Replacement AutoEncoder: A Privacy-Preserving Algorithm for Sensory Data Analysis

Mohammad Malekzadeh, Richard G. Clegg
School of Electrical Engineering and Computer Science
Queen Mary University of London
London, UK
m.malekzadeh, r.clegg@qmul.ac.uk

Hamed Haddadi
Faculty of Engineering
Imperial College London
London, UK
h.haddadi@imperial.ac.uk

***Abstract*—An increasing number of sensors on mobile, Internet of things (IoT), and wearable devices generate time-series measurements of physical activities. Though access to the sensory data is critical to the success of many beneficial applications such as health monitoring or activity recognition, a wide range of potentially sensitive information about the individuals can also be discovered through access to sensory data and this cannot easily be protected using traditional privacy approaches.**

In this paper, we propose a privacy-preserving sensing framework for managing access to time-series data in order to provide utility while protecting individuals' privacy. We introduce *Replacement AutoEncoder*, a novel feature-learning algorithm which learns how to transform discriminative features of multi-variate time-series that correspond to sensitive inferences, into some features that have been more observed in non-sensitive inferences, to protect users' privacy. This efficiency is achieved by defining a user-customized objective function for deep autoencoders. Replacement will not only eliminate the possibility of recognition sensitive inferences, it also eliminates the possibility of detecting the occurrence of them, that is the main weakness of other approaches such as filtering or randomization.We evaluate the efficacy of the algorithm with an activity recognition task in a multi-sensing environment using extensive experiments on three benchmark datasets. We show that it can retain the recognition accuracy of state-of-the-art techniques while simultaneously preserving the privacy of sensitive information. Finally, we utilize the GANs for detecting the occurrence of replacement, after releasing data, and show that this can be done only if the adversarial network is trained on the users' original data.

***Index Terms*—Privacy Protection; Feature Learning; Time-Series Analysis; Wearable Sensors; Activity Recognition;**

I. Introduction

Smart devices around us are becoming more interconnected and a large variety of data are captured by these sensors on a regular basis. From smartphones, to smart watches and wearables, many of our modern electronic devices have the ability to produce data. Since detailed, personally-identifying datasets in their original form often contain sensitive information about individuals, inconspicuous data collection can lead to major personal privacy concerns. As an example, just ambient light sensor data [1] or accelerometer readings [2] are sufficient to extract sequences of entered text on smartphones and consequently to reveal users' passwords. Moreover, Apthorpe *et al.* [3] examine four IoT smart home devices and find that their network traffic rates can reveal potentially sensitive user interactions even when the traffic is encrypted. Therefore, trustworthy analysis of time-series sensory data without any revelation of sensitive information to third parties is a challenging task.

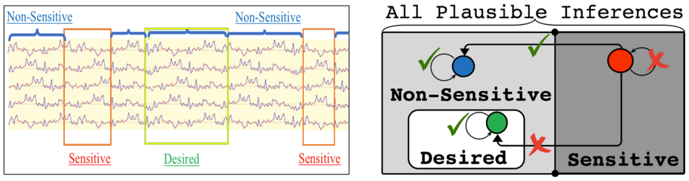

Fig. 1. *(Left)* We assume each section of a multi-variate time-series contains information about a specific inference that is categorized into two classes: *Sensitive* and *Non-Sensitive* (including *Desired*). *(Right)* The ultimate goal is hiding sensitive information by transforming each sensitive section into a non-sensitive (non-desired) one while keep the remaining sections unmodified.

Recent advances in mobile and ubiquitous computing technologies have concurrently accelerated the interest in cloud services. One of the most desired applications of these services is the recognition of current user activity and actuate a response (such as displaying alerts or turning a device on/off). However, omnipresent data gathering and user tracking leads to inherent data security and privacy risks [4]. The main question is how desired statistics of personal time-series can be released to an untrusted third party without compromising the individual's privacy. More precisely, we consider how to transform time-series sensor data in such a way that after release, unwanted sensitive information cannot be inferred, but features the user wishes to share are preserved.

In this paper, thanks to recent advances in feature learning and autoencoder architectures [5], we develop a privacy-preserving platform for the analysis of multichannel time-series data generated by IoT sensors. Autoencoders are neural networks trained to reconstruct their original input, which can be considered as a form of feature extraction algorithm (we will describe it in section IV). We consider existing approaches for preserving inference privacy in time-series data analysis and categorize the inferences that can be made into three disjoint sets: *black-listed* (sensitive for users), *gray-listed* (non-sensitive for users and not desired for applications) and *white-listed* (required to gain utility from third party services). Therefore, we propose a feature-based replacement method to eliminate sensitive information in time-series by transforming them to non-sensitive data (see Figure 1). By applying our method, data utility will be unaffected for specific applications and cloud apps can accurately infer the desired information, while it would be very difficult to detect and almost impossible to recognize a sensitive inference in the shared data. Specifically, our contributions are:

- Inspired by denoising autoencoders [5], we introduce a deep autoencoder architecture which is able to flexibly and adaptively extract useful features from time-series data. This real-time algorithm which we call *replacement autoencoder* (RAE) learns how to replace features of each section of time-series which correspond to sensitive (black-listed) inferences with some values which correspond to non-sensitive (gray-listed) inferences.
- To evaluate the tradeoff between data utility and inference privacy, we apply the algorithm to activity recognition task on three benchmark datasets collected from smart wearable devices. We show how it can retain the recognition accuracy of state-of-the-art techniques [6, 7] for desired inferences, and simultaneously preserve the privacy of data subject by protecting sensitive information.
- Finally, by utilizing the power of Generative Adversarial Networks (GANs) [8], we show that adversaries can detect the presence of data replacement, by distinguishing between real gray-listed data and replaced data (transformed black-listed), only under the assumption that they are able to access the user's gray-listed dataset which was used to train his/her model. Though even under this assumption, recognition of the type of sensitive inferences (without any other side information) is still almost impossible.

Compared to other work on preserving the privacy of sensitive data by anonymization or randomization techniques in databases [9, 10], we consider a new situation dealing with real-time sensory data. In our setting an untrusted cloud application knows the users' identity (thus differential privacy [11] is not applicable) and we want to prevent that application inferring sensitive information from their personal data. The proposed platform enables users to protect their sensitive data, while benefiting from sharing their non-sensitive data with cloud apps.[1]

II. Inference Privacy

A time-series is a collection of observations obtained through repeated measurements over successive equally spaced points in time. The collection of sensory data, from IoT, mobile, or wearable devices, generates time-series measurements of physical quantities such as velocity and acceleration that offers tremendous opportunities for applications. For example, monitoring different physiological parameters using accelerometer and pressure sensor can create time-series for monitoring users' health status [12]. However, malicious programs can secretly collect the time-series and use them to discover a wide range of sensitive information about users' activities [13].

In ubiquitous sensing, the privacy of sensitive inferences is formulated as a tradeoff between users' need to receive utility from third parties with the amount of information inferred from shared time-series [14]. In this setting, it is often observed that some sections of the time-series can be used to infer *sensitive* information about their users' behavior (*black-listed*), and the rest of the sections contains information about *desired* information (*white-listed*) [15]. Here, we introduce a third kind of information in sensory data: those sections of the time-series that contain *non-sensitive* information (*gray-listed*). Therefore, throughout this paper we discuss three kinds of inference (see Figure 1):

- **black-listed Inferences**: sensitive information that users wish to keep private and should not be revealed to third parties. The black-listed inferences are sufficiently sensitive that the user would wish to prevent inference that they have undertaken any activities within this set even if the third party did not know which specific inference.
- **white-listed Inferences**: desired information that users gain utility from sharing with third parties services.
- **gray-listed Inferences**: non-sensitive information that is neutral. It is not useful to the user that this information is detected by third parties yet it is not important for the user to hide it. It may include data that cannot be mapped to a known behavior.

