC's signature field in fingerprint $\boldsymbol{b}$, the DC cannot frame another DC.

Tab. 1 shows a security comparison between our scheme, the ZKWP-based scheme [Wohlgemuth, Echizen, Sonehara et al. (2011)] and the OTWP-based scheme [Backes, Grimm and Kate (2016)]. In terms of data encryption, the ZKWP-based scheme and OTWP-based scheme have to encrypt data outside the scheme, whereas we enable a stream cipher in our scheme. Our scheme enables data encryption to be convenient and efficient on the premise of being provably secure.

Table 1: Security comparison between three schemes

Security Property	ZKWP-based scheme [Wohlgemuth, Echizen, Sonehara et al. (2011)]	OTWP-based scheme [Backes, Grimm and Kate (2016)]	Our Scheme
Data Encryption	Outside scheme	Outside scheme	In scheme
No cross-holding	Strong	Weak	Strong
No framing	Strong	Strong	Stronger

In terms of no cross-holding, our scheme relies on irreversible Paillier encryption computation, whereas the ZKWP-based scheme relies on an irreversible blinding process. These two both have strong security with regard to no cross-holding. However, the OTWP-based scheme relies on a low probability of cheating. OTWP has a cheating probability of $1/2^n$, in which n is the number of data divided parts. Technically, the value of n cannot be too large because multimedia data cannot be divided into too many parts and the time cost of sharing and provenance in the OTWP-based scheme increases very quickly along with increasing n. Thus, the OTWP-based scheme has just weak security with regard to no cross-holding.

In terms of no framing, the ZKWP-based scheme relies on unforgeable commitment. The OTWP-based scheme relies on the unforgeable signature of the DC. Our scheme enhances security with the unforgeable signature of the DC and information certification of the OS. Thus, our scheme exhibits stronger security and more convenience.

7 Performance

We compare our secure data provenance scheme performance with the OTWP-based scheme. For the oblivious transfer watermarking protocol, we adopt the protocol of Naor et al. [Naor and Pinkas (2001)]. For the asymmetric encryption and signature algorithm, we make use of the Paillier cryptosystem [Paillier (1999)]. For our watermark generation and embedding, we use the Cox spread spectrum watermark [Cox, Kilian, Leighton et al. (1997)] and LUT-based watermark embedding algorithm [Celik, Lemma, Katzenbeisser et al. (2008)], respectively. We use the Lenna image as our test multimedia data, which has four different image sizes: 128×128, 256×256, 512×512, 1024×1024. Our fingerprint core has 128 bits including timestamp (40 bits), DP-ID (34 bits), DC-ID (34 bits) and random (20 bits).

In our experiments, we use a powerful server with an Intel Xeon X3430 CPU (4-core, 2.4 GHz) and 8 GB of RAM to act as the OS. Moreover, we use two equivalent common PCs with an Intel i5 2430 M CPU (2-core, 2.4 GHz) and 4 GB of RAM to act as the DP and DC. All these computers are in our experimental LAN. The experiments below are implemented in Java.

7.1 Data outsourcing

We select the length of the encode LUT $\boldsymbol{E}$ as $T = 4096$. Suppose that the test data is the 128×128 Lenna image, then the length of $\boldsymbol{E_{mask}}$ is $128 \times 128 - 4096 - 1 = 12287$. $\boldsymbol{E}$ and $\boldsymbol{E_{mask}}$ are both subject to the same zero-mean Gaussian distribution, whose standard deviation is selected as $\sigma_E = 3000$.

Fig. 7 shows the original image and outsourced image encrypted by our stream cipher. Compared to the original image, the PSNR of outsourced image is only 27.1727, which means great degradation of image quality. Almost no information of original image can be obtained from the encoded data in the right.

