Data Protection: Privacy-Preserving Data Collection With Validation

Jiahui Hou, Dongxiao Liu, *Member, IEEE*, Cheng Huang, *Member, IEEE*, Weihua Zhuang, *Fellow, IEEE*, Xuemin Shen, *Fellow, IEEE*, Rob Sun, *Member, IEEE*, and Bidi Ying, *Member, IEEE*

***Abstract*—The ubiquitous data collection has raised potential risks of leaking physical and private attribute information associated with individuals in a collected dataset. A data collector who wants to collect data for provisioning its machine learning (ML)-based services requires establishing a privacy-preserving data collection protocol for data owners. In this work, we design, implement, and evaluate a novel privacy-preserving data collection protocol. Specifically, we validate the functionality of the data collection protocol on behalf of data owners. First, the ML-based services are not always predefined, it is challenging for a data collector to combat inference of private attributes and user identity from the collected data while maintaining the utility of data. To address the challenge, we reconstruct the data by designing a data transformation model based on the autoencoder and clustering. Second, it is necessary to ensure that the reconstructed data satisfy certain privacy-preserving properties as untrusted data collectors can provide the data transformation models. Therefore, we utilize detection models and design an efficient enclave-based mechanism to validate that the reconstructed data's private attribute estimation probability is bounded by the predefined thresholds. Extensive experiments demonstrate our protocol's effectiveness, such as significantly reducing the accuracy of private attribute detection.**

***Index Terms*—Blockchain, data privacy, edge computing, machine learning.**

I. Introduction

THE increasing availability of connected smart devices (e.g., smartphones and notebooks) has made data collection and data sharing an inseparable and essential part of our daily life. Many of mobile applications (apps) collect data and transfer them to application corporations, referred to as *data collectors*. These mobile application services can benefit from the collected data by feeding them into machine learning (ML) models. For example, based on advanced ML techniques, the collected acceleration and angular velocity data can be used to improve health monitoring [1], [2] and human activities [3]. We consider that a data collector collects data (involving images or time series data). Unfortunately, as data are continuously collected and transmitted from the end user devices to the data collectors, there exist privacy concerns, e.g., leakage of sensitive information and user identity information from the collected data. This is mainly because the collected data contain specific patterns which can be utilized to re-identify and disclose private attributes associated with individuals, such as gender, age and moods, using ML techniques [4], [5], [6].

Traditionally, there are plenty of schemes for privacy-preserving data collection, such as data anonymization [7], even for incremental dataset protection [8], [9]. The differential privacy (DP) and its variants provide a privacy guarantee by adding noise to statistical query responses drawn from a population-scale dataset, such that the presence or absence of an individual in the dataset can be preserved [10], [11]. The statistical utility of data is generally reduced while addressing privacy concerns. Existing works are mainly based on information theory or DP to balance privacy and utility [12], [13], [14]. This work supposes the collected data is used to train an ML task. Adding noise to collected data can result in a significant loss of accuracy for the specific ML task, leading to a lower data utility [15], [16]. In addition, differential privacy methods are not always suitable for protecting data privacy if an adversary can directly analyze the raw data using ML schemes instead of statistical responses to queries. In this work, we assume the collected data will be used to feed an ML-based service. The *utility* of collected data is defined as its ML task accuracy. We focus on data privacy before data collection, aiming at mitigating the risk of inferring (hidden) private attributes from the sanitized data. Inference attacks such as model inversion attacks or membership inference attacks [17], [18] are out of our scope.

To balance utility and privacy: Recently, autoencoder, a type of artificial neuron network, has been applied to achieve data sanitization, transforming the collected data into a new format that is free of user-sensitive information [19]. A simple autoencoder architecture with five fully connected layers is shown in Fig. 1. Mainly, an autoencoder consists of encoder and decoder sub-models, where the former compresses the input into a latent-space representation while the latter reconstructs the output from the latent representation. Recent works utilize an autoencoder to erase sensitive information of the data [20], [21], [22], [23]. Specifically, they reduce the probability of re-identifying user identity and users' private attributes from the collected data,

Manuscript received 22 June 2022; revised 5 October 2023; accepted 8 October 2023. Date of publication 6 November 2023; date of current version 11 July 2024. This work was supported by research grants from Huawei Technologies Canada and from the Natural Sciences and Engineering Research Council (NSERC) of Canada. *(Corresponding author: Dongxiao Liu.)*

Jiahui Hou is with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230026, China (e-mail: jhhou@ustc.edu.cn).

Dongxiao Liu, Cheng Huang, Weihua Zhuang, and Xuemin Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: dongxiao.liu@uwaterloo.ca; cheng.huang@uwaterloo.ca; wzhuang@uwaterloo.ca; sshen@uwaterloo.ca).

Rob Sun and Bidi Ying are with the Huawei Technologies Canada, Ottawa, ON L3R 5A4, Canada (e-mail: Rob.Sun@huawei.com; bidi.ying@huawei.com).

Digital Object Identifier 10.1109/TDSC.2023.3326299

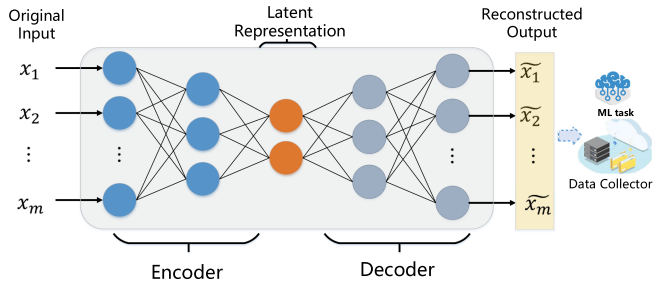


Fig. 1. An example of autoencoder with five fully connected layers. The autoencoder (i.e., a transformation model) that transforms original data from users into a new format before the data are shared with an untrusted data collector.

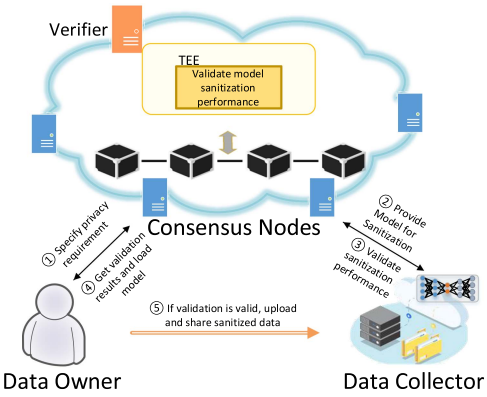


Fig. 2. The architecture of our system. ① The data owner specifies the privacy requirements. ② The data collector generates a data transformation model. The blockchain stores the hash of the transformation model and encrypted data for validation. ③ Data collectors and a verifier utilize TEE and blockchain to achieve sanitization validation. ④ Data owner checks the validity of the result and the transformation model. ⑤ Data owner runs the model on source data ($\mathbf{x}$) and shares sanitized data $\tilde{\mathbf{x}}$ with the data collector.

i.e., *data sanitization*. In this case, an adversary can hardly infer sensitive information from the reconstructed data using ML techniques. The reconstructed data (i.e., sanitized data) can be used to enable a service-specific ML inference. Our sanitization design is the same as the work in [23], which adopts Conditional Variational Autoencoder (CVAE [24]) and introduces the private attributes associated with individuals as conditions. However, in the work of [23], their protocol design does not consider the data utility but improves it by using different hyperparameters. Most of the existing works balance data utility and data privacy based on a predefined ML mechanism. This assumption is strong as data collectors can use different ML mechanisms to improve their services. In this work, we aim to reduce the risk of re-identifying user identity and user's private attributes and preserve the utility *without* a predefined ML model designed for the task. That is, our protocol design can preserve data utility even if the ML mechanism is not predefined. Federated learning [25] is another technique for data privacy, mainly if data are used to train ML models. However, data collectors will be the roles who want to train an ML model to improve their services rather than data owners in our work. That is, we should sanitize data before sharing to data collectors.

Sanitization with validation: Existing on-premise data sanitization techniques rely on performing analysis at the client end, which can be time and resource-consuming for data owners. For example, data owners with weak storage and computation resources can hardly afford to collect data and train an autoencoder-based sanitization model. To support frequent and convenient data collection among multiple data collectors with computing and storage infrastructure, data collectors equipped with expert ML skills should design and train the autoencoder models for achieving data sanitization. Accordingly, data owners will use the well-trained autoencoder model provided by the data collectors to transform their raw data into a new format, referred to as reconstructed data. In this work, we consider a system model as shown in Fig. 2, where the data collector will create and provide an autoencoder-based sanitization model (i.e., data transformation model) to data owners. Here comes another concern: getting rid of private attributes may erase some hidden information associated with the specific task, which can reduce the utility of reconstructed data. As a result, the data collectors have incentives to provide a useless autoencoder model that preserves the utility but can hardly reduce the risk of identification or private attribute inference on the reconstructed data. Hence, validating the privacy-preserving property of a data sanitization method is necessary. In this work, except for designing an autoencoder model for privacy protection, we require validating the sanitization performance of the autoencoder model provided by the data collector. Specifically, given a well-trained ML model for user identity or user's private attribute inference (referred to as detection model), we should validate that the inference accuracy on the reconstructed data is lower than the threshold specified in advance, assuring that the privacy-preserving property of the autoencoder model claimed by the untrusted data collectors is satisfied.

One naïve idea to validate the sanitization performance of the autoencoder model is to share the autoencoder model, detection model, and test data as inputs to a third party that works as a verifier, let the verifier reconstruct the test data using the autoencoder model, and generate the results by running the detection model on the reconstructed test data. Based on the computation results, if the privacy degradation and inference accuracy are bounded, the sanitization performance of the autoencoder model is satisfied from the viewpoint of the verifier. However, such validation result is not convincing from the viewpoint of data owners if the verifier is untrustworthy and may collude with the data collectors. Existing works emphasize the advantages of blockchain to support transparent data sharing that complies with the European General Data Protection (GDPR) requirements [26], [27], [28]. To efficiently validate the sanitization performance without a trusted third party, we adopt the blockchain as a transparent shared ledger among data collectors, data owners, and a verifier, and design a validation mechanism. Recent works have shown that smart contracts can be used to verify ML computation [29]. However, the cost is high, and multiple verifiers are required. We consider that a verifier (such as a computation node on the blockchain) may be interested in the sensitive information used for a validation, and collude with the data collector to pass the validation. Hence we adopt a trusted execution environment (TEE) to provide a secure enclave for validating the sanitization performance of the transformation model from an untrusted verifier.

This work considers a real-life scenario where data owners can hardly train a transformation model. In contrast, data collectors should provide a data transformation model to balance utility and privacy. Hence, we propose a privacy-preserving data collection protocol with sanitization performance validation. We adopt a secure enclave to achieve validation. A careful design is necessary due to the memory limit of the secure enclave, where frequent memory paging is required if we offload an entire detection model to a secure enclave for validation.

Our main contributions in this study are as follows.

- *Data privacy protection:* We propose a data transformation model to prevent re-identifying the user identity and private attributes *without* a predefined ML task for which the data is collected. We concern the utility by using the idea of clustering. The utility of the reconstructed data is thus preserved, which is comparable to the original one as shown in our experiment results.
- *Efficient validation:* We develop a novel and efficient secure enclave-based mechanism to validate the sanitization performance of a data transformation model provided by a data collector. We improve the performance by designing a modified detection model and offloading the correction operations and final validation within the secure enclave, while assigning most validation computations to the data collector.
- *Evaluation:* We implement and evaluate our methodology on real-world datasets, including time-series data (MotionSense [20]) and images (RAF-DB [30]). The task accuracy on sanitized images is degraded by less than 4% while it is increased in some cases.

The rest of this paper is organized as follows. We first review the related work in Section II. We then present the system model, system architecture, and problem formulation in Section III. Section IV further introduces a detailed design of a data transformation model to achieve data sanitization. Section V discusses the sanitization performance validation based on the idea of modifying a detection model. We describe the implementation and performance evaluation of the proposed solutions in Section VI. The conclusion and future work are discussed in Section VII.

II. Related Works

Individuals are willing to keep their data private, while data collectors want to feed the collected data to an AI service. In the following, we present data protection schemes from an outsourcing perspective, including perturbation and data transformation, and discuss federated learning-based techniques.

Data obfuscation and perturbation: Existing works protect the exact data values by adding noise to data [31], [32]. The major challenge is to maintain the quality of the data. That is, the performance of an ML model with obfuscated data should approximate the non-private one. Sang et al. proposed that the values of data can be hidden with random projection and a non-linear function [33]. They have some assumptions about the source data, such as whether attributes are mutually independent, and the information such as means and covariance matrix is public. In addition, an ML model trained on private datasets is likely to expose private information. To prevent adversaries from learning some information about private datasets, adding noise to source data such as using differential privacy (DP) is another option [10]. The DP is widely used in protecting data privacy, with which we can share a dataset with added noise while preserving information of individuals in the dataset [34], [34], [35], [35]. In these works, the main concern is how to preserve the data utility with noise. However, since the task model is unknown in advance, we can hardly quantify the noise added to data, which may result in poor data utility.

