Research Article

Noninteractive Lightweight Privacy-Preserving Auditing on Images in Mobile Crowdsourcing Networks

Juan Zhang,¹ Changsheng Wan ,² Chunyu Zhang,³ Xiaojun Guo,³ and Yongyong Chen²

¹Business School, Nanjing University, Nanjing 210093, China

²School of Cyber Space, Southeast University, Nanjing 210096, China

³School of Information and Engineering, Xizang Minzu University, Xianyang 712082, China

Correspondence should be addressed to Changsheng Wan; wan.changsheng@163.com

Received 17 April 2020; Revised 6 July 2020; Accepted 10 July 2020; Published 25 July 2020

Academic Editor: Clemente Galdi

Copyright © 2020 Juan Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

To determine whether images on the crowdsourcing server meet the mobile user's requirement, an auditing protocol is desired to check these images. However, before paying for images, the mobile user typically cannot download them for checking. Moreover, since mobiles are usually low-power devices and the crowdsourcing server has to handle a large number of mobile users, the auditing protocol should be lightweight. To address the above security and efficiency issues, we propose a novel noninteractive lightweight privacy-preserving auditing protocol on images in mobile crowdsourcing networks, called NLPAS. Since NLPAS allows the mobile user to check images on the crowdsourcing server without downloading them, the newly designed protocol can provide privacy protection for these images. At the same time, NLPAS uses the binary convolutional neural network for extracting features from images and designs a novel privacy-preserving Hamming distance computation algorithm for determining whether these images on the crowdsourcing server meet the mobile user's requirement. Since these two techniques are both lightweight, NLPAS can audit images on the crowdsourcing server in a privacy-preserving manner while still enjoying high efficiency. Experimental results show that NLPAS is feasible for real-world applications.

1. Introduction

Recently, mobile crowdsourcing systems have been widely deployed all over the world, which collect and process data through widely available mobile devices [1]. As discussed in [2], since data typically originate from third parties, falsified data may be reported to the crowdsourcing server. Therefore, to ensure data trust, it is important to check whether the data collected by workers meet the mobile user's requirement before using it [2]. We call this sort of protocol the auditing protocol. At the same time, to avoid economic loss, mobile users may not be allowed to download images before paying for them. Therefore, to check images before downloading them, the auditing protocol should have the noninteractive feature. Moreover, as shown in [2], data uploaded by participants may contain their private and sensitive information. Therefore, the auditing protocol should have the privacy-preserving

feature. Finally, due to limited resources of mobile users, the auditing protocol should be quite efficient. Therefore, a noninteractive lightweight privacy-preserving auditing scheme is needed for the mobile user to check images before downloading them.

Regardless of the technology implemented, a typical “noninteractive lightweight privacy-preserving auditing system (NLPAS)” includes three entities: the “crowdsourcing server (CS)” which stores images, the “mobile user (MU)” who audits images stored on the crowdsourcing server before downloading them, and the worker who collects images and uploads them to the CS. In practice, these entities are involved in two processes (i.e., the uploading process and the auditing process). During the uploading process, the worker collects images and uploads them to the CS. During the auditing process, the MU audits images stored on the CS and then determines whether to download them.

Security has vital significance for NLPAS. To avoid economic loss, the CS is not willing to transport images to the MU before the latter pays for them. On the other hand, the MU is not willing to pay for images before he/she can make sure that these images really meet the requirement. To handle this dilemma, it is reasonable to design a privacy-preserving auditing protocol, which allows the MU to check whether these images meet the requirement before downloading them. Unfortunately, the current security protocols for crowdsourcing systems (i.e., [3–40]) only consider how the CS checks images uploaded by workers. This leads to two issues. First, to earn more money, the CS is not willing to check images and provide true information to the MU. Second, the requirements of multiple MUs may vary, and the CS may not know them. For example, one MU may be interested in the hill in the image, while another MU may be interested in the lion in the image. In this case, it is impossible for the CS to know the requirements of multiple MUs. So, to make sure the images on the CS meet the requirements of the MU, it is desired to design a privacy-preserving auditing protocol for mobile crowdsourcing systems.

Efficiency is another serious concern for NLPAS. Due to limited resources of mobile devices, the MU is seriously concerned about the high computation cost arising from running the auditing protocol. At the same time, the CS will have to handle a lot of auditing requests from multiple MUs, and it is seriously concerned about the computation cost too. So, the newly designed auditing protocol should be lightweight. Taking both security and efficiency into account, we aim to design a noninteractive privacy-preserving lightweight auditing protocol on images in the mobile crowdsourcing system, which extracts features from images and then determines whether these features meet the MU's requirement. An auditing protocol for the mobile crowdsourcing system should fulfill the following requirements:

- (1) *Privacy Preserving.* When auditing images on the CS, the MU must be ensured that his/her requirement is not leaked to the CS or a third-party adversary. Otherwise, the CS may forge wrong information to pass the auditing, resulting in economic loss of the MU. At the same time, the CS must be ensured that the features of images are not leaked to the MU or a third-party adversary too. Otherwise, it will result in economic loss of the CS.
- (2) *Content Privacy.* During auditing, the CS should be ensured that the MU or a third-party adversary cannot extract content of images from exchanged information. Otherwise, the adversary may steal content of images stored on the CS, resulting in economic loss of the CS.
- (3) *Auditing.* After the auditing process, the MU should be able to determine whether the content in an image meets his/her requirement.

- (4) *Computation Cost.* The CS and the MU should be ensured that their computation costs are low when running auditing algorithms.
- (5) *Accuracy.* The CS and the MU should be ensured that the accuracy of the model used for extracting features from images during auditing is high.

Obviously, designing an auditing protocol for NLPAS is a nontrivial task, as the MU has to determine whether the images stored on the CS meet the requirement without downloading them. Recently, security protocols for mobile crowdsourcing systems have focused on image-checking techniques run by the CS. However, there is no protocol considering the image-checking technique established by the MU. Moreover, when focusing on this topic, we notice that there is no security scheme which can be directly used for satisfying all the above requirements. We will present the detailed analysis for arriving at this conclusion in the next section. This becomes a more serious issue, since more and more mobile crowdsourcing systems are being deployed. Motivated by this observation, in this paper, we mainly make three contributions:

- (1) We present a comprehensive set of requirements for auditing protocols in mobile crowdsourcing systems and show some security and efficiency problems of current data checking protocols in mobile crowdsourcing systems.
- (2) We propose a novel auditing protocol called NLPAS, which can check whether the images stored on the CS meet the MU's requirement. However, different from current data checking protocols for mobile crowdsourcing systems, NLPAS allows the MU instead of the CS to check images in a privacy-preserving manner. By doing so, the MU can determine whether the images meet the requirement before paying for them. At the same time, the CS can make sure that those images will not be leaked. To fulfill all these requirements, we will first introduce the "binary convolutional neural network (BCNN)" technique [41] to the newly designed auditing protocol for extracting features of images, and then we will design a novel privacy-preserving Hamming distance computation [42] technique for determining whether the images meet the MU's requirements. Since these two techniques are quite lightweight, the computation cost of NLPAS can be reduced significantly.
- (3) We analyze the security of NLPAS, showing it satisfies requirements (1), (2), and (3) in Section 1. And, we evaluate the efficiency of NLPAS, showing it satisfies requirements (4) and (5) in Section 1. We organize the remainder of this paper as follows. First, we investigate the related work in Section 2. Second, we describe the NLPAS protocol in Section 3, followed by security analysis and efficiency evaluation in Sections 4 and 5, respectively. Finally, in Section 6, we draw our conclusions.

