Providing Occupancy as a Service with Databox

Poonam Yadav, John Moore,
Qi Li, Richard Mortier
University of Cambridge, UK
first.last@cl.cam.ac.uk

Anthony Brown, Andy Crabtree,
Chris Greenhalgh, Derek McAuley
University of Nottingham, UK
first.last@nottingham.ac.uk

Yousef Amar, Ali Shahin Shamsabadi
Queen Mary University of London, UK
first-initial.last@qmul.ac.uk

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

ABSTRACT

Occupancy modelling for efficient energy management of indoor spaces has gained significant recent attention. Unfortunately, many such models rely on copying sensor data to the cloud for third-party services to process, creating risks of privacy breach. Such matters have become particularly pertinent for companies handling data of EU citizens due to provisions of the General Data Protection Regulation (GDPR). In this paper we present an implementation of "Occupancy-as-a-Service" (OaaS) at the edge, inverting the usual model: rather than ship data to the cloud to be processed, we retain data where it is generated and compute on it locally. This effectively avoids many risks associated with moving personal data to the cloud, and increases the agency of data subjects in managing their personal data. We describe the Databox architecture, its core components, and the OaaS functionality. As well as improving the privacy of the occupants, our approach allows us to offer occupancy data to other applications running on Databox, at a granularity that is not constrained by network usage, storage or processing restrictions imposed by third-party services, but is under data subject control.

CCS CONCEPTS

• **Security and privacy** → *Privacy-preserving protocols*; *Domain-specific security and privacy architectures*; • **Computer systems organization** → *Sensors and actuators*;

KEYWORDS

IoT Middleware, Personal Data Management, Indoor Occupancy

ACM Reference Format:
Poonam Yadav, John Moore, Qi Li, Richard Mortier, Anthony Brown, Andy Crabtree, Chris Greenhalgh, Derek McAuley, Yousef Amar, Ali Shahin Shamsabadi, and Hamed Haddadi. 2018. Providing Occupancy as a Service with Databox. In *The 1st ACM International Workshop on Smart Cities and Fog Computing (CitiFog'18), November 4, 2018, Shenzhen, China.* ACM, New York, NY, USA, 6 pages. *https://doi.org/10.1145/3277893.3277894*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CitiFog'18, November 4, 2018, Shenzhen, China
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6051-7/18/11…$15.00
https://doi.org/10.1145/3277893.3277894

1 INTRODUCTION

Accurate occupancy estimation is important for occupancy based energy-efficient control systems such as control of indoor lighting and energy-saving automatic HVAC. Providing accurate occupancy sensing and estimation requires fine-grained time-series data from ambient sensors such as cameras and motion detectors. If this data were exposed to other parties, a considerable amount of private information would be leaked, e.g., occupants' sleeping habits, social interactions, and activity patterns. Using Databox, we implement a privacy-enhanced occupancy information service that provides accurate occupancy estimation without exposing fine-grained time-series data – only the computed occupancy estimation is revealed. Potential benefits of our approach include:

- Improving space utilisation of offices and classrooms by employees and students.
- Improving targeting of advertisements, insurance, security and other services to occupants in hotels/dormitories.
- Improving detection of depression and other diseases based on individual activity patterns.
- Improving energy-efficiency of home automation systems such as HVAC control.

The essence of our approach is to avoid a number of risks and threats by simply not sending raw sensor data to remote cloud-hosted services for processing. Instead, we process data on a set-top box-like device local to the building, and allow other applications to extract only the processed estimation of occupancy according to the data subjects' (i.e., users) preferences. We next discuss related work that has considered the privacy/utility trade-off in sensor data (§2) before describing the architecture of our Databox (§3) and OaaS (§4), presenting an early evaluation (§5), and concluding (§6).

2 RELATED WORK

Sensor data is core to many IoT applications but is known to leak private information [9, 12, 13, 19]. A range of sensors exist which are capable of supporting occupancy prediction [1, 6, 11]. For example, CO_2 measurements are useful for occupancy detection and estimation [7]; fine-grained electric meter data with sampling every 10 seconds leaks more accurate occupants information than sampling every 30 seconds or 1 minute [19]. This work compares CHMM, CRF, HM-SVM, Rule-based for PIR and NaiveBayes, Random Forest, Decision Tree, Multilayer Perceptron, and k-Nearest Neighbor for Smart meter data.

