UNIVERZA NA PRIMORSKEM

FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

DOKTORSKA DISERTACIJA

(DOCTORAL THESIS)

OMOGOČANJE DECENTRALIZIRANE OBDELAVE PODATKOV Z VAROVANJEM ZASEBNOSTI V SENZORSKIH OMREŽJIH

(ENABLING DECENTRALIZED PRIVACY PRESERVING DATA PROCESSING IN SENSOR NETWORKS)

NIKI HROVATIN

KOPER, 2023

UNIVERZA NA PRIMORSKEM

FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

DOKTORSKA DISERTACIJA

(DOCTORAL THESIS)

OMOGOČANJE DECENTRALIZIRANE OBDELAVE PODATKOV Z VAROVANJEM ZASEBNOSTI V SENZORSKIH OMREŽJIH

(ENABLING DECENTRALIZED PRIVACY PRESERVING DATA PROCESSING IN SENSOR NETWORKS)

NIKI HROVATIN

KOPER, 2023

MENTOR: IZR. PROF. DR. JERNEJ VIČIČ
SOMENTOR: PROF. DR. MICHAEL MRISSA

Acknowledgements

Firstly, my sincere appreciation goes to my mentor, Jernej Vičič, and co-mentor, Michael Mrissa. Your invaluable guidance and dedication have profoundly shaped my work throughout this PhD journey. A special acknowledgment goes to Aleksandar Tošić. Collaborating with you has enriched my academic growth immensely.

My acknowledgment wouldn't be complete without expressing my gratitude to both institutions, University of Primorska Faculty of Mathematics, Natural Sciences and Information Technologies, and Innorenew CoE, for all the support.

To my family, your support has been a constant throughout my life, not just this PhD journey. Your belief in me, even during the challenging moments, has been a driving force. For the love, guidance, and unshakable confidence, I am deeply grateful.

To all my friends, your encouragement and friendship have been pillars of support, enhancing every step of my academic and personal adventures. A special mention to Luka Ghira and Simon Regent, with whom I reached the summit of Mont Blanc.

In the intricacies of this journey, there's a thread of unwavering support that's unmistakably yours, Nika. You've been the calm in my academic storms, the clarity amidst choices, and the joy beyond the confines of this PhD. My gratitude to you is as limitless as the dreams we share for our future.

Contents

Abstract

ENABLING DECENTRALIZED PRIVACY PRESERVING DATA PROCESSING IN SENSOR NETWORKS

In an age of rapid technological advancement, sensor networks have emerged as transformative tools, capturing real-time data from the environment, thereby offering unprecedented insights into various physical phenomena. These networks, now part of our modern ecosystems, hold the potential to reshape several sectors, from healthcare and industry to environmental monitoring. Yet, their accelerating adoption is not without challenges. As these networks become deeply integrated in our daily lives, concerns regarding privacy preservation, efficient data processing, and resilience against centralized points of failure become paramount. Guided by these challenges, this thesis delves deep into the realm of sensor networks, specifically aiming to innovate solutions for decentralized privacy-preserving in-network data processing.

Convolutional Neural Networks (CNNs) stand out for their prowess in image processing tasks. Yet their versatility establishes them as viable choice for various machine learning applications within sensor networks. We begin by investigating how the computational load of convolutional neural networks can be distributed across sensor nodes in a grid-shaped architecture. Our findings suggest that by capitalizing on the kernel shape and local node coordination, the convolutional layers of CNNs can be processed in a distributed manner, enhancing privacy and efficiency of the sensor network while reducing the inference response time.

Further, the thesis presents a privacy-preserving protocol that enables sensor network nodes to jointly compute an arbitrary function without disclosing their own private inputs. The computation takes place at data source nodes, while computation instructions and intermediate results move across the network secured by cryptography. The protocol relies on the Onion Routing technique to provide uniformly distributed network traffic and confine the knowledge a foreign actor can gain from monitoring messages traveling the network. We show that the communication protocol is privacy-preserving against the external and internal attacker models, and we validate our protocol implementation using the NS3 network simulator.

In today's expansive machine learning landscape, where robust data protection is paramount, we explored using our privacy-preserving protocol for distributed machine learning. This method not only maximizes privacy by prioritizing in-situ data processing but also enhances efficiency by performing the machine learning directly within the sensor network. Our findings indicate that the results achieved by training machine learning models with our privacy-preserving protocol align closely with those of the traditional centralized training approach. Moreover, simulation studies demonstrate that applying the privacy-preserving protocol for machine learning leads to a practical response time.

In our research, we evaluated the potential of blockchain technology as a solution for sensor networks, aiming to address the inherent need for trust and traceability in data operations. By integrating blockchain with our privacy-preserving protocol, we established a foundation for a decentralized, privacy-preserving, SPOF-free, and secure in-network data analysis framework. Blockchain's consensus protocol eliminates the need for a third-party intermediary, ensuring decentralized trust. Through this integration, we embed a foundational layer of trust right from the most basic tier—the sensing layer, fundamentally enhancing the potential of sensor networks.

Keywords: Sensor Networks, Privacy, Onion Routing, Distributed Computing, Data Aggregation, Multy-Party Computation.

Povzetek

OMOGOČANJE DECENTRALIZIRANE OBDELAVE PODATKOV Z VAROVANJEM ZASEBNOSTI V SENZORSKIH OMREŽJIH

V dobi hitrega tehnološkega napredovanja so senzorska omrežja postala ključna orodja, ki v realnem času zajamejo podatke iz okolja, s čimer omogočajo podroben vpogled v različne fizične pojave. Ta omrežja, ki so danes nepogrešljiv del našega vsakdana, imajo potencial preoblikovanja mnogih sektorjev, od zdravstva do industrije in okoljskega nadzora. Vendar pa njihova hitra rast prinaša tudi izzive. Ko postanejo senzorska omrežja neločljiv del našega življenja, se povečujejo skrbi glede varstva zasebnosti, učinkovite obdelave podatkov in odpornosti proti centraliziranim točkam napak. Vodena z opisanimi izzivi, se ta doktorska naloga poglobljeno posveča senzorskim omrežjem, z osrednjim ciljem razviti inovativne rešitve za decentralizirano obdelavo podatkov z varovanjem zasebnosti v senzorskih omrežjih.

V disertaciji je preučena možnost porazdeljenega računanja konvolucijskih nevronskih mrež (KNM) na vozliščih senzorskega omrežja. Naša analiza razkriva, da z izkoriščanjem oblike jedra in koordinacijo med lokalnimi vozlišči lahko konvolucijske plasti KNM uspešno obdelamo na decentraliziran način znotraj senzorskega omrežja. Tak pristop znatno povečuje zasebnost in učinkovitost senzorskih omrežij ter skrajšuje odzivni čas sistema pri obdelavi KNM.

Nadalje delo predstavlja komunikacijski protokol, ki omogoča vozliščem senzorskega omrežja skupno računanje brez razkritja svojih zasebnih podatkov. Obdelava podatkov se izvaja neposredno na vozliščih, medtem ko navodila in vmesni rezultati, kriptografsko zaščiteni, potujejo skozi omrežje. Naš protokol temelji na tehniki Onion Routing, ki zagotavlja enakomerno porazdelitev omrežnega prometa in zmanjšuje možnost, da bi zunanji akter pridobil občutljive informacije z opazovanjem prometa. Dokazali smo, da je naš komunikacijski protokol varen proti zunanjim in notranjim grožnjam. Dokaze smo potrdili tudi z empiričnim testiranjem komunicijskega protokol s simulatorjem NS3.

Komunikacijski protokol je bil preizkušen tudi z vidika varovanja zasebnosti pri treniranju modelov strojnega učenja. Rezultati kažejo, da so modeli strojnega učenja, trenirani z uporabo našega protokola, v kakovosti primerljivi s tistimi, treniranimi s tradicionalnim centraliziranim pristopom.

Z integracijo tehnologije veriženja blokov (blockchain) in našega komunikacijskega protokola za varovanje zasebnosti smo postavili temelje za decentralizirano ogrodje za analizo podatkov, ki zagotavlja zasebnost in odpravlja centralne točke okvare (SPOF). To ogrodje je ključnega pomena za zanesljivo, varno obdelavo podatkov z upoštevanjem zasebnosti, pri čemer odpira pot za nadaljnje aplikacije in raziskave na področju senzorskih omrežij.

Ključne besede: Senzorska omrežja, Zasebnost, Čebulno usmerjanje, Porazdeljeno računanje, Agregacija podatkov, Večstransko izračunavanje.

Chapter 1

Introduction

Sensor networks are nowadays an integral part of the current technological landscape. The concept gained popularity due to critical advancements in wireless communication, embedded electronics, and sensing technology that led to the development of low-cost, energy-efficient, tiny devices capable of sensing, processing, and transmitting data. In the literature, these small, low-power devices are commonly referred to as sensor nodes. They are usually arranged in large numbers and share a joint (typically wireless, but also sometimes wired) communication medium, thus forming a sensor network [2]. In a traditional sensor network architecture, sensor nodes perform measurements or detect events and transmit the sensed data to sink nodes. A sink node serves as a central point for data collection from the sensor nodes and then usually forwards the data to external systems, such as a base station or a remote server, for persistent storage, processing, and analysis. Nowadays, sensor networks are used in a wide range of applications, from environmental monitoring, healthcare to industrial automation, and have the potential to revolutionize the way we collect and analyze data, as well as how we understand and interact with the surrounding environment.

Despite their many benefits, sensor networks also have several limitations that must be taken into account [2]. One of the main limitations is the limited power and processing capabilities of sensor nodes, which can constrain the amount and complexity of data that can be collected, processed, and transmitted. Additionally, sensor networks can be vulnerable to various types of failures, such as node failures, network partitioning, and communication disruptions, which can lead to data loss, delays, and reduced reliability. Another limitation is the potential for security and privacy breaches, as sensor networks often collect sensitive information and are exposed to various types of attacks [17].

The aforementioned limitations are particularly prominent in the traditional centralized sensor network architecture, where nodes collect and transmit raw data to a remote system outside the sensor network for processing and analysis. Thereby, the current sensor network implementations are transitioning towards a decentralized architecture where the sensor network is sensing, processing, storing, and responding with information instead of data when inquired [13, 6, 19]. This approach is driven by the accelerating advancements in computing power and efficiency of microelectronics and the emergence of edge computing [16]. Edge computing is a distributed computing paradigm that involves performing computational tasks as close as possible to the data source, such as on the "edge" of the network. In the context of sensor networks, edge computing refers to processing data locally on the sensor nodes themselves or on nearby devices. By leveraging the geographical proximity of data and computational tasks, edge computing brings several advantages to sensor networks.

First, by performing data processing and analysis close to the data origin, the amount of data that needs to be transmitted across the network is significantly reduced. This is particularly important in resource-constrained settings since it increases network efficiency and reduces network congestion, leading to a more efficient and reliable system. Moreover, by processing data locally, instead of depending on distant data centers, latency is reduced, enhancing system responsiveness.

Second, edge computing enables distributing processing tasks across multiple nodes, therefore preventing both the centralization of data, and the appearance of a Single Point Of Failure (SPOF) in the network. Furthermore, through redundancy and load balancing algorithms, if a device or node fails, the tasks can be seamlessly redirected to another node; therefore, improving the overall robustness and reliability of the network.

Finally, edge computing globally enhances the privacy and security aspects of sensor networks. By keeping data processing and analysis on the nodes themselves, edge computing minimizes the exposure of sensitive data to potential attackers or eavesdroppers, reducing the risk of data breaches and privacy violations.

Despite the benefits of edge computing for sensor networks, current implementations of distributed computing frameworks have limitations. For instance, there is no universal distributed computing framework that supports arbitrary computation and is secure and efficient while being applicable to sensor networks. While there are some existing distributed computing frameworks for sensor networks that are optimized for specific use cases, such as data aggregation [1], query processing [13] or machine learning [5], they are tailored for specific tasks and could face challenges when applied to different types of computations. Additionally, some frameworks may lack the necessary security features to protect sensitive data in sensor networks, while others may prove too resource-intensive to operate in resource-constrained environments. As a result, there is a need for research into developing new distributed computing solutions that can address these limitations and provide a secure, efficient, and versatile distributed computing platform for sensor networks.

Privacy is a major concern in sensor networks, the problem surges since sensed features relate to people's activities in the monitoring environment; therefore, the collected data is sensitive and deserves adequate security measures [15]. Although applying the edge computing concept and moving computational tasks as close as possible to the data origin can help mitigate potential privacy flaws, privacy is still an open problem since some information must always travel to the network endpoints to inform other parties [17]. Moreover, performing computational tasks as close as possible to the data origin on sensor network nodes can pose risks to data confidentiality. Specifically, in applications where multiple nodes share data for computations. During such tasks, nodes handle and decrypt data from other nodes for processing, giving them access to potentially sensitive information from their peers. The relevance of this issue is particularly emphasized by the sensor network settings, where nodes are typically left unattended after deployment, leaving them vulnerable to tampering by malicious actors. Accordingly, an attacker could tamper a small set of sensor nodes to obtain data from a much larger one.

In the context of sensor networks, this problem was tackled with different approaches. A large number of studies focus on the computational task of in-network data aggregation [20]. The proposed solutions are efficient and effectively preserve privacy; however, the computational task is often limited to simple aggregate functions and barely adaptable to other computational tasks.Other literature derive from the research on

Byzantine fault-tolerant systems describing techniques based on homomorphic encryption [3] or secure multiparty computation [21]. However, fundamental protocols are computationally intensive and induce significant communication overhead, rendering them impractical for use in sensor networks.

Despite the many advancements in cryptography, privacy remains a critical concern in sensor networks, particularly with respect to applications harnessing the edge computing paradigm. Existing solutions have failed to provide an effective, practical, and versatile approach that enables in-network data processing while preserving privacy. Therefore, a significant research gap exists, demanding further investigation.

The primary objective of this thesis revolves around the exploration of decentralized, privacy-preserving approaches for in-network data processing within sensor networks. The research aims towards a framework that supports machine learning and allows machine learning models to be trained and executed on the network nodes while preserving the privacy of sensitive data against eavesdropping and attacks based on node tampering.

While traditional sensor networks often depend on aggregator nodes, this model introduces vulnerabilities. The use of aggregator nodes creates a Single Point Of Failure (SPOF) from both technical and privacy perspectives. While such architecture is scalable and efficient, it can be vulnerable to security breaches or network failures, potentially leading to significant data loss and system downtime. Instead, we focus on a decentralized approach without aggregator nodes, where sensor nodes all have the same role and behave in the same way, eliminating any SPOF.

In this thesis, we introduce two innovative approaches for distributing computational tasks among sensor network nodes to leverage the advantages of the edge computing paradigm and the collective processing power of the network. Central to both strategies is their decentralized architecture enabling sensor networks to handle computationally demanding applications such as machine learning without relying on external systems while minimizing data exposure by processing it as close as possible to its source. To evaluate the benefits and drawbacks of the two approaches, as well as their potential for enabling efficient, effective, and privacy-preserving data processing, we carried out analytical studies and ran simulations using the ns-3 simulator [8].

1.1 In-Network Convolution in Grid Shaped Sensor Networks

The computation load of a deep Convolutional Neural Network (CNN) can be very high, making it infeasible to perform inference on a single node of a sensor network. Moreover, the use of a CNN model sources data from several sensor nodes to perform inference, which leads to sensitive data being gathered at a central processing point, endangering the data privacy of contributing nodes. This poses a significant challenge of how to distribute the computation load of a CNN among sensor nodes for real-time inference processing.

While there are several existing solutions [4, 14] for distributing the computational load of feedforward artificial neural networks on sensor network nodes, the research on distributing the convolutional layers is still lacking. CNNs are known for their effectiveness in various applications that involve grid-shaped data (e.g., image and video analysis); however, their use in the context of grid-shaped sensor networks has not been fully explored.

In the context of this thesis, we designed a SPOF-free solution for distributing the computation load of a CNN's convolutional layer across sensor nodes in a grid-shaped sensor network. Our approach leverages the inherent structure of CNN kernels and local node coordination to perform inference processing close to the source of the data. This minimizes the exposure of sensitive data during transmission across the network. The applicability of this solution is broad, as it has the potential to enhance both privacy and efficiency in any grid-shaped sensor network application that can benefit from the power of CNNs, such as activity tracking, walking gait analysis, fall detection, or various types of automation systems.

1.2 A General Purpose Privacy Preserving Protocol for Sensor Networks

Privacy is a paramount concern in sensor networks due to the collected data being sensitive. Traditional encryption techniques can secure data during transmission; however, applications that process data from multiple nodes have access to cleartext data. Research to date has primarily focused on ensuring confidentiality during in-network data processing through data transformation using cryptography or other obfuscation techniques. This approach enables several nodes to operate on the transformed data without the possibility of extracting private data from other nodes involved in the processing. While this approach has led to the development of several solutions, in solutions considered practical for sensor networks, the technique used to conceal data restricts the set of operations applicable to the transformed data.

To enhance privacy protection in sensor networks, we propose a solution that moves computation rather than sensitive data using layered cryptography. Our approach conveys computation instructions and partial results for in-situ processing on nodes of a sensor network, eliminating the need for sensitive data transfer and enhancing privacy protection against eavesdropping and node tampering. Additionally, our approach enables arbitrary computation to be performed on sensor network nodes, facilitating privacy-preserving machine learning and data retrieval and pushing research toward a decentralized, privacy-preserving, SPOF-free, and secure in-network data analysis framework for sensor networks.

1.3 Research Questions and Hypothesis

RQ-1 and H-1

RQ-1: Can the grid-shaped structure of a sensor network be leveraged to distribute the computational load of CNN's convolutional layers among sensor nodes?

H-1: By leveraging the kernel shape and local node coordination, the convolutional layer of a CNN can be processed in a distributed manner across sensor nodes in a grid-shaped wired sensor network.

Addressed through the following contributions:

- Data about fall events and ordinary daily activities from a sensorized smart floor
- In-Network Convolution in Grid Shaped Sensor Networks

RQ-2 and H-2

RQ-2: How can privacy preservation in sensor networks be enhanced through the anonymous relocation of computation tasks, instead of transferring sensitive data?

H-2: Leveraging layered cryptography can enable anonymous in-situ computation on sensor network nodes, eliminating the need for sensitive data transfer and thereby enhancing privacy protection against eavesdropping and node tampering.