2. Related Work

2.1. Data Trust. Data trust is an essential security problem in mobile crowdsourcing systems. Due to openness, workers in mobile crowdsourcing systems may have different security abilities, resulting in low data trust [3]. Recently, a lot of papers have been published, focusing on data trust in crowdsourcing systems, as illustrated below.

2.1.1. Voting-Based Data Checking. This sort of scheme takes observed results with the most observers as the true data [4–6]. For example, in [4], the authors allowed a number of observers to report their evaluation results on data and took the one as true data if most of the evaluation results were positive. Similarly, the authors in [5] aimed to find conflicts of data obtained from multiple workers, which could be seen as a variation of the voting-based data checking scheme.

2.1.2. Context Information-Based Data Checking. This sort of scheme uses context information such as location information to determine whether the data are true or not. For example, in [7], the authors required workers to upload location information together with data, while the CS used location information to determine whether the data were true or not. Similarly, the authors in [8, 9] used the worker's trajectory for determining whether the data were true or not.

2.1.3. Statistics-Based Data Checking. This sort of scheme [10–13] uses statistic methods for evaluating whether the data are true or not. For example, the authors in [10] used maximum likelihood estimation for checking data. And, the authors in [11] used maximum a posteriori (MAP) estimation for determining whether the data were true or not.

2.1.4. Gold Data Set-Based Data Checking. This sort of scheme uses a gold data set for checking data uploaded by workers [14, 15]. For example, in [15], the authors assigned fully trusted workers with a gold data set for checking data uploaded from other workers.

2.1.5. Data Redundancy Checking. This sort of scheme aims to address data redundancy issues [16, 17]. For example, in [16], the authors designed a scheme to find redundancy data from multiple workers and used redundancy data for estimating missing values.

From the above analysis, it can be seen that the existing schemes mainly focus on data checking performed by the CS and workers. And, they mainly consider whether the data are correct. However, existing schemes did not allow the mobile users to check data before downloading it. Moreover, existing schemes only focus on the correctness of data and do not consider whether the data match the MU's requirement since requirements from multiple mobile users may vary. This leads to a serious issue: The mobile user may waste money on nonmatched data. Therefore, an auditing scheme performed by the MU before data downloading is desired.

2.2. Data Privacy. Data privacy is another important problem in mobile crowdsourcing systems. First, if the data uploaded by workers are leaked, the CS and workers may lose money. More importantly, if the leaked data contain privacy of workers, the adversary may cause them harm. Second, if the data of mobile users are leaked, the adversary may deduce valuable information about mobile users [3]. Therefore, data privacy is a serious concern in crowdsourcing systems. Papers about data privacy in mobile crowdsourcing systems are illustrated below.

2.2.1. Encryption. This sort of scheme aims to encrypt data before uploading them [18–21]. For example, in [18], a homomorphic identity-based encryption algorithm was designed for protecting data uploaded by workers.

2.2.2. Differential Privacy. This sort of scheme adds perturbation to data [22–25]. For example, in [25], the authors added random perturbation to data uploaded by workers.

2.2.3. Location Privacy. This sort of scheme aims to protect location information of workers [26–31]. For example, in [26], the authors used k -anonymity for providing location privacy.

2.2.4. Personal Information Privacy. This sort of scheme aims to protect personal information of workers and mobile users [32–35]. For example, in [35], the authors defined multiple privacy level of personal information.

Recently, several new techniques such as blockchain and fog computing have been introduced to mobile crowdsourcing networks [36–40]. For example, in [36], the authors used blockchain for user authorization. And, in [38], the authors used fog computing for data aggregation and task allocation. Finally, the features of existing schemes are listed in Table 1.

From Table 1, it can be seen that the existing schemes can provide data trust and privacy protection in various ways. However, most schemes only consider one feature, either data trust or privacy preserving. And, no scheme provides both features. Moreover, all existing schemes do not support the noninteractive feature. This leads to a dilemma: on the one hand, to provide privacy protection, the MU cannot get data before paying for it; on the other hand, the MU has to check data before paying for it, to determine whether the data meet the requirement.

Furthermore, for images, this dilemma becomes more serious since the data volume of images is much larger than that of traditional texts. To handle this dilemma, it is desired to design a noninteractive lightweight privacy-preserving auditing protocol on images in mobile crowdsourcing networks, which allows the MU to efficiently determine whether the images meet the requirements without knowing anything about these images.

TABLE 1: Features of existing schemes.

	Data trust	Privacy preserving	Noninteractive
[4–6]	Voting-based data checking	×	×
[7–9]	Context information-based data checking	×	×
[10–13]	Statistics-based data checking	×	×
[14, 15]	Gold data set-based data checking	×	×
[16, 17]	Data redundancy checking	×	×
[18–21]	×	Encryption	×
[22–25]	×	Differential privacy	×
[26–31]	×	Location privacy	×
[32–35]	×	Personal information privacy	×
[36]	Blockchain-based authorization	×	×
[37]	×	Blockchain-based data privacy	×
[38]	×	Fog-based data privacy	×

3. NLPAS: The Protocol

3.1. Preliminaries

3.1.1. Binary Convolutional Neural Networks. A convolutional neural network [41] is a kind of neural networks whose forward propagation operation can be expressed as $Y = f(W \otimes X)$, where Y is the output tensor of the operation, $f(\cdot)$ is a nonlinear function, W and X are the weight tensor and the activation tensor generated by the previous neural network layer, and $\otimes$ is the convolution operation. A binary convolutional neural network is a kind of convolutional neural networks, whose weights $w \in W$ and activations $x \in X$ are 1 bit instead of the floating point. And, the forward propagation operation can be expressed as $Y = f(W \otimes X) = f((mB_W) \otimes nB_X) = f(mnB_W \otimes B_X)$, where m and n are integers, and each $b_W \in B_W$ is computed from the corresponding $w \in W$ as follows:

$$b_W = \text{sgn}(w) = \begin{cases} 1, & w \geq 0, \\ -1, & w < 0. \end{cases} \quad (1)$$

Similarly, each $b_X \in B_X$ is computed from the corresponding $x \in X$ as follows:

$$b_X = \text{sgn}(x) = \begin{cases} 1, & x \geq 0, \\ -1, & x < 0. \end{cases} \quad (2)$$

For bitwise b_W and b_X , the $B_W \otimes B_X$ operation can be efficiently computed using the XNOR-bitcount algorithm defined in [43].

Moreover, since the sgn function is not differentiable during backward propagation, Bengio et al. used the clip function instead of sgn as follows: $\text{clip}(\cdot) = \max(-1, \min(1, \cdot))$ [44]. By using the clip function, the binary convolutional neural network can use the same gradient descent algorithm as that of traditional neural networks to update parameters during training.