Transformation-based method for data privacy: Recently, it is found that sensitive attributes of data (e.g., gender and race) or user-identifiable features can be removed with a one-time data transformation using an autoencoder [20], [23], [36]. In this way, adversaries can hardly estimate the private sensitive attributes from the reconstructed data. Similar to perturbation, data transformation can result in poor utility, with which the accuracy loss of the reconstructed data can be significant. Hence, we need to navigate the tradeoff between data privacy and data utility. For example, Alireza et al. proposed a data anonymization scheme by training deep autoencoders in an adversarial manner to protect data privacy [37]. Based on an autoencoder or a conditional variational autoencoder, we can design a unified sensitive information protection approach that collectively hides multiple attributes and preserves the content information of input data [23], [36]. Osia et al. showed that uploading less sensitive intermediate representations (IRs) rather than original training data is another option to protect data [22]. The sensitive information of the extracted feature is unpredictable since data samples with the same class are represented within a local neighborhood, while the sensitive class borders are disposed of by utilizing an intricate combination of Siamese fine-tuning, dimensionality reduction, and noise addition mechanisms. These works' core challenge is how to define a loss function that regulates the reconstructed data to disregard information about sensitive attributes or user identity and subjects to a minimal distortion to satisfy utility constraints. In our work, we design our data transformation model by utilizing a conditional variational autoencoder but propose a novel scheme without having user identity information in advance.

Data privacy with federated learning: The state-of-the-art federated learning (FL) techniques are widely used to collaboratively train an ML model based on private data [25], [38]. Data are protected as they are kept locally. FL-based techniques study a better trade-off between privacy and communication/computation costs. However, the FL framework requires all participants to train the ML models, which implies that data owners have enough computation and storage resources. In addition, the ML model should be predefined, which is different from our setting. In this work, since we do not make any assumption on the way data collectors use the collected data (e.g., the type of ML techniques) and the ability of data owners, federated learning techniques are not satisfied.

Verifiable ML models: In a machine learning as a service system, machine learning models are shared with users on a pay-as-you-go basis or subscription. The model accuracy claimed by the model owner may be one of the important criteria that users choose to use this ML service. Thus, the verifiability

of computation results is critical to the users. Zhang et al. proved that the estimation results of a model can be turned into statements of zero-knowledge proofs [39]. The model owner can share the estimation results and the corresponding zero-knowledge proofs with the users and make others verify the correctness of the results. Recent works verify the ML computation results based on the blockchain [40], [41]. Zhao et al. achieved on-chain verification of inference results using zero-knowledge succinct arguments of knowledge [41]. These works theoretically prove the range of the model's estimation output, given the range of inputs. However, these approaches usually lack efficiency due to the nonlinearity of ML models. Another idea to verify the ML computation is to design a smart contract for complex computation and make the consensus on multiple verifier nodes [42], [43]. The basic idea is to offload computation to multiple off-chain nodes that work as verifiers, and each of them verifies the results independently. The verifiers disprove the claim and arbitrate the single computation step by using the contract arbitrating, such as in [44]. Different from the existing methods, given detection models, we propose a new idea to validate the sanitization performance of a transformation model based on a key observation.

III. System Model and Problem Definition

A. System Model and Threat Model

System model: There are mainly four parties in our system: *data owners*, *data collectors*, *a verifier* with a secure enclave, and *blockchain*.

Data owners, also referred to as users, generate or own data and are willing to share their data. The data owners are willing to share their data with application corporations or other organizations for multiple services, such as in the health-care sector. However, they are concerned about the privacy of their shared data since private information or user identity information associated with the data may be inferred with ML techniques. Since the data source devices are likely to have weak computation ability, they can hardly sanitize their data for every service, especially when the number of services is large. In our work, data owners are willing to share data but do not have the incentive to train an ML model with other data owners.

Data collectors, usually referred to as application providers, collect source data from data owners. The data collectors are assumed to be technology corporations or organizations that want to collect a large scale of data to improve their services. Data collectors will train ML models to improve their services based on the collected data. The ML-based task is explicit (e.g., improving emotion detection ability), while the ML techniques are not specific. For example, data collectors can use different convolutional neural network (CNN) models, including VGG and Inception networks, *etc* Data collectors are concerned about the utility of collected data. Data collectors are equipped with powerful computation and storage capabilities to train ML models with the collected data for better services. To meet the data privacy requirements of data owners and satisfy the privacy regulations such as the GDPR, each data collector is responsible for creating a data sanitization model to sanitize the collected data, reducing their private attribute inference accuracy.

The *verifier* is a node on the Blockchain, which equips with a secure enclave and can offload encrypted data from the Blockchain within the enclave to validate that a data transformation model satisfies the sanitization performance. The verifier is an enclave-enabled machine where the estimation outputs from the secure enclave will be securely sent to data owners.

Blockchain, referred to as a shared ledger, is composed of many blocks where a new block is added in a chronological order [45]. Blockchain can ensure traceability and tamper resistance of transactions, which is adopted in data management [46]. Secret data for validation, such as test data, true labels, and noise added to a modified detection model, are encrypted and stored on the Blockchain. Interaction among data owners, data collectors and a verifier is through the Blockchain.

Threat model: Data owners are assumed to be honest and are willing to share data with data collectors after data are sanitized with a data transformation model provided and validated by the data collectors. *Data collectors* have the incentive to create a data transformation model with which the data leakage cannot be guaranteed, but data utility is preserved. This is because there can be a positive relationship between sensitive information and information associated with the data utility. Getting rid of more sensitive information from the data can provide better data privacy, while less information related to data utility can be preserved. We utilize a detection model to estimate the ability of the given transformation model on data protection. The detection model used for validation should be private to the data collector. The *verifier* is assumed to be *semi-honest* and will strictly follow the validation specification, but try to learn about the detection models. If a detection model is compromised, it is possible that the data collector will collude with the verifier to pass the validation. Hence, we perturb the detection model with random noises. The only trusted entity on the verifier's platform is the *secure enclave*, which provides secure computation on an untrusted platform. We use the Intel Software Guard Extensions (SGX) for its wide use [47], [48], [49]. The data and code (i.e., noise added to detection models and true labels) are loaded from the blockchain and decrypted within the secure enclave via a secure and authenticated channel. A processor reserved memory (PRM) in the SGX enclave is set aside as a protected memory region, preventing access from outside peripherals. However, the SGX has PRM limit and requires paging and encrypting secret data outside the enclave when running a program that needs more than 90 MB memory [50]. As an ML model size may exceed 90 MB, similar to the work in [51], we require offloading as few ML computations as possible within the enclave to improve the validation efficiency. The *blockchain* as a shared ledger is assumed to be trusted.

B. System Architecture

Our system mainly consists of two execution phases as shown in Fig. 2: generation of data transformation model and validation of model's sanitization performance. We list all notations in Table I. Below we detail these two phases.

Data Transformation Model Generation: Data owners interact with data collectors via the blockchain. A data owner first provides the data privacy requirement, i.e., private attributes (need protecting), privacy thresholds, and labels for target services whose ML techniques are not determined (step ① in Fig. 2). The privacy thresholds specify the data privacy level that data owners want to achieve. Given the data privacy requirements, task goal, and corresponding label information, a data collector creates a data transformation model (step ② in Fig. 2). In this phase, the data collector provides the designed data transformation model to a verifier, and uploads accuracy degradation thresholds and hash of model parameters on the blockchain. The accuracy degradation thresholds are the data protection level claimed by the data collector. Note that the data collector should generate a data transformation model that balances data utility and privacy even without a predefined task model.

Sanitization Performance Validation: A verifier (a node on the chain), together with the data collector, verifies the sanitization performance of the data transformation model (step ③ in Fig. 2). The data collector first downloads test data from the blockchain and uses the transformation model to sanitize the test data. Then, the data collector downloads modified detection models and runs the modified detection models on the sanitized test data. The modified detection model, which is perturbed with random noise, is stored on the verifier's side but provided by a trusted third party, such as a non-profit organization. The added noise is encrypted and stored on the blockchain. To prevent data collectors from learning the modified detection model performance, for each data collector, the noise added to the detection model should be different. The verifier equipped with a secure enclave offloads encrypted noise and true label and receive intermediate results from the data collectors to finally validate the results within the secure enclave. Since the detection model is modified with random noise, a verifier resorts to its secure enclave to correct the results returned by the modified model using the encrypted data from the blockchain. Finally, the verifier checks whether the validation results satisfy the predefined thresholds and sends the results to data owners via blockchain (step ④ in Fig. 2). The data owners receive the data transformation model from the data collector and validate it with hash values downloaded from the blockchain. If all of the validation results are valid, the data owners will sanitize their data with the data transformation model and share sanitized data with the data collector (step ⑤ in Fig. 2).

C. Transformation Model Based on Conditional Variational Autoencoder

Machine Learning Classifier: A K-class machine learning classifier is composed of multiple nonlinear layers, mapping a data sample, $\mathbf{x} \in \mathbb{R}^m$, to a probability vector of length K and generating a predicted label of the data sample. Simply put, the model is formulated as a mapping function with a set of model parameters: $f_n(f_{n-1}(\cdots, f_2(f_1(\mathbf{x}, W_1), W_2) \cdots, W_L)$ and $\mathbf{W} = \{W_1, W_2, \ldots, W_L\}$. Assume the model has L layers, f_l, where $l \in [1, L]$, maps the input of the l-th layer to the output. For hidden layers ($l = 2, \ldots, L-1$), any inputs/outputs of f_l are called intermediate results (IRs). Each connection between the layers of the model has a set of parameters (i.e., weight matrix, W_l).

Task Model and Data Utility. Let us consider a data sample with task label, i.e., $\{(\mathbf{x}, \mathbf{y_T})\}$. The number of class labels is K, $\mathbf{y_T}$ is the true label vector of $\mathbf{x} \in \mathbb{R}^m$, where $\mathbf{y_T} \in \{0, 1\}^K$ and its kth element is 1 if the data sample belongs to the class k. A task model is a K-class classifier, i.e., $\mathcal{F}_T$, defined as an approximation function

$$\mathcal{F}_T : \mathbf{y_T} = \arg\max_k \{\mathcal{F}_T(\mathbf{x})^{(k)}\}_{k=1}^K \tag{1}$$

where $\mathcal{F}_T(\mathbf{x})^{(k)}$ generates the probability of $\mathbf{x}$ belonging to class k, and $\sum_{k=1}^K \mathcal{F}_T(\mathbf{x})^{(k)} = 1$. Our sanitization protocol design concerns utility but does not rely on $\mathcal{F}_T$.

Given an N-element dataset with labels, i.e., $\mathbf{D} = \{(\mathbf{x^{(i)}}, \mathbf{y_T}^{(i)})\}_{i=1}^N$, the accuracy of $\mathcal{F}_T$ is defined as

$$\mathcal{U}(\mathcal{F}_T(\mathbf{D})) = \frac{1}{N} \sum_i^N \mathbb{1}_{\mathcal{F}_T(\mathbf{x^{(i)}}) = \mathbf{y_T}^{(i)}} \tag{2}$$

where $\mathbb{1}_E$ is an indicator function which outputs 1 if E is true, and 0 otherwise. Given a dataset with labels, $\mathbf{D}$, the accuracy of a task model with $\mathbf{D}$ is defined as its *utility*.

Transformation Model. Given collected data, $\mathbf{x} \in \mathbb{R}^m$, a transformation model, $\mathcal{F}$, is defined as

$$\mathcal{F} : \mathbf{x} \to \tilde{\mathbf{x}} \tag{3}$$

where $\tilde{\mathbf{x}} \in \mathbb{R}^m$. The reconstructed data is regarded as *sanitized data*.

The transformation model is designed to sanitize the collected data, and we implement it based on a conditional variational autoencoder (CVAE) used in the work of [23]. This is because using a CVAE-based transformation model can maximally preserve the input data's information content while controlling the latent representation $\mathbf{z}$ and manipulating the private attribute of reconstructed data. In this work, we additionally condition on task-related variables and carefully design a loss function for better utility.