Addressed through the following contributions:

- A General Purpose Data and Query Privacy Preserving Protocol for Wireless Sensor Networks
- PPWSim: Privacy preserving wireless sensor network simulator

RQ-3 and H-3

RQ-3: Is it possible for a decentralized privacy preserving ML approach that executes the ML model in-situ on a sensor node and moves the model across several nodes within the sensor network to achieve comparable results to traditional batch training approaches?

H-3: Conveying a data stream mining model through nodes of a sensor network for training and inference will result in comparable prediction accuracy to a model trained through traditional batch training.

Addressed in the contribution: Privacy-Preserving Data Mining on Blockchain-Based WSNs

RQ-4 and H-4

RQ-4: Is it possible to implement a decentralized, privacy-preserving, SPOF-free, and secure in-network data analysis framework for sensor networks?

H-4: Utilizing layered cryptography for the anonymous conveyance of computation tasks and leveraging blockchain technology as a decentralized fabric of trust can enable the implementation of a decentralized privacy preserving, SPOF-free, and secure in-network data analysis framework.

Addressed in the contribution: Privacy-Preserving Data Mining on Blockchain-Based WSNs

Chapter 2

Published Papers

2.1 Paper 1

Title: Data about fall events and ordinary daily activities from a sensorized smart floor
Authors: Aleksandar Tošić, Niki Hrovatin, Jernej Vičič
Year: 2021
Journal: Data in Brief
DOI: `https://doi.org/10.1016/j.dib.2021.107253`
Link:`https://www.sciencedirect.com/science/article/pii/S2352340921005370`

Data in Brief 37 (2021) 107253

Contents lists available at ScienceDirect

Data in Brief

journal homepage: www.elsevier.com/locate/dib

Data Article

Data about fall events and ordinary daily activities from a sensorized smart floor

Aleksandar Tošić[a,b,*], Niki Hrovatin[a,b], Jernej Vičič[a,c]

[a] Faculty of Mathematics, Natural Sciences and Information Technologies, University of Primorska, Glagoljaška 8, Koper SI-6000, Slovenija
[b] InnoRenew CoE, Livade 6, Izola 6310, Slovenija
[c] Research Centre of the Slovenian Academy of Sciences and Arts, The Fran Ramovš Institute, Novi trg 2, Ljubljana 1000, Slovenija

ARTICLE INFO

Article history:
Received 15 March 2021
Revised 24 May 2021
Accepted 1 July 2021
Available online 6 July 2021

Keywords:
Fall detection
Machine learning
Elderly
Smart floor
Sensor networks

ABSTRACT

A smart floor with 16 embedded pressure sensors was used to record 420 simulated fall events performed by 60 volunteers. Each participant performed seven fall events selected from the guidelines defined in a previous study. Raw data were grouped and well organized in CSV format.
The data was collected for the development of a non-intrusive fall detection solution based on the smart floor. Indeed, the collected data can be used to further improve the current solution by proposing new fall detection techniques for the correct identification of accidental fall events on the smart floor.
The gathered fall simulation data is associated with participants' demographic characteristics, useful for future expansions of the smart floor solution beyond the fall detection problem.

* Corresponding author at: Faculty of Mathematics, Natural Sciences and Information Technologies, University of Primorska, Glagoljaška 8, SI-6000 Koper, Slovenija.
E-mail addresses: aleksandar.tosic@upr.si, aleksandar.tosic@innorenew.eu (A. Tošić).

https://doi.org/10.1016/j.dib.2021.107253

Specifications Table

Subject	Computer Science Applications
Specific subject area	Fall detection, the act of differentiating ordinary daily activities from the accidental fall event.
Type of data	Table
How data were acquired	Data were acquired using the smart floor, a floor surface with embedded Force Sensing Resistors (FSR). The smart floor has 16 FSR sensors linked to an ArduinoMega microcontroller. Fall data is recorded using a Java program on a personal computer linked to the ArduinoMega. The Java program and the Arduino sketch are provided as supplementary material. Additional data was acquired using the questionnaire provided as supplementary material under the file name *questionnaireExample.pdf*.
Data format	Raw
Parameters for data collection	The data gathering was organized as an open event, where anyone could volunteer to contribute. We have not established any constraint on the age or physical traits of participants. Each participant performed 7 different fall events.
Description of data collection	The data were collected at a data gathering event organized in a gym. Each participant was asked to simulate 7 different fall events. The data was recorded using the smart floor [4]. A proprietary program that collects data from sensors and manages the data gathering process was used in the experiment. The program is added as part of the dataset, but it is also accessible on Gitlab: https://gitlab.com/Dormage/smart-floor-fall-detection. The version tag *031194d73413a7bbdb68825236bd96f457735b30* was used in data gathering process. A total of 420 fall events were recorded.
Data source location	Koper - Slovenia
Data accessibility	Repository name: Zenodo Data identification number: https://doi.org/10.5281/zenodo.4605619 Direct URL to data: https://zenodo.org/record/4605619 Instructions for accessing these data: unzip the archive, all data is distributed in folders for easy access. There are two main folders: dataset and program. Data is distributed in csv format, each line represents one experiment (one person simulating the falls).

Value of the Data

- The data is useful for the development of fall detection systems and new methods to recognize accidental fall events among ordinary daily activities. The data can also be used for the development of new techniques for multivariate time-series analyses.
- Accidental fall events are a significant threat to the health and independence of older adults [1]. Approximately 30% of people aged 65 fall each year, and the odds increase for those aged over 70 years [2]. Hence, the development of fall detection systems is crucial to identify a fall event and provide immediate help.
- The provided participant's demographic data acquired through the questionnaire can be used to explore future expansions of the smart floor solution beyond the fall detection problem. A similar solution [3] was developed to identify a person's unique walking gait over a smart mat monitoring system.
- The gathered fall simulation data can be used to investigate fall patterns, and how a person reacts during a fall event.

1. Data Description

We provide the data in two formats. The raw data as result of the data acquisition process is stored in the folder **raw_data**, and the CSV formatted data, which is a user-friendly representation of the raw data. However, an accurate description of the data set is provided only for CSV formatted data. The CSV formatted data is contained in the folder **csv_data**.

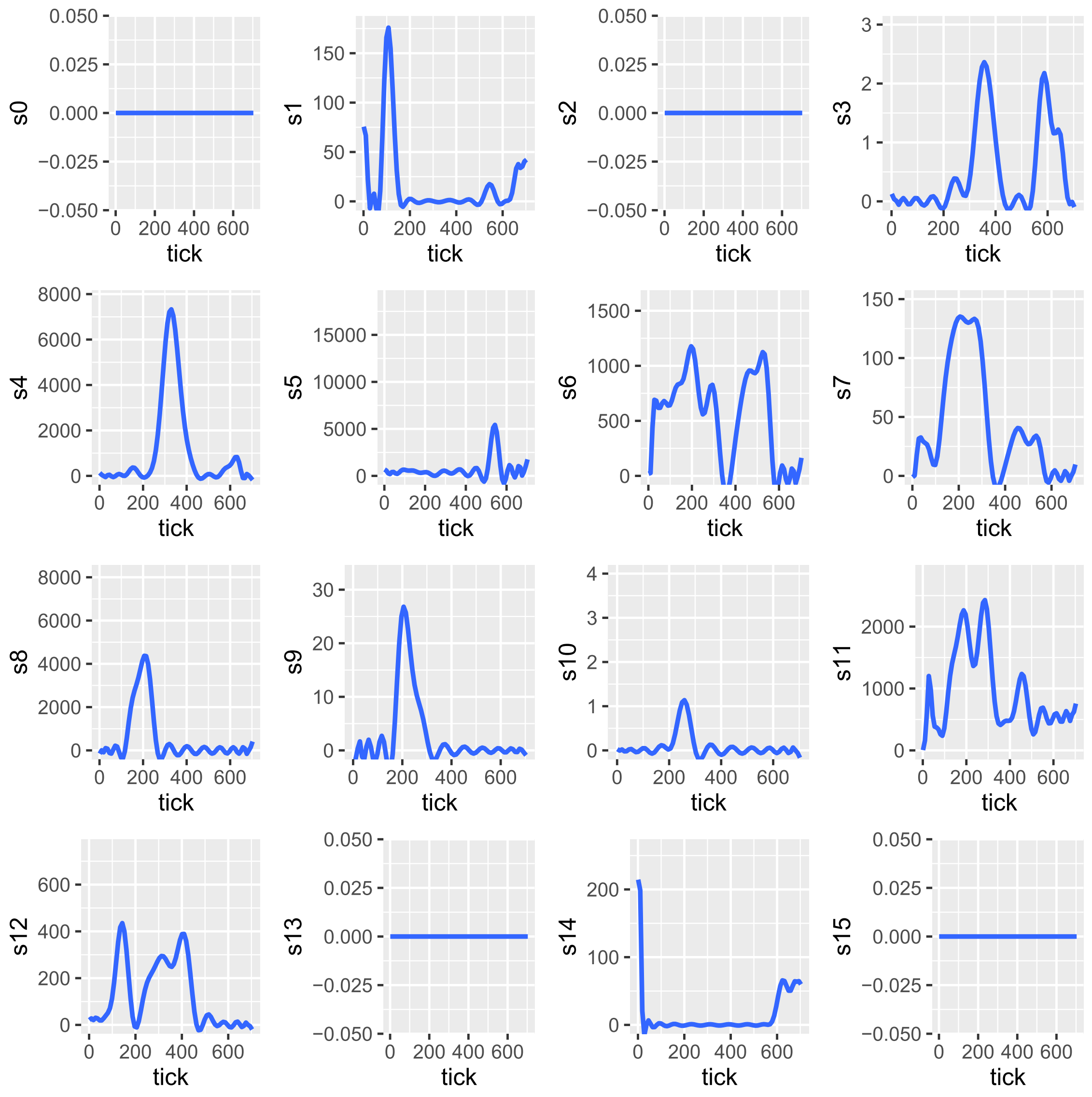


Fig. 1. Visualisation of recorded sensor telemetry during a fall event.

1.1. raw_data

The folder **raw_data** contains raw data obtained as result of the data acquisition process described in Section c. The raw data consists of numerous files. Each file contains the recording of an activity and is associated with a PERSON_ID. The PERSON_ID links the recorded activity with other activities performed under the same PERSON_ID, and with data acquired from the questionnaire.

Pressure sensors are arranged on the smart floor as shown in Fig. 5. An activity recording contains the value of the 16 pressure sensors recorded in a time interval (Fig. 1). Each column represents measurements of a pressure sensor. The first column takes values of the sensor *s*0, the second column takes values of the sensor *s*1, and so on. The rightmost column takes values of the sensor *s*15. Sensor values are increasing proportionally to the force applied on the sensor. Measured values range from 0 to 65535 at the maximum applied force, which is a decimal representation of the 16-bit binary interval provided by the controller in the process of converting the analogue signal from the FRS. The measurements of all 16 pressure sensors is performed at

Table 1
Example of data recorded from 16 pressure sensors contained in the **raw_data** folder.

0 ,	170 ,	0 ,	0 ,	0 ,	0 ,	87 ,	0 ,	345 ,	0 ,	329 ,	0 ,	11 ,	0 ,	0 ,	0
0 ,	174 ,	0 ,	0 ,	0 ,	0 ,	87 ,	0 ,	346 ,	0 ,	319 ,	0 ,	11 ,	0 ,	0 ,	0
0 ,	175 ,	0 ,	0 ,	0 ,	0 ,	87 ,	0 ,	346 ,	0 ,	320 ,	0 ,	9 ,	0 ,	0 ,	0
0 ,	171 ,	0 ,	0 ,	0 ,	0 ,	85 ,	0 ,	349 ,	0 ,	319 ,	0 ,	10 ,	0 ,	0 ,	0
0 ,	167 ,	0 ,	0 ,	0 ,	0 ,	83 ,	0 ,	346 ,	0 ,	327 ,	0 ,	10 ,	0 ,	0 ,	0
0 ,	144 ,	0 ,	0 ,	0 ,	0 ,	15 ,	0 ,	141 ,	0 ,	14965 ,	0 ,	0 ,	0 ,	72 ,	0
0 ,	141 ,	0 ,	0 ,	0 ,	0 ,	19 ,	0 ,	111 ,	0 ,	22146 ,	0 ,	0 ,	0 ,	260 ,	0
0 ,	79 ,	0 ,	0 ,	0 ,	0 ,	5400 ,	0 ,	57 ,	0 ,	17287 ,	0 ,	0 ,	0 ,	325 ,	0
0 ,	0 ,	0 ,	0 ,	0 ,	0 ,	43451 ,	0 ,	15 ,	0 ,	12579 ,	0 ,	0 ,	0 ,	382 ,	0
0 ,	27 ,	0 ,	0 ,	0 ,	0 ,	51240 ,	0 ,	82 ,	0 ,	9981 ,	0 ,	4 ,	0 ,	424 ,	0
0 ,	45 ,	0 ,	0 ,	0 ,	0 ,	25581 ,	0 ,	110 ,	0 ,	7984 ,	0 ,	0 ,	0 ,	452 ,	0
0 ,	21 ,	0 ,	0 ,	0 ,	0 ,	10736 ,	0 ,	115 ,	0 ,	7254 ,	0 ,	0 ,	0 ,	450 ,	0
0 ,	17 ,	0 ,	0 ,	0 ,	0 ,	3879 ,	0 ,	125 ,	0 ,	6922 ,	0 ,	0 ,	0 ,	395 ,	0
0 ,	26 ,	0 ,	0 ,	0 ,	0 ,	1606 ,	0 ,	115 ,	0 ,	6415 ,	0 ,	0 ,	0 ,	421 ,	0
0 ,	30 ,	0 ,	0 ,	0 ,	0 ,	940 ,	0 ,	78 ,	0 ,	5701 ,	0 ,	0 ,	0 ,	414 ,	0
0 ,	27 ,	0 ,	0 ,	0 ,	0 ,	538 ,	0 ,	74 ,	0 ,	5619 ,	0 ,	4 ,	0 ,	408 ,	0
0 ,	24 ,	0 ,	0 ,	0 ,	0 ,	361 ,	0 ,	90 ,	0 ,	5619 ,	0 ,	6 ,	0 ,	423 ,	0
0 ,	25 ,	0 ,	0 ,	0 ,	0 ,	256 ,	0 ,	92 ,	0 ,	5305 ,	0 ,	2 ,	0 ,	433 ,	0
0 ,	4 ,	0 ,	0 ,	0 ,	0 ,	203 ,	0 ,	92 ,	0 ,	4945 ,	0 ,	0 ,	0 ,	428 ,	0
0 ,	7 ,	0 ,	0 ,	0 ,	0 ,	191 ,	0 ,	69 ,	0 ,	4875 ,	0 ,	0 ,	0 ,	420 ,	0

Table 2
Name and description of attributes in the *positiveSet.csv* file.

Attribute	Description
FALL_ID	uniquely identifies the fall in the data file *positiveSet.csv*
PERSON_ID	uniquely identifies the volunteer who has simulated the fall
FALL_CATEGORY	identifies the fall execution
TICK	elapsed time of the recording, each tick counts 10 milliseconds
S0...S15	takes the value of sensors on the smart floor ranging from 0 to 65535

the same time. Measurements are collected every 10 milliseconds. An example of raw data from 16 pressure sensors is provided in Table 1.

1.2. *csv_data*

The folder **csv_data** contains CSV formatted data from the **raw_data** folder. The raw data was not filtered or altered. We just added headers and categorization attributes to enhance the dataset's reusability. Each volunteer has a unique id (PERSON_ID) associated with it's data. The PERSON_ID attribute links a volunteer across the data in the following files:

- *positiveSet.csv*– data about simulated fall events
- *negativeSet.csv*– data about ordinary daily activities
- *testSet.csv*– data about ordinary daily activities that might cause false positives
- *surveyData.csv*– data about participants obtained using a questionnaire

The content of the folder **csv_data** is detailed in the sections below.

1.3. *csv_data/positiveSet.csv*

The file contains CSV formatted data from the raw data in the folder *raw_data/positive*. The file stores 420 simulated fall events recorded in a time interval (Table 4). In Table 2 we give the description of each attribute.

Table 3
Description of different fall executions identified by the FALL_CATEGORY attribute. Ending position of the described fall events is depicted in Fig. 6.

FALL_CATEGORY	Description of the fall execution	Duration (s)
1	forward fall on the knees	5
2	forward fall with forward arm protection	5
3	forward fall ending laying flat	5
4	forward fall on the knees with rotation, ending in the lateral position	5
5	lateral fall ending laying flat	5
6	lateral fall ending laying flat with recovery	10
7	forward fall ending laying flat with recovery	10

Table 4
Example of data contained in positiveSet.csv. The first row details attribute names. The data is similarly structured also for negativeSet.csv and testSet.csv.

fallID	personID	fall_cat.	tick	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
1	1	1	0	0	137	0	0	0	0	92	0	409	0	367	0	23	0	0	0
1	1	1	1	0	139	0	0	0	0	93	0	408	0	367	0	23	0	0	0
1	1	1	2	0	140	0	0	0	0	94	0	409	0	363	0	23	0	0	0
1	1	1	3	0	142	0	0	0	0	93	0	411	0	350	0	23	0	0	0
1	1	1	4	0	142	0	0	0	0	93	0	411	0	340	0	24	0	0	0
1	1	1	5	0	140	0	0	0	0	94	0	411	0	332	0	23	0	0	0
1	1	1	6	0	138	0	0	0	0	94	0	411	0	328	0	23	0	0	0
1	1	1	7	0	139	0	0	0	0	92	0	411	0	329	0	24	0	0	0
1	1	1	8	0	137	0	0	0	0	92	0	411	0	327	0	24	0	0	0
1	1	1	9	0	140	0	0	0	0	92	0	413	0	331	0	24	0	0	0
1	1	1	10	0	139	0	0	0	0	93	0	413	0	332	0	24	0	0	0
1	1	1	11	0	140	0	0	0	0	94	0	413	0	331	0	24	0	0	0
1	1	1	12	0	140	0	0	0	0	96	0	414	0	332	0	24	0	0	0
1	1	1	13	0	141	0	0	0	0	96	0	413	0	335	0	24	0	0	0
1	1	1	14	0	141	0	0	0	0	96	0	413	0	339	0	24	0	0	0
1	1	1	15	0	142	0	0	0	0	96	0	413	0	339	0	24	0	0	0
1	1	1	16	0	144	0	0	0	0	95	0	413	0	342	0	24	0	0	0
1	1	1	17	0	143	0	0	0	0	95	0	413	0	344	0	23	0	0	0
1	1	1	18	0	143	0	0	0	0	96	0	413	0	347	0	24	0	0	0
1	1	1	19	0	144	0	0	0	0	96	0	414	0	351	0	24	0	0	0
1	1	1	20	0	142	0	0	0	0	96	0	414	0	351	0	24	0	0	0
1	1	1	21	0	142	0	0	0	0	96	0	414	0	357	0	24	0	0	0
1	1	1	22	0	142	0	0	0	0	96	0	414	0	359	0	24	0	0	0
1	1	1	23	0	143	0	0	0	0	95	0	416	0	360	0	24	0	0	0
1	1	1	24	0	144	0	0	0	0	95	0	414	0	357	0	23	0	0	0
1	1	1	25	0	144	0	0	0	0	95	0	414	0	360	0	24	0	0	0
1	1	1	26	0	143	0	0	0	0	95	0	413	0	360	0	24	0	0	0
1	1	1	27	0	143	0	0	0	0	96	0	414	0	361	0	24	0	0	0
1	1	1	28	0	143	0	0	0	0	95	0	411	0	350	0	24	0	0	0

Each volunteer simulated 7 different fall events. Each fall was performed following a different fall execution. The attribute FALL_CATEGORY is used to identify the fall execution. In Table 3, we describe the fall execution for each value of the attribute FALL_CATEGORY. The length of the recording time interval is expressed in seconds in the column duration.