3.1.2. Hamming Distance. The Hamming distance [42] is defined below.

Given two n -bit binary vectors $\vec{a} = a_1, a_2, \dots, a_n$ and $\vec{b} = b_1, b_2, \dots, b_n$, where $(a_i \in \{0, 1\}, i = 1, \dots, n)$ and $(b_i \in \{0, 1\}, i = 1, \dots, n)$, the Hamming distance of $\vec{a}$ and $\vec{b}$ is defined as

$$d(\vec{a}, \vec{b}) = \sum_{i=1}^n (a_i \otimes b_i). \quad (3)$$

The above definition shows that the Hamming distance is the total number of different bits between $\vec{a}$ and $\vec{b}$. In other words, to compute the Hamming distance, we need to count the different bits between $\vec{a}$ and $\vec{b}$.

3.2. System Model. The main purpose of NLPAS is to determine whether the image on the CS meets the requirement of the MU in a privacy-preserving manner.

The main idea of NLPAS can be divided into two parts, namely, the feature extracting part and the Hamming distance computation part. For the feature extracting part, the MU first defines a binary convolutional neural network and trains it using a data set according to the user's requirement. Then, the MU extracts a binary vector ($\vec{a}$) from a template image using this binary convolutional neural network, which is used as the requirement feature. Finally, the MU sends the trained binary convolutional neural network to the CS, and the latter extracts a binary vector ($\vec{b}$) from the image to be audited using this trained network, which is used as the auditing feature. Since the binary convolutional neural network is quite lightweight, NLPAS can achieve high efficiency.

For the Hamming distance computation part, the MU and the CS hide the two input vectors ($\vec{a}$ and $\vec{b}$) in a carrier number, respectively. And then, all operations for counting different bits between these two vectors are based on five basic mathematical operations, namely, addition, subtraction, multiplication, division, and modulo operations. Since these five basic mathematical operations are quite lightweight, our scheme can achieve high efficiency.

Based on the feature extracting and Hamming distance computation techniques, the MU compares the Hamming distance of $\vec{a}$ and $\vec{b}$ with a threshold to determine whether the interested image on the CS meets the requirement.

The system model of NLPAS is shown in Figure 1, which includes three phases as described below. And, the notations are listed in Table 2.

FIGURE 1: System model of NLPAS.

TABLE 2: Notations in this paper.

Notation	Description
mode	Binary convolutional neural network model trained by the MU
SK_A	Private parameters of the MU
PK_A	Public parameters of NLPAS
$\vec{a} = a_1, \dots, a_i, \dots, a_n$	Feature vector of the MU
$\vec{b} = b_1, \dots, b_i, \dots, b_n$	Feature vector of the CS
n	Length of the two binary vectors $\vec{a}$ and $\vec{b}$
l	Security strength of NLPAS
cipher A = {C, D}	Ciphertexts generated by the MU
$C = \{c_1, c_2, \dots, c_n\}$	The first set of random numbers that $\vec{a}$ is hidden in
$D = \{d_1, d_2, \dots, d_n\}$	The second set of random numbers that $\vec{a}$ is hidden in
cipher B = {E, F}	Ciphertexts generated by the CS
$d(\vec{a}, \vec{b})$	Hamming distance of $\vec{a}$ and $\vec{b}$
$\{e_i, i = 1, 2, \dots, n\}$	The first set of random numbers that $d(\vec{a}, \vec{b})$ is hidden in
$\{f_i, i = 1, 2, \dots, n\}$	The second set of random numbers that $d(\vec{a}, \vec{b})$ is hidden in
thres	Threshold for determining whether the image on the CS meets the MU's requirement
w	Prime number for counting different bits in $\vec{a}$ and $\vec{b}$
g	Prime number used as a carrier
$P = \{p_1, p_2, \dots, p_n\}$	The first set of random numbers for hiding g
$Q = \{q_1, q_2, \dots, q_n\}$	The second set of random numbers for hiding g
$S = \{s_1, s_2, \dots, s_n\}$	The third set of random numbers for hiding g
$T = \{t_1, t_2, \dots, t_n\}$	The fourth set of random numbers for hiding g
$V = \{v_i = p_i + q_i g, i = 1, 2, \dots, n\}$	The first set of bases for hiding vectors
$U = \{u_i = s_i + t_i g, i = 1, 2, \dots, n\}$	The second set of bases for hiding vectors
J, K	Transitional values for extracting the hamming distance
X, Y	Values that contain the hamming distance

3.2.1. The Initialization Phase. During this phase, the MU defines a binary convolutional neural network model (mode), trains it, and extracts a binary vector $\vec{a}$ from the template image using the mode. Then, the MU generates public and private cryptographic parameters for the NLPAS system. These cryptographic parameters will be used for hiding vectors and extracting results in the following hiding

phase and extracting phase. The initialization algorithm is described as follows.

$\{SK_A, PK_A\} \leftarrow \text{Init}(n, l)$. This algorithm is run by the MU for generating system parameters for NLPAS. It takes as input the length of input vectors (i.e., n) and the security strength of NLPAS (i.e., l -bit) and outputs the set of private and public cryptographic parameters (i.e., SK_A and PK_A).

Then, the MU sends the public parameter (PK_A) and the trained model (mode) to the CS. And, the latter extracts a binary vector b from the interested image stored on it.

After the initialization phase, the MU holds ($SK_A, PK_A, \vec{a}, \text{mode}$) and the CS holds (PK_A, b, mode).

3.2.2. The Hiding Phase. When the MU wants to compute the Hamming distance $d(\vec{a}, b)$ where $\vec{a}$ is known only by the MU and b is known only by the CS, it establishes the hiding process, by running the Hiding A algorithm and sending the results to the CS. The Hiding A algorithm is described below.

$\{\text{cipher A}\} \leftarrow \text{Hiding A}(\vec{a}, SK_A, PK_A)$. This algorithm is run by the MU for hiding the binary vector $\vec{a}$ into a ciphertext. It takes as inputs the MU's binary vector (i.e., $\vec{a}$), the MU's private parameter (i.e., SK_A), and the MU's public parameter (i.e., PK_A) and outputs the ciphertext (i.e., cipher A).

Upon receiving the ciphertext (i.e., cipher A), the CS injects its vector (i.e., b) into cipher A using the Injecting B algorithm and gets the updated ciphertext cipher B. Then, the CS sends cipher B back to the MU, and the Hamming distance of vectors $\vec{a}$ and b is included in cipher B. The Injecting B algorithm is described below.

$\{\text{cipher B}\} \leftarrow \text{Injecting B}(b, \text{cipher A}, PK_A)$. This algorithm is run by the CS for injecting the binary vector b into cipher A. It takes as inputs the CS's binary vector (i.e., b), the MU's ciphertext (i.e., cipher A), and the MU's public parameter (i.e., PK_A) and outputs the updated ciphertext (i.e., cipher B).

After the hiding phase, the MU gets cipher B, and the Hamming distance of $\vec{a}$ and b is hidden in cipher B for being extracted in the following extracting phase.

3.2.3. The Extracting Phase. After receiving the updated ciphertext (i.e., cipher B) from the CS, the MU extracts the Hamming distance from cipher B using the Extra algorithm, which is described below.