For example, consider a step counter application for a smart watch that uses time-series data generated by equipped sensors for counting the owner's steps. These rich time-series can be used for recognizing many activities, such as *Walking, Jogging, Running, Sitting, Lying, Drinking, Smoking, Fighting, and Sleeping*. For an example user white-listed inferences could be *Walking, Jogging, and Running*, black-listed ones could be *Drinking, Smoking, Fighting, and Sleeping*, and finally, gray-listed inferences could be *Sitting and Lying*. It is necessary to point out that, in comparison to static databases, there are also many possible real situations which in sensory data correspond to non-sensitive inferences. Consider the number of different possible orientations you can rotate your hand and corresponding data associated with them measured by equipped sensors in your smart watch [16]. Hence, it would be reasonable to safely replace sensitive sections of time-series with non-sensitive real sections in a way that no information about the replacement operation could leak to the other party.

Generally, there are four privacy-preserving approaches to dealing with this kind of personal time-series: **(A) Randomization**: adding noise to sensitive sections of the time-series to make them harder to understand. **(B) Filtering**: removing sensitive sections of the time-series to avoid any inference. **(C) Mapping**: projection of the original time-series into a lower dimensional space to reduce

[1] All the code and data used in this paper is publicly available and can be obtained from: https://github.com/mmalekzadeh/replacement-autoencoder

the amount of information included in data. **(D) Replacement**: replacement of sensitive sections of the time-series with some non-sensitive data. In the following, we discuss why the first three approaches are not suitable and how we can cover all of their disadvantages by using the fourth approach.

A. Randomization

Generally, in this approach an independent [17] or a correlated [18] noise (additive or multiplicative) is added to time-series for hiding sensitive information. Kargupta *et al.* [17] have empirically shown that independent random noise with low variance preserves very little data privacy since most of the noise can be filtered, and high variance noise completely destroys the utility of data mining. Similarly, Wang *et al.* [19] argued that independent and identically distributed Laplace noise series used in current approaches can be removed from the correlated time-series by utilizing a refinement method (e.g. filtering). To remedy this problem, they proposed a correlated Laplace mechanism which guarantees that the correlation between the noise and original time-series is indistinguishable to an adversary. However, their method does not support non-stationary and high-dimensional time-series. Perturbed sections can be easily detected by adversaries and adding correlated noise to each sensitive section, based on its own structure and features, will be very challenging. The main challenge is to generate a correlated noise that after adding to data, the detection of black-listed sections of the time-series will be impossible. We show how feature-based replacement can answer this challenge.

B. Filtering

In the filtering approach, instead of perturbation or randomization, we can completely remove sensitive sections of time-series data before publishing. For example, MaskIt [20] is a filtering technique that decides whether to release or suppress the current state of the smartphone users (e.g. current location). By doing this, recognition of black-listed activities will be very hard by third parties because no information about them is published. However, detection of time intervals associated with sensitive activities will be very simple, hence a side channel attack is possible. For example if an attacker gets access to user's web browser history, then they might know that currently they are in a pub, so the filtered out data could correspond to "Drinking". In addition, some sections of a multichannel time-series can include both sensitive and desired information, so entirely removing them can greatly reduce the efficiency of data. We discuss how feature learning enables us to efficiently remove sensitive information from multichannel time-series without completely filtering them out.

C. Mapping

In this approach, instead of the original high dimensional time-series, from which inferring unnecessary sensitive information will be easy, a small set of features is extracted from data and published to third parties. This is done in such a way that the data can be only useful for inferring desired (white-listed) information. In fact, feature sharing can be considered as a mapping mechanism to control and reduce the inferences possible from shared sensory data [21]. Laforet *et al.* [22] introduced an optimization method to generate and publish an entirely new time-series from the original ones by considering some predefined constraints on the type of information that can be inferred. Another solution would be to publish just task-relevant high-level features. Recently, Osia *et al.* [23] introduced an algorithm in deep learning to manipulate the extracted features (from the primary layers of neural networks) in a way that protects the data privacy against unauthorized tasks by creating a feature set which is specific to a desired task.

The hardest part of this approach is selecting the best reduced set of features that provide a reasonable tradeoff between utility and privacy. This approach is very dependent on the method used by third parties because we need to know if their services are compatible with this reduced set of features or not. Furthermore, it is impossible to state that a feature extracted from original data only and only contains information about white-listed information and they cannot reveal any information about black-listed activities. We explain how these challenges can be answered by applying a feature-based replacement instead of feature selection.

D. Replacement

As discussed in the previous sections, not only recognition of sensitive activities, but also detection of the presence of such activities in the published time-series can violate a data subject's privacy. The main idea of the replacement approach is replacing sensitive sections of time-series with sections very similar to the non-sensitive sections of the time-series. This approach benefits the main advantages of the previously mentioned approaches while mitigating their disadvantages.

For example, Saleheen *et al.* [15] argues that there is a high degree of correlation between observed human behavior and the data recorded by surrounding physiological sensors. They proposed a Dynamic Bayesian Network (DBN) model that, instead of random substitution, uses user-specific substitution as a mechanism to protect the privacy of black-listed behaviors while preserving the classification accuracy for white-listed behavior. But their model-based algorithm ignored gray-listed inferences, it is offline and assumes the availability of all time-series data before starting the replacement procedure. They also maintain a mapping database, for use in the replacement phase, that stores sensor segment of different length for each possible state. In this paper, we propose a feature-based replacement method that only needs users' personal data to provide an online and robust replacement method for privacy-preserving analysis of sensory data.

III. Mediator Architecture

There are several motivations and preference factors involved when sharing personal sensor data. Users are very conservative in some situations, and less concerned in other cases. For this purpose, we need a personal and customizable

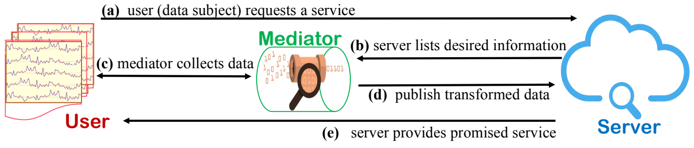
(a) user (data subject) requests a service
Mediator
(b) server lists desired information
(c) mediator collects data
(d) publish transformed data
User
(e) server provides promised service
Server

Fig. 2. High-Level Architecture: a privacy-preserving mediator between cloud servers and data subject. Sensitive sections of personal time-series are transformed with the Mediator, based on the relevant application, and then shared with the third party's server.

framework for users to control their privacy according to their use of each application. Awareness of threats on collected time-series that can jeopardize the users' privacy will help them to choose whether to provide apps with their raw data or applying some transformation before granting access to them. In this section, we discuss the high-level architecture of the *Mediator*: a personal platform that enables cloud services to provide the user with desired services while protecting the privacy of the data subject (see Figure 2).

Currently, there are two major approaches to sharing sensor data for analyzing purposes. One is sharing the data with trusted third parties without any alteration based on non-disclosure agreements, but difficulties in trusting external data processors is the main weakness of this approach. Another solution is using one of the aforesaid methods in Section II) for transforming data before publishing them to the untrusted third parties. The tradeoff between information loss and user privacy is the most challenging part of this approach. Moreover, since in the both approaches data typically leaves users' side and resides in a shared environment such as cloud servers, users no longer have any control over their data. Recently, Haddadi *et al.* [24] developed a personal networked device, enabling individuals to coordinate the collection of their personal data, and to selectively and transiently make those data available for specific purposes.

Considering these challenges, we propose a *privacy by design* solution as an intermediate method that preserves the privacy of sensitive information without the need to trust third parties and simultaneously allow data subject to benefit from cloud services for their desired applications. Figure 2 shows the high-level architecture of the Mediator; *a privacy-preserving platform operates between cloud services and data subject*. Time-Series sensory data are processed with the Mediator based on the relevant application and then shared with the third party's server. In fact, based on an initial agreement between the user and the server, the Mediator intelligently hides sensitive information before releasing data. Generally, the main goal of the Mediator is to prevent the reconstruction of original time-series from transformed ones, yet allow to accurately estimate some statistics despite the transformation.