Figure 7: Images before and after stream cipher encoding

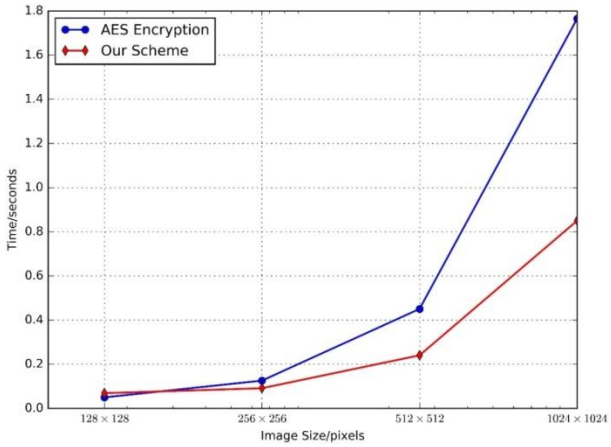

Figure 8: Multimedia encryption computation cost

To evaluate the stream cipher efficiency, we compare the encryption time cost of our stream cipher with the widely used AES encryption algorithm on different data sizes, which is shown in Fig. 8. When the image size is small, our stream cipher computation cost is very close to AES encryption. As image size enlarges, our stream cipher computation cost increases slower than AES. When the image size is 1024×1024, our scheme only costs 48.2% of the time of AES encryption.

For the space cost, if we use px to represent the image pixel unit, then the AES encrypted image has $64\ px$ bits. Alternately, our outsourced data encoded by the stream cipher also has $64\ px$ bits. The space efficiencies of the two are basically equal.

These results shows that our stream cipher enables high-efficiency data outsourcing for the DP.

7.2 Data sharing

In data sharing protocols, after the 128-bit fingerprint is generated, the DC signs the fingerprint core with a 1024-bit signature. Thus, our full fingerprint length is 1152 bits. Our watermark vector is subject to a zero-mean Gaussian distribution, whose standard deviation is $\sigma_w = 1000$. To locate a certain piece of data in the outsourcing storage, the DataID is used as an 80-bit vector.

Fig. 9 shows a comparison between the 128×128 original image and its fingerprinted copy. The PSNR of the fingerprinted copy compared with the original image is 34.2178, which indicates quite high fidelity.

Figure 9: Comparison of original and fingerprinted images

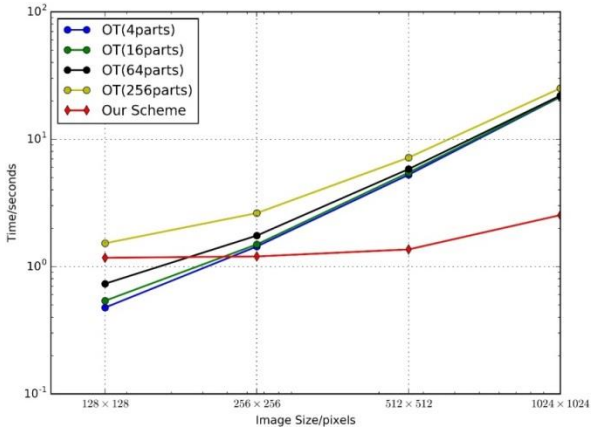

Figure 10: Multimedia sharing computation cost

In a data sharing protocol, the main computation time of our scheme is spent on many homomorphic computations in the Paillier ciphertext domain. We measure the data sharing computation time on different image sizes and make a comparison with the OTWP-based scheme, as shown in Fig. 10. Because the data sharing computation cost of the OTWP-based scheme is positively correlated with the number of parts of data division, we test data divided into 4, 16, 64 and 256 parts for the OTWP-based scheme. To make the figure clear, the logarithmic coordinates are used on the vertical axis. In the figure, as the data size increases, our LUT-based watermark embedding data sharing protocol maintains a slow increase of computation time cost, whereas the OTWP-based scheme's data sharing

computation cost enlarges rapidly. On the 1024×1024 image, our scheme achieves only $10.1\% \sim 11.7\%$ of the computation time cost of data sharing compared to the OTWP-based scheme, which is a significant reduction.