1.4. *csv_data/negativeSet.csv*

The file contains CSV formatted data from the raw data in the folder *raw_data/negative*. The file stores 30 ordinary daily activities recorded in a time interval. In Table 5, we give the description of each attribute.

Table 5
Name and description of attributes in the *negativeSet.csv* file.

Attribute	Description
NEG_ID	uniquely identifies the ordinary daily activity in the data file *negativeSet.csv*
PERSON_ID	uniquely identifies the volunteer who has simulated the ordinary daily activity
NEG_CATEGORY	identifies the type of ordinary daily activity
TICK	elapsed time of the recording, each tick counts 10 milliseconds
S0...S15	takes the value of sensors on the smart floor ranging from 0 to 65535

Table 6
Description of different ordinary daily activities identified by the NEG_CATEGORY attribute.

NEG_CATEGORY	Description of the ordinary daily activity	Duration
1	random walking and random stop	8 minutes
2	empty floor	10 seconds
3	one step forward then waiting 5 seconds in position, repeat	1 minute
4	random walking	10 seconds

Table 7
Name and description of attributes in the *testSet.csv* file.

Attribute	Description
TEST_ID	uniquely identifies the ordinary daily activity in the data file *testSet.csv*
PERSON_ID	uniquely identifies the volunteer who has simulated the ordinary daily activity
TEST_CATEGORY	identifies the type of ordinary daily activity
TICK	elapsed time of the recording, each tick counts 10 milliseconds
S0...S15	takes the value of sensors on the smart floor ranging from 0 to 65535

Table 8
Description of different ordinary daily activities identified by the TEST_CATEGORY attribute.

TEST_CATEGORY	Description of the ordinary daily activity	Duration
1	a chair is positioned on the smart floor, and the volunteer will sit on the chair	5 seconds
2	a volunteer is sitting on a chair positioned on the smart floor, the volunteer will stand up from the chair	5 seconds
3	a volunteer will bend down and catch something on the smart floor	5 seconds
4	a volunteer will jump on the smart floor	5 seconds

The file consists of 4 different types of ordinary daily activities. Each activity type is identified by the attribute NEG_CATEGORY.

In Table 6, we describe the ordinary daily activity type for each value of the attribute NEG_CATEGORY. The length of the recording time interval is expressed in the column duration.

1.5. *csv_data/testSet.csv*

The file contains CSV formatted data from the raw data in the folder *raw_data/test*. The file stores 12 ordinary daily activities recorded in a time interval. In Table 7, we give the description of each attribute.

The file consists of 4 different types of ordinary daily activities. Each activity type is identified by the attribute TEST_CATEGORY. In Table 8, we describe the ordinary daily activity type for each

Table 9
Basic summary of the participants (surveyData.csv).

variable	min	max	mean	median	sd	n	q25	q75
age	12	51	28.63	27	8.07	60	22	33.25
weight	40	120	72.73	72	16.22	60	60	83.5
height	150	197	174.98	176	11.69	60	165.75	183.25
sportActive	1	5	3.1	3	1.1	60	2	4
worried	1	3	1.58	1	0.72	60	1	2
fallEvents	0	4	1.25	1	1.45	60	0	2

Table 10
Example of data contained in (surveyData.csv). The first row details attribute names.

person_ID	sex	age	weight	height	sportActive	worried	fallEvents
1	M	21	95	190	3	1	1
2	M	24	80	188	5	1	0
3	M	23	85	190	4	1	2
4	F	15	50	156	5	1	0
5	F	33	74	160	4	3	1
6	M	22	83	192	3	3	0
7	F	12	40	150	5	1	4
8	M	26	90	182	4	1	4
9	F	20	101	168	2	2	1
10	M	28	93	180	2	1	1

value of the attribute TEST_CATEGORY. The length of the recording time interval is expressed in the column duration.

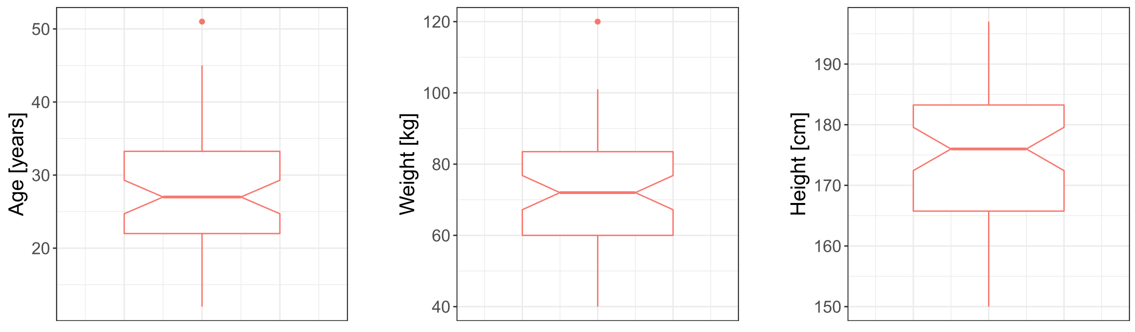
1.6. csv_data/surveyData.csv

The file contains CSV formatted data obtained using a questionnaire. An example copy of the questionnaire is provided under the filename *questionnaireExample.pdf*. Every volunteer fulfilled the questionnaire before the data acquisition. Data from the questionnaire is linked through the attribute PERSON_ID with the data in the following files: *positiveSet.csv*, *negativeSet.csv* and *test-Set.csv*.

The file *surveyData.csv* contains the following basic demographic data: sex (m/f), age (years), weight (kg) and height (cm) (Table 10). The attribute PERSON_ID uniquely identifies the volunteer. The attribute SPORTACTIVE represents the self evaluation of sport activity, ranging from (1-not active) (5-very active). The attribute WORRIED represents anxieties linked with the data acquisition process, ranging from (1-not at all) to (5-very worried). The attribute FALLEVENTS represents the number of fall events experienced by the volunteer during this year ranging from (0 - zero fall events) to (4 - four or more). A summary of the dataset is provided in Table 9 and Fig. 2.

2. Experimental Design, Materials and Methods

Data were acquired using the smart floor displayed in Fig. 3, and described in [4]. The smart floor has 16 embedded Force Sensing Resistor (FSR) sensors linked to analog inputs of an ArduinoMega microcontroller. The ArduinoMega runs the *code/readData.ino* program, which triggers sensor reading every 10 milliseconds. Sensor data is sent to a personal computer linked to the ArduinoMega via serial communication. The java based client records the sensor data provided in *code/dataCollection*. The whole data collection set-up is shown in Fig. 4.



(a) (b) (c)

Fig. 2. The collected demographic data represented in boxplots.

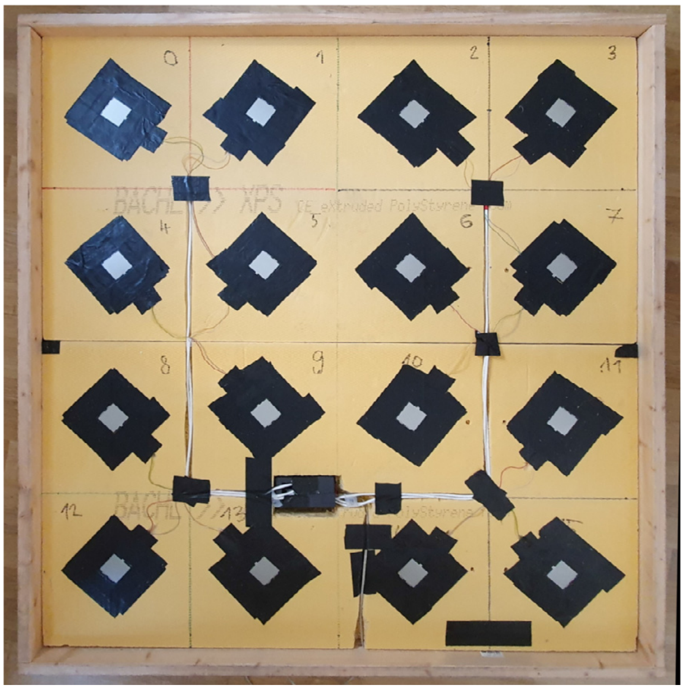


Fig. 3. The smart floor without the laminate layer. In the picture are displayed the 16 FSR pressure sensors, and the enclosure of the ArduinoMega.

The data gathering process was conducted in a properly equipped gym as depicted in Fig. 4. Each participant was informed orally and in written form about the aims of the experiment and possible risk hazards. Adequate protections for elbows and knees were offered to participants. Before the fall simulation, each participant fulfilled the questionnaire *questionnaireExample.pdf*.

Each participant was asked to simulate 7 different fall events on the smart floor surface. The fall events were selected from the article [5], which tackles the problem of fall simulation. Selected fall events are described in Table 3. All simulated fall events were recorded following the next procedure:

(note: we refer to the person that conducts the data collection as the data collector)

1. The participant stand on the right side of the smart floor as shown in Fig. 5, and waits for the signal. The participant must not step on the smart floor.
2. The data collector starts recording with the laptop and signals the participant.
3. The participant simulate the fall, and holds the ending position. The position must be maintained as if a real debilitating fall occurred.

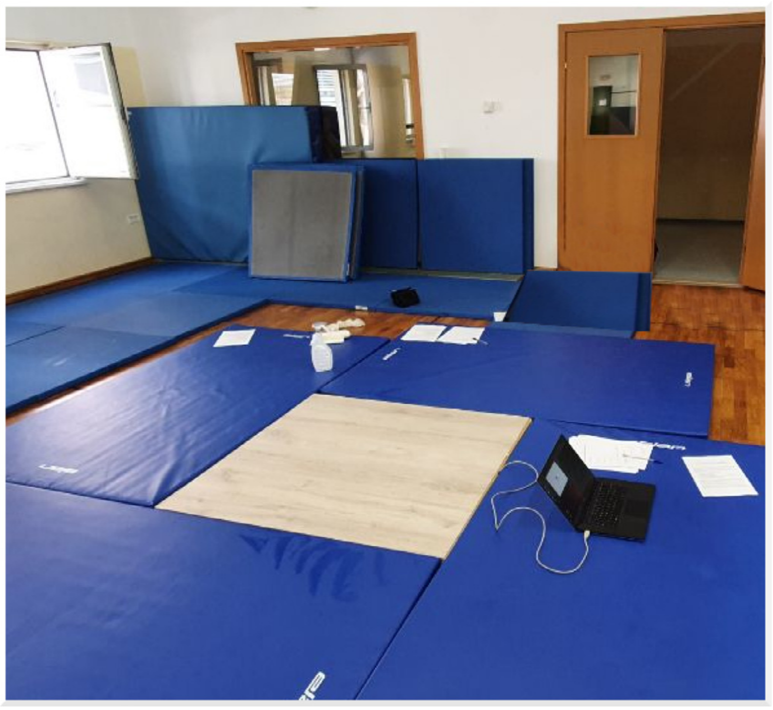


Fig. 4. Data collection set-up: The smart floor is the white square surface surrounded by landing mats. It differs from Fig. 3, because covered with the laminate layer. Notice the laptop linked to the smart floor for data acquisition.

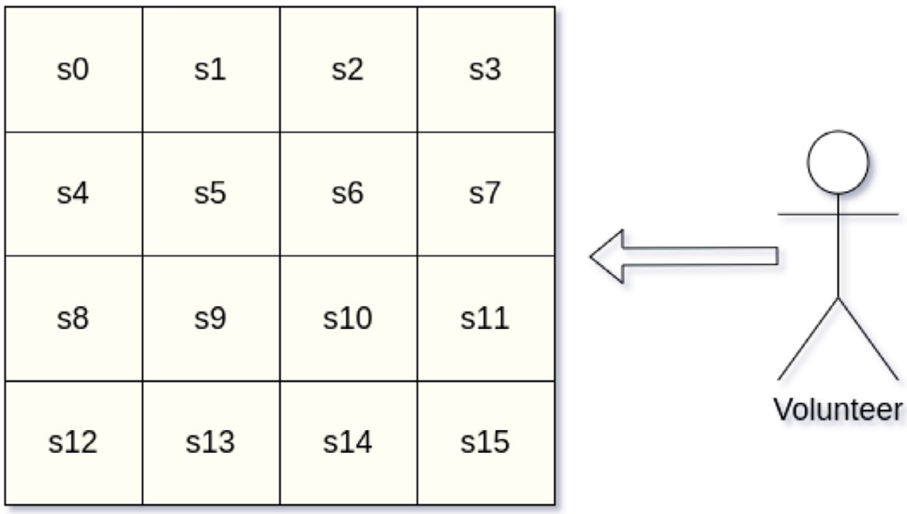
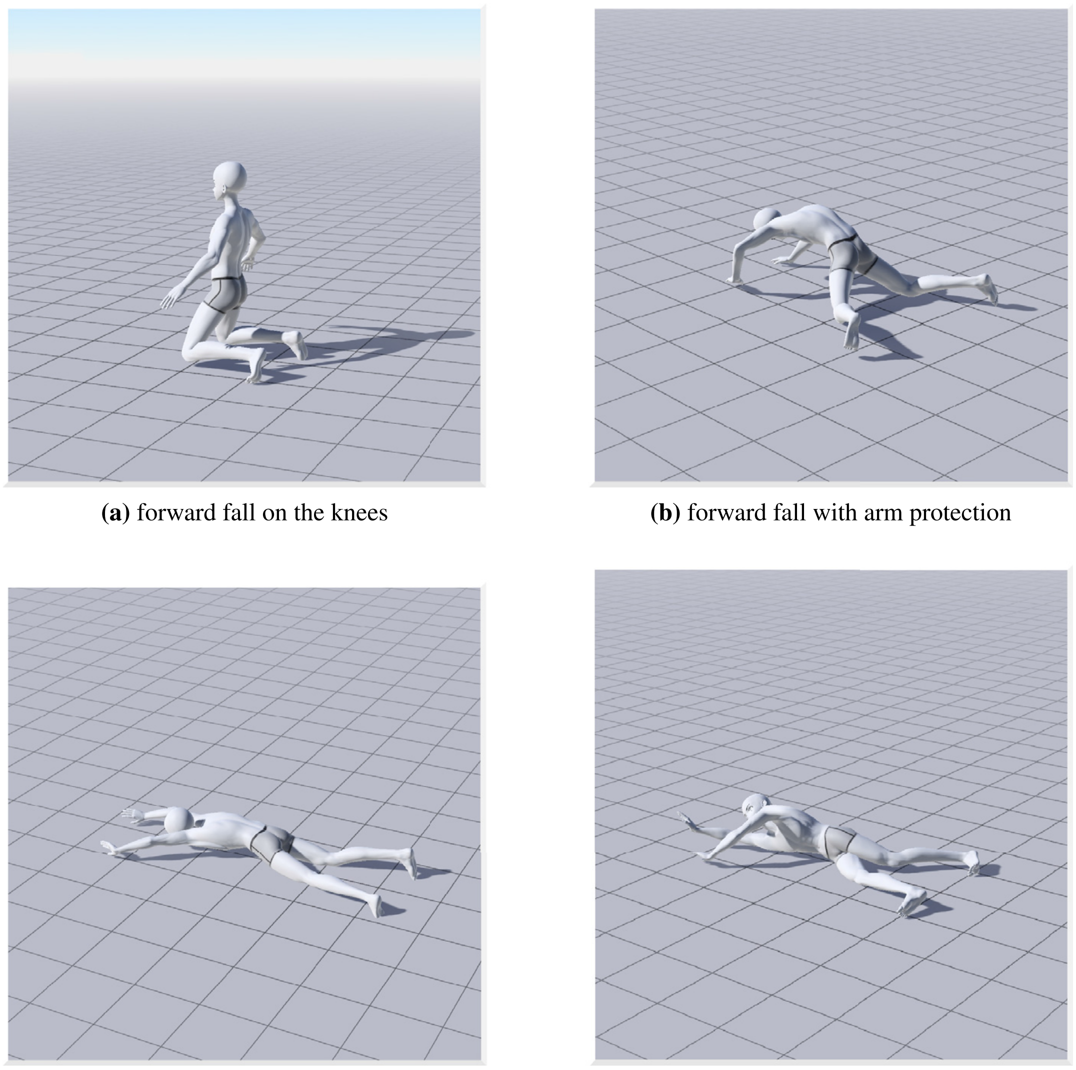


Fig. 5. Participant starting position before each fall event in relation to sensor placement on the smart floor. Sensors on the smart floor are described with the notation from s0 to s15, the same notation is used across all the provided data in CSV files.

4. After the recording interval is expired, the data collector notifies the participant, to release the ending position and leave the smart floor.

Other ordinary daily activities were recorded similarly but without any precondition on the starting position of the participant.