Let θ and ϕ represent the encoder and decoder parameters, respectively. The main idea is to force the distribution, which is generated from the encoder outputs, to be close to a standard normal distribution, $\mathcal{N}(0, I)$, where I is an identity matrix. The data transformation model can condition both the encoder and the decoder with random variable $\mathbf{y}$. Condition $\mathbf{y}$ is the private attribute associated with data samples that data owners want to protect. Specifically, the encoder outputs $\mu_{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$ given $\mathbf{x}$ as input. Latent representation, $\mathbf{z}$ is sampled from a multivariate Gaussian distribution, denoted as $q_\theta(\mathbf{z}|\mathbf{x})$, whose mean is $\mu_{\mathbf{x}}$ and variance is $\sigma_{\mathbf{x}}$. The decoder approximates the encoder input, $\mathbf{x}$, given $\mathbf{z}$ as input.

We design the transformation model without giving a predefined ML model for services. That is, the services (i.e., tasks) are specific, while data collectors can use any ML techniques to improve their services. Given transformation model $\mathcal{F}$, task model $\mathcal{F}_T$, dataset with labels, $\mathbf{D} = \{(\mathbf{x^{(i)}}, \mathbf{y_T}^{(i)})\}_{i=1}^N$, the data utility is preserved if the original data utility, $\mathcal{U}(\mathcal{F}_T(\mathbf{D}))$, is similar to the reconstructed data utility, $\mathcal{U}(\mathcal{F}_T(\tilde{\mathbf{D}}))$, where $\tilde{\mathbf{D}} = \{(\mathcal{F}(\mathbf{x^{(i)}}), \mathbf{y_T}^{(i)})\}_{i=1}^N$.

D. Validation Based on Detection Models

Detection Model: Detection models are ML classifiers to infer private attributes or user identity from data. Let $A_1, A_2, \ldots, A_s$ denote private attributes that data owners want to protect, and let I represent the user identity. Accordingly, we have $|s|$ detection models for private attribute inference and one detection model for user identity inference, i.e., $\{\mathcal{F}_{D_j}\}, j \in \{A_1, \ldots, A_s, I\}$. Let the true labels of the private attributes for each sample (i.e., $\mathbf{x}$) be $\{\mathbf{y_{A_1}}, \ldots, \mathbf{y_{A_s}}\}$, and the identity label of $\mathbf{x}$ be $\mathbf{y_I}$. The detection model is defined as

$$\mathcal{F}_{D_j} : \mathbf{y_j} = \arg\max_k \{\mathcal{F}_{D_j}(\mathbf{x})^{(k)}\}_{k=1}^{K} \tag{4}$$

where K can be the number of class labels of a private attribute, e.g., $|A_s|$, or the number of identity label, $|I|$, which is dependent on the type of $\mathcal{F}_{D_j}$.

We consider that some trustful examining organizations or governments will design the detection model.

Measurement of Data Privacy Protection: We aim to measure to what extent data privacy can be protected with the transformation model. The measurement of data privacy protection is also the criteria to validate that the privacy property of a transformation model is guaranteed. To this end, we quantify the accuracy degradation of detection models on the reconstructed data generated by the transformation model to measure its sanitization performance.

Let N test samples be $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$. Given data transformation model $\mathcal{F}$, and true label $\mathbf{y_j}$ of a data sample, we then have two datasets with labels: $\mathbf{D} = \{(\mathbf{x^{(i)}}, \mathbf{y_j}^{(i)})\}_{i=1}^{N}$ and $\tilde{\mathbf{D}} = \{(\mathcal{F}(\mathbf{x^{(i)}}), \mathbf{y_j}^{(i)})\}_{i=1}^{N}$. Given specific detection model, $\mathcal{F}_{D_j}$ for $j \in \{A_1, \ldots, A_s, I\}$, we define a metric for evaluating the degree of privacy property guaranteed by the data transformation model.

Let accuracy degradation denote as the fraction of test samples that are correctly classified by the detection model while their sanitized versions are misclassified, given by

$$\mathcal{V}_D(\mathcal{F}, \mathcal{F}_{D_j}) = \mathcal{U}(\mathcal{F}_{D_j}(\mathbf{D})) - \mathcal{U}(\mathcal{F}_{D_j}(\tilde{\mathbf{D}})) \tag{5}$$

Given a private attribute detection model, a higher value indicates that the data transformation model can achieve a higher degree of privacy. In the evaluation part, given an ML task model, we also use accuracy degradation to estimate data utility. Differently, a low accuracy degradation implies a high data utility preservation.

Let attack accuracy denote as the detection model accuracy of inferring private attributes from sanitized data, given by

$$\mathcal{V}_A(\mathcal{F}, \mathcal{F}_{D_j}) = \mathcal{U}(\mathcal{F}_{D_j}(\tilde{\mathbf{D}})) \tag{6}$$

The lower value of attack accuracy indicates that the detection model is less capable of inferring the private attributes from the sanitized data.

Validation of Sanitization Performance: We should validate the sanitization performance of a data transformation model provided by a data collector, with which data owners can be assured that data privacy protection claimed by the data collector is satisfied.

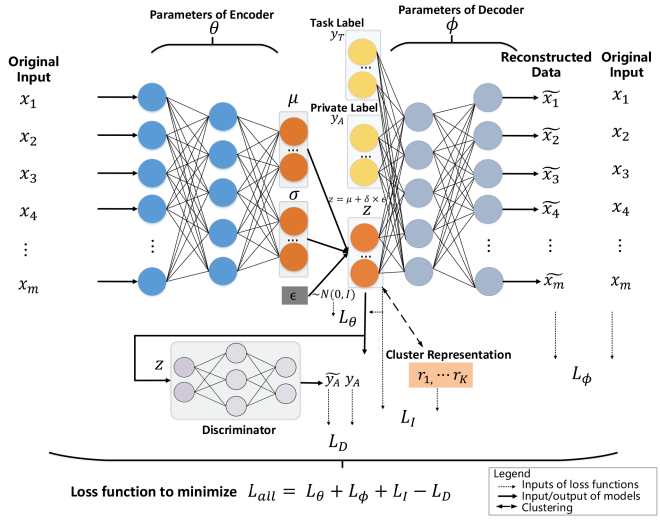


Fig. 3. The architecture of our data transformation model.

Let $\delta_{A_1}, \ldots, \delta_{A_s}, \delta_I$ be accuracy degradation thresholds claimed by a data collector and $\gamma_{A_1}, \ldots, \gamma_{A_s}, \gamma_I$ be thresholds specified by data owners. Given a transformation model, $\mathcal{F}$, $s+1$ detection models, $\{\mathcal{F}_{D_{A_1}}, \ldots, \mathcal{F}_{D_{A_s}}, \mathcal{F}_{D_I}\}$, the data collector and a verifier should validate $\mathcal{F}$ with a given test set $\{(\mathbf{x}, \mathbf{y_{A_1}}, \ldots, \mathbf{y_{A_s}}, \mathbf{y_I})^{(i)}\}_{i=1}^{N}$, with $\mathbf{y_j}$ representing the true label of attributes or user identity for $\mathbf{x}$.

The requirements are given by

$$\mathcal{V}_D(\mathcal{F}, \mathcal{F}_{D_j}) \geq \delta_j \tag{7}$$

$$\mathcal{V}_A(\mathcal{F}, \mathcal{F}_{D_j}) \leq \gamma_j \tag{8}$$

where j represents one specific detection model that estimates either private attribute A_j or user identify I from the data samples, $j \in \{A_1, \ldots, A_s, I\}$. The privacy holds if the above-mentioned requirements (7 and 8) are satisfied.

IV. Privacy-Preserving Data Collection

In this section, we present design details of our data transformation model, which transforms raw data before sharing the data with an untrusted data collector (e.g., an application service provider). To preserve data utility and combat private attribute inference, we design a transformation model based on the idea of [23], adopting a CVAE architecture. Differently, we cluster the latent representation, $\mathbf{z}$, and redesign the loss function for better utility and data privacy (especially for protecting user identity information). In the following, we present the architecture of our data transformation model with a running example and illustrate how we balance privacy and utility by designing loss functions. Given a dataset with labels of private attributes and an associated task, we describe how the data transformation model is trained with the proposed algorithm.

A. Transformation Model Architecture

The architecture of the transformation model is depicted in Fig. 3. For presentation clarity, we assume the number of private

attributes that data owners want to protect is 1, i.e., $s = 1$. For increasing numbers of private attributes, we can repeat the following steps. Let the class label vector of the private attribute be $\mathbf{y_A}$. A discriminator is an ML classifier that generates $\tilde{\mathbf{y_A}}$ as the predicted label vector of a given latent representation $\mathbf{z}$. Each of the discriminator networks is implemented with an ML model and estimates private attribute label given the latent representation, $\mathbf{z}$. The data transformation model consists of an encoder whose trainable parameters are denoted as θ, and a decoder whose trainable parameters are denoted as ϕ.

To address the overfitting concern and make the data transformation model adapt to datasets from new data owners, we adopt a conditional variational autoencoder architecture [23], which generates latent representation from a distribution. We assume there is a posterior distribution (multivariate Gaussian distribution), i.e., $P(\mathbf{z}|\mathbf{x}) \sim \mathcal{N}(0, I)$, given input data $\mathbf{x}$. The encoder generates $\mu_{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$. The latent representation, $\mathbf{z}$, is then sampled from a multivariate Gaussian distribution with mean $\mu_{\mathbf{x}}$ and variance $\sigma_{\mathbf{x}}$. In addition, we condition the decoder on the private attribute and the task attribute such that the decoder input is latent representation $\mathbf{z}$ augmented with $\mathbf{y_T}$ and $\mathbf{y_A}$.

By changing private attributes $\mathbf{y_A}$ and introducing discriminator networks, we combat the private attribute inference attack. We condition the decoder on the task attributes, which aims at maximally passing task-related attributes to the reconstructed data. All attribute information is converted into a numerical label vector. The decoder generates $\tilde{\mathbf{x}}$ (reconstructed data) given $\mathbf{z}$, private attribute label vector $\mathbf{y_A}$, and task attribute label vector $\mathbf{y_T}$.

A discriminator network takes $\mathbf{z}$ as input and generates the estimated private attribute label. The transformation model with a discriminator network aims at ensuring that information of the private attributes can hardly be learned, given latent representation $\mathbf{z}$. If data owners want to protect more private attributes, the decoder will be conditioned on all the specified attributes. Accordingly, more discriminator networks will be added to the transformation model.

Without a predefined task model, existing works lack schemes to combat inference of user identity information from the sanitized data One naïve idea to conceal user identity information is to adopt another discriminator network in the data transformation model, which specifies identity as a sensitive attribute. However, the transformation model is designed by a data collector that does not have data owners' data in advance, without the knowledge of how many individuals are involved and the identity labels of the data. Since the label information of user identity is variant to datasets, e.g., the number of identity classes can be different, where each class represents one specific identity. Without associated label information, the data collector can hardly train a discriminator network to estimate identity from latent representations. Instead, we make transformed data in the same task class be similar with (i.e., close to) each other while separating those in different task classes as far as possible. To this end, we cluster latent representations, making $\mathbf{z}$ with the same task label close to each other and $\mathbf{z}$ with different labels far from each other in the latent space. The identifying information about individuals can be obscured as an adversary can hardly recognize identity labels from similar latent representations. Since latent representations with different task classes are separated from each other, these properties are passed to the reconstructed data by a decoder. In this way, the utility can be preserved as an ML model can easily classify the reconstructed data. Let us consider an activity classification as the task model (what the data are collected for). Suppose data samples from different users but with the same activity label are mapped to different points after using the task model. In that case, an adversary may infer identifying information from the data thereby. Conversely, if all of the samples from different individuals are clustered and mapped to one single point for the same activity, the identifying information will be obscured. Given K-class task model, $\mathbf{r_1}, \ldots, \mathbf{r_K}$ denote the cluster representatives, which represent cluster centroid of latent representations. To improve data utility and hide identity information, we introduce a loss function based on cluster representatives, i.e., maximizing the difference among $\mathbf{r_i}$ and minimizing the difference between latent representations $\mathbf{z}$ and its corresponding cluster representatives $\mathbf{r_i}$.

The data transformation model is trained in an adversarial manner using loss functions. During the training process, we can minimize loss functions that consist of four terms. Specifically, we minimize the distortion between the original data and the reconstructed data, the distance between a learned distribution and the specified multivariate Gaussian distribution, and the distance between latent representations and cluster representatives while maximizing the difference between the estimation of the private attributes from latent representations and their true labels. Accordingly, we define a reconstruction loss, the Kullback–Leibler (KL) divergence loss, a clustering loss, and a discriminator loss in the following.