As represented in figure 2 we have three major actors here:

- **Server**: a third party provides a desired service hosted in the cloud (such as activity recognition, health monitoring, occupancy detection, and smart home management). Here, we assume it owns a machine learning model that uses some of the features of the users' time-series to produce an output related to the requested service.
- **User** : a user owns some sensors that continuously generate time-series data (such as noise, light, oxygen, temperature, acceleration, humidity, motion, and etc.). User wants to benefit from cloud hosted services by sharing their sensory data, but in a way that only desired and non-sensitive information can be inferred.
- **Mediator**: this is generally a machine learning platform. It creates a model for getting three major inputs: (1) original time-series of *User*, (2) list of desired inferences for *Server*, and (3) a personal dataset of *User* including sensitive and non-sensitive data to train Replacement AutoEncoder. After training, the Mediator can transforms User's data in real-time to send out to the *Server*. Mediator divides time-series into a sequence of sections and releases each section separately. *It should definitely be a trusted application that runs on the User side*. Recently, some architectures have been proposed for mediating access to users' personal data, under their control, that can be used for this purpose [24].

In the following, we propose Replacement AutoEncoder as an efficient data transformation algorithm for deploying in the Mediator.

IV. Feature Learning

In this section, we motivate and outline our proposed algorithm for online replacement of sensitive information in time-series with non-sensitive data. We discuss the core idea behind feature learning (also known as representation learning), talk about the properties of *autoencoder* architecture and its usefulness for our purposes, and inspired by *denoising autoencoder*, proposed by Vincent *et al.* [5], introduce a new architecture, *replacement autoencoder*, for real-time privacy-preserving analysis of sensory data. We show how discriminative features extracted by replacement autoencoder can be used to replace sensitive information in personal time-series with some non-sensitive data, without loss of utility for specific tasks. Throughout the paper, we will use the following notation: $\boldsymbol{x}$ is an input vector (a section of time-series) and $\boldsymbol{z}$ is output vector which is generated by neural network. $\boldsymbol{b}$, $\boldsymbol{g}$, and $\boldsymbol{w}$ will respectively denote a *black-listed*, *gray-listed*, and *white-listed* section of time-series. ϕ is an activation function and $L(x, z)$ is a loss function. An activation function (e.g. sigmoid function) of a node in a neural network defines the output of that node given a set of inputs, and a loss function considers the difference between predicted and true values for an instance of input data to measure how well a neural network is working.

A. Autoencoder

The performance of machine learning models is heavily dependent on the type of data representation and the robustness of extracted features on which they are applied. The key aspect of feature learning is the intelligent extraction of features

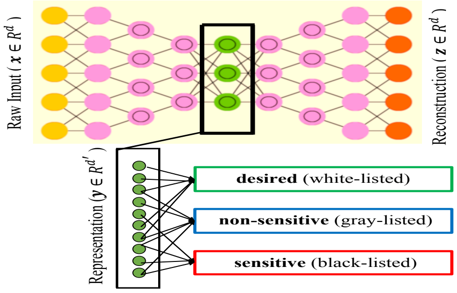
Raw Input ($x \in R^d$)
Reconstruction ($z \in R^d$)
Representation ($y \in R^{d'}$)
desired (white-listed)
non-sensitive (gray-listed)
sensitive (black-listed)

Fig. 3. An autoencoder is a neural network that is trained to encode the input x into some representation $y = f(x)$ so that the output $z = g(f(x))$ is a reconstruction of the input based on the representation. The autoencoder tries to capture underlying explanatory factors and discover a robust and discriminative feature set from training data. Each subset of this feature set may be relevant for each particular inference in our desired task.

from data and discover the informative representation needed for desired task. LeCun *et al.* [25] argue that deep-learning frameworks are in fact representation learning methods with multiple levels of representation, obtained by composing multilayer stack of non-linear modules that each transform the representation at one level into a representation at a higher, slightly more abstract level. Bengio *et al.* [26] have shown that the hidden layers of a multilayer neural network (starting with the raw input) can potentially learn more abstract features at higher layers of representations and reuse a subset of these features for each particular task. The earlier layers of a deep neural network contain more generic features that should be useful for many tasks, but later layers become progressively more specific to the details of the classes relevant to the desired task [27].

As depicted in Figure 3, an autoencoder takes an input vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$, and first maps it to a hidden representation $\boldsymbol{y} = (y_1, y_2, \ldots, y_{d'}) \in \mathbb{R}^{d'}$ through a deterministic mapping $\boldsymbol{y} = f_\theta(\boldsymbol{x}) = \phi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$, parameterized by $\theta = \{\boldsymbol{W}, \boldsymbol{b}\}$. ϕ is an activation function, $\boldsymbol{W}$ is a $d' \times d$ weight matrix, and $\boldsymbol{b}$ is a bias vector.

Let us consider $\boldsymbol{y}$ as a feature set extracted by autoencoder, and each y_i in $\boldsymbol{y}$ to be equivalent to output of a node in the middlemost layer of autoencoder (see Figure 3). The resulting latent representation $\boldsymbol{y}$ is then mapped back to a reconstructed vector $\boldsymbol{z} = g_{\theta'}(\boldsymbol{y}) = \phi(\boldsymbol{W}'\boldsymbol{y} + \boldsymbol{b}')$ in input space ($\boldsymbol{z} \in \mathbb{R}^d$) with $\theta' = \{\boldsymbol{W}', \boldsymbol{b}'\}$. The parameters of the autoencoder are optimized to minimize average reconstruction error $L(\boldsymbol{x}, \boldsymbol{z})$:

$$\theta^*, \theta'^* = \arg_{\theta,\theta'} \min \frac{1}{n} \sum_{i=1}^{n} L(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}) \tag{1}$$

where $\boldsymbol{x}^{(i)}$ is the i^{th} input data in the training dataset and L is a loss function (e.g. squared error $L(\boldsymbol{x}, \boldsymbol{z}) = ||\boldsymbol{x} - \boldsymbol{z}||^2$). In fact, $L(\boldsymbol{x}, \boldsymbol{z})$ is a measure of the discrepancy between $\boldsymbol{x}$ and its reconstruction $\boldsymbol{z}$, over all provided training examples [28].

The autoencoder can include one or more hidden layers that hold parameters which are used to represent the input. A deep autoencoder captures the structure of the data-generating distribution, by constraining the architecture to have a low dimension ($d' < d$ in Figure 3). Generally, the aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction or feature extraction. In an autoencoder, for each type of input (corresponding to a specific class of data), a subset of hidden units might be activated in response to it. In other words, for any given input $\boldsymbol{x}$, only a small fraction of the possible features $\boldsymbol{y}' \subseteq \boldsymbol{y}$ are relevant, thus the representation associated with that input precisely describes which features respond to the input, and how much. Deep architectures can lead to abstract representations in the last layers. In fact, more abstract concepts in the last layers can often be constructed in terms of less abstract ones captured in the early layers. More abstract concepts are generally invariant to most local changes of the input hence the outputs are usually highly nonlinear functions of the input.

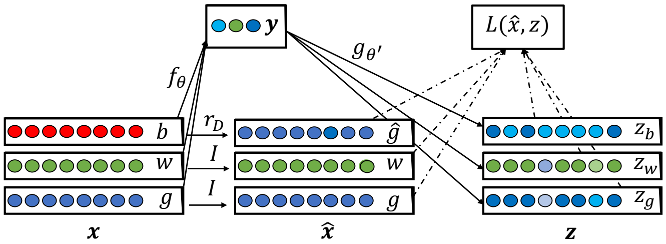

Fig. 4. Replacement Autoencoder: inspired by denoising autoencoder proposed by Vincent *et al.* [5]

B. Replacement Autoencoder

The denoising autoencoder, proposed by Vincent *et al.* [5], is a special kind of autoencoder that receives a corrupted data as input and is trained to predict the original, uncorrupted data as its output. In section IV-A, we described how autoencoder minimizes a loss function $L(\boldsymbol{x}, g(f(\boldsymbol{x})))$ penalizing $g(f(\boldsymbol{x}))$ for being dissimilar from $\boldsymbol{x}$. This encourages $g(f(\boldsymbol{x}))$ to learn to be merely an identity function. A denoising autoencoder instead minimizes $L(\boldsymbol{x}, g(f(\tilde{\boldsymbol{x}})))$, where L is a loss function and $\tilde{\boldsymbol{x}}$ is a copy of $\boldsymbol{x}$ that has been corrupted by some form of noise. Denoising autoencoders must therefore undo this corruption rather than simply copying their input.

