kUBI: A Framework for Privacy and Transparency in Sensor-Based Business Models for Consumers: A Pay-How-You-Drive Example

Christian Roth[0000−0002−1668−5441], Mario Saur, and Dogan Kesdogan

University of Regensburg
Regensburg, Germany
<firstname.lastname>@ur.de

Abstract. Ubiquitous computing has fundamentally redefined many existing business models. The collected sensor data has great potential, which is being recognized by more and more industries, including car insurance companies with Usage-Based Insurance (UBI). However, most of these business models are very privacy-invasive and must be constructed with care. For a data processor, the integrity of the data is particularly important. With *kUBI* , we present a framework that takes into account the interests of the providers as well as the privacy of the users, using the example of Android. It is fully integrated into the Android system architecture. It uses hybrid data processing in both stakeholder domains. Protected enclaves, whose function can be transparently traced by a user at any time, protect company secrets in the hostile environment, i.e. a user's smartphone. The framework is theoretically outlined and its integration into Android is shown. An evaluation shows that the user in the exemplary use case UBI can be protected by *kUBI* .

Keywords: Privacy Enhancing Technology · Transparency Enhancing Technology · Sensor Data · Smartphone · Privacy Framework.

1 Introduction

Ubiquitous computing fundamentally changes our understanding of existing business models. Personal-agents such as smartphones are kept close to a user. They collect sensor data which are of great value for companies because they enable user-specific products. Benndorf & Normann found that the *readiness-to-share* this data is related to monetary aspects [5]. Threats to privacy are knowingly or unknowingly disregarded [13]. As a result, the majority of users are willing to disclose comprehensive information about their digital identity [14]. The loose nature of the data exchange may also be related to the *nothing-to-hide* mentality of the users; as long as no negative consequences are feared, the willingness to share data increases or unjustified data flows are accepted [21]. The consequences of data protection violations are often unclear from the outside, so that this danger is sometimes disregarded. This can also be seen in the lack of awareness when dealing with (sensitive) data. Felt et al. [9] show that the majority of users accept Android permissions requested by applications without question. Permissions once granted are generally not revoked [14]. This can lead to a dangerous spiral, which is exacerbated by observing the results of Weydert et al. [24]: The willingness to share data continues to grow as users are given the opportunity to actively participate in and control the data process. Thus, a Transparency Enhancing Technology (TET) is essential for users to understand which data is shared with a foreign party and for what reason.

Another serious threat is the lack of awareness of privacy attack vectors in smartphones. This is serious, since smartphones have broad tracking capabilities and are part of the private sphere. It is known that sensors such as cameras, microphones or device memory are sensitive [18]. However, literature shows that even on the basis of so-called zero-permission sensors like accelerometers or gyroscopes, which belong to the Inertial Measurement Unit (IMU) family, attacks on privacy are possible. A IMU produces Floating Phone Data (FPD) which are mandatory for sensor-based business models. This motivates the need for a Privacy Enhancing Technology (PET) which is suitable to protect the data of the users.

Contribution In this paper, we propose a PET/TET in form of a holistic framework for balancing privacy and integrity. It takes into account the diverse and conflicting interests of the stakeholders involved. In fact, we

- present threats related to sensor data in smartphones and place them in the context of Usage-Based Insurance (UBI),
- present our comprehensive framework *kUBI* that takes into account the various demands of the stakeholders,
- show how to embed *kUBI* into Android to enable privacy-friendly Pay-How-You-Drive (PHYD), and
- use an existing attack from previous work [20] to show that the proposed procedure can increase privacy and integrity in PHYD.

Structure The remainder of this paper is organized as follows. In Section 2, we list related work and present some basics for this work. Afterwards, Section 4 presents the UBI scenario including the stakeholders. A short insight into the creation of sensor data using Android is given in Section 3. Subsequently, the framework *kUBI* is described in detail and integrity and privacy aspects are considered. An evaluation (c.f. Section 6) shows on the basis of an existing attack and real data that *kUBI* can protect privacy. We conclude the paper in Section 7.

2 Related Work

It is obvious that the current permission system in Android is not sufficient in certain cases [6]. Therefore, there is active research to foster understanding of sensor data usage (e.g. [4]) or to provide protection mechanisms (e.g. [3,7]). Many of the approaches either block sensor access altogether or introduce blurring elements into the collected data to prevent sensor misuse. In the context of many sensor-based business models, this is not a practicable way, because the data lose their meaningfulness. A correct evaluation is no longer possible. Policy-driven systems must support multiple stakeholders [6] to be applicable in the given context. However, a policy might contain sensitive information, which has to be protected. This is a dilemma, since the policy is issued by a business, but for reasons of privacy, it is to be enforced in the domain of the user, without the user knowing the contents.

Furthermore, the right conclusions have to be derived from the data for many business models to be usable. Therefore, Privacy Preserving Data Mining (PPDM) techniques have to be found. In their survey, Hong et al. [12] analyze multiple perturbation methods for time series, although not in the context of UBI. Roth et al. [20] motivated the need for a new privacy-enhanced framework to enable trust and privacy in sensor-based business models, because it was shown that existing PPDM techniques such as anatomization, permutation, and perturbation are insufficient in this area.

When focusing on the context of UBI, there is little work, although these kind of insurance models are on the rise. Troncoso et al. [22] already proposed a privacy-enhanced model for Pay-As-You-Drive (PAYD). However, PAYD is significantly different to PHYD, because here, it is often only the location that is problematic [10]. Thus, the need for privacy-enabled PHYD models still exists, since PHYD-enabled rates are common in the UBI business model.

3 Mobile Application Environment

This section gives a brief overview of sensor data processing.

3.1 Android Sensor Stack

Modern smartphones offer a broad range of sensors. Built-in sensors may measure motion, orientation, and environmental conditions. They are accompanied by virtual sensors which deliver preprocessed data. In the context of mobility, data from a smartphone is often called Floating Phone Data (FPD).

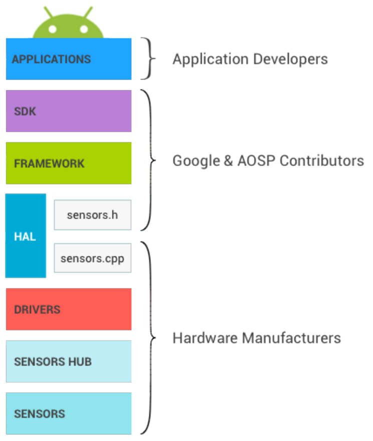

Fig. 1: Layers of the Android Sensor Stack [2].