Reconstruction Loss L_ϕ: A decoder in the data transformation model is designed to generate reconstructed data from original data. Given a latent representation, a private attribute, and a task attribute, the reconstructed data, $\tilde{\mathbf{x}}$, returned by the decoder is expected to be similar as the original data, $\mathbf{x}$. A reconstruction loss is designed to achieve such goal. We denote the reconstruction loss by L_ϕ. To make the reconstructed data close to the original data, L_ϕ is implemented with Mean-Squared Error (MSE), given by

$$L_\phi = ||\mathbf{x} - \tilde{\mathbf{x}}||_2^2 \tag{9}$$

where $\mathbf{x}$ is an input of the encoder while $\tilde{\mathbf{x}}$ is the output of the decoder.

KL-Divergence Loss L_θ: The data transformation model controls latent representations by generating them from a multivariate Gaussian distribution with mean and variance from the encoder. The KL-divergence is adopted to compare the real distribution of $\mathbf{z}$, denoted as $p(\mathbf{z})$, and the generated distribution, $q_\theta(\mathbf{z}|\mathbf{x})$. The KL-divergence loss is defined as

$$L_\theta = D_{KL}[q_\theta(\mathbf{z}|\mathbf{x})|p(\mathbf{z})] \tag{10}$$

where $p(\mathbf{z})$ denotes the real distribution of the latent variables $\mathbf{z}$.

The distribution $P(\mathbf{z}|\mathbf{x})$ is assumed to follow $\mathcal{N}(0, I)$, we then have $p(\mathbf{z}) = \sum_{\mathbf{x}} P(\mathbf{z}|\mathbf{x})P(\mathbf{x}) = \sum_{\mathbf{x}} \mathcal{N}(0, I)P(\mathbf{x}) =$

$\mathcal{N}(0,I)\sum_{\mathbf{x}} p(\mathbf{x}) = \mathcal{N}(0,I)$. The KL-divergence loss is calculated as $-\frac{1}{2}[-\sum_{i=1}^{|z|}\sigma_i^2 - \sum_{i=1}^{|z|}\mu_i^2 + \sum_{i=1}^{|z|}(\log\sigma_i^2 + 1)]$, according to the work in [52], where $\mu_{\mathbf{x}} = \{\mu_i | i = 1, \ldots, |z|\}$, $\sigma_{\mathbf{x}} = \{\sigma_i | i = 1, \ldots, |z|\}$, and $|z|$ is the size of $\mathbf{z}$.

Clustering Loss L_I. To cluster the latent representations with the same task label and make them close to each other, we adopt the idea as [53]. We denote the clustering loss as L_I and minimize L_I by making latent representations close to their cluster representatives while making cluster representatives far away from each other. The number of clusters is the number of task classes, and each cluster representative has the same size as latent representations, i.e., $\mathbf{r_k} \in \mathbb{R}^{|z|}, k = 1, \cdots K$. To constitute a continuous generalization, we use ω to initialize cluster representatives, and the clustering loss is defined as

$$L_I = -\alpha \sum_{i=1}^{K}\sum_{j=1, i\neq j}^{K} \frac{||\mathbf{r_i} - \mathbf{r_j}||_2^2}{K(K-1)} + \sum_{k=1}^{K} \frac{e^{-\omega||\mathbf{z}-\mathbf{r_k}||_2^2} \times ||\mathbf{z} - \mathbf{r_k}||_2^2}{\sum_{i=1}^{K} e^{-\omega||\mathbf{z}-\mathbf{r_i}||_2^2}} \tag{11}$$

where α regulates the trade-off between data utility and user identity privacy. As proved in the work [53], if $\omega \to +\infty$, we have the parameterized softmax coefficient equal to 1. This loss function contributes to improving data utility while hiding identity information as separated reconstructed data with different classes can be correctly classified, but similar reconstructed data with the same class can hardly be identified.

Discriminator Loss L_D: Each discriminator network estimates the specified private attribute label $\mathbf{y_A}$ from a given latent representation $\mathbf{z}$. We denote L_D be discriminator loss. This loss term is maximized during the training process, which indicates the divergence between private attribute and the learned latent representation $\mathbf{z}$ is maximized. In this case, the data transformation model cannot estimate the private attribute correctly from the latent representation. Similarly, the discriminator loss is calculated by MSE, and L_D is given by

$$L_D = ||\mathbf{y_A} - \tilde{\mathbf{y_A}}||_2^2 \tag{12}$$

where $\mathbf{y_A}$ is the true label vector of the private attribute while $\tilde{\mathbf{y_A}}$ is the discriminator output.

Adversarial Loss Function: The total loss function of our model is defined as

$$L_{all}(\theta, \phi|\eta) = -\frac{1}{N}\sum_{i=1}^{N}(\upsilon_1 L_\phi + \upsilon_2 L_\theta + \upsilon_3 L_I - \upsilon_4 L_D) \tag{13}$$

where N is the number of data samples in a dataset. Parameters υ_1, υ_2, υ_3 and υ_4 regulate the trade-off between finding a good transformation for input data and a transformation that suits the discrimination and clustering purpose. Given a pre-trained discriminator model with model parameters, i.e., η, we update transformation model parameters θ and ϕ through minimizing the adversarial loss functions during the training process.

Algorithm 1: Data Transformation Model.

Input: Training data $\mathbf{X}$ with task label $\mathbf{y_T}$ and private attribute $\mathbf{y_A}$, learning rate λ_1, λ_2, clustering factor $\mathbf{\Omega}$, epoch size $\mathbf{E}$, batch size $\mathbf{B}$, trade-off parameters υ_1, υ_2, υ_3, υ_4, cluster number K

Output: Model parameters θ, ϕ and η

```
for ω = 0 to Ω do
  Pretrain the encoder, decoder, and discriminator without clustering loss
  Use latent representations learned from pretraining, initialize r_k with K-means, 1 ≤ k ≤ K
  Initialize θ, φ, η
  for i = 0 to E do
    for b = 0 to B do
      Sample a minibatch: {x_1, ..., x_B}
      ε ⟵ N(0, I)
      μ_x, σ_x ⟵ Encoder(x; θ)
      z ⟵ μ_x + σ_x · ε
      x̃ ⟵ Decoder(z, y_A, y_T; φ)
      ỹ_A ⟵ Discriminator(z; η)
      η ⟵ η − λ1 ▽ L_D (Eq. 12)
      {θ, β} ⟵ {θ, β} − λ2 ▽ L_all(θ, φ|η) (Eq. 13)
    end for
  end for
end for
```

B. Training Process

Algorithm 1 summarizes the main idea of our transformation protocol. We have a training dataset $\mathbf{X}$ with label information $\mathbf{y_A}$ and $\mathbf{y_T}$. The learning rate is denoted as λ_1 and λ_2. We denote Ω as a constant and let $\mathbf{E}$ be the epoch size and $\mathbf{B}$ be the minibatch size. We denote $\upsilon_1, \upsilon_2, \upsilon_3$, and υ_4 as trade-off parameters, which may impact transformation model performance. The number of classes in the desired ML task is the cluster number K. After the training process, we have a data transformation model with well-trained model parameters θ and ϕ.

For each fixed a, we first pre-train the encoder and decoder from scratch without the clustering loss component to speed up the training process. After pretraining, given $\mathbf{z}$ as input, $\mathbf{r_1}, \ldots, \mathbf{r_K}$ are initiated using k-means clustering. The encoder generates μ_x and σ_x. As $\mathbf{z}$ is a random variable sampled from $\mathcal{N}(\mu_{\mathbf{x}}, \sigma_{\mathbf{x}})$, we cannot implement the backpropagation through a random variable. According to the work in [52], instead of sampling $\mathbf{z}$ from $\mathcal{N}(\mu_{\mathbf{x}}, \sigma_{\mathbf{x}})$, we adopt the reparameterization trick to rewrite $\mathbf{z}$ as $\mathbf{z} = \mu_{\mathbf{x}} + \sigma_{\mathbf{x}} \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$. The decoder generates $\mathbf{z}$, given $\mathbf{y_A}$ and $\mathbf{y_T}$ as inputs. The discriminator generates $\mathbf{y_A}$ from latent representation $\mathbf{z}$. Given $\tilde{\mathbf{x}}$ and $\mathbf{x}$, we calculate the L_ϕ using MSE. Given encoder outputs $\mu_{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$, L_θ is calculated. We calculate L_I with $\mathbf{r_1}, \ldots, \mathbf{r_K}$ and L_D with generated $\tilde{\mathbf{y_A}}$ and true label $\mathbf{y_A}$. Model parameters η, θ, ϕ are updated by minimizing the loss function using SGD. The discriminator independently updates its parameter η by minimizing L_D. We then update θ, ϕ by minimizing L_{all} (Equation. 13) during the training process.

Note that a data collector can use its collected data or dataset with labels downloaded from public websites. A trained transformation model is sent to data owners, who will run it locally to sanitize their data.

V. Validation of Data Sanitization

In this section, we present the details of validating the sanitization performance of a data transformation model. Specifically, given a data transformation model, $\mathcal{F}$, a detection model, $\mathcal{F}_{D_j}$, and the corresponding pair of thresholds $\{\delta_j, \gamma_j\}$, for $j \in \{A_1, \ldots, A_s, I\}$, we require validating that $\mathcal{F}$ satisfies the requirements: $\mathcal{V}_D(\mathcal{F}, \mathcal{F}_{D_j}) \geq \delta_j$ and $\mathcal{V}_A(\mathcal{F}, \mathcal{F}_{D_j}) \leq \gamma_j$. In the following, we use $\mathcal{V}_D(\cdot)$ and $\mathcal{V}_A(\cdot)$ for simplicity.

One naïve idea to address this problem is to verify detection model computations on sanitized data based on zero-knowledge succinct arguments of knowledge [41] or contract arbitrating [43]. However, the cost can be high for verifying computations of multiple complicated detection models, and verifiers may be unwilling to undertake heavy computations. Since verifiers are semi-honest, we employ a secure enclave to run detection models on sanitized data and validate the performance with given thresholds. To solve the enclave memory limitation and improve validation efficiency, we offload most validation computations to data collectors and leave few computations to the secure enclave. For presentation clarity, we present validation process with one detection model. When it comes to other private attribute or user identity detection models, we repeat the same validation process but replace the detection model. In the following, we first illustrate our key observation and present why we can offload most validation computations to data collectors based on a modified detection model. Then, we take a modified detection model with three convolutional layers and three fully connected layers as a running example to explain how to achieve validation.

A. Modified Detection Model

Critical Observations: We train different kinds of models to evaluate how a well-trained model will be changed by adding random noise: a neuron network model with 4 fully connected (FC) layers as hidden layers and a VGG model. We select 60% and 100% weights in one hidden layer and modify them with numbers sampled from a normal distribution with mean 0 and standard deviation (SD) 0.1. It is shown in Fig. 5(a) that the accuracy can work as an approximate random guess (i.e., 10% for a 10-class classification), if we modify 100% weights in one hidden layer. As shown in Fig. 5(b), we modify one of the last three hidden convolutional layers in the VGG model, and the accuracy is around 10% if we modify 100% weights in one convolutional layer. Note that our method is different from the standard DP-based methods. We do not add noise to IRs or local gradients during the training process as the work in [54]. Our focus is model inference rather than training. Instead, we aim to protect the detection model by perturbing. We directly perturb the detection models by adding random noise to partial weight parameters. As a result, we combat the utility rather than pursue a decent privacy-utility trade-off by formally guaranteeing DP.

In addition, we check whether sanitized data are affected by the modified detection models. The data are sanitized using our data transformation model. We train two models as detection models for inferring gender information and user identity on mobile sensor data (i.e., MotionSense). The former model is implemented with 7 fully connected layers, while the latter is implemented with 3 convolutional layers and 3 fully connected layers. We generate a modified model by adding noise sampled from a normal distribution (whose mean is 0 and SD is 1) to weights in the next-to-last layer. The results in Fig. 6 show that the modified model performance is affected regardless of giving original data or sanitized data as inputs. The accuracy of the modified model on the original data and the sanitized data are similar, e.g., in the case of gender detection, we have 56.98% and 56.99%, respectively. This is because the modified model cannot have the ability to estimate data correctly but always misclassify all data into the same class.

Insights: These results show that adding noise to model weights can change model intermediate results and predicted results accordingly. Our observation results are the same as the work in [51]. In this way, we thus prevent data collectors from having the original detection models and passing the validation by perturbing the models. To reduce the computation load of verifiers, we offload most inference computations of modified detection models, e.g., computations before the last layer, to a data collector. Afterward, the verifier with a secure enclave corrects IRs and continues to run the rest of the inference computations.

Given a modified model, if a data collector honestly shares the outputs of modified detection models, a verifier can have the actual estimation results after correcting modified IRs with added noise. Otherwise, the verifier cannot correct the estimation results using noise added to model weights beforehand. After the correction process, the estimation results are different from the actual one. In other words, a data collector cannot control the final validation results without knowing added noise. In the following, we show that if data collectors forge IRs, after IRs correction, the model performance works as adding noise to models. The data collectors cannot control the outputs and ensure the forged results can pass the validation.