As an extension of denoising autoencoder, we introduce *Replacement AutoEncoder* (RAE): a machine learning architecture that can be utilized for transforming time-series data in a way that sensitive information will be replaced with non-sensitive data while desired information will be retained in data. Therefore, the main objective is to train an autoencoder to produce a non-sensitive output from a sensitive input, while keeping other types of input unchanged.

As we see in the figure 4, this is achieved by "replacing each black-listed section $\boldsymbol{b}$ with a gray-listed section $\hat{\boldsymbol{g}}$ by means of a stochastic replacement $r_D(\hat{\boldsymbol{g}}|\boldsymbol{b})$" in the training phase. What exactly the RAE is trying to do is implementing a piecewise function:

$$\boldsymbol{z} = RAE(\hat{\boldsymbol{x}} = r_D(\boldsymbol{x})) = \begin{cases} \boldsymbol{x}^{SR} & \text{if } \boldsymbol{x} \text{ is black-listed} \\ \boldsymbol{x} & \text{otherwise} \end{cases} \tag{2}$$

Where we define $\boldsymbol{x}^{SR}$ as a *safe replacement* for $\boldsymbol{x}$. Ideally, we say a replacement is safe if it does not reveal any information about the data replacement operation. Therefore a replaced vector $\boldsymbol{z}$ is as unsafe as the amount of information it reveals about the data replacement method.

Here, as with the basic autoencoder, input vector $\boldsymbol{x}$ will be mapped to an intermediate discriminative representation $\boldsymbol{y} = \boldsymbol{f_\theta}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$ from which we reconstruct a $\boldsymbol{z} = \boldsymbol{g_{\theta'}}(\boldsymbol{y}) = \boldsymbol{\phi}(\boldsymbol{W}'\boldsymbol{y} + \boldsymbol{b}')$. The key aspect of RAE is that in the training phase, it tries to minimize the average *replacement error* $L(\hat{\boldsymbol{x}}, \boldsymbol{z})$ over the training set

$$\theta^*, \theta'^* = \arg_{\theta,\theta'} \min \frac{1}{n} \sum_{i=1}^{n} L(\hat{\boldsymbol{x}}^{(i)}, \boldsymbol{z}^{(i)}) \qquad (3)$$

where, $\hat{\boldsymbol{x}}$ is a replaced version of $\boldsymbol{x}$, which can be white-listed, black-listed or gray-listed data. Again, $\theta = \{\boldsymbol{W}, \boldsymbol{b}\}$ and $\theta' = \{\boldsymbol{W}', \boldsymbol{b}'\}$ are respectively weight matrices and bias vectors for encoding and decoding parts. RAE generates $\boldsymbol{z}$ as close as possible to the partially replaced input $\hat{\boldsymbol{x}}$. It minimizes the criterion in equation 3 instead of equation 1, therefore, the transformed data, $\boldsymbol{z}$, is a function of $\hat{\boldsymbol{x}}$: the result of a stochastic replacement of $\boldsymbol{x}$ (see Figure 4).

The key idea of the Replacement AutoEncoder is in training phase of autoencoder: "we train RAE in a way that it learns how to transform discriminative features that correspond to sensitive inferences, as shown in Figure 3, into some features that have been more observed in non-sensitive inferences, and at the same time, keeps important features of desired inferences unchanged". Intuitively, RAE can be seen as an algorithm for data transformation where the related functions learned automatically from data provided rather than engineered by experts. Note that, autoencoders only be able to reconstruct data similar to what they have been trained on. For example, an autoencoder trained on running activity data generated by an accelerometer sensor would do a rather poor job of reconstructing drinking or smoking activity data generated by the same sensor, because the features it would learn are just useful for reconstructing data correspond to running or similar activities like walking or jogging.

In section VI we discuss the safety of the RAE outputs. We will show that if adversaries have access to the dataset of original gray-listed data which was used to train the RAE, they can utilize it to train a Generative Adversarial Network [8] for distinguishing between a real gray-listed section of time-series and a transformed one. Therefore, although the recognition of the exact type of sensitive inferences without any side information is still impossible, personal data leak could lead to the reduced safety of replacement operation. Finally, an advantage of RAE is that it automatically learns features. For this reason, it is easy to customize them to produce a personal mediator framework, described in section III, that will perform well on a specific type of input. It doesn't require re-engineering, just appropriate personal data for training. We clarify this process by experimental results provided in the section V-B.

TABLE I
ACTIVITY CLASSES AND OTHER PROPERTIES OF EACH DATASET

	#	*Name of Dataset*		
		Opportunity	**Skoda**	**Hand-Gesture**
Activities	**0**	null	null	null
	1	open door1	write notes	open window
	2	open door2	open hood	close window
	3	close door1	close hood	water a plant
	4	close door2	check front door	turn book
	5	open fridge	open left f door	drink a bottle
	6	close fridge	close left f door	cut w/ knife
	7	open washer	close left doors	chop w/ knife
	8	close washer	check trunk	stir in a bowl
	9	open drawer1	open/close trunk	forehand
	10	close drawer1	check wheels	backhand
	11	open drawer2		smash
	12	close drawer2		
	13	open drawer3		
	14	close drawer3		
	15	clean table		
	16	drink cup		
	17	toggle switch		
Subjects		**4 people**	**1 person**	**2 people**
Sampling Rate		**30 Hz**	**98 Hz**	**32 Hz**
Dimension (d)		**113**	**60**	**15**

V. EXPERIMENTS

A. Datasets

Experiments are conducted on three benchmark datasets: Opportunity [29], Skoda [30], and Hand-Gesture [31].

1) Opportunity: This dataset, famously known as the Opportunity Activity Recognition [2], contains naturalistic human activities recorded in a sensor environment where subjects performed daily morning activities. Two types of recording sessions were performed: Drill sessions where the subject performs sequentially a pre-defined set of activities and "activity of daily living" runs (ADL). Each record in the dataset comprises 113 sensory readings and the sampling rate of the sensor signals is 30 Hz [32]. We use Drill and the first four sets of ADLs as the training data, and use the fifth set of ADL as the testing data. The data is composed of the recordings of 4 subjects and there are 18 gestures classes in this activity recognition task (Table I). The null class refers to the either non-relevant activities or non-activities. A large amount of data in this dataset matches the gray-listed *Null* class (about 62%). For this reason it does not present much of a challenge to the RAE. We make the task artificially harder by reducing the amount of gray-listed data by removing a high percentage of the null data. We randomly remove some instances of this class to obtain more realistic results. Therefore, we keep only 25% of the instances of data matching the Null class to make the problem harder for RAE.

2) Skoda: This dataset describes the activities of assembly-line workers in a car production environment[3]. They consider the recognition of 11 activity classes performed in one of the quality assurance checkpoints of the production plant (Table I). In their study, one subject wore

[2] http://www.opportunity-project.eu

[3] http://www.ife.ee.ethz.ch/research/activity-recognition-datasets.html

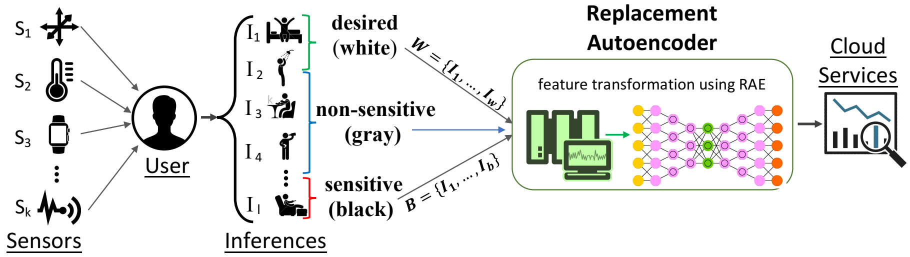

Fig. 5. Replacement Platform: we assume each user knows their list of sensitive inferences and each cloud services announces their list of desired inferences. Based on defined inferences lists, RAE learns how to transform time-windows that correspond to sensitive inferences before sending them out.

19 3D accelerometers on both arms and perform a set of experiments using sensors placed on the two arms of a tester (10 sensors on the *right* arm and 9 sensor on the *left* arm). The original sample rate of this dataset was 98 Hz, but it was decimated to 30 Hz for comparison purposes with the other two datasets. The Skoda dataset has been employed to evaluate deep learning techniques in sensor networks [7], which makes it a proper dataset to evaluate our proposed framework.