We present the sensor stack at the example of Android (Fig. 1; c.f. [2]). Application developers interact with the Android OS via the SDK which represents a high level view of the sensors. The Framework links multiple applications to the Hardware Abstraction Layer (HAL). It introduces multiplexing, enabling multiple applications to access a sensor at the same time. Virtual sensors are also created within this layer. The HAL is the link between Android and a hardware manufacturer's implementation for a concrete sensor. It follows a well defined interface (sensors.h). Lower layers are in the sole responsibility of hardware manufacturers and may be closed source. Android itself is unable to alter any sensor behavior unless defined in sensors.h. For security reasons, the stack is organized bottom-up, i.e. higher levels cannot send data to lower instances. Multiple applications reading data from the same sensor do not interfere. It is simple to register to sensor readings. First, an Android-wide SensorManager gives access to the SensorService which in turn can be used to access various sensors. Then, a SensorEventListener can be used to handle sensor updates. The following example queries the accelerometer and is updated everytime a new SensorEvent is acquired.

```
val sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
val accelerometerSensor: Sensor? = ↩
      sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
val sensorListener = object: SensorEventListener {
    override fun onAccuracyChanged(sensor: Sensor, accuracy: Int) { }
    override fun onSensorChanged(event: SensorEvent) {
        val dT: Long = event.timestamp
        val axisX: Float = event.values.get(0)
        val axisY: Float = event.values.get(1)
        val axisZ: Float = event.values.get(2)
        // further process values
    }

}
sensorManager.registerListener(sensorListener, accelerometerSensor, ↩
      SensorManager.SENSOR_DELAY_FASTEST)
```

3.2 Attacks on Sensor Data

Both prominent mobile operating systems, Android and iOS, protect specific sensors using a sophisticated permission system. These permissions control which app is allowed to use the information by a sensor. However, as of the current Android 11, some sensors are not protected by the user-controlled permission system, which indicates a threat for privacy. A corrupt application can read the sensor data for further processing without the user noticing and never asking for permission. It may be used to gather the needed sensor data. The privacy violation facing the user solely using sensor data from an IMU is extensive:

- **Identification attacks** The first class of attacks wants to identify a driver from a set of drivers (e.g. [17]). In some cases, a learning phase is necessary so that the attack can name the entity afterwards. Typically, this is a closed-set problem. Furthermore, the device used can also be the target of an identification attack in which it is to be uniquely recognized (e.g. [25]).
- **Trajectory reconstruction** In addition, in some cases it is of interest to trace the route taken by a user (e.g. [15]). The motivation can be versatile and ranges from hotspot identification (such as the place of work) to the derivation of movement patterns, as well as the conclusion of characteristics of the driver (e.g. hospital visits indicate an illness).
- **Reconstruct environment** Furthermore, not only can route and driver be deduced from the sensor data, but also the surroundings and means of transport such as bus or train (e.g. [11]).
- **Spoken word reconstruction** Another kind of attack is even more privacy-invasive since it tries to reconstruct words from the accelerometer and thus

can gain additional information without the need to ask for microphone permission. The main motivation for this research is hot-word detection for personal assistants [26], but can also be used as an invasive technology.

Even though this is only a short list, it motivates the need for privacy-protection of frameworks. Deactivating a sensor, however, is not a feasible approach. In our scenario, everyone with access to the raw sensor values can act as an attacker executing one of the mentioned attacks.

4 Usage-Based Insurance

In the following section we clarify the given scenario of UBI [23]. In addition to the well-known payment model based on no-claims classes, the models PHYD [8] and the related but significantly different model PAYD [16] are used in the automotive insurance industry. Both are innovative pricing models in the category UBI. In contrast to PAYD, which places value on generally valid characteristics when calculating the premium, e.g. kilometers driven and the main area covered [23], PHYD takes into account the individual driving behavior of a driver. As with existing vehicle insurance policies, a vehicle is priced, and there is no individual allocation of costs to the respective drivers.

We use UBI as the running example for our framework, although it can be adapted to other sensor-based business models such as health insurance.

4.1 Stakeholders and Their Respective Interests

The UBI scenario introduces two stakeholders, first the policyholder and then the insurer. It is obvious that both parties have different priorities regarding the business model.

The policyholder wants to protect his own privacy. In the PHYD business model, data such as GPS or sensor data from the accelerometer or gyroscope are collected while driving to give multiple insights into a user's driving style. One can argue that this data are critical in terms of privacy. According to e.g. Pfitzmann [19], privacy is conveyed among other things by the fact that users can decide for themselves which data are passed on and to what extent. This is not the case for most PHYD models, since data are gathered solely as defined by the insurer. However, a user only receives a discount if he transmits the data. Thus, he too has an interest in ensuring the integrity of the data.

An insurer on the other side is primarily interested in correct information, i.e, sensor data with integrity. One can further differentiate between two integrity aspects. Firstly, an insurer is interested in correct data (data integrity), so he can derive the correct driver classification for a trip. Secondly, system integrity is important to guarantee a correct workflow according to the given business model.

4.2 Workflow

In Roth et al. [20], PHYD and PAYD products from different major insurance companies in Germany were analyzed. Typically, a vehicle in this context that is driven by several people is insured, one of whom is the policy holder. FPD are recorded using a smartphone or, in rare cases, with a black box. The transmitted raw sensor values denoted as S are usually not analyzed by the insurer itself, but by an external data processor. Aggregated statements are then made available to the insurer. Often certain events in the data are analyzed, so-called maneuvers m, which are patterns in the raw data S. These include, for example, braking or acceleration events. The exact process of information extraction is a protected company secret and is usually not communicated. We assume that the classification process uses a blackbox classification function $\mathcal{C}$ to assign each m to one of μ classes (e.g. aggressive, neutral, passive). The data processor often works for several insurance companies. After analysis of the trips of a certain period of time, a discount is given to the policy holder by the insurance company. The amount of the discount depends on the type of driving.

Interestingly, data are mostly processed by a third party agency. This is often motivated as a privacy foundation since the third party processor only receives the raw sensor data and an identifier independent of customer information. The insurance companies claim that personal data are thus separated from driving data. On that basis, privacy shall be provided, although, this seems like a poor approach to convince a user. Data processing through a third party requires an additional level of trust from the user, especially since some of the processors are not even located in Europe. At the moment, illegal use of the data should be prevented from an organizational and legal point of view by the General Data Protection Regulation (GDPR) in article 5. However, we motivate the need for a technological protection mechanism to, along with others, protect one from privacy attacks. This is underlined by the fact that at this time, there are no TETs or PETs, which make the evaluation of the driving data comprehensible.