(a) forward fall on the knees

(b) forward fall with arm protection

(c) forward fall ending laying flat, lateral fall ending laying flat, lateral fall ending laying flat with recovery, forward fall ending laying flat with recovery

(d) forward fall on the knees with rotation, ending in the lateral position

Fig. 6. All possible ending position of seven simulated fall events described in Table 3.

Ethics Statement

The data gathering process involved the use of human subjects (we were observing actual human falls). Each participant was informed orally and in written form about the aims of the experiment and possible risk hazards. Participants were voluntary, and they could withdraw from the data gathering at any point. Informed consent was obtained from all the participants and in the case of minor participants from their legal guardians. The enclosed copy of the informed consent shows the exact formulation. The authors state that the study does not include anything that the Medical Ethics Committee of Slovenia would cover.

CRediT Author Statement

Aleksandar Tošić: Conceptualization, Software, Investigation, Writing - review & editing, Visualization; **Niki Hrovatin:** Software, Investigation, Writing - review & editing, Validation, Data

Curation; **Jernej Vičič:** Conceptualization, Investigation, Writing - review & editing, Supervision, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships which have, or could be perceived to have, influenced the work reported in this article.

Acknowledgments

Funding for this research is provided by European Commission through the Horizon 2020 project 'Pilots for Healthy and Active Ageing' (Pharaon, Grant agreement no. 857188), the Horizon 2020 'InnoRenew CoE' (Grant Agreement no. 739574), and the Republic of Slovenia from European Union's European Regional Development Fund.

References

[1] S. Chaudhuri, H. Thompson, G. Demiris, Fall detection devices and their use with older adults: a systematic review, J. Geriatr. Phys. Ther. (2001) 37 (4) (2014) 178.
[2] W.H. Organization, W.H.O. Ageing, L.C. Unit, WHO Global Report on Falls Prevention in Older Age, World Health Organization, 2008.
[3] Q. Shi, Z. Zhang, T. He, Z. Sun, B. Wang, Y. Feng, X. Shan, B. Salam, C. Lee, Deep learning enabled smart mats as a scalable floor monitoring system, Nat. Commun. 11 (1) (2020) 1–11.
[4] A. Tošić, J. Vičič, M.D. Burnard, Privacy preserving indoor location and fall detection system, in: Human-Computer Interaction in Information Society : Proceedings of the 22nd International Multiconference Information Society - IS 2019, 2019, pp. 9–12. http://library.ijs.si/Stacks/Proceedings/InformationSociety/2019/IS2019_Volume_H%20-%20HCI.pdf.
[5] N. Noury, P. Rumeau, A. Bourke, G. ÓLaighin, J. Lundy, A proposal for the classification and evaluation of fall detectors, Irbm 29 (6) (2008) 340–349.

2.2 Paper 2

Title: In-Network Convolution in Grid Shaped Sensor Networks
Authors: Niki Hrovatin, Aleksandar Tošić, Jernej Vičič
Year: 2021
Journal: Journal of Web Engineering
DOI: `https://doi.org/10.13052/jwe1540-9589.2114`
Link:`https://journals.riverpublishers.com/index.php/JWE/article/view/12517`

In-Network Convolution in Grid Shaped Sensor Networks

Niki Hrovatin[1,2,*], Aleksandar Tošić[1,2] and Jernej Vičič[1,3]

[1]*University of Primorska, Faculty of Mathematics, Natural Sciences and Information Technologies, Koper, Slovenia*
[2]*InnoRenew CoE, Izola, Slovenia*
[3]*Research Centre of the Slovenian Academy of Sciences and Arts, The Fran Ramovš Institute, Ljubljana, Slovenia*
E-mail: niki.hrovatin@famnit.upr.si; aleksandar.tosic@upr.si; jernej.vicic@upr.si
[*]*Corresponding Author*

Received 16 July 2021; Accepted 20 September 2021;
Publication 23 November 2021

Abstract

Gathering information is the primary purpose of a Sensor Network. The task is performed by spatially distributed nodes equipped with sensing, processing, and communication capabilities. However, data gathered from a sensor network must be processed, and often the collective computation capability of nodes forming the sensor network is neglected in favor of data processing on cloud systems. Nowadays, Edge Computing has emerged as a new paradigm aiming to migrate data processing close to data sources. In this contribution, we focus on the development of a sensor network designed to detect a person's fall. We named this sensor network the smart floor. Fall detection is tackled with a Convolutional Neural Network, and we propose an approach for in-network processing of convolution layers on grid-shaped sensor networks. The proposed approach could lead to the development of a sensor network that detects falls by performing CNN inference processing on the edge. We complement our work with a simulation using the simulator ns-3. The simulation is designed to emulate the communication overhead of the proposed approach applied to a wired sensor network that resembles the smart

Journal of Web Engineering, Vol. 21_1, 75–96.
doi: 10.13052/jwe1540-9589.2114

floor. Simulation results provide evidence on the feasibility of the proposed concept applied to wired grid shaped sensor networks.

Keywords: Sensor networks, edge computing, fall detection, convolutional neural networks, network simulator ns-3.

1 Introduction

A sensor network consists of spatially distributed nodes deployed in a dynamic environment for specific monitoring purposes. Several current and potential applications exist ranging from the military domain, environmental monitoring, healthcare, industrial manufacturing monitoring etc. [21]. Typical sensor networks count hundred or thousand of densely deployed nodes. Nodes are devices constrained in processing and communication capabilities, equipped with one or multiple sensors. Such large networks of sensors provide a detailed view of the environment in which they are deployed.

In this contribution, we consider a sensor network deployed to detect events using a Convolutional Neural Network (CNN) that feeds on input from multiple sensor nodes. Since the CNN inference process requires data from multiple sensor nodes, typical solutions convey all the nodes' data to a central processing point usually in the cloud. Therefore computing capabilities of nodes forming the network are not used. Furthermore, transferring data to remote systems like cloud services could introduce security concerns [24] and a substantial latency between event occurrence and detection [18].

Nevertheless, we have recently witnessed the emergence of edge computing [1, 29], a paradigm aiming to move computation as close as possible to data sources. In the present research, we tackle the problem of distributing CNN processing load across sensor nodes' actual data sources of the sensor network. Precisely we propose a technique applicable only to grid-shaped sensor networks, in which neighboring sensor nodes are interchanging sensor readings to conjointly compute Convolution Layers of a CNN on sensor nodes. Additionally, we discuss applying the proposed technique on our non-intrusive fall detection sensor network dubbed smart floor. The smart floor is a grid-shaped sensor network in which each sensor node is sensing the local force applied on the floor. A CNN is used to recognize if the activity occurring over the smart floor is a person that fell on the floor or just activities of daily living like walking, moving objects, etc.

Although the proposed concept takes advantage of the processing power of sensor nodes, substantial communication overhead is generated since each sensor node is interchanging sensor readings with neighbors. In this study, we address concerns regarding the communication overhead with a simulation based on the ns-3 simulator [11]. The simulation is designed to emulate the communication overhead of the proposed concept applied to a wired grid shaped sensor network that resembles the smart floor. Simulation results show that the communication overhead only leads to severe network congestion if low bit-rate links are used with a large convolution kernel. Furthermore, the examination considers two network topologies, showing a significant difference in the number of packet drops and packet traveling time.

Section 2 gives an overview of the related literature about in-network neural network inference in sensor networks. In Section 3 we discuss fall detection and we give an overview of the current development of the smart floor. In Section 4 we present a technique to in-network compute 3D discrete convolution on sensor nodes of a grid shaped sensor network. Section 5 presents the experimental setup and results of the simulation. Conclusion and future work is given in Section 6.

2 Related Work

Several studies in the Internet of Things (IoT) field are addressing deep learning inference on low-power mobile devices. DeepX [13] is a software accelerator based on runtime layer compression and deep architecture decomposition able to allocate data and model operations optimally across diverse processor types. The Big-Little approach [3] uses small deep learning models that are located on embedded devices (the Little role) to perform inference on a restricted set of events. However, for some critical situations that require a reliable inference, the collected data is also sent to processors in the cloud (the Big role) to perform the inference process on a larger deep learning model.

An interesting approach aiming at the reduction of the communication overhead in IoT networks was proposed in [5]. The proposed architecture moves the deep learning model from the base station towards the data source nodes. However, deep learning models are only partially moved. Models located on data source nodes are small and resource-efficient, designed to output a limited set of codes. The codes are then used to perform the final inference on powerful base-station nodes using large inference models. Although the proposed approach could reduce the communication overhead, it is designed for inference over data sourcing from one device and not from multiple devices, as required in the CNN implemented in the smart floor.

In-network computing involves the processing of data as it travels via the sensor network to the sink node. The sink node in a sensor network is the node that acts as a gateway between the sensor network and external systems and usually has greater processing and storage capabilities than other nodes in the sensor network. In-network computing was already applied to the deep learning concept in [12]. They proposed a distributed deep learning architecture that assigns processing roles to sensor network nodes. Data is gathered on those sensor nodes, processed, and sent to the next node in the chain. Each node in the chain is computing a layer of the convolutional neural network. The sink node is computing the end fully connected layers. The proposed approach effectively distributes a convolutional neural network's computation load among multiple nodes of a sensor network. However, this causes a high load on nodes in the computation chain, especially on the first node that must gather data from all sensing nodes.

Fukushima et al. [8] addressed this problem with a framework for distributed deep learning in wireless sensor networks based on node coordination. They assume that a Wireless Sensor Network (WSN) can resemble a grid structure, and they map CNN neurons to physical nodes of a WSN. Therefore the whole CNN is statically built on the WSN and is trained to work only with that precise deployment of sensor nodes. They demonstrated this approach's feasibility with a simulation and two experiments. Although the research conducted by Fukushima et al. includes a sensor network able to detect fall events, the research does not consider event localization.

In the present contribution, we propose a slightly different approach motivated by the need to detect fall events and the smart floor properties of being grid-shaped, wired, and having a high node density. In our proposed solution, we map to sensor nodes only the convolution layers. A procedure will then extract processed activities from the sensor network and send them to a sink node that will complete the inference using the fully connected layers of the CNN. Therefore, event location can be determined from the location of data reporting nodes. Moreover, the solution differs from the approach proposed by Fukushima et al. since it allows the application of a once trained and well-tested CNN in different network deployments. Training a CNN with neurons built into nodes of a sensor network could lead to slightly different predictions at sensor networks having diverse shapes even if the same training data is used. The root cause of this alteration of predictions is the shape of the building in which the sensor network is placed, physically constraining the shape of the network and the number of senor nodes. Therefore by applying the technique presented by Fukushima et al. the CNNs deployed into different sensor networks will have a different number of neurons and disposition of holes. (Fukushima et al. were referring to holes, as locations of the monitored environment without sensor nodes.) In the fall detection scenario holes might be numerous due to the irregular shapes of rooms and disposition of walls inside buildings. Inference alteration due to changes in the number of neurons in Artificial Neural Networks is well known [2, 28]. And inference alteration due to changes in the number of holes is documented by Fukushima et al.

A fall detection system must ensure that each fall event is detected and correctly reported. Since undetected fall events can have repercussions on the health of the fallen person, each wrongly predicted fall event must be investigated to find the root cause and prevent similar occurrences. Having the exact inference mechanism on different deployments allows the adjustment of every system instance if eventual flaws are discovered.

3 Non-Intrusive Fall Detection Using Smart Floor

Falls are unpredictable accidental events. Common in childhood, rare in adulthood, but a significant problem among the elderly. The report *Fall Prevention in older Age* [20] carried out by the World Health Organization state that approximately 30% of people aged 65 fall each year, and the odds increase for those over 70 years of age. Falls are critical events in the elderly, which can result in severe injuries or fatalities. More than 50% of injury-related hospitalization among people over 65 years is a fall consequence. The research [10] pointed out the need to provide immediate help after a fall event to prevent severe or fatal consequences.

While fall prevention is enhanced by behavioral, environmental, socioeconomic, and biological risk reduction [20], the fall recognition problem is widely addressed with the use of technological solutions. The literature review conducted by Singh et al. [23] investigates different fall detection systems categorizing them into *wearable*, *ambiance*, and *hybrid* systems. Wearable systems are based on accelerometer or gyroscope sensor technology embedded in items or smartphones. Wearable solutions are low cost solutions that can detect a fall effectively; however, the user must actively wear and maintain these systems. Ambiance fall detection systems are embedded in the monitored environment. Image sensors, acoustic sensors, pressure sensors, infrared sensors, radar sensors or a combination of them is used to monitor user activity and detect falls. Ambiance systems provide good fall detection performance, they usually incur higher costs, but they eliminate the active interaction between user and system. Technology acceptance by

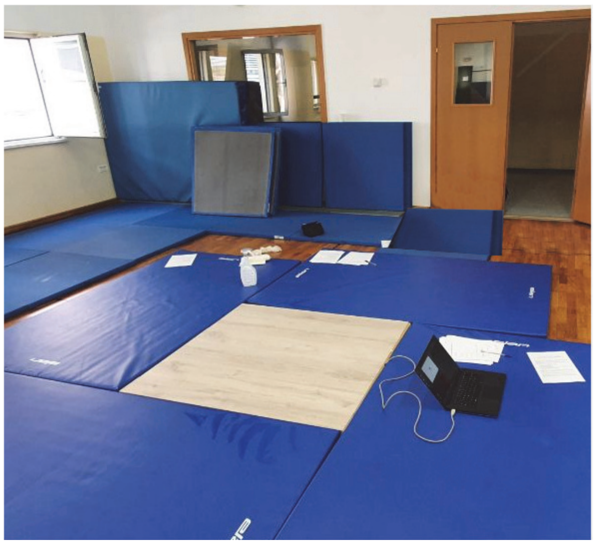


Figure 1 The picture was taken during the data collection event, where fall events were simulated following safety precautions. The smart floor is the white surface surrounded by landing mats.

older adults is a recognized problem, drastically emphasized in monitoring systems [14]. Fall detection technology must be constantly present in the user's life, to assure benefit; hence solutions based on wearable devices or image sensors are considered highly intrusive [6, 23] since users are always aware of the system. A systematic review conducted by Yusif et al. [30], suggests that the main adoption barriers between older people and assistive technologies are: privacy, trust, added value, cost, ease of use, perception of no need, stigma etc. (sorted by importance). The mentioned factors are raising the need to design a fall detection system able to effectively detect falls while being non-intrusive, privacy-aware, and cost-effective.

The literature review conducted by Singh et al. [23] provides many insights about different sensor technologies used in fall detection systems. Analyses suggest that systems based on *passive Infra-red* (PIR) radiation sensors are the only ones achieving a high fall detection rate while being non-intrusive, privacy-aware, and cost-effective. These systems are usually structured as vertical arrays of many PIR sensors positioned on a room wall, designed to detect the fall vertical motion. A PIR fall detection system was developed by Fukushima et al. [8], relying on a 6 $\times$ 6 grid-shaped wireless sensor network able to detect falls using a CNN. The system achieved good fall detection performance. However, such systems are subject to the risk of obstruction of the sensing range if an object is located between the sensor and the monitored user. An interesting alternative to PIR systems highlighted by Singh et al. are floor pressure sensor systems, which can achieve high fall detection rates while being privacy-aware and non-intrusive. The reviewed solutions rely on complex sensing technologies resulting in high implementation prices, like floor pressure sensing using optical fibers [7]. However, a commercial solution based on capacitive sensor technology embedded in the flooring material has already been released on the market under the name SensFloor [25].

In this contribution, we present recent development of our fall detection solution based on the smart floor implementation presented in [27], which relies on widely available and cheap Force-Sensing Resistor (FSR) technology; adequate to be embedded in a vast choice of flooring materials. The smart floor was recently the object of study in a master thesis [19], intending to establish solid foundations for developing a non-intrusive privacy-preserving fall detection system. A dataset [26] consisting of 420 simulated fall events was collected using the smart floor. In Figure 1, the smart floor during the data collection event. The collected data was used to train different machine learning models and results shown the notable accuracy of CNN in distinguishing activities of daily living from simulated fall events. The effectiveness of CNNs applied to floor pressure sensing systems was highlighted by the contribution [22], where a CNN was used to identify the person's unique walking gait over a smart mat monitoring system. The system was also proposed for fall detection, but further analyses must be conducted.

However, our system's uniqueness does not reside in the machine learning component but rather in the end goal of developing a modular tile system, where each tile will be the sensor node of a distributed sensor network designed to detect fall events while being non-intrusive and privacy-preserving.

4 In-Network Convolution In Grid Shaped Sensor Networks

In this section, we first present the sensor network model taken into account to develop our solution. Then we show how to perform the in-network 3D discrete convolution on grid-shaped sensor networks. Furthermore, we discuss applying the presented technique to in-network compute multiple convolution layers and pooling layers on the smart floor. We end with a brief description of our in-network fall detection solution.

4.1 The Sensor Network Model

We consider a sensor network resembling the smart floor described in [19,26,27]. The network consists of two types of nodes, the majority are sensor nodes, which are nodes equipped with sensor technology sensing a physical quantity in time. These nodes are limited in computational and memory capabilities as they are designed to be cheap. The other type of node is named sink node, which purpose is to gather data sensed by sensor nodes, store it, process it, and interact with external systems. The sink node also has greater computational and storage capabilities than sensor nodes.

Sensor nodes are deployed in a plane following a grid structure; each sensor node is equidistant from the closest sensor nodes in cardinal directions. Sensor nodes are linked via a point-to-point link with each sensor node in their neighborhood. We refer to the neighborhood of a sensor node as the eight closest sensor nodes. In each cardinal and intercardinal direction relative to a sensor node, lies one and only one sensor node from its neighborhood. Sensor nodes can directly interchange sensor readings with nodes in their neighborhood. However, sensor nodes can also retrieve sensor readings from nodes outside their neighborhood and communicate with the sink node using a multi-hop routing strategy based on grid coordinates similar to [16]. A sensor node can be uniquely identified in the network using the IP-address or gird coordinates (e.g. (x, y)). For convenience, we assume that the sink node acts like a central coordination authority that knows grid coordinates and IP-addresses of each sensor node in the network. A sensor node can request the sink node to reveal other sensor node's grid coordinates based on the node IP-address or vice-versa.