$d(\vec{a}, b) \leftarrow \text{Extra}(\text{cipher B}, SK_A, PK_A)$. This algorithm is run by the MU for extracting the Hamming distance from the updated ciphertext (i.e., cipher B). It takes as inputs the updated ciphertext cipher B, the MU's private parameter (i.e., SK_A), and the MU's public parameter (i.e., PK_A) and outputs the Hamming distance of $\vec{a}$ and b (i.e., $d(\vec{a}, b)$).

After the extracting phase, the MU gets the Hamming distance of $\vec{a}$ and b (i.e., $d(\vec{a}, b)$). Then, the MU sets a threshold value thres . If $d(\vec{a}, b) \geq \text{thres}$, the interested image stored on the CS does not match the MU's requirement. Otherwise, the interested image on the CS matches the MU's requirement.

In the above system model, the MU's vector $\vec{a}$ is hidden in cipher A using the Hiding A algorithm, which cannot be known by the CS. At the same time, the CS's vector b is hidden in cipher B using the Injecting B algorithm, which cannot be known by the MU. Therefore, NLPAS can achieve the privacy-preserving goal described in Section 1.

In the above system model, the CS's vector b is hidden in cipher B using the Injecting B algorithm, which cannot be known by the MU. Therefore, the MU only knows the

Hamming distance between $\vec{a}$ and b and does not know b and the corresponding interested image. Therefore, NLPAS can achieve the content privacy goal described in Section 1.

3.3. Construction. The construction of NLPAS is a tuple (Init, Hiding A, Injecting B, Extra) of probabilistic polynomial time algorithms as shown in Figure 2, and the details are defined below.

$\{SK_A, PK_A\} \leftarrow \text{Init}(n, l)$. The MU runs this algorithm for generating system parameters for NLPAS as follows. First, the MU generates a l -bit prime number w for counting different bits. Second, the MU generates a large prime number g with the length $2l + 2 + \log_2 n$ as the carrier of NLPAS. Third, the MU generates four sets of positive random numbers for hiding g , namely, $P = \{p_1, p_2, \dots, p_n\}$, $Q = \{q_1, q_2, \dots, q_n\}$, $S = \{s_1, s_2, \dots, s_n\}$, and $T = \{t_1, t_2, \dots, t_n\}$, where $\sum_{i=1}^n p_i < (w - n)/2$, $\sum_{i=1}^n s_i < (w - n)/2$. Fourth, the MU computes two sets of bases for hiding vectors as follows: $V = \{v_i = p_i + q_i g, i = 1, 2, \dots, n\}$ and $U = \{u_i = s_i + t_i g, i = 1, 2, \dots, n\}$. Finally, the MU gets $SK_A = \{g, P, Q, S, T, V, U\}$ and $PK_A = w$.

$\{\text{cipher A}\} \leftarrow \text{Hiding A}(\vec{a}, SK_A, PK_A)$. The MU runs this algorithm for hiding the binary vector $\vec{a} = a_1, \dots, a_n$ into a ciphertext as follows. The MU computes $c_i = v_i + w$ and $d_i = u_i$ for each $a_i = 1$ in $\vec{a}$. Otherwise, $c_i = v_i$ and $d_i = u_i + w$. Then, the MU gets cipher A = $\{C = \{c_1, c_2, \dots, c_n\}, D = \{d_1, d_2, \dots, d_n\}\}$.

$\{\text{cipher B}\} \leftarrow \text{Injecting B}(b, \text{cipher A}, PK_A)$. The CS runs this algorithm for injecting the binary vector $b = b_1, \dots, b_n$ into cipher A as follows. First, the CS computes $e_i = c_i$ and $f_i = w d_i$ for each $b_i = 1$ in b . Otherwise, $e_i = w c_i$ and $f_i = d_i$. Second, the CS computes $E = \sum_{i=1}^n e_i$ and $F = \sum_{i=1}^n f_i$. Finally, the CS gets cipher B = $\{E, F\}$.

$d(\vec{a}, b) \leftarrow \text{Extra}(\text{cipher B}, SK_A, PK_A)$. The MU runs this algorithm for extracting the Hamming distance from the updated ciphertext (i.e., cipher B) as follows. First, the MU computes $J = E \bmod g$ and $K = F \bmod g$. Second, the MU computes $X = J - (J \bmod (w^2))/w^2$ and $Y = K - (K \bmod (w^2))/w^2$. Third, the MU computes $d(\vec{a}, b) = X + Y$.

In the above construction, NLPAS uses only a few simple mathematical operations (i.e., addition, subtraction, multiplication, division, and modulo operations) instead of time-consuming cryptographic operations such as modular exponentiation. Therefore, it enjoys high efficiency. We will further evaluate the efficiency of NLPAS in Section 5.

4. Security Analysis

In this section, we first show that NLPAS is correct and then analyze the security of NLPAS according to the security requirements described in Section 1 (i.e., privacy preserving, content privacy, and auditing).

4.1. Correctness. In the construction in Section 3.3, we use X for counting the bits where $a_i = 1$ and $b_i = 0$. Similarly, we use Y for counting the bits where $a_i = 0$ and $b_i = 1$.

The initialization phase	The hiding phase	The extracting phase
<p>Procedure init algorithm</p> <p>Input: vector length (n), security strength (l).</p> <p>Output: public parameter (PK_A), private parameter (SK_A).</p> <p>Step 1: MU generates two primes w and g.</p> <p>Step 2: MU randomly generates four sets</p> <p>$P = \{p_1, p_2, \dots, p_n\}$, $Q = \{q_1, q_2, \dots, q_n\}$, $S = \{s_1, s_2, \dots, s_n\}$, and $T = \{t_1, t_2, \dots, t_n\}$.</p> <p>Step 3: MU computes</p> <p>$V = \{v_i = p_i + q_i g, i = 1, 2, \dots, n\}$ and $U = \{u_i = s_i + t_i g, i = 1, 2, \dots, n\}$.</p> <p>Step 4: MU gets</p> <p>$SK_A = \{g, P, Q, S, T, V, U\}$ and $PK_A = w$.</p> <p>end procedure Init algorithm</p>	<p>Procedure hiding A algorithm</p> <p>Input: vector ($\vec{a}$), parameters (PK_A and SK_A).</p> <p>Output: cipher A.</p> <p>Step 1: for each $a_i \in \vec{a}$, if $a_i = 1$, MU computes $c_i = v_i + w$ and $d_i = u_i$. Otherwise, MU computes $c_i = v_i$ and $d_i = u_i + w$.</p> <p>Step 2: MU gets cipher $A = \{C = \{c_1, c_2, \dots, c_n\}, D = \{d_1, d_2, \dots, d_n\}\}$.</p> <p>end procedure Hiding A algorithm</p> <p>procedure Injecting B algorithm</p> <p>Input: vector ($\vec{b}$), ciphertext (cipher A), parameter (PK_A).</p> <p>Output: cipher B.</p> <p>Step 1: for each $b_i \in \vec{b}$, if $b_i = 1$, CS computes $e_i = c_i$ and $f_i = w d_i$. Otherwise, CS computes $e_i = w c_i$ and $f_i = d_i$.</p> <p>Step 2: CS computes $E = \sum_{i=1}^n e_i$ and $F = \sum_{i=1}^n f_i$, and cipher $B = \{E, F\}$.</p> <p>end procedure Injecting B algorithm</p>	<p>Procedure extra algorithm</p> <p>Input: ciphertext (cipher B), parameters (PK_A and SK_A).</p> <p>Output: $d(\vec{a}, \vec{b})$.</p> <p>Step 1: MU computes</p> <p>$J = E \bmod g$ and $K = F \bmod g$.</p> <p>Step 2: MU computes</p> <p>$X = J - (J \bmod(w^2))/w^2$ and $Y = K - (K \bmod(w^2))/w^2$.</p> <p>Step 3: MU computes</p> <p>$d(\vec{a}, \vec{b}) = X + Y$.</p> <p>end procedure Extra algorithm</p>

FIGURE 2: Construction of NLPAS.