Then, we analyze the communication cost of the three steps. In the fingerprint generation step, the DC sends 2 messages whose total volume is $1024 \times 2 + 34 + 80 + 1024 \times 3 + 34 = 5268$ bits; the OS sends 2 messages, whose total volume is $1024 \times 2 + 34 + 1024 \times 3 + 34 = 5188$ bits. In the authorization granting step, the DP sends 1 message, whose total volume is $1024 \times T + 1024 \times 2 + 34 = 1024\,T + 2082$ bits. In the final step of data decoding, the DC sends 1 message, which includes $80 + 34 + 1024 = 1138$ bits in total; the OS also sends 1 message, which includes $64\,px + 1024$ bits. In summary, the DC sends 6406 bits; the OS sends $64\,px + 6212$ bits; and the DC sends $1024\,T + 2082$ bits. For our example of the image with size 128×128 and $T = 4096$, the communication cost of the OS is 128.75 KB, and the communication cost of the DC is 512.25 KB, which is practical in outsourcing scenario.

In general, data sharing in our scheme has high computational efficiency and practical communication efficiency.

7.3 Data provenance

Because the Cox spread spectrum watermark is a non-blind watermark algorithm, data provenance should only be conducted by the party that can access the original data. Thus, it has to be the DP itself or a particular server that is fully trusted by the DP.

The correctness of provenance is ensured first by the Cox robust watermark. Moreover, the generation matrix $\boldsymbol{G}$, which is repetition code or Gaussian random code, can assure that $G^T G$ is a strict diagonal dominance matrix. Thus, either an MF or PI receiver can extract the correct fingerprint from a robust watermark.

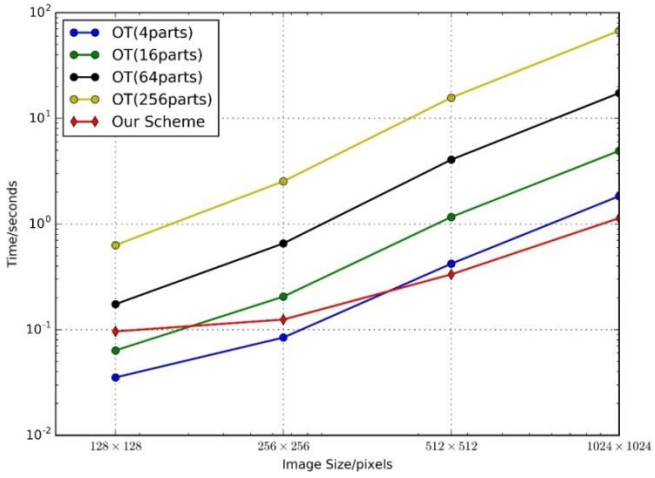

Figure 11: Data provenance tracking computation cost

Here, we implement the process in Fig. 6 to compare the provenance time with the OTWP-based provenance scheme, as shown in Fig. 11. As stated before, the logarithmic coordinates are also used here on the vertical axis. Our scheme shows a slower increasing

trend along with increasing image size and achieves better efficiency, which costs only $1.7\% \sim 61.9\%$ of the time.

For the communication cost of data provenance, the data volume from the DP is $1152 + 1024 = 2176$ bits; the data volume from the DC is $64\,px + 1024$ bits. Taking the 128×128 image as an example, the communication data volume of the DC in the provenance phase is 128.13 KB.

In summary, data provenance in our scheme has higher time efficiency and reasonable communication efficiency.

8 Conclusion

We present a multimedia data provenance scheme that can track the provenance of leaked multimedia data securely and efficiently. We regulate system participants and security definitions, and propose an improved LUT-based fingerprinting protocol. Simultaneously, we present a stream cipher embedded in our scheme to provide a secure and efficient data outsourcing algorithm. We prove that multimedia data provenance can satisfy our security regulations and achieve better efficiency performance.