4.2 In-network 3D Convolution on Grid-shaped Sensor Network

The 2D discrete convolution technique is widely used in image processing [9] to apply smoothing filters, image sharpening, identify edges, and classify images in conjunction with machine learning methods [15]. Images are represented as matrices of $n \times m$ pixels with variable intensities. The convolution operation will slide a kernel of size $k \times k, k < m$ over each pixel of the image, performing an elementwise multiplication between the kernel and the covered part, summing up results into one single output value. The kernel repeats this process for every pixel it slides over, generating a new 2D matrix.

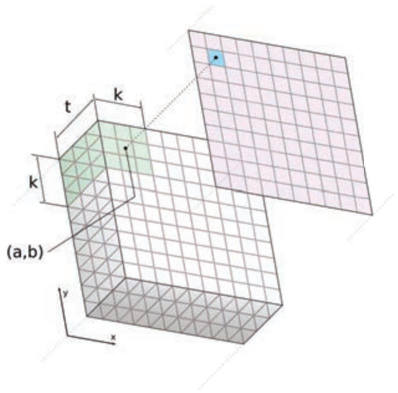


Figure 2 The figure presents the 3D convolution operation. Each square in the white plane represents a sensor node. The green area indicates sensor readings needed to compute a fragment of the convolution operation on the node a, b using the kernel of size $k \times k \times t$. The red grid is the result of the convolution operation over the grid of sensor nodes, highlighted with blue color the convolution result of sensor node on coordinates a, b.

Our proposed distributed solution is based on the assumption of the sensor network's physical grid structure introduced in Section 4.1. The grid-shaped sensor network resembles a matrix on which to perform the convolution operation with a kernel. However, we will not slide the kernel over the matrix of sensor nodes. Instead, each sensor node will compute a fragment of the entire convolution operation.

Since fall events occur over a surface during a time interval, the application of 3D convolution to the smart floor will certainly enhance prediction accuracy as 3D CNNs [17] can operate on planar and temporal dimensions.

To perform the 3D discrete convolution operation on the grid shaped sensor network, we take into account the two planar dimensions and a time dimension detailing the change in sensor values over time. Sensor nodes share the same kernel with all other sensor nodes in the network. For convenience, we describe the procedure only for kernels of size $k \times k \times t$. $k \times k$ the planar dimensions, and t the time dimension as it is shown on Figure 2.

To compute a fragment of the convolution operation, a sensor node located in the grid-shaped sensor network at coordinates (x, y) will first gather t sensor readings from each sensor node located at coordinates $(x + i, y + j)$, where $-\lfloor \frac{k}{2} \rfloor \leq i, j \leq \lfloor \frac{k}{2} \rfloor + v$. Figure 3 shows such operation. If the kernel dimension k is an odd number, the variable $v = 0$, otherwise $v = -1$. If sensor node coordinates are outside the physical network, zero padding is applied.

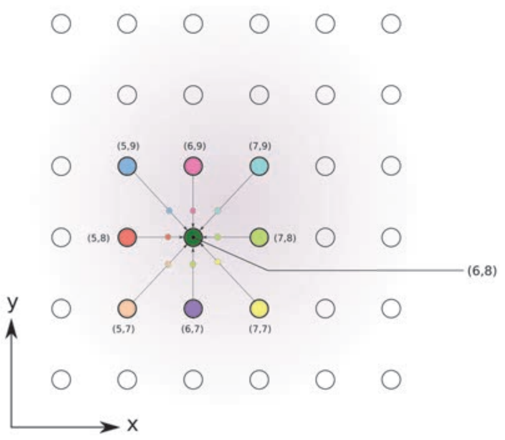


Figure 3 The figure illustrates how a sensor node located at grid coordinates $(6, 8)$ gathers data from other sensor nodes to compute a fragment of the convolution operation with a kernel of size $3 \times 3 \times t$. Sensor nodes are drawn with circles, and data from diverse sensor nodes is represented with a different color.

Then the sensor node will perform the elementwise multiplication between the kernel, its sensor value, and sensor values collected from local sensor nodes, results are summed up in one single output value. We refer to this value as a fragment, and to local sensor nodes as the sensor nodes whose values were used to compute the fragment. Since each sensor node knows its and local sensor nodes' location as coordinates in a grid-shaped sensor network, fragments are computed respecting the same kernel orientation on all sensor nodes. Therefore, each sensor node holds a fragment of the resulting convolution operation between the grid-shaped sensor network and the selected kernel.

4.2.1 Multi-layer convolutional neural network

The proposed technique can be applied to multi-layer CNN such that the first convolution layer is computed on sensor readings as described in the upper section. Outputs from the first layer reside on sensor nodes, maintaining the grid structure. Hence it is possible to compute the next convolution layer following the same principle but on previous layer outputs instead of sensor readings. This concept was already proposed by Fukushima et al. [8], where a multi-layer CNN was built on a wireless sensor network.

They proposed the application of this concept also to in-network compute the pooling layer. **Pooling layers** are used to reduce the dimension of data by combining multiple values into one single value using the average or max function. However, considering our the smart floor needs, the pooling layer will be implemented only on the time dimension, as we want to maintain as much planar information as possible. Hence the pooling layer can be computed on sensor nodes without the need for data from other sensor nodes.

4.2.2 Communication overhead

The in-network processing of convolution layers requires a noticeable communication overhead. Each sensor node computes a fragment of the convolution layer by gathering data from other sensor nodes. Hence the communication overhead is increased locally to sensor nodes involved in the computation. Fukushima et al. demonstrated the manageability of this concept with two experiments, and they provided a simulation to determine how kernel size affects the communication overhead. The simulation revealed that kernels $k \times k \times t$ of size $k \leq 5$ does not generate a concerning communication overhead.

Our research considers the communication overhead generated in wired sensor networks with a simulation using the network simulator ns-3. Simulation results are presented in Section 5.

4.3 Fall Detection Using In-network Convolution

Our proposed fall detection solution is based on the grid-shaped sensor network model introduced in Section 4.1, a CNN designed to process part of the inference on sensor nodes, and a procedure to extract the field of interest for further processing at sink nodes. Since the sensor network will be deployed in many rooms or an entire building, the whole system must be modular and adaptable to fit any room shape and deployment extensions. Therefore, we will not train a CNN on the whole sensor network, like were done by Fukushima et al. [8], as this will generate different CNNs at each deployment which might produce slightly different inference outcomes.

Our system is designed to work on a CNN trained on the smart floor described in Section 3, composed of 16 pressure sensors. This limits the CNN input to readings from only 16 sensor nodes structured like a 4×4 grid. Hence, we need a mechanism to extract activities happening on the sensor network and feed them in the CNN for inference.

The activity extraction starts from sensor nodes which are constantly sensing the pressure applied on them. However, they will proceed with further processing only if they identify a significant change in the sensed pressure, which signals an activity on them. Triggered sensor nodes will in-network compute convolution layers of the CNN using the technique explained in Section 4.2. Adjacent convolution layer results are grouped via node coordination, and the grouped result is sent to a sink node. The sink node will complete the inference by processing convolution layer results with the CNN's fully connected layers. The exact event location can be determined from the location of nodes reporting the grouped result.

5 Evaluation

This section presents results obtained from a simulation using the ns-3 simulator [11]. Via a simulation, we aim to evaluate whether the proposed concept of in-network computing convolution layers in wired grid shaped sensor networks generates a communication overhead that leads to a severe network congestion. Precisely in the simulation, we try to model the smart floor. Additionally, we compare two network topologies, the *plain grid* in which sensor nodes are linked with four neighboring nodes, one link in each cardinal direction, and the *diagonal grid* in which each sensor nodes are linked with eight neighboring nodes, one link in each cardinal and intercardinal direction. Topologies are illustrated in Figure 4. Although the two network topologies have numerous redundant links, we selected them since our end goal is the development of the smart floor as a modular tile system, where each tile will be a sensor node able to link with adjacent sensor nodes through wired point to point links embedded in the tiles. To perform this investigation, we examine how kernel size and link bitrate affects the traveling time (TT) of datagrams in the network. We define the TT of packets in the network as the elapsed time between the issuing of the packet by a sensor node, and the receipment of the packet from the destination node.

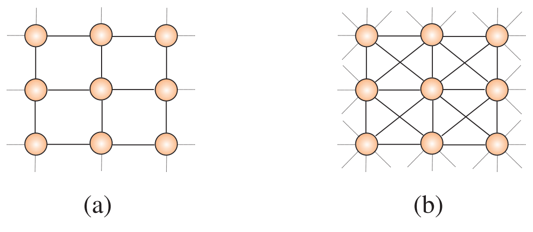


Figure 4 Network topologies considered in the simulation: *plain grid* in (a), *diagonal grid* in (b).

5.1 Experimental Setup

The simulated network is designed to resemble the smart floor, a grid shaped sensor network in which each sensor node is sensing the force applied on it (see Section 3). Sensor nodes are measuring the applied force 100 times per second [26], converting the force into an integer value. Therefore, a smart floor sensor node is generating 400B/s of data at fixed-length encoding using 32b integers. In the simulated network, we approximate this bitrate using packets of 512B. Each sensor node in the simulation generating a batch of packets each second, and sending one packet of the batch to each of its local sensor nodes. We refer to local sensor nodes of a node at coordinates (x, y) as the sensor nodes whose coordinates are in the range $(x \pm \lfloor \frac{k}{2} \rfloor, y \pm \lfloor \frac{k}{2} \rfloor)$, (x, y) excluded, k one dimension of a $k \times k$ kernel. Clocks of all nodes are synced, and a batch identifier is applied to packets. Packets are not sent all at the same time, but each packet is sent at a randomly chosen time in a $1 - \Delta t$ second time interval (Δt computed from the kernel size and link bitrate to allow the packets to reach the destination before the next batch of packets commences issuing). Packets that do not reach the destination within one second after their batch starts issuing are counted as lost packets.

The simulated network consists of a 15×15 grid of sensor nodes. Nodes are linked following two network topologies the *plain grid*, and the *diagonal grid*. Links between nodes are full-duplex point-to-point links of bitrate dr. dr being an independent variable of values $dr = \{0.125, 0.25, 0.5, 1, 5, 10\}$–Mbps. Bitrate values of dr were chosen to approximate bitrate values of commonly used communication standards in IoT and sensor networks (e.g., RS-232, UART, USB, and Ethernet). Each sensor node acts also as a router. Routing tables of nodes are statically computed at network deployment using the Dijkstra Shortest Path First algorithm [4]. A simulation was ran for each combination of independent variables network topology, link bitrate dr, and kernel size k. Kernel size values of $k = \{3, 5, 7\}$ should be appropriate since, on the smart floor, a larger kernel size will cover an area much larger than the interested detection area of a fall event. (The smart floor includes 16 pressure sensors deployed on a surface of 1.2 m^2.)

Data were collected only from one sensor node during the simulation, the one located in the middle of the grid, precisely at coordinates $(7, 7)$. Only this node was monitored since it is located in the central portion of the network, where the communication overhead is condensed. The recorded data include TT of packets reaching the monitored node and packet loss throughout the whole network. Figures 5 and 6 display scatter plots of

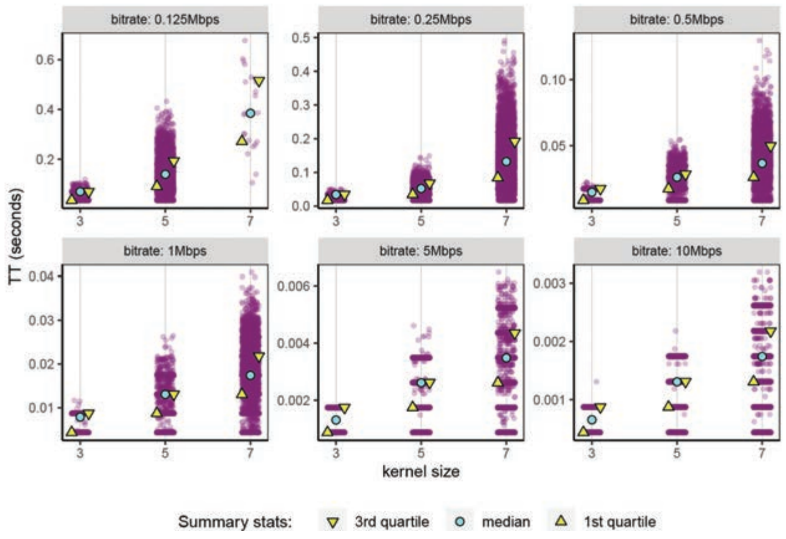


Figure 5 Scatter plots of packets TT, at combinations of independent variables: topology *plain grid*, bitrate $dr = \{0.125, 0.25, 0.5, 1, 5, 10\}$–Mbps and kernel size $k = \{3, 5, 7\}$. Data from the node at coordinates $(7, 7)$.

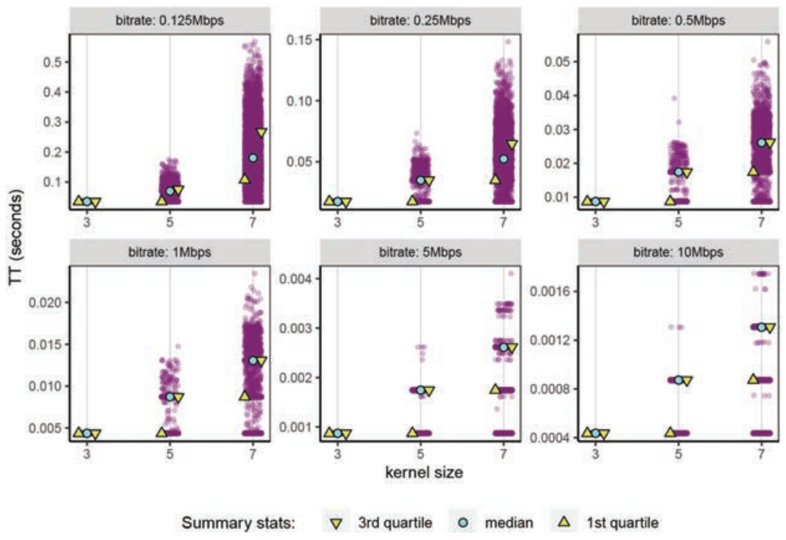
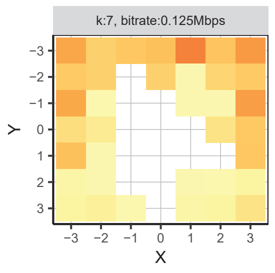
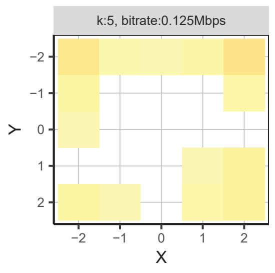


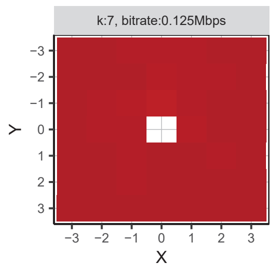
Figure 6 Scatter plots of packets TT, at combinations of independent variables: topology *diagonal grid*, bitrate $dr = \{0.125, 0.25, 0.5, 1, 5, 10\}$–Mbps and kernel size $k = \{3, 5, 7\}$. Data from the node at coordinates $(7, 7)$.



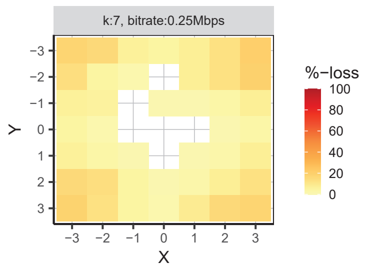
(a)



(b)



(c)



(d)

Figure 7 Percentage of packets sourcing from sensor nodes at coordinates $(7 + X, 7 + Y)$ that do not reach the monitored node within one second after their batch starts issuing. The monitored node is at coordinates $(7, 7)$. Figure (a) *diagonal grid* topology. Figures (b),(c),(d) *plain grid* topology. We have not observed packet loss on the monitored node in other combinations of considered independent variables topology, bitrate, and kernel size.

packet TT. In the former, the network is deployed following the *plain grid* topology, and in the latter, the network is deployed following the *diagonal grid* topology. Table 1 presents summary statistics of packet TT and packet loss. Figure 7 display the percentage of packets sourcing from sensor nodes local to the monitored node that do not reach the monitored node.