3) Hand-Gesture: In this dataset, sensory data about recognizing hand gestures from body-worn accelerometers and gyroscopes is recorded[4]. There are two subjects perform hand movements with 8 gestures in regular daily activity and with 3 gestures in playing tennis, so in total, there are 12 classes (Table I). Every subject repeated all activities about 26 times. Similar to the other datasets, the null class refers to the periods of no specific activity. The sampling rate of dataset is 32 Hz and each record has 15 real valued sensor readings in total.

For all datasets, we fill up the missing values using linear interpolation and normalize each sensor's data to have zero mean and unit standard deviation. For Skoda and Hand-Gesture datasets, we randomly select 80% of time-windows for training phase and the remaining 20% is used for the tests.

B. Experimental Settings

We consider Human Activity Recognition problem (HAR) and a setting in which there are a list of $|c|$ different inference classes: $I = \{I_1, ..., I_i, ..., I_j, ..., I_c\}$. We divide inferences into three distinct categories:

1) *desired* activities: $I_w = \{I_1, I_2, ..., I_i\}$
2) *sensitive* activities: $I_b = \{I_{i+1}, I_{i+2}, ..., I_j\}$
3) *non-sensitive* activities: $I_g = \{I_{j+1}, I_{i+2}, ..., I_c\}$

where $i < j$. We assume when a service is requested from third parties, they announce the list of required inferences to be drawn to make the service works properly. We put these inferences (activities) in the list of desired inferences I_w. Moreover, we assume users know what kind of inferences are non-sensitive for them. Therefore, users announce a list of non-sensitive inferences I_g (e.g. *null* classes in datasets). In our experiments, we put the remaining inferences into sensitive inferences I_b. Thus, in our experiments we utilize datasets to train RAE in this way:

(a) Following [6], we consider a sliding window with size d and step size w to segment the time-series of k sensors into a set of sections. Specifically, a two-dimensional matrix containing k rows of time-series of which each row consists of d data points. For all datasets we set $d = 30$ and $w = 3$. Moreover, the activity label of each sliding window is assigned by the most-frequently appeared label in all d records. In this way, we have a training dataset contains labeled sections of time-series. We divide training dataset into three disjoint sets, by considering I_b, I_w, and I_g in each experiment.

(b) We train the RAE for performing the *feature-based replacement*. As we said in section IV, autoencoders try to learn an approximation to the identity function, so as to output $\boldsymbol{z}$ that is similar to input $\boldsymbol{x}$. Thus, for implementing the idea of Replacement AutoEncoder, we set $X = W \cup G \cup B$ and $Z = W \cup G \cup \hat{G}$, where $X = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$ is input data in the training phase of RAE, and $Z = \{z^{(1)}, z^{(2)}, \dots, z^{(n)}\}$ is the corresponding output. W, G, and B are respectively data instances corresponding to white-listed, gray-listed, and black-listed inferences. $\hat{G}$ is a randomly selected subset of G sections with the same size with the B. By replacing B with $\hat{G}$, RAE learns how to replace sections of a time-series that contain sensitive data with non-sensitive data (implementation of equation 3).

(c) We implemented RAE using Keras[5] framework. The activation function for input and output layers is $linear$ and for all of the hidden layer is Scaled Exponential Linear Unit ($selu$) [33]. Since we normalize data to have a zero mean and unity standard deviation, $selu$ does not filter out negative numbers, thus it is a better choice than Rectified Linear Unit ($relu$). The loss function in training phase is chosen to be Mean Squared Error (mse), because we want to retain the overall structure of reconstructed sections. The dimension of RAE input is $inp = [\ k \times d, 1\ ]$, and it has five hidden layers with size $inp/2, inp/8, inp/16, inp/8, inp/2$ respectively (except for Hand-Gesture dataset, which

[4] https://github.com/andreas-bulling/ActRecTut

[5] https://keras.io/

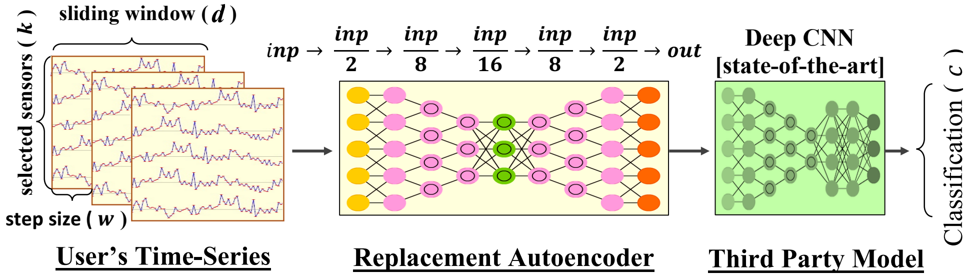

Fig. 6. A stacked autoencoder as a mediator : in all experiments in this paper, $w = 3$ and $d = 30$, and k depends on the number of sensors in each dataset.

TABLE II
F_1 SCORES FOR SKODA DATASET (IN ALL THE TABLES, OF_1 STANDS FOR ORIGINAL DATA AND TF_1 STANDS FOR TRANSFORMED ONE)

Hand	List of Inferences (Table I)	OF_1	TF_1
Left	$I_w = \{4, 8, 9, 10\}$	97.92	96.32
	$I_b = \{1, 5, 6, 7\}$	**96.24**	**0.00**
	$I_g = \{0, 2, 3\}$	94.34	93.42
Left	$I_w = \{2, 3, 5, 6, 7, 9\}$	96.52	93.23
	$I_b = \{4, 8, 10\}$	**97.88**	**0.00**
	$I_g = \{0, 1\}$	93.86	94.85
Right	$I_w = \{1, 4, 10\}$	97.56	94.9
	$I_b = \{2, 3, 8, 9\}$	**97.97**	**0.00**
	$I_g = \{0, 5, 6, 7\}$	92.33	88.23
Right	$I_w = \{2, 3, 5, 6, 7, 9\}$	95.76	91.06
	$I_b = \{4, 8, 10\}$	**97.39**	**0.00**
	$I_g = \{0, 1\}$	94.31	92.39

TABLE III
F_1 SCORES FOR HAND-GESTURE DATASET

Subject	List of Inferences (Table I)	OF_1	TF_1
#1	$I_w = \{1, 2, 3, 4, 9, 10, 11\}$	94.11	90.15
	$I_b = \{5, 6, 7, 8\}$	**95.75**	**0.26**
	$I_g = \{0\}$	95.04	96.54
#1	$I_w = \{1, 3, 4, 5, 6, 7\}$	95.23	90.45
	$I_b = \{2, 8, 9, 10, 11\}$	**94.53**	**0.62**
	$I_g = \{0\}$	95.04	97.46
#2	$I_w = \{1, 3, 4, 5, 6, 7, 8\}$	97.21	93.30
	$I_b = \{2, 9, 10, 11\}$	**92.54**	**0.71**
	$I_g = \{0\}$	95.89	97.53
#2	$I_w = \{2, 3, 5, 6, 7, 9\}$	96.10	92.13
	$I_b = \{4, 8, 10\}$	**96.96**	**0.52**
	$I_g = \{0, 1\}$	95.70	97.56

is $inp/2, inp/3, inp/4, inp/3, inp/2$, because of low dimensionality of data). All the experiments have been performed on 30 epochs with batch size 128. (see Figure 6)

To evaluate the performance of time-series transformation by RAE, we implemented one of the best methods for HAR that have been proposed recently [6] as a third party model. They build a Convolutional Neural Network to investigate the multichannel time-series data. We compare the performance of third party model on transformed time-series with original ones and show that utility will retain for all of the activities included in white-listed inferences, while recognizing an activity correspond to black-listed inferences is near impossible. Both the original and transformed time-series are given to the third party model for classification task and

TABLE IV
F_1 SCORES FOR OPPORTUNITY DATASET

Subject	List of Inferences (Table I)	OF_1	TF_1
#1	I_w={9,10,11,12,13,14,15,16,17}	71.75	64.32
	I_b={1,2,3,4,5,6,7,8}	**79.15**	**0.21**
	I_g={0}	88.93	89.70
#1	I_w={1,2,3,4,5,6,7,8,15,17}	76.87	75.93
	I_b={9,10,11,12,13,14}	**71.49**	**1.32**
	I_g={0,16}	84.44	82.08
#3	I_w={9,10,11,12,13,14,16}	74.92	77.07
	I_b={1,2,3,4,15,17}	**76.16**	**0.92**
	I_g ={0,5,6,7,8}	84.98	81.58
#3	I_w={1,2,3,4,5,6,7,8,15,17}	70.32	65.05
	I_b={9,10,11,12,13,14,16}	**74.92**	**6.31**
	I_g={0,1}	93.72	92.95

F_1 scores are calculated in Table II, Table III, and Table IV. The results show that utility will retain for desired activities, while recognizing sensitive activities is near impossible. The most valuable result is that third party's model misclassifies all transformed sections correspond to black-listed inferences (B) into the gray-listed inferences (G). Therefore, the false-positive rate on white-listed inferences is very low (see Figure 7).