Modified Model Generation. Let $\mathcal{F}_D$ be the original detection model with L layers, and test dataset be $\mathbf{X}$, we have $\mathcal{F}_D(\mathbf{X}) = Y$. We generate the modified detection model by adding noise, $\mathbf{R}$, from a normal distribution to model weights in the $\{L-1\}$th layer. To avoid a data collector can learn the performance of a modified detection model based on multiple validation processes, the modified detection model is different for each validation process. We define the modified model as $\tilde{\mathcal{F}}_D = \{\tilde{F}_{D_{L-1}}; F_{D_L}\}$. Let the output of the $\{L-2\}$th layer be $\mathbf{X}_{L-1}$, the intermediate results be M', we have $\tilde{F}_{D_{L-1}}(X) = M'$. We denote M as the actual IRs returned by the original detection model (without modification). Given the same data input, IRs before the $\{L-1\}$th layer, e.g., $\mathbf{X}_{L-1}$, are the same as the IRs from the original detection model if we only modify model weights in the $\{L-1\}$th layer. A detection model is partitioned as two components, computations before the $\{L\}$th layer, i.e., $\tilde{F}_{D_{L-1}}$ are offloaded to a data collector. The rest of the computations, involving computations of an activation function and the last fully connected layer, are executed by the verifier within the secure enclave. The verifier corrects M' with $\mathbf{R}$ and

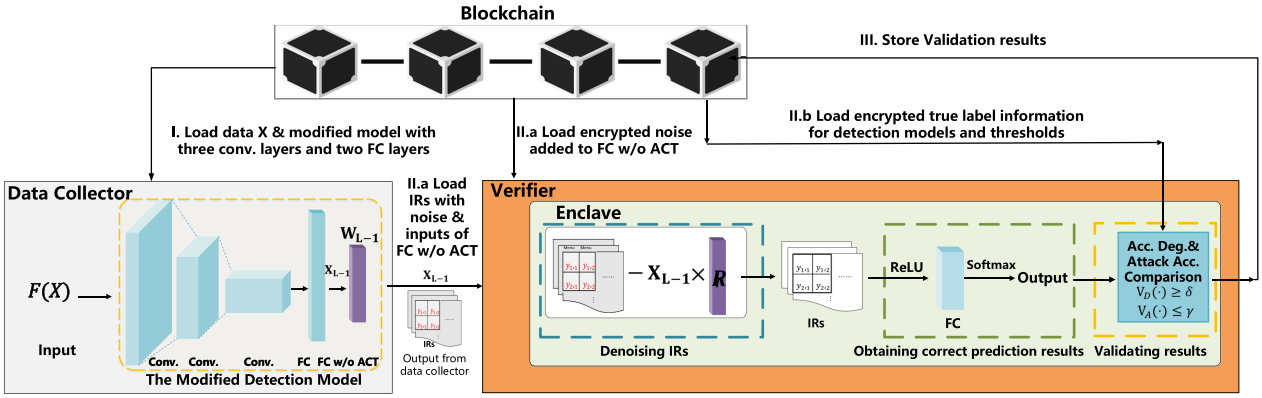
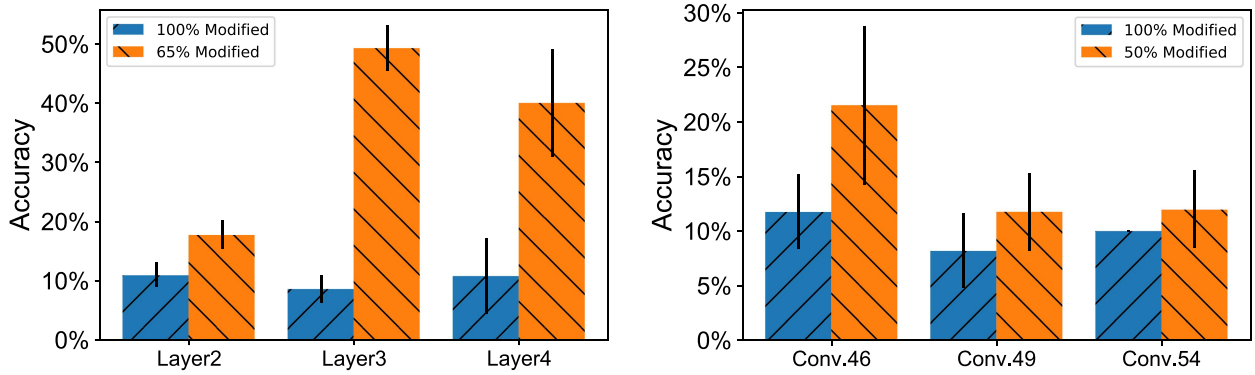


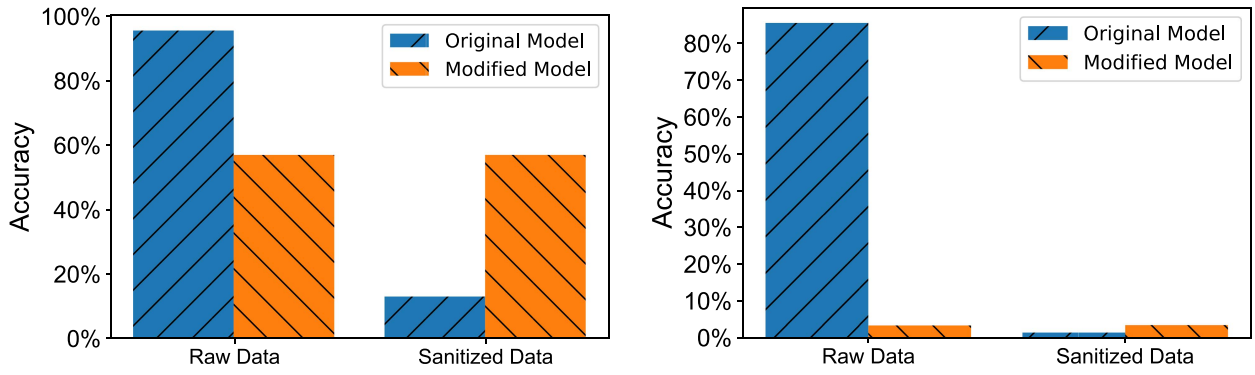
Fig. 4. Workflow of validating the sanitization performance of a given transformation model. I) Downloading a modified detection model modified with random values, $\mathbf{R}$, in the $\{L-1\}$th layer and running the modified model. II) Denoising IRs using $\mathbf{R}$, and input of the $\{L-1\}$th layer, $\mathbf{X}_{L-1}$, (II.a). Computing $\mathcal{V}_D(\cdot)$ and $\mathcal{V}_A(\cdot)$ and validating the results with thresholds (II.b). Storing the results on the blockchain (III).



(a) Accuracy degradation of GNN model.

(b) Accuracy degradation of CNN model.

Fig. 5. Effect of model modification on different model architectures.



(a) Acc. of detection model to infer gender.

(b) Acc. of detection model to infer user identity.

Fig. 6. Effect of model modification on sanitized data.

$\mathbf{X}_{L-1}$, and has M within the secure enclave. After correction, the output of $F_{D_L}(M)$ is Y. We present the details in the following.

B. Data Sanitization Validation Process

As shown in Fig. 4, we present the validation process and use a detection model with 3 convolutional layers and 3 fully connected layer as a running example, i.e., $L = 6$. In summary, we offload most computations to a data collector but correct the IRs and get validation outputs within the enclave. The validation process is mainly composed of two components. I) A modified model without activation function (e.g., ReLU) is offloaded to the data collector. The modified model is with added secret values, $\mathbf{R}$, in the $\{L-1\}$th fully connected layer. The data collector runs the modified model locally. II) The verifier first denoises IRs using encrypted data $\mathbf{R}$ and input $\mathbf{X}_{L-1}$. The input

data and IRs are loaded from the data collector, while encrypted data is from the blockchain (II.a). Given true labels of test data, the verifier computes $\mathcal{V}_D(\cdot)$ and $\mathcal{V}_A(\cdot)$ and validates the result with the thresholds (II.b). Finally, the validation results are stored in the blockchain.

Offloading Computations of Modified Detection Model (Phase I): A data collector first downloads the test data, $\mathbf{X}$, and a modified detection model, $\tilde{\mathbf{F}}_D$, from the blockchain. The noise $\mathbf{R}$ added to the model is regarded as secret data and stored in the blockchain beforehand. Specifically, W'_{L-1} denotes the modified weights. The activation function connected to the $\{L-1\}$th fully connected layer is calculated by the verifier after IRs correction. The data collector sanitizes the test data using its designed transformation model $\mathcal{F}$. The output is the input of the modified detection model, i.e., $\mathcal{F}(X)$. The data collector then runs the detection model with input $\mathcal{F}(X)$. Let the output of $\tilde{F}_{D_{L-1}}(X)$ be M', and $\mathbf{X}_{L-1} \times \mathbf{W}'_{\mathbf{L-1}} = M'$. The output M' and the input of the $\{L-1\}$th layer, i.e., $\mathbf{X}_l$, are sent to the verifier afterward.

IRs Correction (Phase II.a): In this phase, one off-chain verifier downloads IRs $\mathbf{X}_{L-1}$, M' and encrypted noise $\mathbf{R}$ (added to the weights in the fully connected layer) from the blockchain. The verifier utilizes the secure enclave to execute the matrix multiplication between $\mathbf{R}$ and $\mathbf{X}_{L-1}$ within the enclave. As the weights are modified with $\mathbf{R}$, we load the results of $\mathbf{X}_{L-1} \cdot (\mathbf{W}_{L-1} + \mathbf{R})$, simplified as M', and correct M' within the enclave. We decrypt $\mathbf{R}$ that are privately stored on the blockchain and compute $M' - \mathbf{X}_{L-1} \cdot \mathbf{R}$ within the enclave. Let the result be M. This process is proceeded to obtain the "denoised output", i.e., removing the effect of the added noise. The rest computations (involving a ReLU activation function and a fully connected layer) are executed within the enclave. Specifically, the verifier runs the activation function within the enclave, followed by the fully connected layer and a softmax function. The softmax function generates the probability for each class (e.g., private attribute class), and we can label the test data with the label class whose probability is the highest. After correction, the estimated label of the modified model is the same as the one generated from the original detection model.

Calculating Validation Results (Phase II.b): After correcting the estimation results of the modified detection model, the verifier decrypts the true labels downloaded from the blockchain and calculates $\mathcal{V}_A(\cdot)$ based on Equation 5. To calculate $\mathcal{V}_D(\cdot)$, we require the detection model accuracy on both the original test data and the reconstructed data. To this end, the data collector and the verifier repeat Phase I and Phase II, but the data collector takes the original test data as inputs in Phase I. We then compare $\mathcal{V}_D(\cdot)$ and $\mathcal{V}_A(\cdot)$ with the pair of predefined thresholds, i.e., $\{\delta, \gamma\}$, within the secure enclave.

Outputs (Phase III): If the requirement is satisfied, the result is set to be valid. The updated state is stored on the blockchain. The data owners receive the validation results. If the result is valid, data owners will download the data transformation model from the data collector and its hash value from the blockchain. The data owners calculate the hash of weights in the transformation model and compare them with the hash values from the blockchain. If the result is matched, the data owners will sanitize their data before sharing them with the data collector.

Validation Analysis: We analyze that a forged IRs uploaded by the data collector can have the same performance as adding noise to IRs in the detection model. Based on our previous observation, if the IRs of the last few layers are added with noise, the estimation results will be different.

Given test data, $\mathbf{X}$, a modified model, $\tilde{\mathcal{F}}_D = \{\tilde{F}_{D_{L-1}}; F_{D_L}\}$, a detection model, $\mathcal{F}_D$, and a transformation model, $\mathcal{F}$, if the data collector forges the intermediate results and upload $\hat{M}$ instead of $\tilde{F}_{D_{L-1}}(\mathbf{X}) = M'$, we will have $\mathcal{F}_D(\mathcal{F}(\mathbf{X})) \neq \tilde{\mathcal{F}}_D(\hat{M})$.

Given input $\mathbf{X}_{L-1}$, added noise $\mathbf{R}$, we have $M = M' - \mathbf{X}_{L-1} \cdot \mathbf{R}$. If the data collector wants to upload a forged $\hat{M}$ to pass the validation, we have $\hat{M} - \mathbf{X}_{L-1} \cdot \mathbf{R}$ instead of M, which is the input of F_{D_L}. We can rewrite $(\hat{M} - \mathbf{X}_{L-1} \cdot \mathbf{R})$ as $M + \mathbf{R}_L$ where $\mathbf{R}_L = \hat{M} - \mathbf{X}_{L-1} \cdot \mathbf{R} - M$. As adding noise to weights in one hidden layer results in the outputs (i.e., IRs) are added with noise. The modified IRs make the model performance change heavily based on our previous observation. The data collector can hardly estimate the outputs after correction as $\mathbf{R}$ is unknown. The final estimation result is different from the actual one with $\mathbf{R}_L$.