5.2 Experimental Results

From plots in Figures 5 and 6 can be seen, that the data is not normally distributed. We believe that the reason behind the non-normal distribution of packet TT is that packets are traveling through multiple nodes to reach the destination, some packets traveling through more nodes than other packets due to kernel size. Moreover, we notice that at lower datarates ($dr \leq 0.5$), packet TT begin to spread. A possible cause for the spread of TT is traffic

Table 1 Summary statistics of packet TT recorded at the node at coordinates $(7,7)$. The column **%** packet loss expresses the packet loss of the whole network at selected independent variables

Network Topology	Kernel Size	Link Bitrate (Mbps)	Min TT (s)	Avg TT (s)	Max TT (s)	Std TT (s)	1st Quartile TT (s)	Median TT (s)	3rd Quartile TT (s)	% Packet Loss
Plain grid	3	0.125	0.03482	0.05443	0.11952	0.01915	0.03488	0.06963	0.06976	
		0.25	0.01741	0.0266	0.05191	0.00906	0.01744	0.03432	0.03488	
		0.5	0.0087	0.01313	0.02246	0.00439	0.00872	0.01478	0.01744	
		1	0.00435	0.00656	0.0117	0.00219	0.00436	0.00789	0.00872	
		5	0.00087	0.00131	0.00175	0.00044	0.00087	0.00131	0.00174	
		10	0.00044	0.00065	0.00131	0.00022	0.00044	0.00065	0.00087	
	5	0.125	0.03482	0.14719	0.43254	0.07272	0.09082	0.13952	0.19282	1.7
		0.25	0.01741	0.05089	0.14889	0.02183	0.03488	0.05232	0.06825	
		0.5	0.0087	0.02324	0.05463	0.0093	0.01744	0.02611	0.02843	
		1	0.00435	0.01122	0.02645	0.00443	0.00872	0.01306	0.0131	
		5	0.00087	0.00219	0.00461	0.00085	0.00174	0.00261	0.00262	
		10	0.00044	0.00109	0.00218	0.00042	0.00087	0.00131	0.00131	
	7	0.125	0.10553	0.39668	0.67717	0.15938	0.27151	0.38481	0.51607	86.9
		0.25	0.01741	0.14409	0.49108	0.0792	0.08454	0.13234	0.19204	5
		0.5	0.0087	0.03858	0.12956	0.0174	0.02616	0.03668	0.04988	
		1	0.00435	0.01658	0.04086	0.00686	0.01306	0.01741	0.0218	
		5	0.00087	0.0031	0.00646	0.00123	0.00261	0.00348	0.00435	
		10	0.00044	0.00154	0.00318	0.00061	0.00131	0.00174	0.00218	
Diagonal grid	3	0.125	0.03482	0.03487	0.03494	4e-05	0.03482	0.03488	0.03488	
		0.25	0.01741	0.01743	0.01747	2e-05	0.01741	0.01744	0.01744	
		0.5	0.0087	0.00872	0.00874	1e-05	0.0087	0.00872	0.00872	
		1	0.00435	0.00436	0.00437	1e-06	0.00435	0.00436	0.00436	
		5	0.00087	0.00087	0.00087	1e-06	0.00087	0.00087	0.00087	
		10	0.00044	0.00044	0.00044	1e-07	0.00044	0.00044	0.00044	
	5	0.125	0.03482	0.0667	0.17552	0.02454	0.03494	0.06976	0.07564	
		0.25	0.01741	0.03059	0.07342	0.00979	0.01744	0.03488	0.03494	
		0.5	0.0087	0.01483	0.03921	0.00442	0.00872	0.01744	0.01744	
		1	0.00435	0.00736	0.01476	0.00214	0.00436	0.00872	0.00872	
		5	0.00087	0.00146	0.00262	0.00041	0.00087	0.00174	0.00174	
		10	0.00044	0.00073	0.00131	0.00021	0.00044	0.00087	0.00087	
	7	0.125	0.03482	0.19328	0.56775	0.10669	0.10756	0.18028	0.26753	8
		0.25	0.01741	0.05129	0.14832	0.02147	0.03488	0.05232	0.06473	
		0.5	0.0087	0.02204	0.05592	0.00785	0.01744	0.02611	0.02621	
		1	0.00435	0.01054	0.02346	0.00353	0.00872	0.01306	0.01308	
		5	0.00087	0.00205	0.0041	0.00066	0.00174	0.00261	0.00262	
		10	0.00044	0.00102	0.00175	0.00033	0.00087	0.00131	0.00131	

congestion; packets are waiting for links to be freed. In Figure 7 can be seen, that at kernel size $k = 7$ and bitrate $dr = 0.125$ Mbps, there is a considerable difference in packet loss between *plain grid* and *diagonal grid* topology. The difference can be noticed also in Table 1. Furthermore, in Table 1 can be seen that severe network congestion occurs only at plain grid topology at parameters: $k = 7$ and bitrate $dr = 0.125$ Mbps with packet loss ratio at 87%. A two-tailed Mann-Whitney U test showed that there is a significant difference in packet TT between a network deployed following the *plain grid* topology and one deplyoed following the *diagonal grid* topology. We ran the test at bitrate $dr = 10$ Mbps and kernel size $k = \{3, 5, 7\}$. Test results: $k = 3$ (W = 176082, p-value $< 2.2e^{-16}$), $k = 5$ (W = 1516573, p-value $< 2.2e^{-16}$),$k = 7$ (W = 5879408, p-value $< 2.2e^{-16}$).

6 Conclusion and Future Work

In this contribution, we depicted the future development of our smart floor as a **non-intrusive fall detection system**. We gave a brief overview of the fall detection problem and how it is solved using technology solutions, highlighting our system's potential.

We proposed a technique to partially process the CNN inference on sensor nodes of a grid-shaped sensor network, and we discussed the application of this technique to our system. The proposed solution that will fit the smart floor needs is able to adapt to changes in the sensor network topology, does not require a new CNN training for each deployment, effectively reduces computation load on the end system, and the activity extraction procedure cuts the data transfer overhead in the sensor network.

Our research is complemented by a simulation designed to emulate the communication overhead of the proposed technique in grid-shaped wired sensor networks. Simulation results show that severe network congestion occur at *plain grid* topology when large convolution kernels are used ($k > 5$) in networks with low bitrate links ($dr \leq 0,125$ Mbps). Interestingly, at the same configuration parameters, the severe network congestion was not observed at *diagonal grid* topology.

Considerably more research will need to be done on the activity extraction procedure since the end objective is to develop a privacy-preserving system, and the activity extraction procedure could potentially provide location privacy by removing any linkage to the source of the data through the grouping of data in anonymous ready to process chunks.

An advanced activity extraction procedure could be applied to coordinate sensor nodes and in-network perform the whole CNN inference processing. Therefore, the sensor network will communicate with external systems only to report falls.

Acknowledgements

The authors gratefully acknowledge the European Commission for funding the InnoRenew project (Grant Agreement #739574) under the Horizon2020 Widespread-Teaming program and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Regional Development Fund). They also acknowledge the Slovenian Research Agency ARRS for funding the project J2-2504.

References

[1] Ejaz Ahmed, Arif Ahmed, Ibrar Yaqoob, Junaid Shuja, Abdullah Gani, Muhammad Imran, and Muhammad Shoaib. Bringing computation closer toward the user network: Is edge computing the solution? *IEEE Communications Magazine*, 55(11):138–144, 2017.

[2] F Arifin, H Robbani, T Annisa, and NNMI Ma'Arof. Variations in the number of layers and the number of neurons in artificial neural networks: Case study of pattern recognition. In *Journal of Physics: Conference Series*, volume 1413, page 012016. IOP Publishing, 2019.

[3] Elias De Coninck, Tim Verbelen, Bert Vankeirsbilck, Steven Bohez, Pieter Simoens, Piet Demeester, and Bart Dhoedt. Distributed neural networks for internet of things: The big-little approach. In *Internet of Things. IoT Infrastructures*, pages 484–492. Springer International Publishing, 2016.

[4] Stuart E Dreyfus. An appraisal of some shortest-path algorithms. *Operations research*, 17(3):395–412, 1969.

[5] Rong Du, Sindri Magnusson, and Carlo Fischione. The internet of things as a deep neural network. *IEEE Communications Magazine*, 58(9):20–25, 2020.

[6] Le Fang, Yu Wu, Chuan Wu, and Yizhou Yu. A non-intrusive elderly home monitoring system. *IEEE Internet of Things Journal*, 2020.

[7] Guodong Feng, Jiechao Mai, Zhen Ban, Xuemei Guo, and Guoli Wang. Floor pressure imaging for fall detection with fiber-optic sensors. *IEEE Pervasive Computing*, 15:40–47, 03 2016.

[8] Yuta Fukushima, Daiki Miura, Takashi Hamatani, Hirozumi Yamaguchi, and Teruo Higashino. Microdeep: In-network deep learning by micro-sensor coordination for pervasive computing. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 163–170. IEEE, 2018.

[9] Rafael C Gonzalez, Richard E Woods, and Barry R Masters. Digital image processing third edition. *Pearson Prentice Hall*, pages 743–747, 2008.

[10] R Jan Gurley, Nancy Lum, Merle Sande, Bernard Lo, and Mitchell H Katz. Persons found in their homes helpless or dead. *New England Journal of Medicine*, 334(26):1710–1716, 1996.

[11] Thomas R Henderson, Mathieu Lacage, George F Riley, Craig Dowell, and Joseph Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 14(14):527, 2008.

[12] Nada B Jarah. Deep learning in wireless sensor network. *Journal of Al-Qadisiyah for computer science and mathematics*, 13(1):Page–11, 2021.

[13] Nicholas D Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 1–12. IEEE, 2016.

[14] F Le Deist and M Latouille. Acceptability conditions for telemonitoring gerontechnology in the elderly: optimising the development and use of this new technology. *Irbm*, 37(5-6):284–288, 2016.

[15] Zewen Li, Wenjie Yang, Shouheng Peng, and Fan Liu. A survey of convolutional neural networks: analysis, applications, and prospects. *arXiv preprint arXiv:2004.02806*, 2020.

[16] Wen-Hwa Liao, Jang-Ping Sheu, and Yu-Chee Tseng. Grid: A fully location-aware routing protocol for mobile ad hoc networks. *Telecommunication systems*, 18(1):37–60, 2001.

[17] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz. Hand gesture recognition with 3d convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–7, 2015.

[18] Mithun Mukherjee, Rakesh Matam, Constandinos X Mavromoustakis, Hao Jiang, George Mastorakis, and Mian Guo. Intelligent edge computing: Security and privacy challenges. *IEEE Communications Magazine*, 58(9):26–31, 2020.

[19] Hrovatin Niki. *Neintruzivna identikacija padcev s pomočjo pametnih tal: magistrsko delo*. PhD thesis, Univerza na Primorskem, Fakulteta za matematiko, naravoslovje in informacijske tehnologije, 2020.

[20] World Health Organization, World Health Organization. Ageing, and Life Course Unit. *WHO global report on falls prevention in older age*. World Health Organization, 2008.

[21] Kazi Chandrima Rahman. A survey on sensor network. *Journal of Computer and Information Technology*, 1(1):76–87, 2010.

[22] Qiongfeng Shi, Zixuan Zhang, Tianyiyi He, Zhongda Sun, Bingjie Wang, Yuqin Feng, Xuechuan Shan, Budiman Salam, and Chengkuo Lee. Deep learning enabled smart mats as a scalable floor monitoring system. *Nature communications*, 11(1):1–11, 2020.

[23] Anuradha Singh, Saeed Ur Rehman, Sira Yongchareon, and Peter Han Joo Chong. Sensor technologies for fall detection systems: A review. *IEEE Sensors Journal*, 20(13):6889–6919, 2020.

[24] Ashish Singh and Kakali Chatterjee. Cloud security issues and challenges: A survey. *Journal of Network and Computer Applications*, 79:88–115, 2017.

[25] Axel Steinhage and Christl Lauterbach. Sensfloor® and navifloor®: Robotics applications for a large-area sensor system. *International Journal of Intelligent Mechatronics and Robotics (IJIMR)*, 3(3):43–59, 2013.

[26] Aleksandar Tošić, Niki Hrovatin, and Jernej Vičič. Data about fall events and ordinary daily activities from a sensorized smart floor. *Data in Brief*, 37:107253, 2021.

[27] Aleksandar Tošić, Jernej Vičič, and Michael David Burnard. Privacy preserving indoor location and fall detection system. In *Human-Computer Interaction in Information Society : proceedings of the 22nd International Multiconference Information Society – IS 2019*, pages 9–12, 2019.

[28] Sun-Chong Wang. Artificial neural network. In *Interdisciplinary computing in java programming*, pages 81–100. Springer, 2003.

[29] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. A survey on the edge computing for the internet of things. *IEEE access*, 6:6900–6919, 2017.

[30] Salifu Yusif, Jeffrey Soar, and Abdul Hafeez-Baig. Older people, assistive technologies, and the barriers to adoption: A systematic review. *International journal of medical informatics*, 94:112–116, 2016.

Biographies

Niki Hrovatin was bor in Trieste, Italy, in 1994. He received the M.S. degree in Computer Science from the University of Primorska, Koper, Slovenija, in 2020. He is currently a Teaching Assistant and Ph.D. student at University of Primorska, and a Research Assistant at InnoRenewCoE, Izola, Slovenija. His current research interests include sensor networks, distributed systems, and blockchain.

Aleksandar Tošić. PhD candidate, teaching, and research assistant at University of Primorska, and Innorenew CoE. His fields of research are distributed, and decentralized systems, Peer to Peer networks, and distributed ledger technologies.

Jernej Vičič. Associate professor and research associate at the University of Primorska and Research Centre of the Slovenian Academy of Sciences and Arts. His research interests are quite broad ranging from language technologies to distributed systems.

2.3 Paper 3

Title: PPWSim: Privacy preserving wireless sensor network simulator
Authors: Niki Hrovatin, Aleksandar Tošić, Jernej Vičič
Year: 2022
Journal: SoftwareX
DOI: `https://doi.org/10.1016/j.softx.2022.101067`
Link: `https://www.sciencedirect.com/science/article/pii/S2352711022000516`

SoftwareX 18 (2022) 101067

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

PPWSim: Privacy preserving wireless sensor network simulator

Niki Hrovatin [a,b,*], Aleksandar Tošić [a,b], Jernej Vičič [a,c]

[a] University of Primorska, Faculty of Mathematics, Natural Sciences and Information Technologies, Slovenia
[b] InnoRenew CoE, Slovenia
[c] Research Centre of the Slovenian Academy of Sciences and Arts, The Fran Ramovš Institute, Slovenia

A R T I C L E I N F O

Article history:
Received 1 December 2021
Received in revised form 23 March 2022
Accepted 24 March 2022

Keywords:
Wireless Sensor Networks
NS-3
Onion routing
Edge computing

A B S T R A C T

The Wireless Sensor Network (WSN) is a monitoring system consisting of a large number of wireless devices forming a multi-hop network without infrastructure. WSN nodes are generally cheap resource-constrained devices detecting environmental features using sensors. Although well-established communication protocols for Internet systems can provide reliable and secure communication between peers, applying similar solutions on WSNs could overwhelm the resource-constrained technology WSNs consist of. Moreover, the broadcast nature of wireless communication could reveal contextual information to external actors eavesdropping on RF signals. Therefore, it is of particular concern to research communication protocols designed explicitly for WSN to preserve network resources and privacy. In this paper, PPWSim is presented, a simulator developed to study a privacy-preserving communication protocol for edge data processing in WSNs. The simulator is based on the simulation environment nsnam NS3; therefore, it can be easily extended or used as a framework for developing other WSN simulators.

Code metadata

Current code version	Version 1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-21-00231
Code Ocean compute capsule	https://codeocean.com/capsule/84f180e8-2f28-4a10-8004-47568ae1704f/tree
Legal Code License	GNU General Public License version 2
Code versioning system used	GIT
Software code languages, tools, and services used	C++, Python, nsnam NS3, Libsodium, Google Protobuf
Compilation requirements, operating environments & dependencies	Linux, autoconf, automake, libxml2 libxml2-dev, g++, git, libsodium-dev, sqlite3, libtool, make, pkg-config, python3, python3-dev, python3-setuptools
If available Link to developer documentation/manual	https://nikih94.github.io/
Support email for questions	niki.hrovatin@famnit.upr.si

1. Motivation and significance

Wireless Sensor Networks (WSN) commonly consist of a large number of sensing nodes deployed in a dynamic environment for specific monitoring purposes. Nodes must self-configure and collaborate with other nodes to autonomously form a network without infrastructure where data can flow from sensor nodes to external actors interacting with sink nodes. Sensor nodes are low-cost devices equipped with sensors and constrained in processing, bandwidth, storage, and energy resources. On the other hand, sink nodes are not severely resource-constrained, and their role is to act as a gateway to external systems requiring data from sensor nodes. However, these particular WSN characteristics of low-cost devices, multi-hop routing, and no infrastructure that shape the versatility of WSNs make them subject to various attacks [1].

The large number of sensor nodes composing a WSN allows very granular monitoring of the environment; nonetheless, a large number of sensing nodes imply large amounts of data, and data must be processed to extract valuable information. However, typical solutions convey all the sensed data to a central processing point, usually in the cloud. Therefore, missing the opportunity to take advantage of the computing power of nodes

[*] Corresponding author at: University of Primorska, Faculty of Mathematics, Natural Sciences and Information Technologies, Slovenia.
E-mail address: niki.hrovatin@famnit.upr.si (Niki Hrovatin).

https://doi.org/10.1016/j.softx.2022.101067

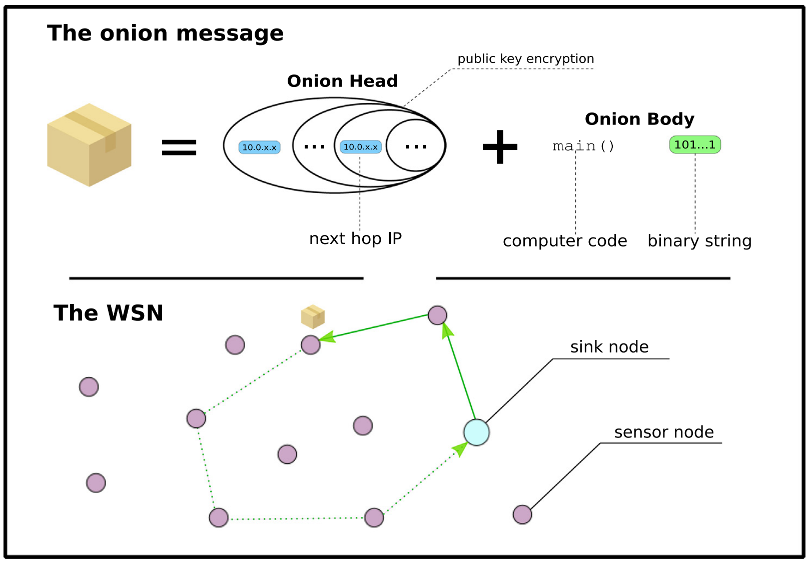


Fig. 1. The figure displays the onion message and its circuit-like path in the WSN. The onion message path is encoded in the onion head. The onion body carries computer code and a binary string used to store an aggregated value. The computer code executes at sensor nodes in the onion message path, embedding the execution result in the carried binary string.

forming the WSN and overloading the multi-hop network raised by resource-constrained technology. Several privacy-preserving data aggregation solutions emerged [2–5] to preserve privacy against attackers, reduce the communication overhead, and employ sensor node processing capabilities. These solutions work by aggregating sensor readings from multiple nodes along routing paths as data travel toward the sink node. The feasibility, and effectiveness of these solutions are usually tested in simulation environments before deployment. However, even popular network simulators such as NS3 [6] lack a generalized framework for WSN simulations that would serve as a common denominator for reproducibility of research results. In this paper, we present a simulation framework for WSN using NS3. We validate our approach using the framework to implement a general purpose data and query privacy preserving protocol for wireless sensor networks using onion routing [7].

2. Problem overview

PPWSim was developed to study the implementation of the privacy-preserving communication protocol [7] in a WSN and: **(a)** assess the response time, **(b)** conduct a scalability study, **(c)** determine if the network topology significantly affects the protocol performance. The software released with this manuscript is an extended version allowing broader manipulation of the simulation environment for future research.