C. Visualization

It is a very challenging task to visualize multi-channel time-series, especially in our high-dimensional setting. In this section, in order to better understand the idea of Replacement AutoEncoder, we present and discuss some perceptible examples.

The MNIST database of handwritten digits [34], is the most famous dataset for evaluating learning techniques and pattern recognition methods. It contains $60,000$ training images and $10,000$ testing images. As an explicit example, in our setting, we consider 0 as a kind of non-sensitive inference (gray-listed), other even numbers $(2, 4, 6, 8)$ as a set of sensitive inferences (black-listed), and all of the odd numbers $(1, 3, 5, 7, 9)$ as desired information we want to keep unchanged. Figure 8 shows the output of the RAE when it receives some samples from test set as input. RAE transformed all of the images categorized as sensitive information into images that are undoubtedly recognized as 0 by humans and also image recognition algorithms. This is just a simple example to build an intuitive sense of how our method can learn the piecewise function described in equation 2 to selectively censor sensitive information in personal data.

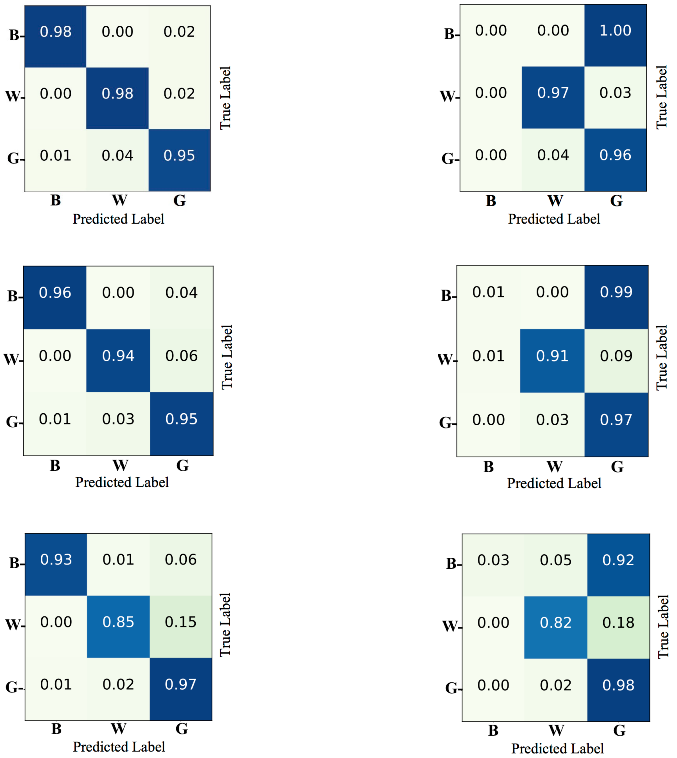

Fig. 7. Confusion Matrix: (Left) Original time-series . (Right) Transformed time-series. It shows that, after transformation, almost all of the black-listed activities (B) are recognized as gray-listed ones (G). (Top) Results relate to Skoda dataset (left hand) for these lists: $I_W = \{2,3,5,6,7,9\}$, $I_B = \{4,8,10\}$, $I_G = \{0,1\}$. (Middle) Results relate to Hand-Gesture dataset (subject 1) for these lists: $I_W = \{1,2,3,4,9,10,11\}$, $I_B = \{5,6,7,8\}$, $I_G = \{0\}$. (Bottom) Results relate to Opportunity dataset (subject 1) for these lists: $I_W = \{1,2,3,4,5,6,7,8,15,17\}$, $I_B = \{9,10,11,12,13,14\}$, $I_G = \{0,16\}$

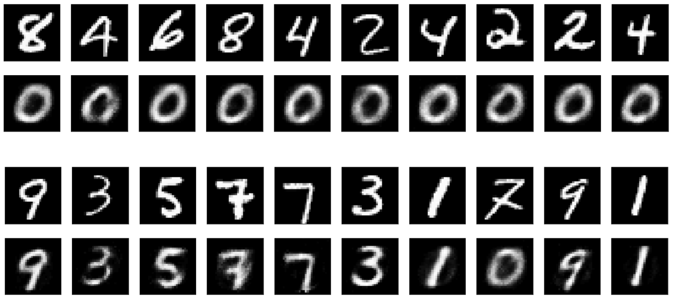

Fig. 8. Outputs of *replacement autoencoder* in MNIST handwritten digits [34]: (Top) Considering 0 as *non-sensitive* (gray-listed) and information other even numbers as *sensitive* (black-listed) ones. (Bottom) Considering all the odd numbers as *desired* information.

For visualizing time-series data which are used in the experiments, we implement t-Distributed Stochastic Neighbor Embedding (t-SNE) [35]. At the moment, it is one of the most common algorithm for 2D visualization of high-dimensional datasets. t-SNE is a technique for dimensionality reduction and a well known strategy for visualizing similarity relationships, between different classes in the dataset, in high-dimensional data. It uses PCA to compress high-dimensional data (e.g. 1800 for Skoda dataset [30]) into a low-dimensional space (e.g. 100 dimensional), then maps the compressed data to a 2D plane that can be plotted easily. Since t-SNE cares about preserving the local structure of the high-dimensional data, we use it to describe how the RAE would push sensitive data points, in the original data space, into areas that correspond to non-sensitive inferences, and at the same time preserve the separability and structures of desired data points [6].

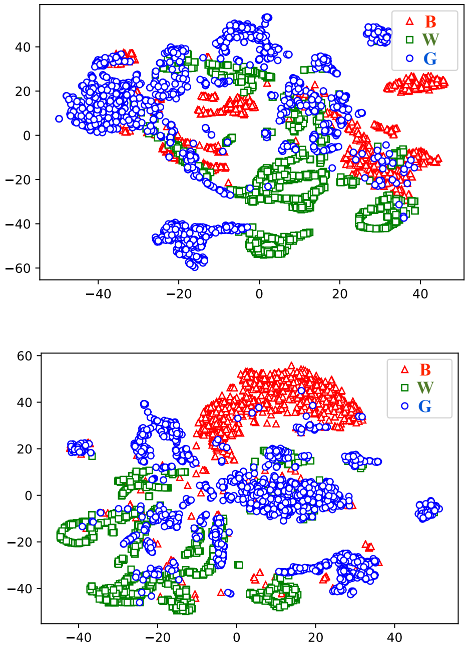

Fig. 9. The effect of applying feature-based transformation using replacement autoencoder on time-series: feature space relationships among different classes of activities for Skoda dataset (first row of table II). (Top) t-SNE on original time-series. (Bottom) t-SNE on transformed time-series.

Figure 9 shows the feature space relationships among different classes of activities for Skoda dataset. Regarding the scatter plots depicted in Figure 9, black-listed data points in original time-series (top plot) have their own structures and clearly separable from other classes. But, in the transformed version of time-series (bottom plot) all of the black-listed data points have been moved into another area of feature-space which is more closely related to gray-listed data points. They also lose their recognizable patterns and all of them mapped to almost the same area of the feature space. On the other hand, it maintains particular patterns of white-listed data points, thus causes no harm to the recognition accuracy of desired activities. Note that in this example, we reduce the dimensionality from 1800-d to just 2-d, which leads to huge information loss. t-SNE has two tunable parameters, *initial dimension* (in this experiment it is equal to 100), and *perplexity* (here is 60). Perplexity is a measure for information that is defined as 2 to the power of the Shannon entropy.