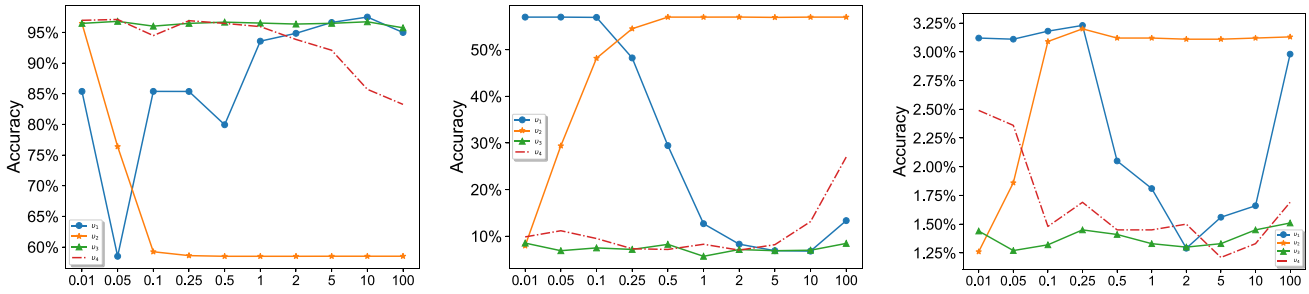
Discussion: We illustrate the validation process using CNN models. For other types of neuron networks, the validation process is the same. By designing a modified detection model, most of the computations except the classification computations are offloaded to data collectors. The verifier with a secure enclave will correct the IRs and output the validation results. Within a secure enclave, added noise will be decrypted and used to correct the IRs. To reduce correction computations within the enclave, we adopt the same idea as the work in [51], only weights with large values are added with noise.

VI. Performance Evaluation

A. Setup

The verifier is equipped with Intel(R) Core(TM) $i7-8700$ CPU @ 3.40 GHz processor with 16 GB of RAM and runs the computations on Ubuntu 18.04 environment with the Intel SGX support. The collector is equipped with a TELSA P100 GPU with 16 GB. ML models, including detection models and the transformation model, are built with TensorFlow framework [55] while the computation within the SGX enclave is implemented with C++ and Python. We translate the model structure that is built with the TensorFlow framework since TensorFlow is unfeasible inside the SGX enclave. We built convolution blocks based on Deep Neural Network Library (DNNL), which supports convolution, depthwise, standard, and activation operations. The forward pass of CNN models is implemented with C++. The intermediate results with noise and layer-wise inputs are passed to the enclave using C++.

Datasets & Models: We train our data transformation model and detection models using the MotionSense dataset [20] which consists of the accelerometer and gyroscope sensor data from smart devices. The sampling rate is set to be 50 Hz. The dataset has 12 features, including gravity, attitude, rotation rate, and



(a) Acc. of inferring activity. (b) Acc. of inferring gender. (c) Acc. of inferring user identity.

Fig. 7. Effect of trade-off parameters.

acceleration in three dimensions. The dataset contains data from 24 subjects, including 14 males and 10 females. There are 6 different activities, and we consider climbing up and down the stairs, walking, and jogging in our case. We consider activity detection is the task goal. We also evaluate our sanitization metrics on an image dataset (RAF-DB [30]), which has a large-scale facial expression dataset with around 30 K facial images downloaded from the Internet. We consider emotion detection as the task goal. The private attributes are age and gender.

We implement the encoder and decoder with 4-layer neural networks for time-series sensor data. CNN architectures are adopted for sanitizing image data. The hidden layer is instantiated with the ReLU activation function. We set the number of latent variables to 10 in the MotionSense dataset. The discriminator network is instantiated as a neuron network with 4 layers.

We use the CNN model with 3 convolutional layers and 3 dense layers to train detection models to estimate sensitive attributes. Max pooling and dropout layers are applied to hidden dense layers to avoid overfitting. We also implement a CNN model with 3 convolutional layers and 3 dense layers for activity recognition, which is used to evaluate the data utility of the reconstructed data. These models are used to evaluate the sanitization performance and data utility of the reconstructed data generated from our data transformation model.

B. Parameter Setting

In the following, we train and evaluate our data transformation model with different settings of model trade-off parameters, dimensions of latent representations, and clustering parameters.

We first study the impact of trade-off parameters by evaluating the accuracy of the activity inference model, gender inference model, and user identity inference model on the reconstructed data. The reconstructed data are generated by the data transformation model with different setting of υ_1, υ_2, υ_3, and υ_4. These parameters show the ability of loss component L_ϕ, L_θ, L_D, and L_I to affect data transformation model performance. For each trade-off parameter, we consider 10 values from 0.01 to 100. As shown in Fig. 7, L_D and L_I have less impact on the data transformation model performance compared with L_ϕ and L_θ as the trade-off parameters υ_3 and υ_4 do not heavily change detection model performance accordingly. Based on the performance of different inference models, we preserve data utility if we set υ_1 to be 5. If υ_2 is larger than 1, the L_θ will be minimized to 0 during the training process. To preserve data utility and make accuracy degradation of detection models, i.e., the gender detection model and the user identity detection model, exceed 70%, we set υ_2 to be 0.01. Taken together, we train our data transformation model by setting υ_1 to be 5, υ_2 to be 0.01, υ_3 to be 0.25 and υ_4 to be 0.5 as it can achieve 96.47% activity accuracy, which preserves data utility, while achieving 7.17% and 1.45% for gender and user identity detection.

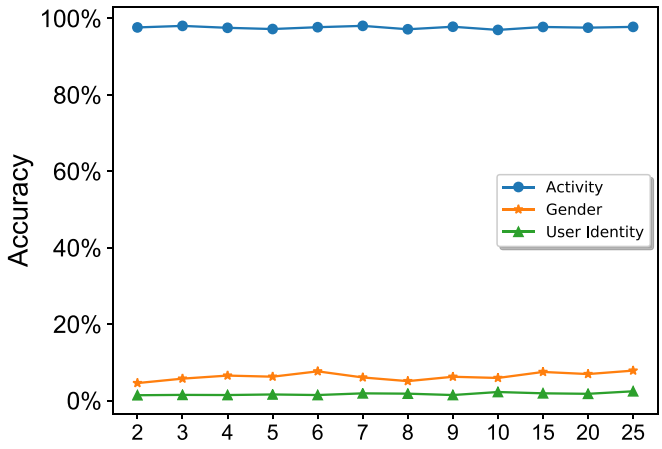


Fig. 8. Effect of latent representation dimension.

We further study the impact of varying latent representation dimension, and take 11 different dimension values from 2 to 25. We evaluate the impact of the dimension by running a task model and two different detection models on the reconstructed data. As shown in Fig. 8, the model accuracy does not change heavily with varying latent representation dimension, where the difference is less than 3%. The dimension seems to have less impact on our data transformation model. In the following, we set the dimension to be 5.

In Fig. 9, we evaluate the task model accuracy under different settings of ω, which affects the clustering loss. We update ω from a small value and increase with a gentle slope by following the recursive sequence $\omega_{T+1} = 2^{\frac{1}{\log(T)^2}}$ as the work in [53], and T is the epoch size. We train the data transformation model by taking

TABLE I
TABLE FOR NOTATIONS

Variables	Explanation
$\mathbf{X} = \{\mathbf{x}\}_{i=1}^{N}$	A dataset with N samples, each sample has multi-features and can be represented as m-element vector, *i.e.*, $\mathbf{x} \in \mathbb{R}^m$
$\mathcal{F}_T$	A task model to which the collected data will be fed, which is used to evaluate data utility
$\mathcal{F}$	A data transformation model consists of an encoder and a decoder to reconstruct data
$\mu_{\mathbf{x}}, \sigma_{\mathbf{x}}$	An output of the encoder for $\mathbf{x}$
$\mathbf{z}$	A latent representation sampled from a multivariate Gaussian distribution $\mathcal{N}(\mu_{\mathbf{x}}, \sigma_{\mathbf{x}})$
$\bar{\mathbf{x}}$	An output of the decoder called reconstructed data, $\bar{\mathbf{x}} \in \mathbb{R}^m$
θ, ϕ	Trainable model parameters of encoder and decoder
$\mathcal{F}_{D_j}$	Detection models for estimating the label of private attribute and user identity from a data sample, j represents attribute should be protected, $j \in \{A_1, \cdots, A_s, I\}$
$\mathbf{y}, \mathbf{y_T}, \mathbf{y_A}, \mathbf{y_I}$	The data label vector, task label vector, private attribute label vector, user identity label vector of a data sample, $\mathbf{y} \in \{0,1\}^K$ and its k_{th} element is 1 if the data sample belongs to the class k.
$K, \lvert A_s \rvert, \lvert I \rvert$	Number of class labels in task model, detection models for private attribute A_s and user identity inference
δ_j, γ_j	Thresholds claimed by a data collector and specified by data owners, respectively, $j \in \{A_1, \cdots, A_s, I\}$
L	The number of layers for an ML model
W_{L-1}, W'_{L-1}	Model weights and modified weights of an ML model in $\{L-1\}_{th}$ layer

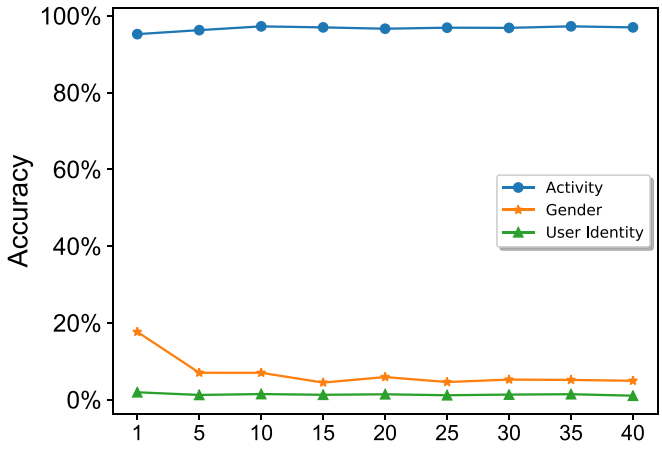


Fig. 9. Effect of different clustering parameter settings.

the iteration number of ω from 1 to 40 and evaluate the detection model and task model performance on the reconstructed data. The difference is quite small based on the results shown in Fig. 9. We set the initial value of ω to be 0.25 rather than 0. The user identity accuracy is decreased slightly, which is 1.92% for setting the iteration number as 1 while 1.03% for 40. The results indicate the data transformation model maps points with the same class close to each other such that the detection accuracy is low. For the gender detection model, the accuracy is around 4% to 5% if we choose an iteration number that is no less than 15. To train our data transformation model, we set the initial value of ω to be 0.25 and set the iteration number to be 15. The accuracy of sensitive inference on gender is 4.45%. The accuracy of user identity recognition is 1.27%.

C. Performance of Transformation Model

Based on the above results, we train our data transformation model by setting the trade-off parameter υ_1 as 5, υ_2 as 0.01, υ_3 as 0.25, and υ_4 is set as 0.5, the initial value of ω as 0.25, and the iteration number of ω as 15. We evaluate the sanitization performance of such data transformation model in the following. In the work of [23], they modify the private attributes or generate them randomly before using the decoder to synthesize a new version of data. In our work, we adopt the same idea and use a deterministic modification of the private attributes for a low reidentification accuracy.

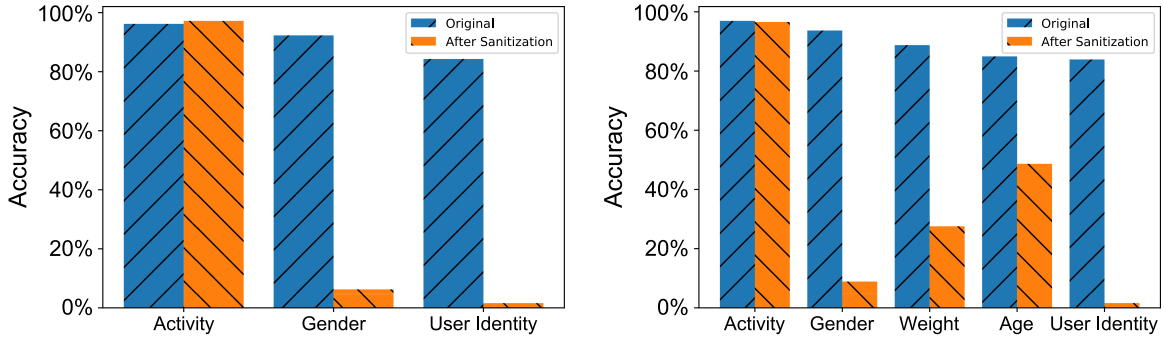
Data Sanitization Performance on MotionSense: We denote gender information as a private attribute and use the gender detection model to evaluate the sanitization performance of our transformation model. We use the activity recognition model to evaluate the utility. To present the sanitization performance of our data transformation model, we compare the accuracy of both sanitized and original data. The performance results are shown in Table II, which emphasizes that the data utility is preserved and improved using our protocol. The activity accuracy of the original data is 96.2%, while the accuracy of the reconstructed data achieves 96.9%.