In the interest of data economy and expediency, only data that actively allow pricing within the framework of a UBI rate should be collected by the insurer. If further information can be read out of the data, allowing an increase in knowledge, this should be viewed critically by the client in the sense of privacy. Consequently, the question arises whether an insurer will misuse data to derive further information. A PET should prevent such kinds of attacks.

5 *kUBI*

There is obvious need for a privacy-enhanced version to process sensor data. Our proposed pattern named *kUBI* is device agnostic, hence not bound to a specific implementation. Furthermore, it is flexible to enable multiple business models which rely on sensor data.

5.1 Potential Strategies

A holistic pattern for privacy-enhancing existing business models has to take into account the contradicting requirements of the stakeholders. We now discuss four different strategies which are possible in the given context.

1. **Processor-based** All data are collected by a user using a (blackbox) device or smartphone application and submitted as raw data to the processor. The user can neither control nor is there technical evidence that the data is only used for a specific use case, resulting in a less trustful model. Hence, this model is a classical example of *undercover-agent trust*, where an insurer wants to protect his interests from the legitimate user of the device. This is the per-se standard model in the context of UBI.
2. **User-based** Since a user always trusts his local device (called *personal agent trust*), it is desirable that a user collects and processes the data in his trusted domain. To enforce privacy, only results or aggregated partitions of data are transferred (after approval) to a processor. Integrity is harder to validate on the processor side.
3. **Balanced** The processor defines the needed service quality via a policy. The user anonymizes data w.r.t service policy and only sends aggregated results. Furthermore, a user can ensure within his anonymization that his own private goals are reached. Hence, integrity can be controlled by the process while privacy is solely controlled by a user.
4. **Trustee-based** Trust is moved from the processor to a trustee, however, this is a limited enhancement for the user compared to the processor-based approach since trustworthiness in a model is not increased.

5.2 Privacy Enhanced Model

According to the balanced strategy, we now present our pattern for privacy-enhanced business models relying on and processing sensor data from users. The pattern is designed w.r.t Pfitzmann et al. [19] and has two different zones. One is the local trust zone of a user (*User Domain*) and the other the domain of the processor (*Business Domain*). Nobody trusts the zone of his counterpart with one exception: the user's trust zone has an isolated environment called *Hostile Domain* (i.e. a Trusted Execution Environment (TEE)) which can execute hostile but authenticated code in a well defined manner. The user is unable to see or tamper with functions in this isolated environment. Even though we show our model's feasibility at the example of PHYD, it can be ported to other use cases as well.

kUBI protects users' privacy by establishing k-anonymity in a data set. Recall that in the PHYD model, a vehicle is insured, the rate of which is calculated by the classification of the trips made with this vehicle. The classification of a trip is done by the classification of maneuvers. The way the maneuvers are recognized in the sensor data stream is the same for all drivers, but each driver shows an individual behavior during a maneuver. This allows an unnecessary identification of the driver. *kUBI* first performs Complex Event Processing to identify these maneuvers in a continuous data stream. It then replaces the detected maneuvers in such a way that although the correct classification of the maneuvers (by the insurer) is still ensured, the entropy of the maneuvers is reduced to such an extent that no conclusions can be drawn about the driver. This is achieved by using reference maneuvers of the same class as drop-in replacements for recognized, driver-derived maneuvers. Performing the whole trip

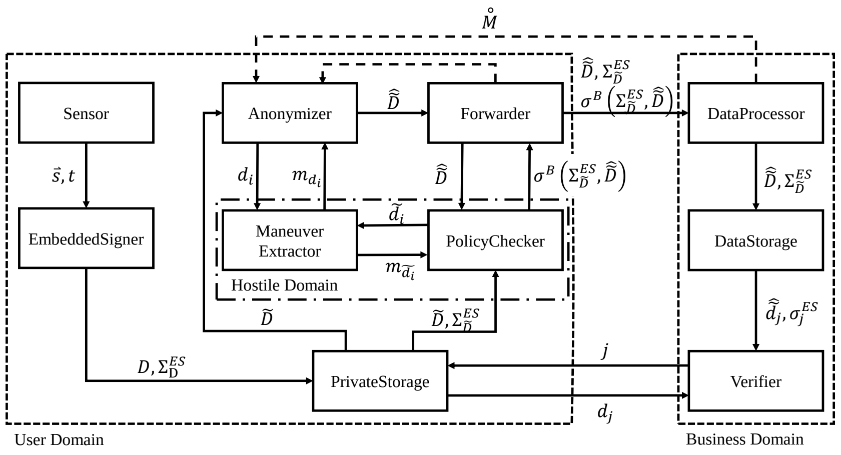
$\mathring{M}$
Sensor
$\vec{s}, t$
EmbeddedSigner
Anonymizer
$\widehat{\widetilde{D}}$
Forwarder
$\widehat{\widetilde{D}}, \Sigma_{\widetilde{D}}^{ES}$
$\sigma^B\left(\Sigma_{\widetilde{D}}^{ES}, \widehat{\widetilde{D}}\right)$
DataProcessor
d_i
m_{d_i}
$\widehat{\widetilde{D}}$
$\sigma^B\left(\Sigma_{\widetilde{D}}^{ES}, \widehat{\widetilde{D}}\right)$
$\widehat{\widetilde{D}}, \Sigma_{\widetilde{D}}^{ES}$
Maneuver Extractor
$\widetilde{d}_i$
PolicyChecker
$m_{\widetilde{d}_i}$
Hostile Domain
DataStorage
$\widehat{\widetilde{d}}_j, \sigma_j^{ES}$
$\widetilde{D}$
$\widetilde{D}, \Sigma_{\widetilde{D}}^{ES}$
D, Σ_{D}^{ES}
PrivateStorage
j
Verifier
d_j
User Domain
Business Domain

Fig. 2: Pattern for privacy enhanced business models relying and processing sensor data from users.

classification in the user domain and only forward that data to an insurer is not feasible since e.g. additional or historical knowledge is needed and computation resources are limited.

The proposed framework *kUBI* is organized in several modules, as illustrated in Fig. 2, which are either placed in the users domain (i.e. mobile phone) including the hostile environment, or the business domain (i.e. insurer). We consider *kUBI* being a PET because a user can actively decide what is sent and establish anonymity himself. We also consider it aTET because the user sees the data including evaluation, i.e. maneuvers, which are passed on.