[6] Using codes implemented in https://lvdmaaten.github.io/tsne

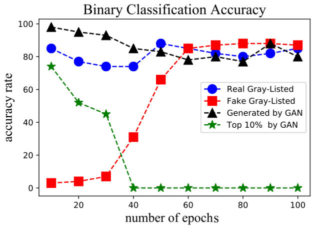

Fig. 10. GAN's accuracy in distinguishing different kinds of real, fake and generated sections (the Skoda dataset - first row of table II).

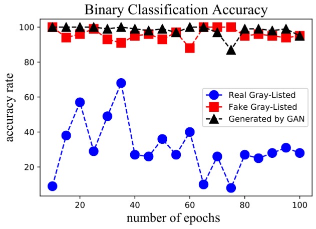

Fig. 11. GAN's accuracy in detecting real gray-listed data by having access to other users' data (the Opportunity dataset - first row of table IV).

VI. Threat Model

We assume adversaries know the algorithm which is used for transforming time-series, they have access to the user's transformed time-series data, and also they have access to a dataset contain data corresponding to all the real gray-listed data which the user uses to train his/her model. If an attacker can determine when gray-listed data from RAE is "fake" (replacing black-listed data) then RAE is no more useful than filtering. Hence it is important for us to show that to what extent the replacement gray-listed data that RAE writes over black-listed data is indistinguishable from the real gray-listed data output by RAE.

For this purpose, we use a Generative Adversarial Network (GAN) [8]. GANs are neural networks that learn to create synthetic data similar to some known input data. GANs consist of two models: a *generative* model G and a *discriminative* model D. The model D is basically a binary classifier that determines whether a given input looks like a real input from the dataset or like an artificially created data. The model G takes random input values and uses feedback from the classifier D to produce convincing data that the classifier D can't distinguish from real data. Thus, we use the GAN for evaluating the output of the RAE as follows.

First, we assume that adversaries will try to detect whether the current sections, recognized as a non-sensitive activity, are real or fake. Thus, they train a GAN on the obtained gray-listed dataset, based on the code implemented in [36]. In this GAN, G learns to produce sections very similar to real gray-listed sections and D learns to determine whether the given sections, recognized as non-sensitive data, are real gray-listed sections or fake ones. Second, after training the GAN, we separate the discriminative part D and give it as input: (i) *real* gray-listed sections, (ii)*fake* gray-listed sections, (iii)*randomly generated* gray-listed sections by G, and (iv)*the top 10% randomly generated* gray-listed sections as rated by discriminator D. We measure binary classification accuracy rate for these four categories. The results are shown in Figure 10.

The results demonstrate that the output of the RAE is almost as similar as the random data generated by G but not quite as good as the best data generated by G. We see that when we give D more time to learn, it eventually distinguishes between real gray-listed sections and fake ones. Therefore, if the user's non-sensitive data which is used to train his/her model is leaked out, it will be easy for adversaries to distinguish between real and fake non-sensitive data. Thus, in this situation, the safety of our proposed replacement method will be reduced to the safety of filtering approach.

We also conducted another experiment to investigate whether the privacy of user i can be compromised by having access to the original gray-listed data of another user j. Figure 11 shows the accuracy of the binary discriminator D for distinguishing between real and fake gray-listed data of subject #3 when D have been trained on gray-listed data from subject #1. Results show that D cannot recognize real gray-listed data as real, and its error rate is about 70%. Though it recognizes all fake gray-listed data as fake, it is not valuable because it recognizes real data as fake as well. Thus, we see that a GAN can be a good discriminator only if adversaries gain access to real data from the attacked user and data from other users is not sufficient. Note that in this experiment the list of black-listed and gray-listed inferences is the same for both data subjects.

We can see from Figure 10 that the GAN performance in classifying data began to improve slowly from epoch 10 to epoch 30 and stabilized by epoch 70. By contrast even after 100 epochs in Figure 11 there is no consistent pattern of improvement for the GAN without access to the individuals gray-listed data and it seems unlikely that the GAN will ever learn to distinguish this data.

VII. Discussion and Related Work

Historically, Westin [37] has defined *information privacy* as "the right to select what personal information about me is known to what people". Recently, Ziegeldorf et al. [4] extends this definition for *IoT privacy* by "having individual control over the processing of personal information and awareness of the subsequent use of them". Here, we discuss the recent progress in the protection of users' privacy in time-series data analysis and compare them with our approach.

Anonymization. In the past decade, there has been some remarkable progress in the field of preserving privacy in tabular data publishing, such as *k-anonymity* [38] and *l-diversity* [39]. These approaches, more known as *anonymization* methods, mainly protect the identity of the data owners by coarse-graining methods. They try to retain essential information for analysis but suppress or generalize other identity attributes. In this paper, instead of anonymity, we consider a situation where untrusted third parties know users' identity and we introduce a method which prevents inferring sensitive information from their personal time-series.

Randomization. An approach for protecting sensitive inferences is *data randomization* which perturbs data by multiplicative or additive noise, or other randomized processes. Zhang et al. [40] developed a time-series pattern based noise generation strategy (called noise obfuscation) for privacy protection on the cloud. It generates and injects noise service requests into real ones to ensure that their occurrence probabilities are about the same so that service providers cannot distinguish which requests are real ones. To protect the user from inference attacks on his private data, Erdogdu et al. [41] proposed a method that sequentially randomizes samples from the time-series prior to their release according to a stochastic process, called the privacy mapping. For example, a household is willing to give the aggregate electrical load to the service provider, but wishes to keep the information related to their eating patterns private, in particular the microwave usage which can also be inferred from the aggregate load. Allard et al. [42] proposed Chiaroscuro, a solution for clustering personal time-series that are distributed on personal devices with privacy guarantees. Chiaroscuro allows the participating devices to collaborate privately by combining encryption with differential privacy [11]. Generally, because of high temporal and spatial granularity of time-series and strong correlation between their samples, when general data randomization approaches are extended to the dynamic case of time-series data, scalability challenges arise and maintaining utility often becomes challenging [41]. For this reason, we focus on a scalable method for effectively hiding sensitive information by only replacing sensitive sections of time-series without perturbing the desired and non-sensitive data.

Deep Learning Frameworks: Liu et al. [43] proposed a collaborative privacy-preserving deep learning system in mobile environment, which enables multiple sites to learn deep learning model only by sharing partial parameters with each other. In their approach, the privacy is protected by keeping data in local and use a parameter selection mechanism to only share small fraction of the parameters to the server at each round. Phan et al. [9] enforce ϵ-differential privacy by injecting noise into the objective functions of the deep autoencoders at every layer and training step. By considering the need to protect sensitive inferences, we introduce a novel algorithm for feature learning which can be utilized for real-time privacy-preserving time-series publishing. It uses a new objective function for learning a deep autoencoder by automatically extraction of discriminative features from input time-series .

Discussion: Here we discuss some important advantages of RAE for privacy-preserving analysis of sensory data.

- *No sensitive activities can be inferred*: RAE is trained to transform blacklisted sections into the same-size sections that are very similar to gray-listed activates. Therefore, on the cloud side, only non-sensitive gray-listed activities can be inferred.
- *No loss in utility*: autoencoder transformed white-listed sections with the least amount of changes, because it is trained to copy these sections as exactly as possible. In addition, it will learn to transform both gray-listed and black-listed activities to gray-listed ones. So, the amount of false positive (wrongly recognizing an activity as white-listed) is near to zero. This is very important to the quality of services provided to the user.
- *No detection of sensitive intervals*: in *noise addition* and *filtering*, third parties can easily detect time intervals corresponding to sensitive activities. They can use this knowledge alongside other side information to conclude about the type of related activity. In our feature-based replacement approach it is very hard to distinguish between sensitive and non-sensitive time interval, unless adversaries get access to the original gray-listed data which have used for training RAE.
- *No hand selected feature set*: in the *mapping* (feature publishing) approach we need to go through the reduced representation of a time-series and divide them into white-listed and black-listed feature sets. In feature-based replacement we publish transformed version of time-series with the same dimensionality of original data and have not concerned about separating features. In fact RAE can automatically transform features correspond to black-listed data while retaining the features correspond to white activities unchanged.