Ang et al [2] attempted to compute the number of people in the room via training the different classifiers on the data extracted

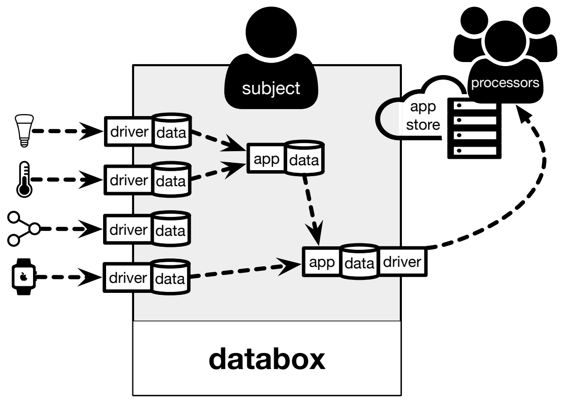


Figure 1: Databox architecture.

from ambient sensors. They used ambient sensors instead of camera sensors for occupancy detection in order to preserve the privacy of people. They estimated the number of people in the room as a linear function of time, humidity, CO_2, sound rate and illumination. Their results have shown that CO_2 rate, illumination level and sound rate are the most dominants sensors for recognising indoor human occupancy.

Although a threat-model has been proposed [14] it makes assumptions that a single sensor per room is used and that the attacker has access to the database with the goal being to derive information about the occupants at the target location.

IoT applications typically do not provide metrics related to privacy risk associated with sensor data generated by users. To address this, Singh et al proposed a system which allows an application to tradeoff its functionalities and privacy risks as instructed by the user [18]. However, there are many applications where finding this tradeoff conflicts with providing the required functionalities expected by the user. For example, introducing noise in occupancy data to prevent privacy disclosure degrades the efficiency and usefulness of the data [10].

Zhao et al [21] presented a fog computing solution, where computation is performed at the edge of the network, to detect occupancy and human activity using low-cost motion sensing and wireless networking devices. To make use of the occupancy data generated by this solution by other applications running in the cloud, a complete privacy preserving solution is required.

Similarly, edge computing enabling smart cameras with built-in face and object recognition capabilities filter unwanted events data locally to reduce the network traffic.[1,2]

We are concerned with the scenario where the system preserves privacy by supporting computation across the raw, fine-grained data locally, only disclosing the results of this processing which will typically have reduced associated risks [5, 15, 16]. This provides utility for applications consuming occupancy data while reducing privacy risks.

3 DATABOX

Databox, depicted in Figure 1, is a device designed to collect and mediate access to personal data. We envisage it being instantiated in the form-factor of a set-top box or similar; we prototype it using a Raspberry Pi 3 Model B+. All components are encapsulated as Docker containers.[3] It hosts third-party computations in the form of *Apps*, while external devices such as sensors interface to the Databox via *Drivers* responsible for interacting with the external device through reads and writes to an associated *Store*, a lightweight time-series database. For example, a Driver might collect data from a CO_2 sensor and write those data into its associated Store.

Data is isolated within Databox as each Driver may only write to its own Store, and Apps must request permission on installation to be able to access a Store. If the user grants permission, the App receives a set of access tokens (formatted as Macaroons [3]) which it can subsequently present to the Store for verification that the requested access is permitted by the user. Data may only be communicated to a third-party service through a managed component, the *Export Driver* subject to the user granting that permission when installing the App. All communication between components takes place between containers using a CoAP-inspired protocol [4] implemented over ZeroMQ.[4] Data can be shared between a Driver and a App through the database component within a Store or can be relayed through a Store using our middleware layer, Zest. Figure 2 illustrates the latter where a Driver capturing images is able to send them to an App for facial recognition and receive back the results using Zest's notification system. In summary, this sequence involves:

(1) App observes requests on Store endpoint */notification/request/image_capture/**
(2) Driver POSTs image to Store endpoint */notification/request/image_capture/001*
(3) App carries out image processing and POSTs result to Store endpoint */notification/response/image_capture/001*
(4) Driver receives notification containing processed image

The Store acts as a broker between Driver and App to facilitate the communication. This takes place over well defined endpoints on the Store which means both Driver and App can be controlled through access tokens and have no direct way of communicating with one another. The ability to mediate access between Apps/Drivers within Databox using Macaroons is a novel feature within our architecture.