In addition, we compare our work with the state-of-the-art method, i.e., ObscureNet [23]. Our method outperforms the ObscureNet in preserving data utility while providing good privacy protection. In this work, the accuracy approximates a random guess or even is lower, i.e., 47.4% and 1.3% on detecting gender information and individuals from the reconstructed data, respectively. For gender detection, the accuracy can drop to be approximately 7% if we deterministically modify the gender attributes before the decoder generates a sanitized version of the data sample. However, from an information-theoretic point of view, an extremely low accuracy in identifying information can be as good as a high accuracy in a binary case. Attackers can simply invert the predictions and get a high accuracy of detecting the correct labels. As a result, for a binary case such as gender detection, we adopt our sanitization method without the modification process of private attributes. The accuracy turns out to be 47.4%, which approximates a random guess (i.e., 50% for two classes) accuracy in detecting correct labels. However, we still pursue a low attack accuracy for multi-class private attributes and adopt deterministic modification of private attributes. The low accuracy shows that the detection models can hardly re-identify the reconstructed data's private attributes and user identity. For example, given the reconstructed data, we can hardly determine whether this data belongs to females or males. In addition, the individual who generates or owns this data is also unknown. Our results show that our sanitization method performs better in preserving data utility by introducing a clustering loss and clustering the sanitized data into different task classes. An ML task model can accurately classify sanitized data as they have been clustered into different task classes in advance.

In addition, we compare our sanitization model on both privacy and utility with the state-of-the-art works: ObscureNet [23] and work of Zhang et al. [36]. We test sanitization performance with multiple private attributes, including age, gender protection, and user identity. We have grouped age into five classes. The results are shown in Table III, where our work outperforms existing works on both data utility and data privacy.

TABLE II
COMPARISON OF UTILITY PRESERVATION

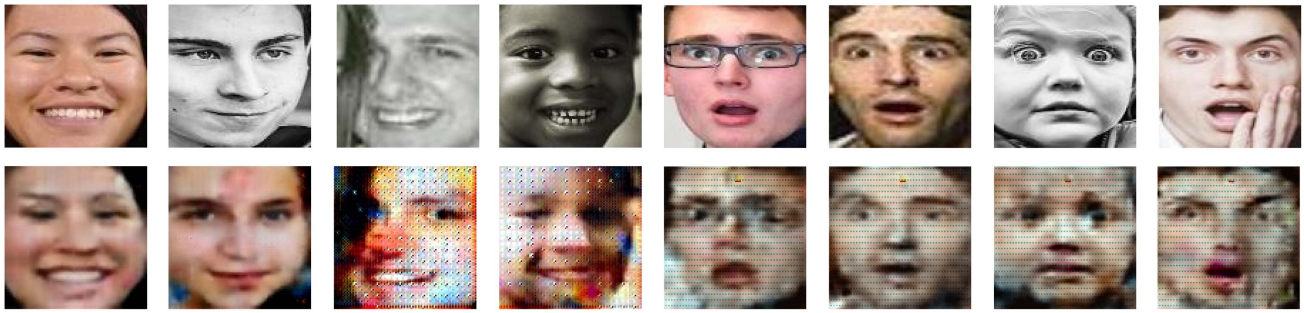
Method	Climbing Down the Stairs	Climbing Up the Stairs	Walking	Jogging	Overall Activity	Gender	Identification
The Original	95.6%	93.2%	98.7%	97.3%	96.2%	92.3%	84.3%
ObscureNet [23]	84.0%	92.7%	98.3%	97.1%	93.3%	13.5%	1.4%
Ours	87.6%	97.6%	98.3%	99.7%	96.9%	47.4%	1.3%



(a) Accuracy with one attribute protection.

(b) Accuracy with three attributes protection.

Fig. 10. Effect of adding more private attributes.



(a) Reconstructed images for happiness detection.

(b) Reconstructed images for surprise detection.

Fig. 11. Effect of image sanitization.

TABLE III
ACCURACY COMPARISON ON MOTIONSENSE

Method	Activity	Age	Gender	ID
The Original	96.2%	78.3%	92.3%	84.3%
ObscureNet [23]	93.3%	17.7%	13.5%	1.4%
Zhang [36]	92.3%	9.1%	56.8%	7.5%
Ours	96.9%	3.8%	47.4%	1.3%

The results show that the utility can be preserved by conditioning on task-related variables and introducing the clustering loss. For attributes that are grouped into more classes, we have lower accuracy. If we group age information into five classes, the age detection accuracy of sanitized data comes to 3.8% while we have 14.7% Our method can outperform existing works since we add an additional clustering module (adding another component to the loss function), which makes data samples with the same task attribute hard to be distinguished from each other.

More Private Attribute Inference: We also evaluate our data transformation model performance when extending the number of private attributes. Specifically, we denote gender, weight, and age as private attributes and train detection models for inferring gender, weight, and age attributes from the reconstructed data. As shown in Fig. 10, even if we enlarge the number of private attributes, we have accuracy degradation on all private attributes. The accuracy degradation on weight and age attributes is not as heavy as gender. This may be because their detection model accuracy is below 90%, which is less than the gender detection model accuracy.

Data Sanitization Performance on RAF-DB: We also estimate our sanitization protocol on image data. Our transformation model adopts a CNN architecture. The reconstructed results are shown in Fig. 11. The first row shows the original facial images, where the left shows the happy emotion and the right shows the surprised emotion. The second row shows the reconstructed images. As shown, the reconstructed facial images look different from the original ones, combing user identity re-identification while the emotional features are maintained. We also compare our sanitization model on both privacy and utility with ObscureNet [23]. We take age and gender as private attributes. We also reconstruct ObscureNet with CNN architectures. The sanitization model is denoted as ObscureNet$_{CNN}$. The comparison results are shown in Table IV, where our work outperforms existing works. For some tasks, the sanitized data

TABLE IV
PROTECTION PERFORMANCE COMPARISON ON RAF-DB

Method	Surprise	Happiness	Sadness	Age	Gender
The Original	79.12%	88.34%	81.61%	68.33%	80.33%
ObscureNet [23]	46.47%	86.4%	31.84%	49.33%	57.7%
ObscureNet$_{CNN}$	88.2%	84.3%	76.1%	17.1%	55.9%
Ours	95.2%	86.2%	77.7%	16.3%	43.1%

TABLE V
ACCURACY COMPARISON WITH FORGED INPUTS

Method	N(0,1)	N(0,2)	N(2,2)	N(0.2, 0.8)	[-4.5, 12.9]
Gender Acc	54.21%	47.92%	56.53%	56.91%	56.5%
ID Acc	3.35%	3.34%	4.13%	3.06%	3.62%

have higher accuracy than the original data. The results state the effectiveness of our protocol.

D. Validation With Forged Inputs

We reconstruct the MotionSense dataset using our sanitization protocol. In this experiment, we use two detection models to evaluate the sanitized data. Table V shows the accuracy of detection models if data collectors give forged inputs. We use different metrics to generate forged inputs, such as sampling from a normal distribution. In our example, we generate a normal distribution with mean as input data's mean, e.g., 0.2, and STD as input data's STD, e.g., 0.8. In addition, we sample data from the interval by analyzing the minimum, i.e., -4.5, and the maximum of data, i.e., 12.9. After correction, model accuracy in gender detection have similar values, which are around 50%, and all of the model accuracy for user identity detection are around 3% to 4%. The results also approximate the random guess as the class number is 2 and 24, respectively. This is because the detection models always estimate inputs to the same class given forged data as input. If we modify the private attribute, the real gender and ID detection accuracy on sanitized data are much lower than random guesses. The results are different from the one given forged inputs. In this way, it is easy to identify the forged IRs as the model accuracy on the forged data may work as adding noise to IRs, performing as a random guess.

E. Latency of Validation Within the Enclave

The latency of validation within the enclave is incurred by running one of the ML model layers. We then measure the latency of the layer's convolutional operations with different weight sizes. We evaluate the performance with an increasing number of weights in each convolutional layer and study the overhead incurred by using the SGX enclave. We also run inference on a native GPU but without the SGX to study the costs incurred using the SGX. From the experimental results, we have an acceptable overhead compared to running inference out of the enclave. It takes 0.1 s for two convolutional layers with the size of $512 \times 512 \times 3 \times 3$ within the enclave while running the one block (containing two layers in VGG) outside the enclave takes 0.003 s. In Fig. 12, we detail the running time of one specific hidden layer within the enclave. We set the weight sizes of last few layers based on design of VGG and MobileNet models. The weight size in the hidden layer is from 24,576, 59,904, 235,104, 653,952, and 965,952 parameters, respectively. As shown in Fig. 12, our protocol can provide an efficient model validation since the costs within the enclave are acceptable.

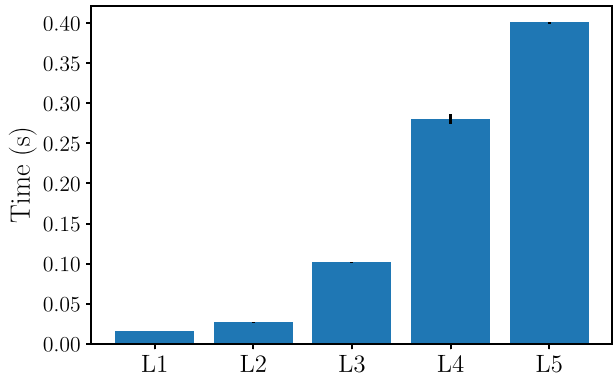


Fig. 12. Time of running hidden layers within the enclave.

Memory Assumption: Data owners' computation ability can be weak. Thus we offload most of the computations to data collectors in our protocol. The well-trained data sanitization model is shared with data owners, which implies data owners only require running data transformation (which takes around 1.6 seconds) rather than training the data transformation model (which takes several hours). The data transformation model for images is around 69 MB, while the model for time-series data is 2 MB, such that most mobile devices can store and execute.

VII. CONCLUSION

We have proposed a privacy-preserving data collection protocol with sanitization performance validation. We have designed a data transformation model to sanitize data samples while preserving data utility. In addition, we have adopted the blockchain and the secure enclave to validate the sanitization performance of the data transformation model efficiently. Experiments have demonstrated the performance of our proposed protocol. We believe that our work can make massive applications benefit from utilizing the wealth of user data and promote transparent data management in the future since we provide an efficient and effective protocol to protect data privacy during the data collection process. This work still has certain limitations and spaces to improve. If secret data on the blockchain are disclosed, data collectors may forge the validation and successfully pass the validation. We leave the study of defending against such data leakage as our future work.

REFERENCES

[1] H. F. Nweke, Y. W. Teh, G. Mujtaba, and M. A. Al-Garadi, "Data fusion and multiple classifier systems for human activity detection and health monitoring: Review and open research directions," *Inf. Fusion*, vol. 46, pp. 147–170, 2019.
[2] P. H. C. Chen, Y. Liu, and L. Peng, "How to develop machine learning models for healthcare," *Nat. Mater.*, vol. 18, no. 5, pp. 410–414, 2019.
[3] J. Hou et al., "SignSpeaker: A real-time, high-precision smartwatch-based sign language translator," in *Proc. Mobile Comput. Netw.*, 2019, pp. 1–15.
[4] R. Alouf, H. Haddadi, and D. Boyle, "Privacy-preserving voice analysis via disentangled representations," in *Proc. ACM Cloud Comput. Secur. Workshop*, 2020, pp. 1–14.

[5] M. Zhao, F. Adib, and D. Katabi, "Emotion recognition using wireless signals," in *Proc. Mobile Comput. Netw.*, 2016, pp. 95–108.

[6] H. Zhu, S. Samtani, H. Chen, and J. F. Nunamaker Jr, "Human identification for activities of daily living: A deep transfer learning approach," *J. Manage. Inf. Syst.*, vol. 37, no. 2, pp. 457–483, 2020.

[7] A. Anjum, et al., "τ-safety: A privacy model for sequential publication with arbitrary updates," *Comput. Secur.*, vol. 66, pp. 20–39, 2017.