Data are generated using a Sensor $\mathcal{S}$ such as the accelerometer and then directly forwarded to an EmbeddedSigner unit $\mathcal{E}$ which aggregates multiple sensor values into a data block d. This unit signs each data block value to prevent changing sensor values in a block afterwards. The signature and the respective value is then forwarded to a persistent PrivateStorage $\mathcal{PS}$ and is forwarded from here in the form of a trip to two modules. The Anonymizer $\mathcal{A}$ is responsible for replacing driven maneuvers with a suitable reference maneuver. The maneuver identification is performed by a ManeuverExtractor $\mathcal{M}$, which is initialized by the insurer and located in the Hostile Domain. A PolicyChecker $\mathcal{P}$ (also in the hostile domain) checks the correct functioning of $\mathcal{A}$. It confirms an integer anonymization by issuing a signature σ and passes it to the forwarder in the user's domain. According to Pfitzmann et al. [19], the forwarder decides whether the anonymized data blocks $\widetilde{\widetilde{D}}$ will be forwarded to the data processor. The data processor checks the signature created by the $\mathcal{P}$, processes and stores the data and has the possibility to verify the integrity of the transmitted data with a Verifier $\mathcal{V}$.

5.3 Components

Sensor A PHYD app requests data from specific sensors. For the sake of simplicity we only use the accelerometer. A Sensor $\mathcal{S}$ continuously generates a data stream of vectors consisting of several sensor values (say for 3 coordinate axes x, y, z) and a timestamp t, denoted as $\vec{s} = [s_x, s_y, s_z, s_t]^T \in \mathbb{R}^4$. Outputs are chronologically organized by the timestamp $s_{i,t}$ ($\vec{s}_i < \vec{s}_{i+1}$, i.e. the i-th vector was created before the $i+1$-th)[1]. As shown in Section 3, a sensor is a virtual device which can be accessed via the Sensor Event API of Android. It outputs Sensor Events (i.e. $\vec{s_S}$) in the described and defined shape. To enable the integration of our framework into existing applications, the shape of the data is not altered.

EmbeddedSigner An application registers at a Signed Sensor Event API to receive Signed Sensor Events. Signed Sensor Events are extended versions of the existing Sensor Events with additional information to provide integrity. A Signed Sensor Event is a concatenation of multiple $\vec{s}$ based on a timestamp δt_{SSE} (e.g. 1 second). We call such a Signed Sensor Event data block with i elements,

$$d_k = \{[\vec{s}_1, \dots, \vec{s}_i], \sigma_k^{ES} \mid s_{l,t} \in (t_{Start}, t_{Start} + \delta t_{SSE}) \text{ for } l = 1, 2, \dots, i\}$$

which starts at t_{Start} and includes all $\vec{s}$ generated up to Δt_{SSE}. Data blocks are non overlapping, i.e. the $k+1$-th data block's first element is $\vec{s}_{i+1}$. Each d_k also has, next to its payload, a signature $\sigma_{d_k}^{\mathcal{E}} = (\mathcal{H}_{\mathcal{E}}(\vec{s}_1 \| \dots \| \vec{s}_i))^{d_{ES}}$ of that payload. The signature is created by using a secure and publicly known hash function $\mathcal{H}_{\mathcal{E}}$ to hash the concatenation of all Sensor Events in that data block and then signing the hash value using the EmbeddedSigner's private key $d_{\mathcal{E}}$. Furthermore, let id be a function defined as $id(d_k) = \{0,1\}^{32}$ which deterministically creates a unique ID for a data block (independent of the payload).

The EmbeddedSigner and its Signed Sensor Event API should be used as a drop-in replacement for the current Sensor Event API and be provided by Android itself for wide and easy adoption. The Android Sensor Stack architecture enables the separation of data acquisition by means of the physical sensor and data processing in the application, hence compatibility of the proposal is ensured.

It is meaningful to place the EmbeddedSigner, which is provided by the OS, in a secure and tamper-proof enclave. For example, Trusty [1] can be used here. Trusty is an Android specific implementation of a TEE for various practical purposes and runs on the same processor of the end device, but according to the TEE definition independent of userland applications. With the Keystore API, Android already offers secure key management, used to create and verify signatures. It is thus suitable for the EmbeddedSigner. *kUBI* can use digital certificates whose root CA is Google itself and whose device-specific private key $d_{\mathcal{E}}$ is deposited once by Google in the Keystore API of the respective device. The device's public certificate can be freely distributed, e.g. for a data processor to verify signatures.

[1] We use {} to denote such an ordered set and [] to specify an unordered list.

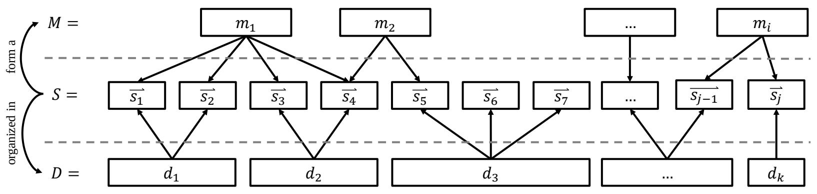

Fig. 3: Relationship of D, S and M. S form M and are organized in D.

PrivateStorage The PrivateStorage is a persistent storage which holds all data blocks D of a device, including the set Σ^{ES} of all signatures $\sigma^{ES}_{d_{1,\dots,|D|}}$. It also serves as a buffer, since a trip is only classified by the insurer's app after it has been completed. It concatenates multiple data blocks $d_i, \dots, d_j$ to a trip $\widetilde{D} \subset D$ which is subject to classification and thus anonymization with the respective signature set $\Sigma^{ES}_{\widetilde{D}} = \{\sigma^{ES}_{d_i} \| \dots \| \sigma^{ES}_{d_j}\}$ for that trip.

ManeuverExtractor According to the setting (c.f. Section 4), maneuvers are interesting in terms of trip classification. For instance, many hard braking maneuvers occurring in a trip indicate aggressive or inattentive driving. Typically, all maneuvers M are derived by analyzing patterns in the sensor data stream and e.g. thresholds are not communicated by an insurer. *kUBI* takes this into account by placing a customizable ManeuverExtractor into the hostile domain of the system. The ManeuverExtractor works as-a-service by sticking to blackbox principles to protect the insurer's business case (*undercover-agent trust*). This provides secrecy, one important requirement from the stakeholder analysis. Hence, the ManeuverExtractor extracts maneuvers from the data block stream using a confidential process. However, the user is able to control in and output to enhance trust: $\mathcal{M} : \{d_1, \dots, d_i\} \longrightarrow \{m_{d_1}, \dots, m_{d_i}\} = \widetilde{M}$ with $\forall d \in \widetilde{D}$. The output is a list of maneuvers crafted by these data blocks including derivable start and end timestamps.