Available Stores are registered on installation in a HyperCat catalogue, through which they can later be discovered by other Apps.[5] Each store provides a RESTful API supporting JSON, text and binary data. Underlying storage is implemented using the Irmin [8] system using a *git*-structured backend. This supports a key design goal of Databox, to provide accountability of data stored and accessed via the commit history of the git-based storage system which provides

[1] *https://www.silklabs.com/*
[2] *https://vivacitylabs.com/*
[3] *https://docker.com/*
[4] *http://zeromq.org/*
[5] *https://hypercat.io*

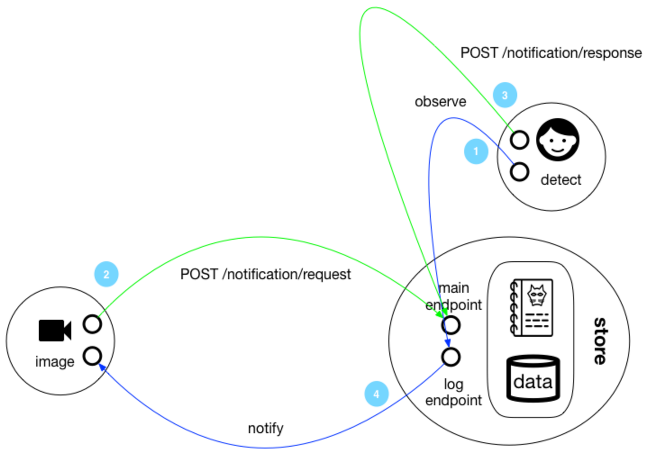


Figure 2: Asynchronous communication between Driver and App.

a detailed account of all mutations to data. Stores can also be replicated and reconciled using git's *clone*, *branch* and *merge* operations. To supplement this information we also use our middleware to publish audit information in real-time to any App permitted to receive it.

Building a times series database on top of a version control system introduces performance complexities. Figure 3 describes the architecture we use to deal with the performance implications of writing frequent amounts of sensor data to a Git. We achieve this by implementing in-memory buffering of incoming messages. Each queue represents an individual data source (sensor) and can be used to serve the most recent read requests directly from memory without disk access. The queue is also used to buffer up writes into chunks that form shards on disk reducing the frequency of commits to Git. A time-oriented index is maintained for each shard. The index is also stored to disk. The in-memory queue has two tunable properties. One represents the maximum size of the queue and the other represents the number of items of the queue to shard when the maximum queue size is reached. Setting these properties depends on the write frequency of the data source (sensor) which is described in the HyperCat. Section 5 provides performance metrics for computation required by OaaS.

4 OCCUPANCY-AS-A-SERVICE

Inside Databox, we implement Occupancy-as-a-Service (OaaS) as a Databox application which provides occupancy data as a service to other applications (e.g., *overcrowd* and *secure entry*) running in Databox as illustrated in figures 4 and 5.

The current version of OaaS makes use of three sensors – camera, motion-sensor, and CO_2 sensors. An App will typically go through the following processing tasks:

(1) Capture an image from the IP camera every unit time specified by the Databox owner or on demand – when motion-sensor detects some human motion.
(2) Detect if there is a face in the frame.
(3) Verify the face in the camera and save the outcome in the App's Store.
(4) Capture CO_2 data from CO_2 sensor periodically and calculate occupancy prediction using pre-trained learning model.
(5) Provide occupancy data in a range of formats for other apps to use, e.g., a simple occupancy visualisation app might display the known, unknown, and total faces detected in the house in the last week using standard time-series functions (last, since, range) to query historical occupancy data.

To enable event driven processing, OaaS uses Databox's middleware *observe* and *notification* functionalities to get data from the motion-sensor local Store. OaaS requests to *observe* an endpoint on the motion-sensor local Store and gets a *notification* every time data is updated.