The communication protocol design is characterized by messages containing a layered object made of several encryption layers similar to the one employed in the Onion Routing [8]. Besides the layered object, messages defined by our protocol also contain a payload. By our protocol design, the payload consists of information related to edge data processing (general-purpose computer code) and a binary string whose purpose is to carry an aggregated value. In the rest of the manuscript, the software, and the documentation, we refer to the *onion head* as the layered object, the *onion body* as the message payload, and the *onion message* as the message consisting of the onion head and the onion body. The onion message is depicted in Fig. 1.

The communication protocol leads computer code and a partial result through WSN nodes, each node in the message path executing the computer code and contributing with its sensed value to the partial result. When the message returns back to the issuer sink node, it contains the required aggregated value computed on sensor readings of nodes specified at message construction.

The simulator implementation differs from the technique presented in [7] by the following aspects: *(1)* the simulator does not constrain processing time of the onion message at sensor nodes, *(2)* the simulator forwards padding bytes instead of computer code, *(3)* onion head layers does not include symmetric encryption keys (the onion message is processed alike on each sensor node in the onion message path).

3. Software description

To pursue research questions raised by the conceptualized privacy-preserving data aggregation protocol, we developed a simulator based on the NS3 [6] simulation environment. The simulator allows the user to manipulate various parameters affecting the traveling time of onion messages. We refer to the traveling time of onion messages as the elapsed time between issuing the onion message from the sink node and the return of the issued onion message to the issuer sink node.

3.1. Software functionalities

This section describes software functionalities and parameters affecting the simulator output. To facilitate the use of the simulator, we implemented the setup of simulation parameters through a configuration file. Moreover, parameters `simulation name`, `simulation seed`, `routing`, `topology`, and `number of nodes in the network` can be set through command-line arguments. An accurate description of the software parameters can be found in the software repository.

The simulator constructs a network consisting of s sensor nodes and one sink node. Sensor nodes are deployed on a plane based on one of the following deployment schemes: the *grid* *topology* (GT) and the *random disc topology* (RDT). In the former, sensor nodes are deployed according to a grid structure; each sensor node is equidistant from the closest sensor nodes in cardinal directions. In the latter, sensor nodes are randomly deployed on a disc-shaped plane. The sink node is deployed in the center of the network in both GT and RDT. Both deployment schemes allow the user to maintain the same average node density at diverse values of s.

The wireless communication in the simulated network adheres to the IEEE 802.11n standard for local wireless networks. Our simulator allows the choice between the 2.4 GHz and 5 GHz carrier frequency and various modulation coding schemes [9].

Each node in the WSN has installed the IP protocol, and messages are transmitted over the TCP protocol to ensure a reliable in-order data delivery. Since WSNs are usually characterized by a small Maximum Segment Size (MSS) and Maximum Transmission Unit (MTU) [10], the simulator allows to set these parameters to the required value. Messages are routed using a routing algorithm for multi-hop ad-hoc wireless networks provided by the NS3 environment. We implemented a simplified procedure for the installation of the routing algorithm by setting the specific parameter. The supported routing algorithms for multi-hop ad-hoc wireless networks are: Ad Hoc On-Demand Distance Vector (AODV) [11], Dynamic Source Routing (DSR) [12], Optimized Link State Routing Protocol (OLSR) [13] and Destination-Sequenced Distance Vector (DSDV) [14].

Our simulator implements cryptographic operations using the Libsodium library [15]. Layers of the onion head are encrypted by applying the *Sealed box*, a public-key cryptosystem based on the Curve25519 [16] with the key length of 256b. Therefore, each encryption layer increases the onion head size by 48bytes due to the shared secret.

Onion messages are executing in the WSN as explained in Section 2; moreover, the simulator allows to control the following properties of onion messages:

- *Onion message path length*: number of sensor nodes to include in the onion message path.
- *Uniform onion head size*: the onion head size is maintained uniform by adding padding to the onion head when a layer of the onion head is decrypted.
- *No onion body*: onion messages will not include the onion body.
- *Onion body size*: set the onion body size to emulate the transportation of computer code in the onion body.
- *Data aggregation*: the onion body contains a value, and each sensor node aggregates its sensor reading to the value carried in the onion body.

Due to the high error rate of wireless multi-hop networks and the usually large size of onion messages, we implemented a watchdog timer to interrupt the forwarding of an onion message if it does not reach the next hop within the selected time frame. Furthermore, the watchdog timer will also trigger the issuing of a new onion message having the same properties as the interrupted one.

3.1.1. Simulator output

The simulator output is written on the STDOUT and on an output file. An example of the output file content is listed as given in Box I.

The output file contains a description of the parameters used to run the simulation and four CSV headers. The CSV headers describe the data rows of the output file. In the following paragraph, we describe the data output of the simulation.

The ONION_DETAILS rows include the traveling time of onion messages, which is measured from the issuing by the sink node until the onion message returns to the sink node. ONION_ROUTING rows enclose the time an onion message travels from one node in the onion message path to the next node in the onion message path. Both ONION_DETAILS and ONION_ROUTING include details about the onion message size in bytes and the `onionId` an identifier to uniquely identify an onion message among data rows. TIMEOUT rows notify interrupted onion messages. The NODE_DETAILS rows report (X,Y) coordinates of nodes able to communicate with the sink node (sensor nodes listed in the $\langle IP, public-key\rangle$ structure, as explained in Section 3.2). Moreover, if OLSR routing is selected, also the node degree is reported. We refer to the node degree as the number of one-hop neighbors of a node.

Besides the mentioned output file, each simulation run additionally produces statistics about the communication traffic generated during the simulation. The communication traffic is measured at MAC and application layers separately. Therefore, allowing to compare the ratio between the data transmitted at the application layer and the total communication overhead generated in the network.

3.2. Illustrative example

The simulator provides a simulation environment for the privacy-preserving data aggregation protocol described in Section 2, allowing the user to adjust parameters described in Section 3.1 by setting a configuration file. The simulator is launched by executing one of the following commands in the NS3 home directory. The second command demonstrates the use of command-line arguments.

```
$ ./waf −−run onion−routing−wsn
$ ./waf −−run "onion−routing−wsn −−a_simNum=0 −−a_name=test
−−a_routing=olsr −−a_topology=grid −−a_nodeNumber=10 "
```

At simulator start-up, the configuration file is read, and simulation parameters are set up. Then, the network is constructed according to the selected network topology, and applications are installed on nodes. Since in the RDT, nodes are deployed at random locations in the available disc-shaped space, some nodes might not be able to communicate with the sink node. Therefore, the simulator works in two phases; the first phase is meant to identify sensor nodes reachable by the sink node.

In the first phase, sensor nodes start up sequentially to prevent network congestions, each sensor node sending its public key to the sink node. The sink node holds a list of reachable nodes consisting of pairs $\langle IP, public - key\rangle$, each entry corresponding to a sensor node able to communicate with the sink node.

In the second phase, the sink node starts constructing an onion message by selecting sensor nodes to include in the onion message path. Sensor nodes are randomly selected from the list of reachable nodes. Onion messages are issued by the sink node sequentially; after an onion message returns at the sink node, the following onion message is issued. Onion messages are processed on sensor nodes conditional to parameters set in the configuration file and mentioned in Section 3.1. The simulator ends when all onion messages specified in the configuration file complete their execution.

3.3. Software architecture

The software architecture is presented using the UML class diagram in Fig. 2 to describe the structure and the UML component diagram in Fig. 3 to show the interaction with NS3 modules. At simulator start-up, the WsnConstructor component reads parameters defined in the configuration file using the ConfigStore module, creates nodes and network devices using the Network module, builds the network topology using the Mobility module,

```
——————————————————————————Simulation description——————————————————————————
Simulation name: test_OLSR_GRID_10_0
Simulation randomstream setup, run: 1, seed:1
Total sensornodes: 10 and 1 sink node
Wireless: IEEE 802.11n at 2.4GHz, , DataMode: HtMcs1, ControlMode: HtMcs1, MTU:1280, MSS:536
Network topology: GRID with row size: 3.
Distance between nodes on X—axis: 60m. Distance between nodes on Y—axis: 60m. Sink node located at x:60,y:60
Routing: OLSR
Routing setup time: 60s, nodes are starting sequentially with 200ms interval, onion starts at: 67s
Onion path lengths: 5 10 15 repeated each path length 1 times.
Simulation started at: Mon Nov 22 15:14:57 2021
——csv headers——
onion_details,sim_name,sim_num,num_of_nodes,topology,routing,onion_id,packet_size,onion_head_size
            ,onion_body_size,onion_path_length,sent_at,recv_at,query_time_to_return
onion_routing,sim_name,sim_num,num_of_nodes,topology,routing,onion_id,send_ip,recv_ip,packet_size
            ,onion_head_size,onion_body_size,sent_at,recv_at,hop_time
timeout,sim_name,sim_num,num_of_nodes,topology,routing,onion_id,onion_path_length,abort_time
node_details,sim_name,sim_num,num_of_nodes,topology,routing,coord_x,coord_y,node_degree
—————————————————————————————Simulation output—————————————————————————————
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,1,10.1.0.1,10.1.0.7,411,267,134,67.800000,67.802884,0.002884
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,1,10.1.0.7,10.1.0.4,411,267,134,67.802884,67.817742,0.014858
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,1,10.1.0.4,10.1.0.9,411,267,134,67.817742,67.835166,0.017424
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,1,10.1.0.9,10.1.0.6,411,267,134,67.835166,67.850985,0.015819
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,1,10.1.0.6,10.1.0.10,411,267,134,67.850985,67.872124,0.021139
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,1,10.1.0.10,10.1.0.1,411,267,134,67.872124,67.873909,0.001785
onion_details,test_OLSR_GRID_10_0,0,10,grid,olsr,1,411,267,134,5,67.800000,67.873909,0.073909
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,2,10.1.0.1,10.1.0.10,671,527,134,68.373909,68.378262,0.004352
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,2,10.1.0.10,10.1.0.9,671,527,134,68.378262,68.396232,0.017970
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,2,10.1.0.9,10.1.0.11,671,527,134,68.396232,68.413327,0.017095
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,2,10.1.0.11,10.1.0.4,672,528,134,68.413327,68.421543,0.008216
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,2,10.1.0.4,10.1.0.7,672,528,134,68.421543,68.423928,0.002385
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,2,10.1.0.7,10.1.0.5,672,528,134,68.423928,68.444033,0.020105
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,2,10.1.0.5,10.1.0.4,672,528,134,68.444033,68.461734,0.017701
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,2,10.1.0.4,10.1.0.2,672,528,134,68.461734,68.480690,0.018956
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,2,10.1.0.2,10.1.0.5,671,527,134,68.480690,68.497158,0.016468
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,2,10.1.0.5,10.1.0.9,671,527,134,68.497158,68.517373,0.020215
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,2,10.1.0.9,10.1.0.1,671,527,134,68.517373,68.521615,0.004242
onion_details,test_OLSR_GRID_10_0,0,10,grid,olsr,2,671,527,134,10,68.373909,68.521615,0.147706
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.1,10.1.0.9,931,787,134,69.021615,69.022961,0.001346
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.9,10.1.0.11,931,787,134,69.022961,69.025259,0.002298
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.11,10.1.0.6,931,787,134,69.025259,69.037589,0.012330
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.6,10.1.0.9,932,788,134,69.037589,69.043895,0.006306
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.9,10.1.0.2,932,788,134,69.043895,69.047897,0.004002
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.2,10.1.0.8,932,788,134,69.047897,69.064188,0.016291
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.8,10.1.0.7,932,788,134,69.064188,69.070067,0.005879
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.7,10.1.0.3,932,788,134,69.070067,69.073157,0.003089
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.3,10.1.0.2,932,788,134,69.073157,69.075896,0.002739
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.2,10.1.0.8,932,788,134,69.075896,69.080590,0.004695
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.8,10.1.0.11,932,788,134,69.080590,69.085924,0.005333
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.11,10.1.0.4,932,788,134,69.085924,69.091843,0.005919
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.4,10.1.0.3,932,788,134,69.091843,69.094292,0.002449
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.3,10.1.0.2,931,787,134,69.094292,69.096980,0.002688
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.2,10.1.0.4,931,787,134,69.096980,69.100846,0.003866
onion_routing,test_OLSR_GRID_10_0,0,10,grid,olsr,3,10.1.0.4,10.1.0.1,931,787,134,69.100846,69.105257,0.004411
onion_details,test_OLSR_GRID_10_0,0,10,grid,olsr,3,931,787,134,15,69.021615,69.105257,0.083641
node_details,test_OLSR_GRID_10_0,0,10,grid,olsr,10.1.0.1,60,60,8
node_details,test_OLSR_GRID_10_0,0,10,grid,olsr,10.1.0.2,0,0,3
node_details,test_OLSR_GRID_10_0,0,10,grid,olsr,10.1.0.3,60,0,5
node_details,test_OLSR_GRID_10_0,0,10,grid,olsr,10.1.0.4,120,0,3
node_details,test_OLSR_GRID_10_0,0,10,grid,olsr,10.1.0.5,0,60,5
node_details,test_OLSR_GRID_10_0,0,10,grid,olsr,10.1.0.6,60,180,4
node_details,test_OLSR_GRID_10_0,0,10,grid,olsr,10.1.0.7,120,60,5
node_details,test_OLSR_GRID_10_0,0,10,grid,olsr,10.1.0.8,0,120,5
node_details,test_OLSR_GRID_10_0,0,10,grid,olsr,10.1.0.9,60,120,7
node_details,test_OLSR_GRID_10_0,0,10,grid,olsr,10.1.0.10,120,120,4
node_details,test_OLSR_GRID_10_0,0,10,grid,olsr,10.1.0.11,0,180,3
————————————————————— Simulation end: Mon Nov 22 15:15:00 2021
```

Box I.

sets up the wireless communication using the WiFi module, installs the internet stack using the Internet module, sets up routing using the corresponding module, installs and starts applications on nodes using Network and Core modules. The WsnConstructor component creates one instance of classes OutputManager and OnionValidator passing them to SensorNode and Sink applications. The OutputManager instance manages the output of the simulation writing on the stdout and the output file. The OnionValidator instance allows checking if the onion message reached the next-hop node within the watchdog timer.

In the first phase of the simulation, instances of the class SensorNode send its public-key to the Sink node by calling Handshake(). When the Sink node receives a handshake message, it registers the sensor node in the m_nodeManager <IP,public-key> structure.

All messages traveling the WSN are constructed with the Protobuf library [17] using the ProtoPacket class.

In the second phase of the simulation, the instance of the class Sink creates an onion message by calling SelectRoute() and PrepareOnion(). The method SelectRoute() randomly selects sensor nodes to include in the onion message path from the m_nodeManager <IP,public-key> structure. The method PrepareOnion() uses an instance of the class OnionManager to create the onion head. The OnionManager class extends the OnionRouting class with the implementation of EncryptLayer() and DecryptLayer() using the Libsodium library. The abstract class OnionRouting allows onion head creation and layer decryption while omitting the implementation of encryption/decryption functions. The constructed ProtoPacket instance containing the onion head and the onion body is then sent to a sensor node by calling the function

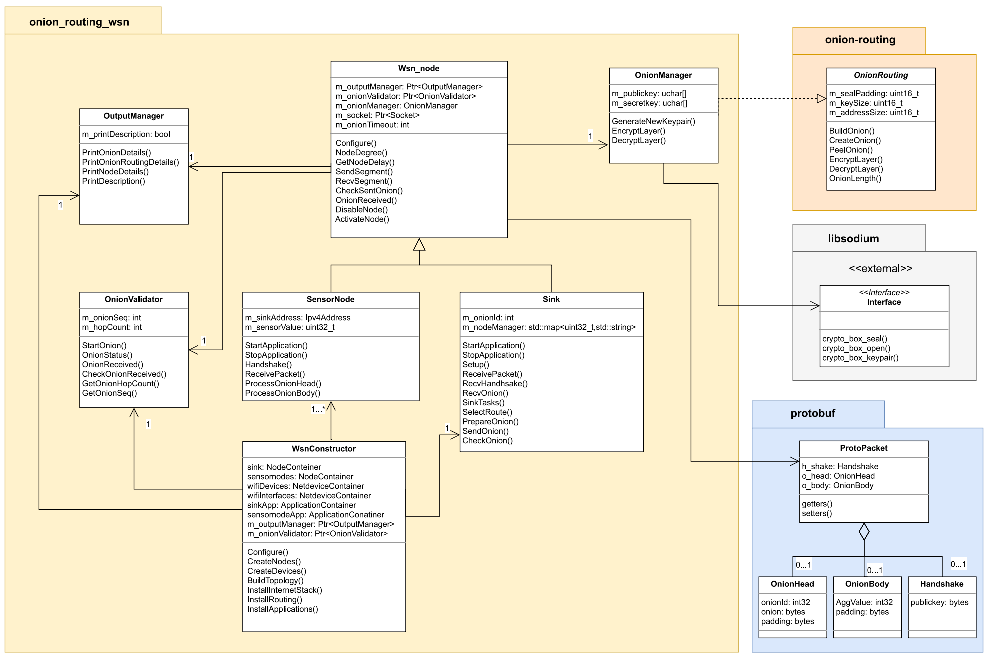


Fig. 2. UML class diagram of the main software components.

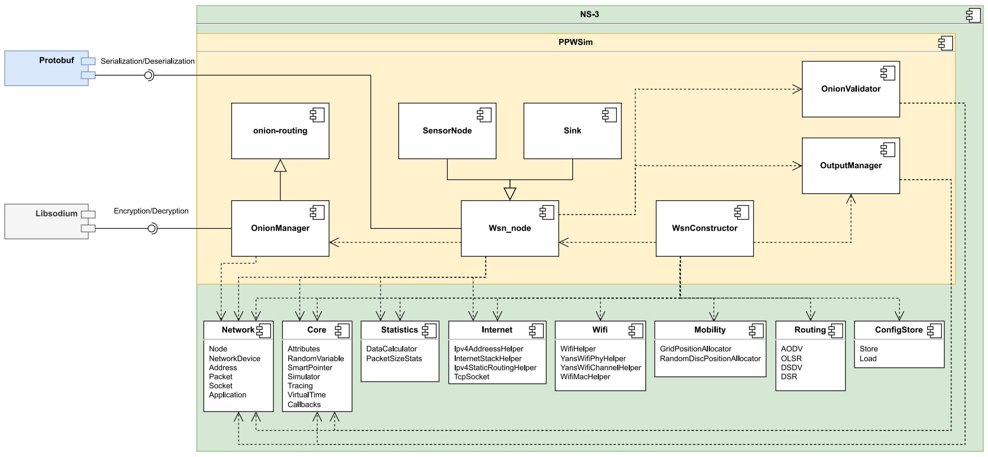


Fig. 3. UML component diagram showing the interaction of PPWSim components with modules of the NS3 environment. Modules of the NS3 environment include denominations of utilized classes.