Figure 3 illustrates the relationship of maneuvers M, data blocks D and single sensor values S. The EmbeddedSigner concatenates multiple sensor values into data blocks which, in turn, can form maneuvers. However, a sensor value does not need to form a maneuver, e.g. if the driver is only going straight without accelerating or braking. $len_s(m)$ defines the number of sensor values that form a maneuver, while $len_d(m)$ counts the number of data blocks. It is likely that $\exists m_a, m_b \in M : len_s(m_a) \neq len_s(m_b)$ holds. Further, we define a transformation function $\mathcal{T}^z(x^*) = y^*$ where $x, y \in [s, m, d]$ and z is the destination type. All elements are time series, but are different subsets. $\mathcal{T}$ transforms units accordingly to Fig. 3. For instance, $\mathcal{T}^s(m) = [s_1, s_2, \dots]$ transforms a maneuver to a list of sensor values that form that maneuver.

Anonymizer Since maneuvers are used to categorize a trip, their integrity is important. However, at the same time, maneuvers help to identify a driver as shown in [20]. Furthermore, sending raw data to foreign domains is critical, thus the Anonymizer module is responsible for balancing the mentioned interests.

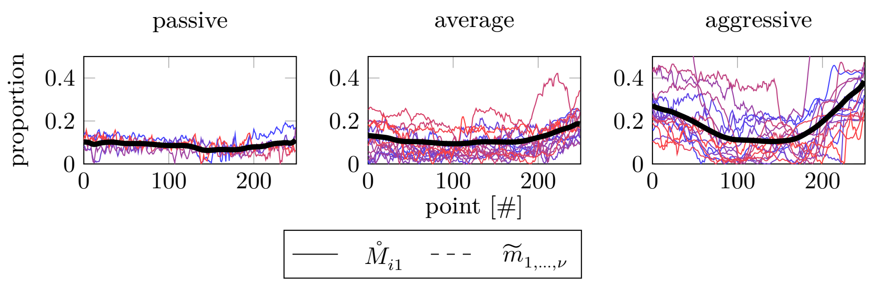

Fig. 4: Reference maneuvers $\mathring{M}_{i1}$ with $i = 1, \dots, \mu = 3$ are representative for a category. Shown are maneuvers of type *braking* for a single s (λ is set to 250).

kUBI achieves this by replacing identified $\widetilde{M}$ in a privacy-friendly, but comprehensible way with so-called reference maneuvers using a one-way function with similar properties like a hash function: $\mathcal{A} : \widetilde{D} \longrightarrow \widetilde{\widetilde{D}}$. It is easy to replace a driven maneuver with a reference maneuver using $\mathcal{A}$. However, it is very hard to reconstruct the original maneuver from a reference maneuver.

A reference maneuver is representative for a maneuver type of a class. *kUBI* assumes that a reference maneuver is created on the basis of real maneuvers by overlaying them (which is also common in speech recognition), e.g. using Soft-DTW barycenter, although it is basically independent of that because reference maneuvers $\mathring{M}$ are provided by the data processor (i.e. insurer). $\mathring{M}$ is a $\mu \times \nu$ matrix where μ is the number of classes an insurer uses to categorize a trip and ν is the number of reference maneuvers selectable for replacement of a maneuver class. We recall that in the given scenario, $\mu = 3$ classes exist. Figure 4 illustrates how reference maneuvers can be constructed from previous, globally recorded maneuvers of a category in the data-processor's domain using the weighted average.

The ANONYMIZER processes any maneuver $\widetilde{M}_{i=1,\dots,\mu;j=1,\dots,\nu}$ to select a suitable reference maneuver to replace m. One speaks in the following of the anonymous maneuver $\widetilde{\widetilde{m}}$, once the original maneuver is replaced. In order to perform the selection, we use Dynamic Time Warping (DTW). DTW can compare two time series by finding the so-called optimal warping path, denoted as $dtw(m_1, m_2)$. The warping path can be interpreted as a similarity between two time series, i.e. two maneuvers m_1, m_2, although maneuvers are previously resampled to same length and normalized to enable comparison. Let $\mathcal{R} : \mathbb{R}^{* \times 4} \longrightarrow \mathbb{R}^{\lambda \times 4}$ be a resampling function $\mathcal{R}$ to transform any given maneuver $m \in M$ of arbitrary length so that $len_s(m) = \lambda$ holds afterwards. The (already normalized) $\mathring{M}_{ij}$ with the minimal warping path is selected for replacement, transformed to a data block and respective sensor values. The result of the ANONYMIZER is a set $\widetilde{\widetilde{D}}$ which contains the trip but all maneuvers have been replaced (and thus anonymized to provide k-anonymity) with a corresponding reference maneuver. However, since the maneuver (and its corresponding data blocks resp. sensor values) has to be adjusted to fit into the data stream, we call it $\hat{\tilde{d}}$. Note that the characteristics of a trip, its maneuver class distribution and order have not been altered at all, allowing to draw the same conclusions by the data processor.

Forwarder Trust in the system is established, among other things, by the fact that the data is not sent to the insurer through an app provided by the insurer, but is under the control of the user (personal-agent trust [19]). If a user decides to submit data to a data processor, he needs a valid signature $\sigma^{B}(\Sigma^{ES}_{\widetilde{D}}, \widehat{\widetilde{D}})$ to prove an anonymization process as it was agreed on with the data processor. Therefore, he forwards the anonymized trip $\widehat{\widetilde{D}}$ to a PolicyChecker to have its correctness confirmed.