Use Cases

overcrowd is an application that sends out a notification if the total number of people in an indoor monitored space for a certain period of time exceeds a threshold. For example, a landlord could limit the total number of tenants that are allowed in rented accommodation, both for security and insurance purposes. This App requests access to following OaaS API's:

(1) Latest occupancy count as per calculated by Oaas per unit interval: */ts/oaas/occupancy_count/latest*
(2) All occupancy counts since <timestamp>: */ts/oaas/occupancy_count/since/<timestamp>*
(3) All occupancy counts between two intervals: */ts/oaas/occupancy_count/range/<start>/<end>*

secure entry is another application which notifies when an unknown person (intruder) enters the monitored place. *secure entry* requires a camera and motion-sensor. When motion is detected, it triggers the camera to take pictures and if it detects any unknown faces [17, 20], it sends a notification to an external URL. *secure entry* as a stand-alone App could leak privacy information of known people and their movement activity in the house if motion-sensor data and camera images are sent to the cloud for processing. Databox through OaaS restricts the *secure entry* App to access data from its own Store components and preserves privacy by providing only intruder entry event information. OaaS gets camera and motion-sensor data and stores processed information to its own Store, and *secure entry* App must request permission on installation to be able to access the OaaS Store:

(1) The *secure entry* App requests to observe the OaaS Store endpoint */notification/request/intruder_detected/**
(2) OaaS POSTs image of the intruder with time-stamp to its Store endpoint */notification/request/intruder_detected/001*
(3) The *secure entry* App carries out local processing and POSTs result to its Store endpoint */notification/response/intruder_detected/001*
(4) The *secure entry* App's associated Driver receives notification containing processed image and exports results to an external URL.

In this way, the ***secure entry*** App exports only intruder data and preserves other occupants privacy.

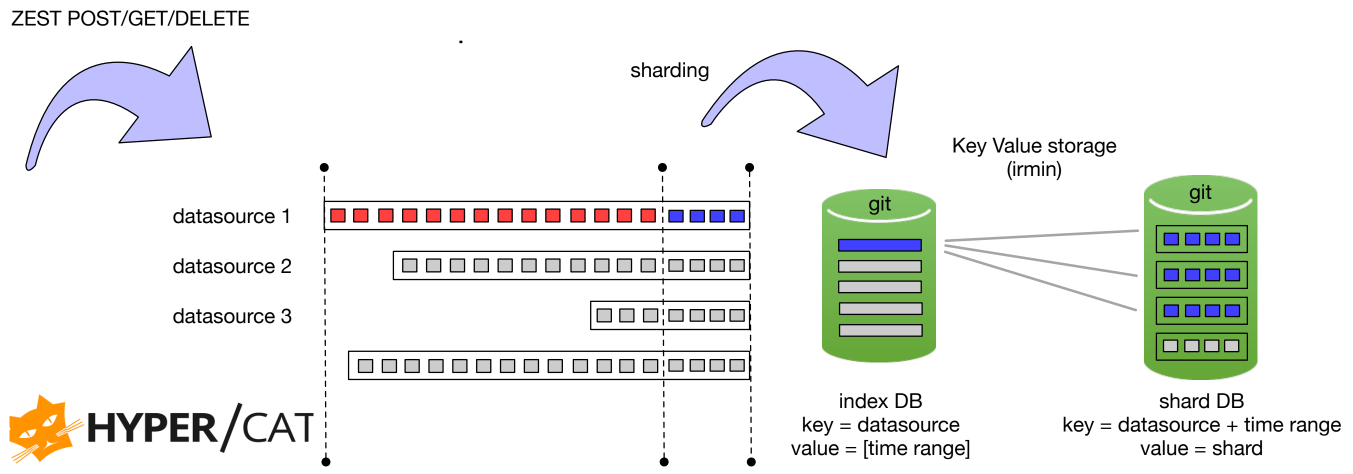


Figure 3: Implementing time-series over *git*. The three datasources correspond to three different sensors (data generators) in Databox.

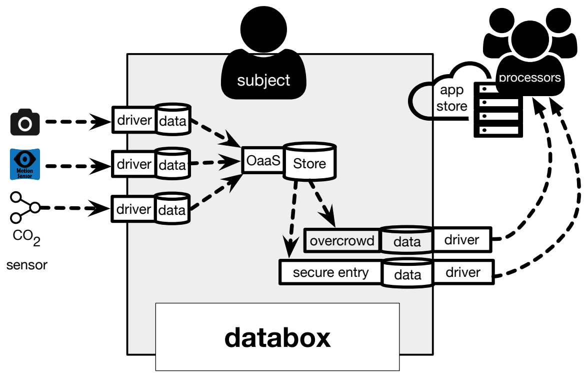


Figure 4: Interaction of OaaS and the Databox Apps, *overcrowd* and *secure entry*.