Therefore, the Hamming distance of $\vec{a}$ and $\vec{b}$ can be computed as $d(\vec{a}, \vec{b}) = X + Y$.

In this section, we shall show that X can really be used for counting the bits where $a_i = 1$ and $b_i = 0$. And, the meaning of Y can be analyzed in a similar way.

We start analyzing the meaning of X from the variable e_i as follows. First, according to the Injecting B algorithm, e_i can be written as

$$e_i = \begin{cases} c_i, & b_i = 1, \\ w c_i, & b_i = 0. \end{cases} \quad (4)$$

Second, taking the value of c_i in the Hiding A algorithm into consideration, e_i can be further written as

$$e_i = \begin{cases} v_i + w, & (a_i = 1, b_i = 1), \\ v_i, & (a_i = 0, b_i = 1), \\ w(v_i + w), & (a_i = 1, b_i = 0), \\ w v_i, & (a_i = 0, b_i = 0). \end{cases} \quad (5)$$

Third, taking the value of v_i in the Init algorithm into consideration, e_i can be written as

$$e_i = \begin{cases} p_i + q_i g + w, & (a_i = 1, b_i = 1), \\ p_i + q_i g, & (a_i = 0, b_i = 1), \\ w(p_i + q_i g + w), & (a_i = 1, b_i = 0), \\ w(p_i + q_i g), & (a_i = 0, b_i = 0). \end{cases} \quad (6)$$

Fourth, considering all the four conditions (i.e., $(a_i = 1, b_i = 1)$, $(a_i = 1, b_i = 0)$, $(a_i = 0, b_i = 1)$, and $(a_i = 0, b_i = 0)$) together, we can compute

$$\begin{aligned} E = \sum_{i=1}^n e_i &= \sum_{a_i=1, b_i=1} (p_i + q_i g + w) + \sum_{a_i=1, b_i=0} (w(p_i + q_i g + w)) \\ &+ \sum_{a_i=0, b_i=1} (p_i + q_i g) + \sum_{a_i=0, b_i=0} (w(p_i + q_i g)) \\ &= \left(\sum_{a_i=1, b_i=0} 1 \right) w^2 + \left(\sum_{a_i=1, b_i=1} 1 + \sum_{b_i=0} p_i w \right) + \sum_{a_i=0, b_i=1} p_i \\ &+ \left(\sum_{b_i=1} q_i + \sum_{b_i=0} w q_i \right) g. \end{aligned} \quad (7)$$

Fifth, since the length of w is l -bit, the length of $(\sum_{(a_i=1, b_i=0)} 1) w^2$ should be no more than $\log_2 n + 2l$. Since $\sum_{i=1}^n p_i < (w - n)/2$, the length of $(\sum_{(a_i=1, b_i=1)} 1 + \sum_{b_i=0} p_i) w$ should be no more than $2l - 1$, and the length of $\sum_{(a_i=0, b_i=1)} p_i$ should be no more than $l - 1$. Therefore, the length of $(\sum_{(a_i=1, b_i=0)} 1) w^2 + (\sum_{(a_i=1, b_i=1)} 1 + \sum_{b_i=0} p_i) w + \sum_{(a_i=0, b_i=1)} p_i$ should be no more than $\log_2 n + 2l + 2$. That is to say, $(\sum_{(a_i=1, b_i=0)} 1) w^2 + (\sum_{(a_i=1, b_i=1)} 1 + \sum_{b_i=0} p_i) w + \sum_{(a_i=0, b_i=1)} p_i < g$. So, we get

$$\begin{aligned} J = E \bmod g &= \left(\sum_{a_i=1, b_i=0} 1 \right) w^2 + \left(\sum_{a_i=1, b_i=1} 1 + \sum_{b_i=0} p_i \right) w \\ &+ \sum_{a_i=0, b_i=1} p_i. \end{aligned} \quad (8)$$

Sixth, since the length of $(\sum_{(a_i=1, b_i=1)} 1 + \sum_{b_i=0} p_i) w$ is no more than $2l - 1$ and the length of $\sum_{(a_i=0, b_i=1)} p_i$ is no more than $l - 1$, the length of $(\sum_{(a_i=1, b_i=1)} 1 + \sum_{b_i=0} p_i) w + \sum_{(a_i=0, b_i=1)} p_i$ is no more than $2l - 1$. That is to say,

$(\sum_{(a_i=1, b_i=1)} 1 + \sum_{b_i=0} p_i)w + \sum_{(a_i=0, b_i=1)} p_i < w^2$. So, we get $J \bmod (w^2) = (\sum_{(a_i=1, b_i=1)} 1 + \sum_{b_i=0} p_i)w + \sum_{(a_i=0, b_i=1)} p_i$. Finally, we get

$$X = \frac{J - J \bmod (w^2)}{w^2} = \frac{(\sum_{(a_i=1, b_i=0)} 1)w^2}{w^2} = \sum_{a_i=1, b_i=0} 1. \quad (9)$$

This is really the total number of bits where $a_i = 1$ and $b_i = 0$. Similarly, we can learn that Y is the total number of bits where $a_i = 0$ and $b_i = 1$. Therefore, the Hamming distance of $\vec{a}$ and $\vec{b}$ is $d(\vec{a}, \vec{b}) = X + Y$.

From the above discussion, we can see that the main idea of privacy-preserving Hamming distance computation includes two points. First, we hide the information of $\vec{a}$ and $\vec{b}$ in a big prime g . Second, the different bits (e.g., $a_i = 1$ and $b_i = 0$) are counted in an independent part of E , e.g., $(\sum_{(a_i=1, b_i=0)} 1)w^2$, which can be extracted using several modulo operations.

4.2. Privacy Preserving. The privacy-preserving requirement is to ensure that the adversary cannot extract $\vec{a}$ or $\vec{b}$ transmitted in the hiding phase.

We first consider the privacy-preserving requirement for $\vec{a}$, where the adversary can be anybody who is able to get cipher A including the CS.

From the Hiding A algorithm defined in Section 3.3, it can be seen that $a_i \in \vec{a}$ is hidden in c_i and d_i . Since p_i, q_i, s_i, t_i , and g are random numbers known only by the MU, c_i and d_i are random numbers too.