Although our scheme cannot prevent data leakage from occurring, reactive accountability is ensured by our scheme. The system model and security model are both suitable for practical data outsourcing circumstances, which makes our provenance scheme convincible for a neutral entity, such as a judge.

Acknowledgement: The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. The work is supported by the National Key Research and Development Program of China (No. 2016YFB0800402) and the National Natural Science Foundation of China (No. U1405254, No. U1536207).

References

Backes, M.; Grimm, N.; Kate, A. (2014): Lime: Data lineage in the malicious environment. *International Workshop on Security and Trust Management*, pp. 183-187.

Backes, M.; Grimm, N.; Kate, A. (2016): Data lineage in malicious environments. *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 178-191.

Bianchi, T.; Piva, A. (2013): Secure watermarking for multimedia content protection: A review of its benefits and open issues. *IEEE Signal Processing Magazine*, vol. 30, no. 2, pp. 87-96.

Bianchi, T.; Piva, A. (2014): TTP-free asymmetric fingerprinting protocol based on client side embedding. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3987-3991.

Buneman, P.; Khanna, S.; Tan, W. (2001): Why and where: A characterization of data provenance. *International Conference on Database Theory*, pp. 316-330.

Celik, M. U.; Lemma, A. N.; Katzenbeisser, S.; van der Veen, M. (2008): Lookup-tablebased secure client-side embedding for spread-spectrum watermarks. *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 475-487.

Cox, I. J.; Kilian, J.; Leighton, F. T.; Shamoon, T. (1997): Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1673-1687.

Hasan, R.; Sion, R.; Winslett, M. (2009): The case of the fake picasso: Preventing history forgery with secure provenance. *7th USENIX Conference on File and Storage Technologies*, vol. 9, pp. 1-14.

Ko, R. K.; Kirchberg, M.; Lee, B. S. (2011): From system-centric to data-centric logging accountability, trust & security in cloud computing. *IEEE Defense Science Research Conference and Expo*, pp. 1-4.

Kuribayashi, M.; Tanaka, H. (2005): Fingerprinting protocol for images based on additive homomorphic property. *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2129-2139.

Memon, N. D.; Wong, P. W. (2001): A buyer-seller watermarking protocol. *IEEE Transactions on Image Processing*, vol. 10, no. 4, pp. 643-649.

Naor, M.; Pinkas, B. (2001): Efficient oblivious transfer protocols. *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 448-457.

Paillier, P. (1999): Public-key cryptosystems based on composite degree residuosity classes. *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 223-238.

Prins, J.; Erkin, Z.; Lagendijk, R. L. (2007): Anonymous fingerprinting with robust QIM watermarking techniques. *EURASIP Journal on Information Security*, vol. 2007, no. 20.

Sundareswaran, S.; Squicciarini, A.; Lin, D.; Huang, S. (2011): Promoting distributed accountability in the cloud. *IEEE International Conference on Cloud Computing*, pp. 113-120.

Tharaud, J.; Wohlgemuth, S.; Echizen, I.; Sonehara, N.; Muller, G. et al. (2010): Privacy by data provenance with digital watermarking-a proof-of-concept implementation for medical services with electronic health records. *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 510-513.

Wohlgemuth, S.; Echizen, I.; Sonehara, N.; Muller, G. (2010): Tagging disclosures of personal data to third parties to preserve privacy. *IFIP International Information Security Conference*, pp. 241-252.

Wohlgemuth, S.; Echizen, I.; Sonehara, N.; Muller, G. (2011): On privacy-compliant disclosure of personal data to third parties using digital watermarking. *Journal of Information Hiding & Multimedia Signal Processing*, vol. 2, no. 2, pp. 270-281.

Yang, Z.; Wang, W.; Huang, Y. (2017): Ensuring reliable logging for data accountability in untrusted cloud storage. *IEEE International Conference on Communications*, pp. 1966-1971.