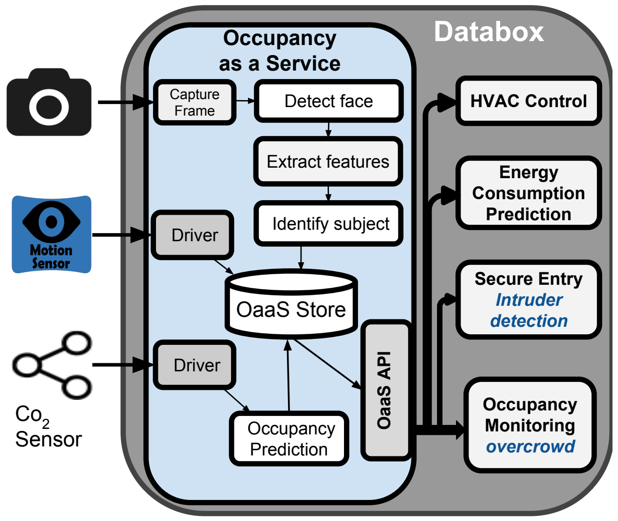


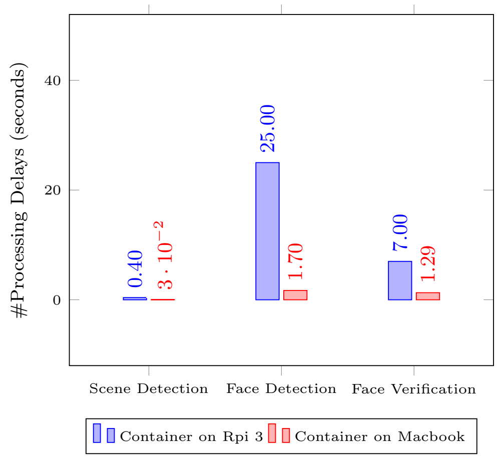
Figure 5: Occupancy-as-a-Service on Databox, providing accurate and privacy-filtered occupancy data via the OaaS API to other Databox-hosted applications.

5 EVALUATION

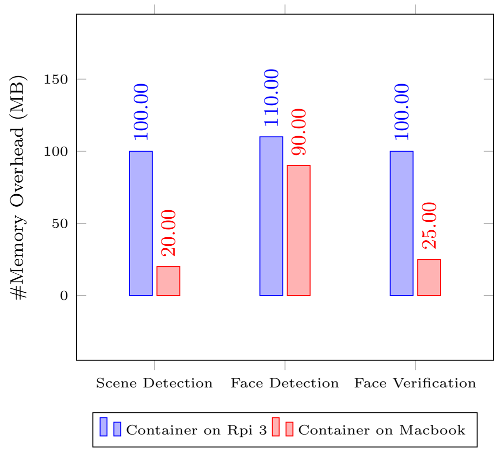
Our preliminary evaluation focuses on the processing requirements of the OaaS App as well as examining the low-level requirements for supporting high-frequency read and writes from our time-series database.

Figures 6a and 6b summarise the processing time and memory overheads associated with each stage of scene detection, face detection and face verification when the OaaS App is run on a Raspberry Pi 3, with comparison against a MacBook Pro. As might be expected, compute intensive activities such as face recognition require significant processing time on the Pi. Nonetheless, the results are provided in a sufficiently timely fashion to be useful, and asynchronous user interface techniques can be used to alleviate perceived delay.

Figures 7a and 7b show how our storage implementation performs on a Raspberry Pi 3 when reading and writing time-series data. Each test involves container-to-container communication instigated by a driver writing to a Store, and an App reading from a Store. This end-to-end test thus involves communication across our middleware layer. A write test involves writing a single data point (value) to storage 10,000 times, while a read test involves a time-series query for the last 100 values repeated 1000 times. The results show encouraging performance for both memory consumption and CPU utilisation across both tests. Further work will examine this performance against other well known time series implementations suitable for our target hardware.