Moreover, since the length of g is much longer than p_i, s_i , and w , the lengths of c_i and d_i are determined by $q_i g$ and $t_i g$. Since q_i and t_i are random numbers, the lengths of c_i and d_i are randomly determined by q_i and t_i , regardless of the value of a_i (i.e., 0 or 1). Therefore, for $a_i = 1$ (i.e., $c_i = v_i + w = p_i + q_i g + w$) and $a_j = 0$ (i.e., $c_j = v_j = p_j + q_j g$), it may be $c_i < c_j$. Similarly, for $a_i = 1$ and $a_j = 0$, it may be $d_i > d_j$. That is to say, the set of c_i and d_i including w may or may not be bigger than the set of c_i and d_i without w . So, the adversary cannot get the value of a_i from c_i and d_i by determining that a bigger random c_i represents 1, or a smaller random d_i represents 1. In other words, without knowing the set of random secrets (p_i, q_i, s_i, t_i, g) , the adversary can extract a_i from c_i or d_i only with a negligible probability.

Furthermore, if the length of $\vec{a}$ is n , the probability that the adversary can get $\vec{a}$ is $(1/2)^n$.

We then consider the privacy-preserving requirement for $\vec{b}$, where the adversary can be anybody who is able to get cipher B including the MU.

From the Injecting B algorithm defined in Section 3.3, it can be seen that b_i is hidden in E and F using the addition operations. Knowing the result of addition, the adversary can extract b_i only with a negligible probability. Therefore, the privacy of $\vec{b}$ is ensured by the addition operation.

Moreover, assuming the MU is the adversary who wants to extract $\vec{b}$ from cipher $B = \{E, F\}$, the MU has to solve the

two equations $E = (\sum_{(a_i=1, b_i=0)} 1)w^2 + (\sum_{(a_i=1, b_i=1)} 1 + \sum_{b_i=0} p_i)w + \sum_{(a_i=0, b_i=1)} p_i + (\sum_{(b_i=1)} q_i + \sum_{b_i=0} w q_i)g$ and $F = (\sum_{(a_i=0, b_i=1)} 1)w^2 + (\sum_{(a_i=0, b_i=0)} 1 + \sum_{b_i=1} s_i)w + \sum_{(a_i=1, b_i=0)} s_i + (\sum_{(b_i=0)} t_i + \sum_{b_i=1} w t_i)g$. Since the MU knows $(p_i, q_i, s_i, t_i, g, w)$, these two equations can be treated as two linear equations with n unknown numbers (i.e., $(b_1, \dots, b_n)$). Therefore, when $n > 2$, there are a number of solutions for them. In addition, for n -bit $\vec{b}$, the number of solutions is $(1/2)^n$. That is to say, the probability that the MU can extract $\vec{b}$ from cipher $B = (E, F)$ is $(1/2)^n$.

From the above discussion, it can be seen that the adversary cannot extract $\vec{a}$ or $\vec{b}$ transmitted in the hiding phase. Therefore, NLPAS can achieve the privacy-preserving goal.

4.3. Content Privacy. The content privacy requirement is to ensure that the adversary cannot extract content of the interested image stored on the CS from cipher B . The content of the interested image is included in $\vec{b}$. Since the adversary cannot extract $\vec{b}$ from cipher B as shown in the previous subsection, the content of the interested image stored on the CS cannot be extracted. Therefore, NLPAS can achieve the content privacy goal.

4.4. Auditing. The auditing requirement is to ensure that the MU can determine whether the content in the image stored on the CS meets the MU's requirement. This is ensured by the Hamming distance. If the Hamming distance of $\vec{a}$ and $\vec{b}$ is smaller than the threshold, the MU can determine that the content in the image is the one he/she needs. Otherwise, the content in the image is not needed by the MU. Moreover, the correctness of Hamming distance computation is ensured in Section 4.1.

5. Efficiency Evaluation

As shown in Section 3, NLPAS includes two parts: feature extracting using the binary convolutional neural network and similarity computation using the privacy-preserving Hamming distance. Therefore, we mainly evaluate the computation costs consumed in these two parts. Moreover, for the feature extracting part, we will evaluate the accuracy of the trained model (i.e., mode).

5.1. Accuracy. To provide a benchmark of efficiency evaluation, we used the MNIST data set [45] and LeNet [46] for comparing the accuracy of the binary convolutional neural network with that of the full-precision convolutional neural network.

MNIST [45] is a data set of handwritten digits, which contains a training set of 60,000 examples and a test set of 10,000 examples. And, all examples in the training and test data sets are 28×28 binary images.

LeNet [46] is a convolutional neural network with three convolutional layers, two subsampling layers, two full connection layers, an input layer, and an output layer.

For implementation, we used the BMXnet [47], which provided basic binarization operations for convolutional neural networks. After experimentation, we got the results as shown in Table 3.

From Table 3, it can be seen that

- (1) The accuracy of the binary LeNet is slightly lower than that of the full-precision LeNet. The accuracy reduced by using the binary LeNet is around $0.99 - 0.97/0.99 \approx 2\%$.
- (2) The model size of the binary LeNet is much lower than that of the full-precision LeNet. The memory saved by binary LeNet is around $4.6 - 0.2/4.6 \approx 95.7\%$.

In other words, by using the binary convolutional neural network instead of the traditional full-precision convolutional neural network, the accuracy is only slightly reduced, but the memory is largely saved. Therefore, the binary convolutional neural network is quite suitable for the mobile crowdsourcing network, where mobile devices are with limited storage resources. The above evaluation shows that NLPAS fulfills the fifth requirement listed in Section 1 (i.e., the accuracy requirement).

5.2. Computation Costs. The computation cost of NLPAS includes the time cost consumed by the binary LeNet model and those consumed by mathematical operations. To test these time costs, we conducted our experiment on a laptop with an Intel i7-4770hq processor and an ubuntu-18.04 operating system. Then, we used OPENSSE [48] as the cryptographic library.

For the binary LeNet, we take the features extracted by the last full-connection layer as the input vectors (i.e., $\vec{a}$ and $\vec{b}$). Therefore, the vector length is $n = 84$ [46]. To provide a basic security level, we set $l = 256$, and the length of g is $\log_2 n + 2l + 2 = 521$. To make sure $\sum_{i=1}^n p_i < (w - n)/2$ and $\sum_{i=1}^n s_i < (w - n)/2$, we set the lengths of p_i and s_i to be 500 bit. Then, we set the lengths of q_i and t_i to be 683 bit, so that the lengths of v_i and u_i are around 1024 bit.

After the initial settings, we can count the mathematical operations in the hiding and extracting phases as listed in Table 4. From Table 4, it can be seen that all mathematical operations are run over 1024 bit and 512 bit fields.

Then, we tested the time costs consumed by these mathematical operations on the above laptop, and the average results of running them for 1,000,000 times are shown in Table 5. From Table 5, it can be seen that the time costs of mathematical operations are at the μs level.

Taking the results in Table 5 into Table 4, we can get the computation costs of algorithms in NLPAS, as shown in Table 6. From Table 6, it can be seen that the computation cost of mathematical operations on the MU (i.e., time costs of Hiding A and Extra) is much lower than that on the CS (i.e., Injecting B). Therefore, NLPAS is suitable for mobile crowdsourcing networks, where MU is with limited computation resources.