[8] J.-W. Byun, T. Li, E. Bertino, N. Li, and Y. Sohn, "Privacy-preserving incremental data dissemination," *J. Comput. Secur.*, vol. 17, pp. 43–68, 2009.

[9] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li, "Secure anonymization for incremental datasets," in *Proc. 3rd Workshop Secure Data Manage.*, 2006, pp. 48–63.

[10] C. Dwork, "Differential privacy: A survey of results," in *Proc. Int. Conf. Theory Appl. Models Comput.*, Springer, 2008, pp. 1–19.

[11] F. Liu, "Generalized Gaussian mechanism for differential privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 4, pp. 747–756, Apr. 2019.

[12] Y. Wang, X. Wu, and D. Hu, "Using randomized response for differential privacy preserving data collection," in *Proc. EDBT/ICDT Workshops*, 2016, pp. 0090–6778.

[13] M. E. Gursoy, A. Tamersoy, S. Truex, W. Wei, and L. Liu, "Secure and utility-aware data collection with condensed local differential privacy," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2365–2378, 2019.

[14] S. Biswas and C. Palamidessi, "TPRIVIC: A privacy-preserving method for incremental collection of location data," 2022, *arXiv:2206.10525*.

[15] W. Liao, J. He, S. Zhu, C. Chen, and X. Guan, "On the tradeoff between data-privacy and utility for data publishing," in *Proc. IEEE Int. Conf. Parallel Distrib. Syst.*, 2018, pp. 779–786.

[16] N. Papernot, M. Abadi, Ú. I. J. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–16.

[17] S. Mehnaz, S. V. Dibbo, E. Kabir, N. Li, and E. Bertino, "Are your sensitive attributes private? Novel model inversion attribute inference attacks on classification models," in *Proc. 31st USENIX Secur. Symp.*, 2022, pp. 4579–4596.

[18] A. E.R. Prince and D. Schwarcz, "Proxy discrimination in the age of artificial intelligence and big data," *Iowa Law Rev.*, vol. 105, pp. 1257–1318, 2020.

[19] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Found. Trends Mach. Learn.*, vol. 12, no. 4, pp. 307–392, 2019.

[20] M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi, "Mobile sensor data anonymization," in *Proc. IEEE Int. Conf. Internet Things Des. Implementation*, 2019, pp. 49–58.

[21] M. Malekzadeh, R. G. Clegg, and H. Haddadi, "Replacement autoencoder: A privacy-preserving algorithm for sensory data analysis," in *Proc. IEEE Int. Conf. Internet Things Des. Implementation*, 2018, pp. 165–176.

[22] S. A. Osia et al., "A hybrid deep learning architecture for privacy-preserving mobile analytics," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4505–4518, May 2020.

[23] O. Hajihassnai, O. Ardakanian, and H. Khazaei, "ObscureNet: Learning attribute-invariant latent representation for anonymizing sensor data," in *Proc. IEEE Int. Conf. Internet Things Des. Implementation*, 2021, pp. 40–52.

[24] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Proc. Neural Inf. Process. Syst.*, 2015, pp. 3483–3491.

[25] T. Li, A. Kumar Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[26] M. He, J. Ni, D. Liu, H. Yang, and X. Shen, "Private, fair, and verifiable aggregate statistics for mobile crowdsensing in blockchain era," in *Proc. IEEE Int. Conf. Commun. China*, 2020, pp. 160–165.

[27] P. Voigt and A. Von dem Bussche, *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Berlin, Germany: Springer, 2017.

[28] X. Shen et al., "Blockchain for transparent data management toward 6G," *Engineering*, vol. 8, pp. 74–85, 2021.

[29] H. Zhang, P. Gao, J. Yu, J. Lin, and N. Xiong, "Machine learning on cloud with blockchain: A secure, verifiable and fair approach to outsource the linear regression for data analysis," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 6, pp. 3956–3967, Nov.-Dec. 2022.

[30] S. Li and W. Deng, "Reliable crowdsourcing and deep locality-preserving learning for unconstrained facial expression recognition," *IEEE Trans. Image Process.*, vol. 28, no. 1, pp. 356–370, Jan. 2019.

[31] T. Zhang, Z. He, and R. B. Lee, "Privacy-preserving machine learning through data obfuscation," 2018, *arXiv: 1807.01860*.

[32] L. Lyu, Y. W. Law, S. M. Erfani, C. Leckie, and M. Palaniswami, "An improved scheme for privacy-preserving collaborative anomaly detection," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, 2016, pp. 1–6.

[33] Y. Sang, H. Shen, and H. Tian, "Effective reconstruction of data perturbed by random projections," *IEEE Trans. Comput.*, vol. 61, no. 1, pp. 101–117, Jan. 2012.

[34] M. E. Gursoy, A. Tamersoy, S. Truex, W. Wei, and L. Liu, "Secure and utility-aware data collection with condensed local differential privacy," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2365–2378, Sep./Oct. 2021.

[35] Q. Wang, Y. Zhang, X. Lu, Z. Wang, Z. Qin, and K. Ren, "Real-time and spatio-temporal crowd-sourced social network data publishing with differential privacy," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 4, pp. 591–606, Jul./Aug. 2018.

[36] D. Zhang, L. Yao, K. Chen, Z. Yang, X. Gao, and Y. Liu, "Preventing sensitive information leakage from mobile sensor signals via integrative transformation," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4517–4528, Dec. 2022.

[37] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," 2015, *arXiv:1511.05644*.

[38] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *J. Healthcare Inform. Res.*, vol. 5, pp. 1–19, 2021.

[39] J. Zhang, Z. Fang, Y. Zhang, and D. Song, "Zero knowledge proofs for decision tree predictions and accuracy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 2039–2053.

[40] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2438–2455, Sep./Oct. 2021.

[41] L. Zhao, Q. Wang, C. Wang, Q. Li, C. Shen, and B. Feng, "VeriML: Enabling integrity assurances and fair payments for machine learning as a service," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 10, pp. 2524–2540, Oct. 2021.

[42] J. D. Harris and B. Waggoner, "Decentralized and collaborative AI on blockchain," in *Proc. IEEE Int. Conf. Blockchain*, 2019, pp. 368–375.

[43] Z. Zheng et al., "Agatha: Smart contract for DNN computation," 2021, *arXiv:2105.04919*.

[44] K. Wüst, S. Matetic, S. Egli, K. Kostiainen, and S. Capkun, "ACE: Asynchronous and concurrent execution of complex smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 587–600.

[45] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Bus. Inf. Syst. Eng.*, vol. 59, no. 3, pp. 183–187, 2017.

[46] X. Shen et al., "Data management for future wireless networks: Architecture, privacy preservation, and regulation," *IEEE Netw.*, vol. 35, no. 1, pp. 8–15, Jan./Feb. 2021.

[47] V. Costan and S. Devadas, "Intel SGX explained.," *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.

[48] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens, "Plundervolt: Software-based fault injection attacks against Intel SGX," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 1466–1482.

[49] S. Fei, Z. Yan, W. Ding, and H. Xie, "Security vulnerabilities of SGX and countermeasures: A survey," *ACM Comput. Surveys*, vol. 54, no. 6, pp. 1–36, 2021.

[50] T. Lee et al., "Occlumency: Privacy-preserving remote deep-learning inference using SGX," in *Proc. Mobile Comput. Netw.*, 2019, pp. 1–17.

[51] J. Hou, H. Liu, Y. Liu, Y. Wang, P.-J. Wan, and X.-Y. Li, "Model protection: Real-time privacy-preserving inference service for model privacy at the edge," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 6, pp. 4270–4284, Nov./Dec. 2022.

[52] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learn. Representations*, 2014, pp. 1–14.

[53] M. M. I. Fard, T. Thonet, and E. Gaussier, "Deep k-means: Jointly clustering with k-means and learning representations," *Pattern Recognit. Lett.*, vol. 138, pp. 185–192, 2020.

[54] R. Liu, Y. Cao, H. Chen, R. Guo, and M. Yoshikawa, "Flame: Differentially private federated learning in the shuffle model," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 10, 2021, pp. 8688–8696.

[55] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. USENIX Symp. Operating Syst. Des. Implementation*, 2016, pp. 265–283.

Jiahui Hou received the BE degree in computer science and technology from the University of Science and Technology of China, China, in 2015 and the PhD degree from the Department of Computer Science, Illinois Institute of Technology, USA. She is a professor with the School of Computer Science and Technology, University of Science and Techonology of China, Hefei, China. She was postdoctoral research fellow with the Department of Electrical and Computer Engineering, the University of Waterloo from 2021 to 2022. Her research interests include privacy and security issues in Artificial Intelligence of Things and AI.

Dongxiao Liu (Member, IEEE) received the PhD degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada in 2020. He is a postdoctoral research fellow with the Department of Electrical and Computer Engineering, University of Waterloo. His research interests include security and privacy in intelligent transportation systems, blockchain, and mobile networks.

Cheng Huang (Member, IEEE) received the BEng and MEng degrees in information security from Xidian University, China, in 2013 and 2016 respectively, and the PhD degree in electrical and computer engineering from the University of Waterloo, ON, Canada in 2020. He is currently a postdoctoral research fellow with the Department of Electrical and Computer Engineering, University of Waterloo. His research interests are in the areas of applied cryptography, cyber security and privacy in the mobile network.

Weihua Zhuang (Fellow, IEEE) received the BSc and MSc degrees in electrical engineering from Dalian Marine University, China, and the PhD degree in electrical engineering from the University of New Brunswick, Canada. She is a University professor and a Tier I Canada research chair in Wireless Communication Networks with the University of Waterloo, Canada. Her research focuses on network architecture, algorithms and protocols, and service provisioning in future communication systems. She is the recipient of 2021 Women's Distinguished Career Award from IEEE Vehicular Technology Society, 2021 Technical Contribution Award in Cognitive Networks from IEEE Communications Society, 2021 R.A. Fessenden Award from IEEE Canada, and 2021 Award of Merit from the Federation of Chinese Canadian Professionals in Ontario. She was the editor-in-chief of *IEEE Transactions on Vehicular Technology* from 2007 to 2013, general co-chair of 2021 IEEE/CIC International Conference on Communications in China (ICCC), technical program chair/co-chair of 2017/2016 IEEE VTC Fall, Technical Program Symposia Chair of 2011 IEEE Globecom, and IEEE Communications Society distinguished lecturer from 2008 to 2011. She is an elected member of the Board of Governors and the president of IEEE Vehicular Technology Society. She is a fellow of the Royal Society of Canada, Canadian Academy of Engineering, and Engineering Institute of Canada.

Xuemin Shen (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is a University professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular networks. He is a registered professional engineer of Ontario, Canada, an Engineering Institute of Canada fellow, a Canadian Academy of Engineering fellow, a Royal Society of Canada fellow, a Chinese Academy of Engineering foreign member, and a distinguished lecturer of IEEE Vehicular Technology Society and Communications Society. He received "West Lake Friendship Award" from Zhejiang Province in 2023, President's Excellence in Research from the University of Waterloo in 2022, the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT) in 2021, the R.A. Fessenden Award in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, James Evans Avant Garde Award in 2018 from IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from IEEE Communications Society (ComSoc), and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He serves/served as the general chair for the 6G Global Conference'23, and ACM Mobihoc'15, Technical Program Committee chair/co-chair for IEEE Globecom'24, 16 and 07, IEEE Infocom'14, IEEE VTC'10 Fall, and the chair for the IEEE ComSoc Technical Committee on Wireless Communications. He is the president of the IEEE ComSoc. He was the vice president for Technical & Educational Activities, vice president for Publications, Memberat-Large on the Board of Governors, Chair of the distinguished lecturer Selection Committee, and member of IEEE Fellow Selection Committee of the ComSoc. He served as the editor-in-chief of *IEEE IoT Journal*, *IEEE Network*, and *IET Communications*.

Rob Sun (Member, IEEE) is a principal engineer with Huawei Technologies Canada Company Ltd. His work primarily focuses on the advancement of the NG wireless, including 5G/6G and WiFi/IoT security architecture and standardization. He was also vice chair of IEEE PMP (Privacy Management Protection) Task Group which was to set out the best practices for protecting personal privacy information, and support efficient, adaptable and innovative approaches for privacy governance. He was regarded as one of the core contributors to the standardizations of a series of NG Wi-Fi security protocols and certifications, including the most recent Wi-Fi WPA3 protocol suites. He also co-authored a few books on wireless security technologies.

Bidi Ying (Member, IEEE) is currently working in Huawei Technologies Canada Company, Ltd., as a senior network architecture engineer. Her main research is about security and privacy in wireless network. Before that, she was working in University of Ottawa. During the past 15 years, she has published more than 200 papers in top conferences and reputable journals.