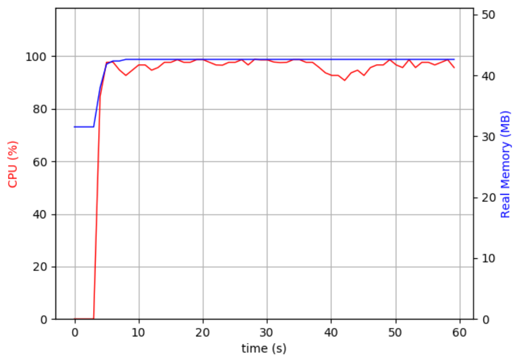


(a) CPU.

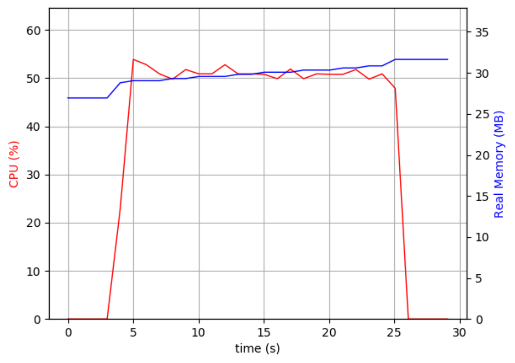


(b) Memory.

Figure 6: Resource consumption of the OaaS App.



(a) Read.



(b) Write.

Figure 7: Store performance.

6 CONCLUSIONS

In this paper we have presented Occupancy-as-a Service (OaaS) implemented as an app on Databox, an edge computing solution supporting movement of data processing code to data, avoiding exposure of data to third-party and the associated potential for breach of privacy. To achieve this, Databox isolates data and computation within containers, mediating access to data using tokens known as Macaroons. In addition, Databox supports data provenance and audit of data access through a storage solution built on top of the git version control system.

Supporting such features and computing at the edge introduces challenges such as utilising the limited processing capabilities of the hardware available. We presented brief evaluation of performance of our OaaS App within a Databox environment on a Raspberry Pi 3. Our initial findings indicate encouraging performance at both App and System (Store) levels. We believe these results show it is possible to implement applications such as occupancy modelling using edge-computing to provide greater privacy for users and without the network usage, storage or processing restrictions typically imposed by traditional cloud-hosted solutions.

ACKNOWLEDGMENTS

Work funded by EPSRC grants EP/N028260/1, EP/M001636/1 and EP/M02315X/1.

REFERENCES

[1] K. Akkaya, I. Guvenc, R. Aygun, N. Pala, and A. Kadri. 2015. IoT-based occupancy monitoring techniques for energy-efficient smart buildings. (2015), 58–63.
[2] Irvan Bastian Arief Ang, Flora Dilys Salim, and Margaret Hamilton. 2016. Human Occupancy Recognition with Multivariate Ambient Sensors. In *CoSDEO: Contact-free Ambient Sensing*. 6.
[3] Arnar Birgisson, Joe Gibbs Politz, òlfar Erlingsson, Ankur Taly, Michael Vrable, and Mark Lentczner. 2014. Macaroons: Cookies with Contextual Caveats for

Decentralized Authorization in the Cloud. In *Network and Distributed System Security Symposium*.

[4] C. Bormann, A. P. Castellani, and Z. Shelby. 2012. CoAP: An Application Protocol for Billions of Tiny Internet Nodes. *IEEE Internet Computing* 16, 2 (March 2012), 62–67. *https://doi.org/10.1109/MIC.2012.29*

[5] Andy Crabtree, Tom Lodge, James Colley, Chris Greenhalgh, Kevin Glover, Hamed Haddadi, Yousef Amar, Richard Mortier, Qi Li, John Moore, Liang Wang, Poonam Yadav, Jianxin Zhao, Anthony Brown, Lachlan Urquhart, and Derek McAuley. 2018. Building accountability into the Internet of Things: the IoT Databox model. *Journal of Reliable Intelligent Environments* 4, 1 (01 Apr 2018), 39–55. *https://doi.org/10.1007/s40860-018-0054-5*