PolicyChecker By defining a policy Φ, the data processor can determine the extent to which the sensor values may deviate between the original and the resulting trip. By specifying a policy, the data quality of the trip is guaranteed despite the anonymization of the sensor data. It is an important building block to balance privacy and integrity. First, it verifies that the anonymized trip $\widehat{\widetilde{D}}$ was created on behalf of the real trip data $\widetilde{D}$. This can be validated by executing a specific Φ. Recall that each $\vec{\mathcal{S}} \subset \widetilde{D}$ has a timestamp which cannot be altered due to the signature $\Sigma^{ES}_{\widetilde{D}}$. Hence, this combination is used to prove integrity to a data processor and is transmitted to him via a signature from its PolicyChecker. Let $\mathcal{P}$ be a function that creates a valid signature if Φ holds: $\mathcal{P} : \widehat{\widetilde{D}}, \widetilde{D}, \Sigma^{\mathcal{E}}_{\widetilde{D}} \to \sigma^{\mathcal{P}}(\Sigma^{ES}_{\widetilde{D}}, \mathcal{H}_{\mathcal{P}}(\widehat{\widetilde{D}}))$ where $\mathcal{H}_{\mathcal{P}}$ is a secure hash function. The signature is essential for further integrity checks within the framework as seen in Data Processor/Verifier. For reasons of trust, PolicyChecker is implemented in the hostile domain within the user domain. A user can not alter or analyze the behavior of that blackbox. However, he controls the input and output parameters which is an important criterion for user acceptance. The box is unable to send any information using a side-channel to the data processor. The output is verifiable by a user and cannot contain any hidden information. Similar to the ManeuverExtractor, Trusty can be used.

Since the PolicyChecker is flexible in terms of the applied policy, we give an example of a potential Φ. An anonymization may be correct if 1. The number of maneuvers in $\widehat{\widetilde{D}}$ equals $\widetilde{D}$, 2. Input $\widetilde{D}$ from the PrivateStorage can be used to verify if the distribution of maneuver types in the anonymized trip $\widehat{\widetilde{D}}$ equals $\widetilde{D}$, and 3. Each $d \in \widetilde{D}$ has a valid signature $\sigma^{ES}(d)$, i.e. no recorded trip is used to deceive the PHYD system (replay attack); datablocks contain $\vec{s}$ which in turn have a timestamp s_t signed into $\sigma^{ES}(d)$.

For privacy reasons and to clearly separate the domains, the PolicyChecker does not have access to the Anonymizer, hence is unable to anonymize a given maneuver in $\widetilde{D}$. Within this framework, the design of the policy is therefore deliberately limited.

DataProcessor/Verifier Once the data processor receives data $\widetilde{\widetilde{D}}$ from a user along with a corresponding signature $\sigma^{\mathcal{P}}\left(\Sigma^{\mathcal{E}}_{\widetilde{D}}, \mathcal{H}_{\mathcal{P}}\left(\widetilde{\widetilde{D}}\right)\right)$ crafted by the processor-controlled PolicyChecker, he needs to validate $\sigma^{\mathcal{P}}$ in the first place. This ensures that the data has not been modified and comes from an accepted domain. Therefore, once he has received the data, he checks that the signed and acknowledged hash $\mathcal{H}_{\mathcal{P}}(\widetilde{\widetilde{D}})$ matches the data blocks that were transmitted. Thus, a user must submit the anonymized data $\widetilde{\widetilde{D}}$ that was checked by the PolicyChecker, otherwise the data processor can detect this. The signatures $\Sigma^{\mathcal{E}}_{\widetilde{D}}$ as well as the trip $\widetilde{\widetilde{D}}$ will be saved and evaluated. To further increase integrity or in case of suspicion of fraud by the user, the data processor has another tool at his disposal. Recall that the Anonymizer performed an anonymization operation $\widetilde{D} \longrightarrow \widetilde{\widetilde{D}}$, which is very hard to reverse. The Verifier is used to perform a knowledge proof, where the user must prove that he is the originator of the transmitted data, otherwise fraud is assumed. Therefore, the Verifier selects up to μ percent of random elements from the received trip $\widetilde{\widetilde{D}}$ (denoted as prove-set $P \subset \widetilde{\widetilde{D}}$) and requests raw data blocks from the prover, i.e. the user. μ is a security parameter to balance integrity and anonymity. Consequently, a trip is only considered valid if $\forall \tilde{\tilde{d}}_i \in P : \exists d_i \left[\mathcal{PS} \leftarrow d_i \wedge id(d_i) = id\left(\tilde{\tilde{d}}_i\right)\right]$ holds, i.e. a user can submit the raw data block for a given anonymized data block. Note that the ID of data block cannot be altered in the process since it is part of the data blocks signature thus lookup form PersistentStorage is done using *id*. In addition, the the Verifier also needs to verify that the given maneuver was not manipulated in terms of any sensor readings or timestamp since the *id* is not bound to a data block's payload. $\Sigma^{\mathcal{E}}_{\widetilde{D}}$ holds all signatures $\sigma^{\mathcal{E}}_{d_{1,\dots,|\widetilde{D}|}}$ of the data blocks as processed by the EmbeddedSigner. Hence, he calculates—using $\mathcal{H}_{\mathcal{E}}$—the hash of every received d_i and verifies if that signature is part of $\Sigma^{\mathcal{E}}_{\widetilde{D}}$. The proof is completed once the user can submit all needed requested data blocks and if each datablock is part of the trip which is verifiable thanks to the signatures.

5.4 Basic Design Decisions

kUBI was designed to balance integrity and privacy. These aspects are integrated at several points in the design.

Integrity The pattern design implements three different proofs which are needed for a privacy-balanced business model. A user has also an interest in integrity because he only gets a discount once an insurer accepts the sent and anonymized data, thus fraudulent behavior is not beneficial.

P1 Data has not been tampered with: A trustworthy EmbeddedSigner unit processes each sensor value as soon as it is generated on a lower level of the Android Software Stack. Each $\vec{s}$ is then hashed and signed by this entity, using its pri-

vate key $d_{\mathcal{E}}$. A user altering some values of a $\vec{s}$ cannot create a valid signature, eventually being detectable by the data processor's VERIFIER.
P2 Data has not been modified out of boundaries: A data processor rejects submitted items from a user unless a valid signature $\sigma^B\left(\Sigma^{ES}_{\widetilde{D}}, \mathcal{H}_{\mathcal{P}}\left(\widetilde{D}^*\right)\right)$ is shown. This in turn is generated by the POLICYCHECKER according to conditions of the data processor. This protected control unit allows the data processor to specify the Quality-of-Service (QoS).
P3 User has produced the data: A user has to prove knowledge of raw data blocks for any anonymized datablock on request by the VERIFIER. Although a user may use data generated by another device, this can easily be handled by using device-dependent credentials in the POLICYCHECKER to generate unique signatures σ^B.