The time costs of the binary LeNet and the full-precision LeNet are shown in Table 7, where the results are average values of running the feature extracting process for 1,000,000

TABLE 3: Accuracy comparison.

	Accuracy	Model size
Binary LeNet	0.97	0.2 MB
Full-precision LeNet	0.99	4.6 MB

TABLE 4: Number of mathematical operations in NLPAS.

	Hiding A	Injecting B	Extra
$+$ (1024 – bit)	84	166	0
$\times$ (1024 – bit)	0	84	0
mod (1024 – bit)	0	0	2
mod (512 – bit)	0	0	2
$\div$ (512 – bit)	0	0	2
$-$ (512 – bit)	0	0	2
$+$ (512 – bit)	0	0	2

TABLE 5: Time costs of mathematical operations (unit: μs).

$+$ (1024 – bit)	0.22
$\times$ (1024 – bit)	1.69
mod (1024 – bit)	2.19
mod (512 – bit)	0.98
$\div$ (512 – bit)	1.03
$-$ (512 – bit)	0.16
$+$ (512 – bit)	0.14

TABLE 6: Computation costs of algorithms in NLPAS (unit: μs).

Hiding A	Injecting B	Extra
18.48	178.48	9.00

TABLE 7: Computation costs of algorithms in NLPAS (unit: μs).

Binary LeNet	Full-precision LeNet
56.6	1435.2

times. From Table 7, it can be seen that the computation cost of feature extracting in NLPAS can be largely reduced by using the binary convolutional neural network instead of the full-precision convolutional neural network.

The above evaluation shows that NLPAS fulfills the fourth requirement listed in Section 1 (i.e., the computation cost requirement).

5.3. Implementation of NLPAS. To make sure that NLPAS can work well, we implemented it. In our experimental environment, there were one laptop and one computer. The laptop acts as the MU, and the computer acts as the CS. The result shows that the total running time in the auditing protocol is approximately 0.3 ms. Therefore, NLPAS is feasible for being deployed in the real world.

6. Conclusions

In this paper, we have proposed a noninteractive lightweight privacy-preserving auditing protocol on images in mobile crowdsourcing networks called NLPAS. NLPAS allows the

mobile user to audit images stored on the crowdsourcing server without downloading them. Moreover, to achieve high efficiency, this paper introduced the binary convolutional neural network technique to the newly proposed auditing protocol and designed a novel privacy-preserving Hamming distance computation algorithm using basic mathematical operations. Experimental results show that NLPAS is feasible for real-world applications.

In this paper, we mainly focused on the privacy-preserving issue of the newly designed auditing protocol for mobile crowdsourcing networks. However, several more issues are to be addressed in the future. First, NLPAS does not consider the integrity of transmitted messages. Therefore, a new security protocol is needed to prevent these messages from being tampered by adversaries. Second, NLPAS used the binary convolutional neural network for extracting a binary vector from images. However, in many scenarios, feature vectors may be extracted using full-precision neural networks, which are not binarized. Therefore, a new technique is needed to convert the full-precision feature vector to a binarized one. To address these issues, future works are needed.

Data Availability

The data used to support the findings of this study are available at <http://yann.lecun.com/exdb/mnist/>.

Conflicts of Interest

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

Acknowledgments

This paper was supported by the NSFC (nos. 71402070 and 61101088), the NSF of Jiangsu Province (no. BK20161099), and the Jiangsu Provincial Key Laboratory of Computer Network Technology.

References

- [1] J. Howe, "The rise of crowdsourcing," *Wired Magazine*, vol. 14, no. 6, pp. 1–4, 2006.
- [2] D. He, M. S. Chan, and M. Guizani, "User privacy and data trustworthiness in mobile crowd sensing," *IEEE Wireless Communications*, vol. 22, no. 1, pp. 28–34, 2015.
- [3] W. Feng, Z. Yan, H. Zhang et al., "A survey on security, privacy and trust in mobile crowdsourcing," *IEEE Internet of Things Journal*, vol. 5, no. 4, 2017.
- [4] L. R. Varshney, "Privacy and reliability in crowdsourcing service delivery," in *Proceedings of the 2012 Annual SRII Global Conference*, San Jose, CA, USA, July 2012.
- [5] J. Ren, Y. Zhang, K. Zhang, and X. Shen, "SACRM: social aware crowdsourcing with reputation management in mobile sensing," *Computer Communications*, vol. 65, pp. 55–65, 2015.
- [6] A. Etuk, T. J. Norman, C. Bisdikian, and M. Srivatsa, "A trust assessment framework for inferencing with uncertain streaming information," in *Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM'2013)*, pp. 475–480, San Diego, CA, USA, March 2013.
- [7] R. W. Ouyang, M. Srivastava, A. Toniolo, and T. J. Norman, "Truth discovery in crowdsourced detection of spatial events," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 4, pp. 1047–1060, 2016.
- [8] B. Kantarci and H. T. Mouftah, "Trustworthy crowdsourcing via mobile social networks," in *Proceedings of the 2014 IEEE Global Communications Conference*, pp. 2905–2910, Austin, TX, USA, December 2014.
- [9] B. Kantarci and H. T. Mouftah, "Mobility-aware trustworthy crowdsourcing in cloud-centric internet of things," in *Proceedings of the 2014 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, Funchal, Portugal, June 2014.
- [10] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Proceedings of the International Conference Pervasive Computing (PERVASIVE'12)*, pp. 138–155, Helsinki, Finland, May 2012.
- [11] R. W. Ouyang, L. M. Kaplan, A. Toniolo, M. Srivastava, and T. J. Norman, "Aggregating crowdsourced quantitative claims: additive and multiplicative models," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1621–1634, 2016.
- [12] T. Kubota and M. Aritsugi, "How many ground truths should we insert? having good quality of labeling tasks in crowdsourcing," in *Proceedings of the IEEE Conference Computer Software and Applications Conference (COMPSAC'15)*, pp. 796–805, Taichung, Taiwan, July 2015.
- [13] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Y. Zhao, "Defending against sybil devices in crowdsourced mapping services," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services—MobiSys'16*, pp. 179–191, Singapore, June 2016.
- [14] C. Prandi, S. Ferretti, S. Mirri, and P. Salomoni, "A trustworthiness model for crowdsourced and crowdsensed data," in *Proceedings of the Conference Trustcom/BigDataSE/ISPA*, pp. 1261–1266, Helsinki, Finland, August 2015.
- [15] G. Drosatos, P. S. Efraimidis, I. N. Athanasiadis, E. D'Hondt, and M. Stevens, "A privacy-preserving cloud computing system for creating participatory noise maps," in *Proceedings of the IEEE Annual Conference Computer Software and Applications (COMPSAC)*, Article ID 581586, Izmir, Turkey, July 2012.
- [16] C. Meng, W. Jiang, Y. Li et al., "Truth discovery on crowd sensing of correlated entities," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems—SenSys'15*, pp. 150–163, Seoul, South Korea, November 2015.
- [17] T. Zhou, Z. Cai, K. Wu, Y. Chen, and M. Xu, "FIDC: a framework for improving data credibility in mobile crowdsensing," *Computer Networks*, vol. 120, pp. 157–169, 2017.
- [18] F. G. MntherMark and P. ManulisAndreas, "Privacy-enhanced participatory sensing with collusion resistance and data aggregation," in *Proceedings of the Cryptology and Network Security (CANS'14)*, pp. 321–336, Hong Kong, China, December 2014.
- [19] G. Zhuo, Q. Jia, L. Guo, M. Li, and P. Li, "Privacy-preserving verifiable data aggregation and analysis for cloud-assisted mobile crowdsourcing," in *Proceedings of the Annual IEEE Conference Computer Communications (INFOCOM'16)*, pp. 1–9, San Francisco, CA, USA, April 2016.
- [20] S. Blasco, J. Bustos-Jimenez, G. Font, A. Hevia, and M. Grazia Prato, "A three-layer approach for protecting smart-citizens