VIII. Conclusions and Future Directions

The protection of users' privacy in time-series analysis is a very challenging task, especially when time-series are gathered from different sources. Invasion of privacy arises when disseminated data can be used to draw sensitive inferences about the data subject. In this paper, we have focused on the inference privacy and considered time-series generated by sensors embedded into mobile and wearable devices. We have introduced Replacement AutoEncoder: a feature learning algorithm which learns how to transform discriminative features that correspond to sensitive inferences, into some features that have been more observed in non-sensitive data, and at the same time, keeps important features of desired inferences unchanged to benefit from cloud services. We evaluate the efficacy of the method with an activity recognition task using experiments on three benchmark datasets and show that it can preserve the privacy of sensitive information while simultaneously retaining the recognition accuracy of state-of-the-art techniques .

We believe that the approach of learning privacy-related features from time-series data will enable us to develop efficient methods for data transformation and help us to enrich existing IoT platform with a robust privacy-preserving time-series data analytics component. Thus, in the future, we will focus on extending the mediator framework by letting users dynamically define their personal privacy policies and inferences that applications are allowed to access. We will also investigate theoretical frameworks and other mathematical tools to determine a bound on sensitive information that remains in the transformed time-series. Beside GANs, we will pay attention to other possible attacks and appropriate responses to them as well as comparing the achieved utility-privacy tradeoff of replacement with other privacy approaches, such as randomization or filtering. Another direction for continuing this research could be using Long Short Term Memory (LSTMs) networks [44] which are capable of learning long-term dependencies in data to capture the discriminative features of time-series.

Acknowledgment

This work was kindly supported by the Life Sciences Initiative at Queen Mary University London and a Microsoft Azure for Research Award. Hamed Haddadi was partially funded by EPSRC Databox grant (Ref: EP/N028260/1).

References

[1] R. Spreitzer, "Pin skimming: Exploiting the ambient-light sensor in mobile devices," in *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*. ACM, 2014, pp. 51–62.

[2] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "Accessory: password inference using accelerometers on smartphones," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM, 2012, p. 9.

[3] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic," *DAT'16*, 2016.

[4] J. H. Ziegeldorf, O. G. Morchon, and K. Wehrle, "Privacy in the internet of things: threats and challenges," *Security and Communication Networks*, vol. 7, no. 12, pp. 2728–2742, 2014.

[5] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th ICDM*. ACM, 2008, pp. 1096–1103.

[6] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition." in *IJCAI*, 2015, pp. 3995–4001.

[7] F. J. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.

[9] N. Phan, Y. Wang, X. Wu, and D. Dou, "Differential privacy preservation for deep auto-encoders: an application of human behavior prediction." in *AAAI*, 2016, pp. 1309–1316.

[10] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[11] C. Dwork, "Differential privacy: A survey of results," in *TAMC*. Springer, 2008, pp. 1–19.

[12] Y. Nam, Y. Kim, and J. Lee, "Sleep monitoring based on a tri-axial accelerometer and a pressure sensor," *Sensors*, vol. 16, no. 5, p. 750, 2016.

[13] M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, "Touchsignatures: identification of user touch actions and pins based on mobile sensor data via javascript," *Journal of Information Security and Applications*, vol. 26, pp. 23–38, 2016.

[14] S. Chakraborty, K. R. Raghavan, M. P. Johnson, and M. B. Srivastava, "A framework for context-aware privacy of sensor data on mobile systems," in *Proceedings of the 14th Workshop on HotMobile*, ser. HotMobile '13. New York, NY, USA: ACM, 2013, pp. 11:1–11:6. [Online]. Available: http://doi.acm.org/10.1145/2444776.2444791

[15] N. Saleheen, S. Chakraborty, N. Ali, M. M. Rahman, S. M. Hossain, R. Bari, E. Buder, M. Srivastava, and S. Kumar, "mSieve: differential behavioral privacy in time series of mobile sensor data," in *Proceedings of the ACM UbiComp*, 2016, pp. 706–717.

[16] S. Shen, H. Wang, and R. Roy Choudhury, "I am a smartwatch and i can track my user's arm," in *Proceedings of the 14th MobiSys*. ACM, 2016, pp. 85–96.

[17] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, "Random-data perturbation techniques and privacy-preserving data mining," *Knowledge and Information Systems*, vol. 7, no. 4, pp. 387–414, 2005.

[18] Y.-S. Moon, H.-S. Kim, S.-P. Kim, and E. Bertino, "Publishing time-series data under preservation of privacy and distance orders," in *International Conference on Database and Expert Systems Applications*. Springer, 2010, pp. 17–31.

[19] H. Wang and Z. Xu, "CTS-DP: Publishing correlated time-series data via differential privacy," *Knowledge-Based Systems*, vol. 122, pp. 167–179, 2017.

[20] M. Götz, S. Nath, and J. Gehrke, "MaskIt: Privately releasing user context streams for personalized mobile applications," in *Proceedings of the ACM SIGMOD*, 2012, pp. 289–300.

[21] S. Chakraborty, K. R. Raghavan, M. B. Srivastava, C. Bisdikian, and L. M. Kaplan, "Balancing value and risk in information sharing through obfuscation," in *IEEE FUSION, 15th International Conference on*, 2012, pp. 1615–1622.

[22] F. Laforet, E. Buchmann, and K. Böhm, "Individual privacy constraints on time-series data," *Information Systems*, vol. 54, pp. 74–91, 2015.

[23] S. A. Osia, A. S. Shamsabadi, A. Taheri, H. R. Rabiee, N. Lane, and H. Haddadi, "A hybrid deep learning architecture for privacy-preserving mobile analytics," *arXiv preprint arXiv:1703.02952*, 2017.

[24] H. Haddadi, H. Howard, A. Chaudhry, J. Crowcroft, A. Madhavapeddy, D. McAuley, and R. Mortier, "Personal data: thinking inside the box," in *Proceedings of The Fifth Decennial Aarhus Conference on Critical Alternatives*. Aarhus University Press, 2015, pp. 29–32.

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[26] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[27] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[28] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[29] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán, and D. Roggen, "The opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 2033–2042, 2013.

[30] P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Troster, "Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection," *Lecture Notes in Computer Science*, vol. 4913, p. 17, 2008.

[31] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys*, vol. 46, no. 3, pp. 33:1–33:33, 2014. [Online]. Available: https://github.com/andyknownasabu/ActRecTut

[32] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha *et al.*, "Collecting complex activity datasets in highly rich networked sensor environments," in *INSS, Seventh International Conference on*. IEEE, 2010, pp. 233–240.

[33] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-Normalizing neural networks," *NIPS*, 2017.

[34] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[35] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[36] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[37] A. F. Westin, "Privacy and freedom," *Washington and Lee Law Review*, vol. 25, no. 1, p. 166, 1968.

[38] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[39] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," in *IN ICDE*, 2006.

[40] G. Zhang, X. Liu, and Y. Yang, "Time-series pattern based effective noise generation for privacy protection on cloud," *IEEE Transactions on Computers*, vol. 64, no. 5, pp. 1456–1469, 2015.

[41] M. A. Erdogdu, N. Fawaz, and A. Montanari, "Privacy-utility trade-off for time-series with application to smart-meter data," in *AAAI 2015 Workshop on Computational Sustainability*, 2015.

[42] T. Allard, G. Hébrail, F. Masseglia, and E. Pacitti, "Chiaroscuro: Transparency and privacy for massive personal time-series clustering," in *Proceedings of the ACM SIGMOD*, 2015, pp. 779–794.

[43] M. Liu, H. Jiang, J. Chen, A. Badokhon, X. Wei, and M.-C. Huang, "A collaborative privacy-preserving deep learning system in distributed mobile environment," in *CSCI*. IEEE, 2016, pp. 192–197.

[44] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.