Trust Our pattern provides trust at four significant positions.
T1 Box only outputs user verifiable signature: There is no hidden information in the POLICYCHECKER's output since it is easily comprehensible by a user using $d^{\mathcal{P}}$ to verify the given signature, created using a known hash function $\mathcal{H}_{\mathcal{P}}$ and user controlled input data.
T2 Box is in a secure enclave and cannot be tampered with: Trustee guarantees that sensitive processes are carried out in the protected environment. A user cannot create a signature for the POLICYCHECKER because he does not have the cryptographic material, i.e. the secret key $d_{\mathcal{P}}$. Trustee makes sure that it cannot be read.
T3 User controls which data to forward: The paradigm of user personal-agent trust was chosen over undercover-agent trust. Each information is processed in the local domain of the user. Every data transmission is solely controlled by a user by relying on a forwarding engine.
T4 User can choose freely from policy defined values: A user can select any $\mathring{M}_{ij}$ to replace a maneuver from a trip $\widetilde{D}$. As long as the POLICYCHECKER verifies the integrity, the data processor accepts a value. It is comprehensible for a user if policy Φ does not hold.

5.5 Modified Android Implementation

Based on the presented model, the following is an exemplary implementation of the API for working with sensors as shown in Section 3. It is ensured that due to the interchangeability of the components, a fast adaptation to existing applications is possible.

```
val signedSensorManager = getSystemService(Context.SIGNED_SENSOR_SERVICE) as ↩
    SignedSensorManager
val accelerometerSensor: Sensor? = ↩
    sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
val signedSensorListener = object: SignedSensorEventListener {
    override fun onAccuracyChanged(sensor: Sensor, accuracy: Int) {
    }
    override fun onSensorChanged(event: SignedSensorEvent) {
        val dT: Long = event.timestamp
        val axisX: Float = event.values.get(0)
        val axisY: Float = event.values.get(1)
        val axisZ: Float = event.values.get(2)
        val datablockId: UUID = event.datablockId
        // use sensor values
    }
    override fun onDatablockComplete(id: UUID, signature: ByteArray) {
    }
}
signedSensorManager.registerListener(signedSensorListener, accelerometerSensor, ↩
    SensorManager.SENSOR_DELAY_FASTEST)
```

One can see that all API calls are the same as in the Android reference implementation. However, every SignedSensorEvent also holds a data block ID. Each time a data block is completed, e.g. after 1 second, onDatablockComplete is called. A developer can then get the data block's ID to map all sensor values to it. Furthermore, he gets a signature of that data block (which is the output of $\mathcal{H}_{\mathcal{E}}$). Applications that do not require the integrity protection features can discard the optional information accordingly. The separation of onSensorChanged and onDatablockComplete ensures that real-time processing is still possible when a new sensor value arrives.

6 Evaluation

Even if the framework is designed generically and the structure is also conceivable for other scenarios, it should nevertheless be shown in the evaluation that *kUBI* can increase the anonymity of individual drivers of a vehicle within its system boundaries, using the proposed idea of replacing maneuvers with reference maneuvers and the given PHYD scenario.

6.1 Identification Attack

The identification attack from Roth et al. [20] was used in this work to assess the quality of the proposed framework. We used the same setting, i.e. same data set and same parameter settings. In the context of that work it could already be shown that classical anonymization methods are not sufficient to protect drivers in the complex context of PHYD. The presented attack is based on supervised machine learning and assigns with the help of DTW and k-Nearest Neighbor a maneuver of the types braking, acceleration and cornering to a driver. It is very robust against noise and other environmental influences.

6.2 Anonymization

An example trip based on real-world data was anonymized using the proposed approach. A $\mathring{M}$ of shape 3×1 served the ANONYMIZER as a basis to replace the real maneuvers against reference maneuvers. From Figure 5, one can see all speed recording of a trip. Maneuvers were extracted using the known approach and data blocks were recreated using sensor values from the respective reference maneuvers. We classified the trip before and after anonymization using the same classification pipeline. The class of each maneuver stayed the same as intended but the sensor readings are, as shown, much smoother removing any identifying behavior of a user while e.g. accelerating ultimately reducing entropy. At

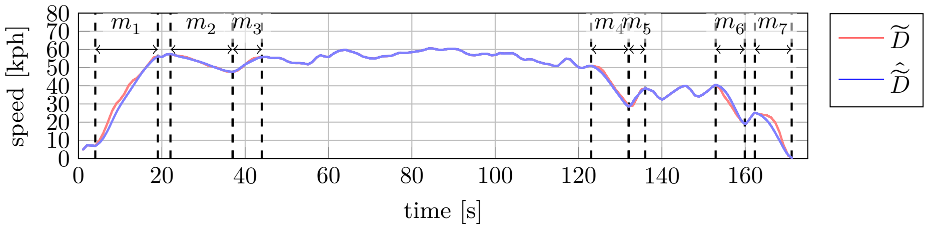

Fig. 5: Comparison of a trip from raw data and its anonymized version.

Table 1: Accuracy for prediction using the identification attack and data set from [20].

(a) Data set without anonymization

Actual \ Predicted	A	B	C	D	E	F
A	0.58	0.06	0.08	0.20	0.05	0.10
B	0.02	0.75	0.03	0.11	0.04	0.05
C	0.02	0.13	0.59	0.10	0.15	0.01
D	0.09	0.04	0.02	0.71	0.05	0.09
E	0.06	0.08	0.03	0.08	0.73	0.03
F	0.05	0.03	0.05	0.10	0.03	0.74

(b) Anonymized data set using *kUBI*

Actual \ Predicted	A	B	C	D	E	F
A	0.44	0.05	0.05	0.09	0.11	0.26
B	0.17	0.29	0.27	0.10	0.08	0.10
C	0.16	0.13	0.30	0.02	0.11	0.28
D	0.27	0.15	0.09	0.29	0.09	0.10
E	0.13	0.21	0.14	0.14	0.30	0.09
F	0.27	0.06	0.16	0.04	0.07	0.39

the same time, the needed classification can still be drawn from the data. Our framework will enable data processors to roll their own classification method using the adaptable ManeuverExtractor and verify correct results using also a data processor controlled PolicyChecker.

6.3 Privacy

Drivers are considered anonymous if the probability of a true prediction is less than or equal to the probability of a false prediction of his maneuvers for at least one other driver (k-anonymity). Section 6.3 illustrates the accuracy for prediction using the identification attack and data set from [20] a confusion matrix. Table 1a shows the results for the non-anonymized data set. It clearly states that the driver is almost in every case identifiable using the illustrated attack, as the other drives can be excluded with a very high probability. However, using our proposed PET *kUBI* , k-anonymity of at least 2 drivers can be ensured as Table 1b shows. Furthermore, results are very dense, thus the anonymity can be estimated even higher, since the prediction probability of many drivers is around 20 %.