[6] A. Ebadat, G. Bottegal, D. Varagnolo, B. Wahlberg, and K. H. Johansson. 2015. Regularized Deconvolution-Based Approaches for Estimating Room Occupancies. *IEEE Transactions on Automation Science and Engineering* 12, 4 (Oct 2015), 1157–1168. *https://doi.org/10.1109/TASE.2015.2471305*

[7] T. Ekwevugbe, N. Brown, V. Pakka, and D. Fan. 2013. Real-time building occupancy sensing using neural-network based sensor network. In *2013 7th IEEE International Conference on Digital Ecosystems and Technologies (DEST)*. 114–119. *https://doi.org/10.1109/DEST.2013.6611339*

[8] T. Gazagnaire, A. Chaudhry, A. Madhavapeddy, R. Mortier, D. Scott adn D. Sheets, G. Tsipenyuk, and J. Crowcroft. 2014. Irmin: a branch-consistent distributed library database. In *OCaml User and Developer Workshop*.

[9] Sarthak Grover and Roya Ensafi. 2016. *https://freedom-to-tinker.com/2016/01/19/who-will-secure-the-internet-of-things/s*. (2016).

[10] Ruoxi Jia, Roy Dong, Sastry S. Shankar, and Costas J. Spanos. 2017. Privacy-Enhanced Architecture for Occupancy-based HVAC Control. In *In Proceedings of e 8th ACM/IEEE International Conference on Cyber-Physical Systems, Pittsburgh, PA USA*. 10.

[11] Khee Poh Lam, Michael Höynck, Bing Dong, Burton Andrews, Yun shang Chiou, Diego Benitez, and Joonho Choi. 2009. Occupancy detection through an extensive environmental sensor network in an open-plan office building. In *Proc. of Building Simulation 09, an IBPSA Conference*.

[12] Yi Liang, Zhipeng Cai, Qilong Han, and Yingshu Li. 2017. Location Privacy Leakage through Sensory Data. *Security and Communication Networks* (2017), 12. *https://doi.org/10.1155/2017/7576307*

[13] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. 2014. Gyrophone: Recognizing Speech from Gyroscope Signals. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, San Diego, CA, 1053–1067. *https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/michalevsky*

[14] Philipp Morgner, Christian Müller, Matthias Ring, Björn Eskofier, Christian Riess, Frederik Armknecht, and Zinaida Benenson. 2017. Privacy Implications of Room Climate Data. In *ESORICS 2017*. 324–343.

[15] Richard Mortier, Jianxin Zhao, Jon Crowcroft, Liang Wang, Qi Li, Hamed Haddadi, Yousef Amar, Andy Crabtree, James Colley, Tom Lodge, Tosh Brown, Derek McAuley, and Chris Greenhalgh. 2016. Personal Data Management with the Databox: What's Inside the Box?. In *Proceedings of the ACM Workshop on Cloud-Assisted Networking (CAN'16)*. ACM, New York, NY, USA, 49–54. *https://doi.org/10.1145/3010079.3010082*

[16] Databox Project. 2016. EPSRC Project on Privacy-Aware Personal Data Platform. *http://www.databoxproject.uk/*. (2016).

[17] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 815–823. *https://doi.org/10.1109/CVPR.2015.7298682*

[18] Rayman Preet Singh, Benjamin Cassel, S. Keshav, and Tim Brecht. 2016. TussleOS: Managing Privacy Versus Functionality Trade-Offs on IoT Devices. In *Computer Communication Review, 2017*.

[19] Kevin Ting, Richard Yu, and Mani Srivastava. 2013. Inferring Occupancy from Opportunistically Available Sensor Data. In *BuildSys'13*. 1–2.

[20] Poonam Yadav. 2015. Face Prediction Model for an Automatic Age-invariant Face Recognition System. *CoRR* abs/1506.06046 (2015). arXiv:1506.06046 *http://arxiv.org/abs/1506.06046*

[21] Yang Zhao, Jeff Ashe, David Toledano, Brandon Good, Li Zhang, and Adam McCann. 2016. Occupancy and Activity Monitoring with Doppler Sensing and Edge Analytics: Demo Abstract. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM (SenSys'16)*. ACM, New York, NY, USA, 322–323. *https://doi.org/10.1145/2994551.2996543*