- privacy in crowdsensing projects,” in *Proceedings of the International Conference of the Chilean Computer Science Society (SCCC’15)*, pp. 1–5, Santiago, Chile, November 2015.
- [21] C. Miao, W. Jiang, L. Su et al., “Cloud-enabled privacy-preserving truth discovery in crowd sensing systems,” in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems—SenSys’15*, pp. 183–196, Seoul, South Korea, November 2015.
- [22] J. Chen, H. Ma, and D. Zhao, “Private data aggregation with integrity assurance and fault tolerance for mobile crowdsensing,” *Wireless Networks*, vol. 23, no. 1, pp. 131–144, 2015.
- [23] S. Wang, L. Huang, M. Tian, W. Yang, H. Xu, and H. Guo, “Personalized privacy-preserving data aggregation for histogram estimation,” in *Proceedings of the IEEE Conference Global Communications (GLOBECOM’15)*, pp. 1–6, San Diego, CA, USA, December 2015.
- [24] L. R. Varshney, A. Vempaty, and P. K. Varshney, “Assuring privacy and reliability in crowdsourcing with coding,” in *Proceedings of the Information Theory and Applications Workshop (ITA’14)*, pp. 1–6, San Diego, CA, USA, February 2014.
- [25] H. Jin, L. Su, H. Xiao, and K. Nahrstedt, “Inception,” in *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing—MobiHoc’16*, pp. 341–350, Paderborn Germany, July 2016.
- [26] Y. Wu, Y. Wu, H. Peng, H. Chen, and C. Li, “Magiccrowd: a crowd based incentive for location-aware crowd sensing,” in *Proceedings of the IEEE Conference Wireless Communications and Networking (WCNC’16)*, pp. 1–6, Doha, Qatar, April 2016.
- [27] L. Pournajaf, L. Xiong, and V. Sunderam, “Dynamic data driven crowd sensing task assignment,” *Procedia Computer Science*, vol. 29, pp. 1314–1323, 2014.
- [28] L. Pournajaf, L. Xiong, V. Sunderam, and S. Goryczka, “Spatial task assignment for crowd sensing with cloaked locations,” in *Proceedings of the IEEE International Conference Mobile Data Management (MDM’14)*, pp. 73–82, Brisbane, Australia, July 2014.
- [29] H. To, G. Ghinita, and C. Shahabi, “A framework for protecting worker location privacy in spatial crowdsourcing,” *Proceedings of the VLDB Endowment*, vol. 7, no. 10, pp. 919–930, 2014.
- [30] L. Zhang, X. Lu, P. Xiong, and T. Zhu, “A differentially private method for reward-based spatial crowdsourcing,” in *Proceedings of the Springer International Conference Applications and Techniques in Information Security (ATIS’14)*, pp. 153–164, Melbourne, Australia, November 2015.
- [31] D. Christin, F. Engelmann, and M. Hollick, “Usable privacy for mobile sensing applications,” in *Proceedings of the International Workshop On Information Security Theory And Practice (WISTP’14)*, pp. 92–107, Heraklion, Greece, 2014.
- [32] I. Krontiris and T. Dimitriou, “Privacy-respecting discovery of data providers in crowd-sensing applications,” in *Proceedings of the IEEE International Conference Distributed Computing in Sensor Systems (DCOSS’13)*, pp. 249–257, Cambridge, MA, USA, May 2013.
- [33] J. Ren, Y. Zhang, K. Zhang, and X. Shen, “Exploiting mobile crowdsourcing for pervasive cloud services: challenges and solutions,” *IEEE Communications Magazine*, vol. 53, no. 3, pp. 98–105, 2015.
- [34] Y. Gong, L. Wei, Y. Guo, C. Zhang, and Y. Fang, “Optimal task recommendation for mobile crowdsourcing with privacy control,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 745–756, 2016.
- [35] Y. Gong, Y. Guo, and Y. Fang, “A privacy-preserving task recommendation framework for mobile crowdsourcing,” in *Proceedings of the IEEE Conference Global Communications Conference (GLOBECOM’14)*, pp. 588–593, Austin, TX, USA, December 2014.
- [36] H. Ma, E. X. Huang, and K.-Y. Lam, “Blockchain-based mechanism for fine-grained authorization in data crowdsourcing,” *Future Generation Computer Systems*, vol. 106, pp. 121–134, 2020.
- [37] C. Lin, D. He, S. Zeadally et al., “SecBCS: a secure and privacy-preserving blockchain-based crowdsourcing system,” *Science China-Information Sciences*, vol. 63, no. 3, 2020.
- [38] H. Wu, L. Wang, and X. Guoliang, “Privacy-aware task allocation and data aggregation in fog-assisted spatial crowdsourcing,” *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 589–602, 2020.
- [39] D. Belli, S. Chessa, B. Kantarci et al., “Toward fog-based mobile crowdsensing systems: state of the art and opportunities,” *IEEE Communications Magazine*, vol. 57, no. 12, pp. 78–83, 2019.
- [40] W. Liu, X. Wang, and W. Peng, “Secure remote multi-factor authentication scheme based on chaotic map zero-knowledge proof for crowdsourcing internet of things,” *IEEE Access*, vol. 8, pp. 8754–8767, 2020.
- [41] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, “Binary neural networks: a survey, pattern recognition,” 2020.
- [42] M. Norouzi, D. J. Fleet, and R. Salakhutdinov, *Hamming Distance Metric Learning: Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA, 2012.
- [43] M. Rastegari, V. Ordonez, J. Redmon et al., “XNOR-net: imagenet classification using binary convolutional neural networks,” in *Proceedings of the European Conference on Computer Vision*, Springer, Amsterdam, Netherlands, 2016.
- [44] Y. Bengio, N. Leonard, and A. Courville, “Estimating or propagating gradients through stochastic neurons for conditional computation,” 2013, <https://arxiv.org/abs/1308.3432>.
- [45] Y. LeCun, “The MNIST database of handwritten digits,” 1998, <https://yann.lecun.com/exdb/mnist/>.
- [46] Y. Lecun, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [47] Hasso Plattner Institute, “Xnor enhanced neural nets,” 2019, <https://github.com/hpi-xnor>.
- [48] Openssl.org, “Openssl-1.0.1e.tar.gz,” 2013 <http://www.openssl.org/source/>.