7 Conclusion

In this paper, it was first shown using the example of UBI that sensor-based business models can benefit from powerful user devices, but that this is accompanied by significant risks for privacy. An analysis of the stakeholders shows the different interests. Based on this, a holistic framework called *kUBI* was presented, which as PET allows to balance integrity and anonymity as primary protection goals. The framework was presented in detail and a potential extension in the Android software stack was described.

We could prove in our evaluation the suitability of the proposed anonymization method and the feasibility of the model using real-world data and an existing identification attack. The results of our privacy-friendly PHYD approach are promising because *kUBI* establishes k-anonymity of at least 2 even in this complex scenario. The risk for side-channel attacks based on raw data, by e.g. an insurance company, are significantly reduced.

For further work, it is planned to further optimize the framework. Thus, the integer swapping of data points can possibly compensate for other attacks described in this work. In addition, various fraud possibilities will not only be theoretically examined, as is the case here, but will also be implemented in practice as a distributed system. The generalization of the framework shall be advanced by means of further use cases.

References

1. Android: Trusty TEE | Android Open Source Project (2016), https://source.android.com/security/trusty
2. Android: Sensors | Android Open Source Project (2020), https://source.android.com/devices/sensors
3. Bai, X., Yin, J., Wang, Y.P.: Sensor Guardian: prevent privacy inference on Android sensors. No. 1, Springer International Publishing (12 2017)
4. Bal, G., Rannenberg, K., Hong, J.I.: Styx: Privacy risk communication for the Android smartphone platform based on apps' data-access behavior patterns. Computers and Security pp. 187–202 (9 2015)
5. Benndorf, V., Normann, H.T.: The Willingness to Sell Personal Data. vol. 120, pp. 1260–1278 (10 2018)
6. Bugiel, S., Heuser, S., Sadeghi, A.R.: Flexible and fine-grained mandatory access control on android for diverse security and privacy Policies. In: Proceedings of the 22nd USENIX Security Symposium. pp. 131–146 (2013)
7. Chakraborty, S., Raghavan, K.R., Johnson, M.P., Srivastava, M.B.: A framework for context-aware privacy of sensor data on mobile systems. In: Proceedings of the 14th Workshop on Mobile Computing Systems and Applications - HotMobile '13. p. 1. ACM Press, New York, New York, USA (2013)
8. EY: Introducing 'Pay How You Drive' (PHYD) Insurance - Insurance that rewards safe driving (2016)
9. Felt, A.P., Ha, E., Egelman, S., Haney, A., Chin, E., Wagner, D.: Android permissions: User attention, comprehension, and behavior. In: SOUPS 2012 - Proceedings of the 8th Symposium on Usable Privacy and Security (2012)
10. Greaves, S., De Gruyter, C.: Profiling driving behaviour using passive global positioning system (GPS) Technology. Operations, Transport and Safety: Outside the Square: Institute of Transpo

11. Hemminki, S., Nurmi, P., Tarkoma, S.: Accelerometer-based transportation mode detection on smartphones. In: SenSys 2013 - Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems. Association for Computing Machinery (2013)
12. Hong, S.k., Gurjar, K., Kim, H.s., Moon, Y.s.: A Survey on Privacy Preserving Time-Series Data Mining. 3rd International Conference on Intelligent Computational Systems (ICICS'2013) (2013)
13. Kang, R., Dabbish, L., Fruchter, N., Kiesler, S.: "My data just goes everywhere": User mental models of the internet and implications for privacy and security. In: Symposium on Usable Privacy and Security (SOUPS) 2015. pp. 39–52 (2015)
14. Kreuter, F., Haas, G.C., Keusch, F., Bähr, S., Trappmann, M.: Collecting Survey and Smartphone Sensor Data With an App: Opportunities and Challenges Around Privacy and Informed Consent. Social Science Computer Review (12 2018)
15. Li, Z., Pei, Q., Markwood, I., Liu, Y., Pan, M., Li, H.: Location Privacy Violation via GPS-agnostic Smart Phone Car Tracking. IEEE Transactions on Vehicular Technology pp. 1–1 (2018)
16. Litman, T.A.: Pay-As-You-Drive Pricing For Insurance Affordability. Victoria Transport Policy Institute **10**(June), 19 (2011)
17. Martínez, M.V., Echanobe, J., Del Campo, I.: Driver identification and impostor detection based on driving behavior signals. IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC pp. 372–378 (2016)
18. Mylonas, A., Theoharidou, M., Gritzalis, D.: Assessing privacy risks in android: A user-centric approach. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 8418 LNCS, pp. 21–37 (2014)
19. Pfitzmann, A., Pfitzmann, B., Schunter, M., Waidner, M.: Trusting mobile user devices and security modules. Computer **30**(2), 61–68 (1997)
20. Roth, C., Aringer, S., Petersen, J., Nitschke, M.: Are Sensor Based Business Models a Threat toPrivacy: The Case Of Pay-How-Your-DriveInsurance Models. In: The 17th International Conference on Trust, Privacy and Security in Digital Business - TrustBus2020. Bratislava, Slovakia (2020)
21. Solove, D.J.: Nothing to hide: The false tradeoff between privacy and security (2011)
22. Troncoso, C., Danezis, G., Kosta, E., Preneel, B.: PriPAYD: Privacy friendly pay-as-you-drive insurance. In: WPES'07 - Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society. vol. 8, pp. 742–755 (9 2007)
23. Tselentis, D.I., Yannis, G., Vlahogianni, E.I.: Innovative Insurance Schemes: Pay as/how You Drive. Transportation Research Procedia **14**, 362–371 (2016)
24. Weydert, V., Desmet, P., Lancelot-Miltgen, C.: Convincing consumers to share personal data: double-edged effect of offering money. Journal of Consumer Marketing (2019)
25. Zhang, J., Beresford, A.R., Sheret, I.: SensorID: Sensor Calibration Fingerprinting for Smartphones. In: Proceedings of the 40th IEEE Symposium on Security and Privacy (SP). IEEE (2019)
26. Zhang, L., Pathak, P.H., Wu, M., Zhao, Y., Mohapatra, P.: AccelWord: Energy efficient hotword detection through accelerometer. In: MobiSys 2015 - Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services. pp. 301–315. Association for Computing Machinery, Inc, New York, New York, USA (5 2015)