Wsn_node::SendSegment() through a NS3::TcpSocket provided by the Internet module.

Instances of the class SensorNode that receive the onion message are notifying the OnionValidator instance by calling OnionReceived. The onion message is processed using functions ProcessOnionHead() and ProcessOnionBody(). The onion message is then sent to the next-hop node, and the SensorNode instance sets the watchdog timer. When the timer elapses, the SensorNode instance will verify that the next node received the onion message by calling CheckOnionReceived() of the OnionValidator instance. If the next node does not receive the onion message, the onion message forwarding will be interrupted by setting the m_onionSeq=0 of the OnionValidator instance. The OnionValidator instance is shared between all nodes; therefore, the Sink instance recognizes the interrupted onion message and starts issuing a new onion message with equal properties to the interrupted one.

The Wsn_node class also provides methods DisableNode() and ActivateNode() for disabling and re-enabling a WSN node. The disable function acts at the physical layer, making the node unreachable for other nodes and unable to participate in the routing.

4. Impact

The PPWSim is in the base a WSN simulator with an added usecase, the data and origin privacy-preserving protocol based on onion routing. Experiments requiring the basic WSN network as well as experiments requiring the additional privacy-preserving characteristics can be simulated on the proposed software.

The WsnConstructor component allows effortless manipulation of generic WSN properties through a configuration file; furthermore, the component is structured in layers corresponding to the ISO/OSI model [18], allowing intuitive extension of functionalities. The Wsn_node component represents a generic WSN node that provides transport layer services, data interchange through Protobuf [17] objects, access to node location information, node deactivation/reactivation, and centralized management of the simulator's output through the OutputManager component. Therefore, the Wsn_node component can be extended to model sensor node or sink node behavior of specific WSN applications, and the Protobuf integration facilitates the development of communication protocols. The OnionRouting class provided by the PPWSim is encryption agnostic and can be re-used to develop other simulators using the onion routing technique [8]. Moreover, PPWSim is based on the NS3 simulation environment, allowing further extension with components provided by the NS3. The NS3 provides: error models to simulate noisy channels, PCAP traces useful for studying location privacy protection in WSNs [19], mobility models to simulate specific scenarios requiring the motion of nodes [20], and many other tools.

The authors were not able to find any existing simulation that could be adaptable to special requirements, such as the privacy preservation that was the main goal of the research described in [7].

An important portion of the scientific research needs to be supported by empirical evaluation. New scientific ideas can be simulated in specially developed software, but the implementation time along with the inevitable software production problems can deter scientists from pursuing further research. Simulation environments, such as NS3, help in rapid development of specialized simulation environments. After a thorough research of the available opensource tools, the authors were unable to find a suitable solution for at the time of writing for the data and origin privacy-preserving WSN network simulating environment. The proposed solution is based on an industry leading simulation environment incorporating the proposed properties.

The software has been used on a few projects at the authors' home institutions University of Primorska and Innorenew CoE. The most notable projects were "Intelligent floor" presented in [21] and "Mrakova domačija historical site", presented in [22]. Already published results presented in [23] are based on the presented software simulator.

5. Conclusions

This paper presents PPWSim, a simulator based on the NS3 [6] simulation environment. PPWSim was designed specifically to study the communication protocol presented in [7]. The simulator allows the user to set simulation parameters affecting the traveling time of messages in the network. Besides future investigations on the solution described in [7], the presented software could serve as a framework for developing other simulators in the scope of WSNs or simulators using the onion routing technique [8].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors gratefully acknowledge the European Commission for funding the InnoRenew project (Grant Agreement #739574) under the Horizon2020 Widespread-Teaming program and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Regional Development Fund). They also acknowledge the Slovenian Research Agency ARRS for funding the project J2-2504.

References

[1] Chan H, Perrig A. Security and privacy in sensor networks. Computer 2003;36(10):103–5. http://dx.doi.org/10.1109/MC.2003.1236475.
[2] Conti M, Zhang L, Roy S, Di Pietro R, Jajodia S, Mancini LV. Privacy-preserving robust data aggregation in wireless sensor networks. Secur Commun Netw 2009;2(2):195–213. http://dx.doi.org/10.1002/sec.95.
[3] Bista R, Chang J-W. Privacy-preserving data aggregation protocols for wireless sensor networks: a survey. Sensors 2010;10(5):4577–601. http://dx.doi.org/10.3390/s100504577.
[4] Zhang L, Zhang H, Conti M, Di Pietro R, Jajodia S, Mancini LV. Preserving privacy against external and internal threats in WSN data aggregation. Telecommun Syst 2013;52(4):2163–76. http://dx.doi.org/10.1007/s11235-011-9539-8.
[5] Xu J, Yang G, Chen Z, Wang Q. A survey on the privacy-preserving data aggregation in wireless sensor networks. China Commun 2015;12(5):162–80. http://dx.doi.org/10.1109/CC.2015.7112038.
[6] Henderson TR, Lacage M, Riley GF, Dowell C, Kopena J. Network simulations with the ns-3 simulator. In: ACM SIGCOMM, Vol.14. (14):2008, p. 527.
[7] Hrovatin N, Tošić A, Mrissa M, Vičič J. A general purpose data and query privacy preserving protocol for wireless sensor networks. 2021, arXiv:2111.14994.
[8] Syverson PF, Goldschlag DM, Reed MG. Anonymous connections and onion routing. In: Proceedings. 1997 IEEE symposium on security and privacy (cat. no. 97CB36097). IEEE; 1997, p. 44–54. http://dx.doi.org/10.1109/SECPRI.1997.601314.
[9] Paul T, Ogunfunmi T. Wireless LAN comes of age: Understanding the IEEE 802.11 n amendment. IEEE Circuits Syst Mag 2008;8(1):28–54. http://dx.doi.org/10.1109/MCAS.2008.915504.
[10] Bettoumi B, Bouallegue R. LC-DEX: Lightweight and efficient compressed authentication based elliptic curve cryptography in multi-hop 6lowpan wireless sensor networks in HIP-based Internet of things. Sensors 2021;21(21):7348. http://dx.doi.org/10.3390/s21217348.
[11] Perkins C, Belding-Royer E, Das S. RFC3561: Ad hoc on-demand distance vector (AODV) routing. Tech. rep., RFC Editor; 2003, http://dx.doi.org/10.17487/RFC3561.
[12] Johnson DB, Maltz DA, Broch J, et al. DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. Ad Hoc Netw 2001;5(1):139–72.
[13] Clausen T, Jacquet P, Adjih C, Laouiti A, Minet P, Muhlethaler P, Qayyum A, Viennot L. Optimized link state routing protocol (OLSR). Network working group, 2003, URL https://hal.inria.fr/inria-00471712.
[14] Perkins CE, Bhagwat P. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. ACM SIGCOMM Comput Commun Rev 1994;24(4):234–44. http://dx.doi.org/10.1145/190809.190336.
[15] Libsodium the sodium crypto library. 2021, https://libsodium.gitbook.io/doc/ (Accessed: 28 May 2021).
[16] Bernstein DJ. Curve25519: new Diffie-Hellman speed records. In: International workshop on public key cryptography. Springer; 2006, p. 207–28. http://dx.doi.org/10.1007/11745853_14.
[17] Google protocol buffers: Google's data interchange format. 2021, https://github.com/google/protobuf/ (Accessed: 22 Nov 2021).
[18] Zimmermann H. OSI reference model-the ISO model of architecture for open systems interconnection. IEEE Trans Commun 1980;28(4):425–32. http://dx.doi.org/10.1109/TCOM.1980.1094702.
[19] Jiang J, Han G, Wang H, Guizani M. A survey on location privacy protection in wireless sensor networks. J Netw Comput Appl 2019;125:93–114. http://dx.doi.org/10.1016/j.jnca.2018.10.008.

[20] Mehto A, Tapaswi S, Pattanaik K. Virtual grid-based rendezvous point and sojourn location selection for energy and delay efficient data acquisition in wireless sensor networks with mobile sink. Wirel Netw 2020;26(5):3763–79. http://dx.doi.org/10.1007/s11276-020-02293-4.

[21] Tošić A, Hrovatin N, Vičič J. Data about fall events and ordinary daily activities from a sensorized smart floor. Data Brief 2021;37:107253. http://dx.doi.org/10.1016/j.dib.2021.107253.

[22] Mrissa M, Tošić A, Vičič J, Burnard M. Distributed ledgers and decentralized WoT architectures. In: Integrating sustainability and health in buildings through renewable materials, innorenew coe international conference. 2020, p. 39.

[23] Hrovatin N, Tošić A, Vičič J. In-network convolution in grid shaped sensor networks. J Web Eng 2022;21:1–22. http://dx.doi.org/10.13052/jwe1540-9589.2114.

2.4 Paper 4

Title: A General Purpose Data and Query Privacy Preserving Protocol for Wireless Sensor Networks
Authors: Niki Hrovatin, Aleksandar Tošić, Michael Mrissa, Jernej Vičič
Year: 2023
Journal: IEEE Transactions on Information Forensics and Security
DOI: `https://doi.org/10.1109/TIFS.2023.3300524`
Link: `https://ieeexplore.ieee.org/abstract/document/10198325`

IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 18, 2023 4883

A General Purpose Data and Query Privacy Preserving Protocol for Wireless Sensor Networks

Niki Hrovatin, Aleksandar Tošić, Michael Mrissa, and Jernej Vičič

***Abstract*— The large number of devices characterizing Wireless Sensor Networks (WSNs) provide the benefits of observing, tracking, and recording everything; nonetheless, the cumulative computing power of those devices is typically not utilized, and the few implementations taking advantage of it neglect privacy or are application-specific. This manuscript describes a privacy-preserving protocol that enables WSN nodes to jointly compute an arbitrary function without disclosing their own private inputs. The computation takes place at data source nodes, while computation instructions and intermediate results move across the network secured by cryptography. The protocol relies on the Onion Routing technique to provide uniformly distributed network traffic and confine the knowledge a foreign actor can gain from monitoring messages traveling the network. We show that the communication protocol is privacy-preserving against the external and internal attacker models, and we validate our protocol implementation using the NS3 network simulator.**

***Index Terms*— Wireless sensor network, privacy, onion routing, distributed computing, data aggregation.**

I. Introduction

IN THE last decade, the cost reduction of sensor production and microelectronics has contributed to the development of large scale Wireless Sensor Networks (WSNs). WSNs are composed of dozens or hundreds of sensing nodes interlinked via radio signaling and meant to be easily deployed, self-configurable, and low cost. Moreover, nowadays, WSN nodes are powerful enough to store a short history of sensed data and process it. Therefore, the edge computing paradigm, which consists in moving computations as close as possible to data sources applies also to WSNs. Information fusion [1], declarative query processing [2], and inference [3] were all considered for in-network execution.

Manuscript received 24 October 2022; revised 30 May 2023 and 12 July 2023; accepted 24 July 2023. Date of publication 1 August 2023; date of current version 11 August 2023. This work was supported in part by the European Commission for funding the InnoRenew Project through the Horizon2020 Widespread-Teaming Program and the Republic of Slovenia (Investment Funding of the Republic of Slovenia and the European Regional Development Fund) under Grant 739574 and in part by the Slovenian Research Agency ARRS under Project J2-2504. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Edgar Weippl. *(Corresponding author: Niki Hrovatin.)*

Niki Hrovatin, Aleksandar Tošić, and Michael Mrissa are with the InnoRenew CoE, 6310 Izola, Slovenia, and also with the Faculty of Mathematics, Natural Sciences and Information Technologies, University of Primorska, 6000 Koper, Slovenia (e-mail: niki.hrovatin@famnit.upr.si).

Jernej Vičič is with the Research Centre of the Slovenian Academy of Sciences and Arts, The Fran Ramovš Institute, 1000 Ljubljana, Slovenia, and also with the Faculty of Mathematics, Natural Sciences and Information Technologies, University of Primorska, 6000 Koper, Slovenia.

Digital Object Identifier 10.1109/TIFS.2023.3300524

However, in a typical WSN, data is moving through a wireless multi-hop network without infrastructure, nodes composing a WSN are generally low-cost and resource-constrained devices; therefore, it is challenging to secure the network and guarantee adequate privacy [4]. Moreover, in-network processing tasks require that several WSN nodes contribute with their sensed data and processing power to the joint evaluation of a function; therefore, the privacy of nodes contributing to the processing is also threatened by other nodes participating. Research to date in the WSN field has focused on providing confidentiality by transforming the data using cryptography [5] or other obfuscation techniques [6] and allowing several nodes to operate over the transformed data without the possibility of extracting the private data from other nodes involved in the joint computation. This approach led to the development of several solutions [7]; nonetheless, in solutions considered practical for WSNs, the technique used to conceal data restricts the set of operations applicable to the transformed data. There is some evidence that anonymous communication could provide privacy for joint computation tasks [8]; however, due to the high privacy requirements of some applications and the data being typically self-descriptive, this category of approaches has been neglected.

In this manuscript, instead of allowing a set of WSN nodes to evaluate a function over obfuscated data, we exploit anonymity to conceal nodes involved in the computation, establishing a strong separation between context and data. The WSN is a system of devices sensing features of the surrounding environment. The data collected by sensor nodes describe events or changes; the sensor node's location reveals where the phenomena occurred. Therefore, the WSN data loses its descriptive utility without enough contextual information. To achieve the separation between data and context, we design a novel communication protocol inspired from the Onion Routing [9] protocol that allows only specific nodes to participate in the joint evaluation of a function while concealing them among a set of nodes performing the onion routing operation without accessing the computation. Nodes participating in the joint computation receive a message enclosing computation instructions and a partial result. Privacy is secured as the technique conceals the context of the partial result from participating sensor nodes. Although the data, represented by the partial result, can be observed by other nodes involved in the computation, the context – such as information on the number, identity, and location of nodes contributing to the partial result – remains hidden. Therefore, without access to key contextual information, the partial result loses its significance for all nodes obtaining it, except for the message origin, which defined the message path and, as such, inherently possesses knowledge of the context.

Throughout the manuscript, we are looking into the specific case of indoor air quality monitoring since it receives an increasing interest as it contributes to reducing the environmental impact of buildings, to improving building occupants' well-being, and to enhancing future building design [10], [11]. WSN deployed for indoor air quality monitoring relies on sensors collecting data about temperature, CO_2, VOC, PM, relative humidity, etc.

Indeed, a large WSN relying on high quality transducers can reach prohibitive costs. Air quality transducers are particularly expensive, and low cost options are raising interest [12]. A common practice to reduce the overall WSN cost is the dense deployment of nodes equipped with low-cost transducers. Reading quality is then improved by aggregating data from multiple nodes. However, in typical WSN implementations, the data is aggregated only at network endpoints resulting in high communication overhead since each node needs to report its sensed value and missing the opportunity to take advantage of the nodes' computing capabilities when conveying the whole processing load on the end system.

There are several techniques for aggregating sensor readings in-network as data moves towards the sink nodes [13]. These techniques also provide privacy preservation focusing on data confidentiality and are designed to operate in periodic reporting settings. Whereas if these techniques are applied in the on-demand setting to retrieve data from the building's individual locations, the resulting network traffic could be informative. In particular, wireless communications use the radio frequency spectrum to broadcast signals over the air; transmitted signals can be intercepted, then analyzed and backtracked to illicitly acquire information. Several studies focus on this complication considering location privacy protection [14] to conceal the location of sink nodes or detected events in WSNs deployed in unattended environments.

However, the adoption of WSNs for indoor monitoring is increasing [15], [16], [17], [18], raising the need for further privacy measures against traffic analysis when aggregating data in-network in on-demand settings. Traffic analysis attacks [19] are of particular concern in building monitoring. These attacks allow adversaries to remain unnoticeable, simply listening to network traffic, and extracting features like message size, frequency, processing time, etc. By associating these features with facts or secrets, machine learning techniques can infer important details about the monitored environment, which could potentially lead to the compromise of building security.

This context motivates the need to combine privacy preservation and distributed data computing on WSN nodes to employ node's computing capabilities without revealing sensor readings or sensitive contextual information.

In this paper, we describe the following contributions:

1) We present a communication protocol that allows WSN nodes to jointly compute an arbitrary function over their inputs while ensuring the privacy of inputs and operating without revealing significant contextual information.
2) The communication protocol is based on a novel use of the Onion Routing [9] technique.
3) We provide privacy preservation analysis showing that the communication protocol is secure against the external and internal attacker models.
4) We provide results of the privacy-preserving protocol simulated using the NS3 simulator [20].

The rest of this paper is organized as follows: Section II gives a brief overview of the privacy preserving protocol. Section III reviews related work and highlights the originality of our solution. Section IV details the general-purpose data and query privacy preserving protocol. Section V gives privacy preservation analyses. Section VI presents results of the privacy preserving communication protocol simulated using the network simulator NS3. Section VIII concludes the manuscript and gives guidelines for future work.

II. Solution Overview

This manuscript describes a communication protocol based on the Onion Routing [9] technique for anonymous communication over a computer network. We similarly employ messages structured into encryption layers, such that a layer can be decrypted only by the targeted node revealing an inner encryption layer addressed to another node in the network. Therefore, message decryption is carried out gradually by leading the layered message across WSN nodes following the precise order given at message construction. Encryption layers are not enclosing only the inner layer, but also additional secret information revealed only to the node decrypting that layer. Path details and encryption keys are in this way conveyed to in-path nodes. Path details are delivered in encryption layers to not disclose the whole message path, such that a node receiving the message can identify only the previous sender and the next receiver of the message. Encryption key pairs, however, are delivered only to a subset of nodes in the message path. Moreover, and differently from the traditional onion routing [9], encryption keys are not used to establish an anonymous channel. Instead, encryption keys give access to the payload accompanying the layered object. Please note that pairs of symmetric encryption keys include distinct keys; however, pairs are chained through layers of the layered object, as can be seen from Fig. 1. Therefore, nodes in the message path serve as the anonymity set[1] for nodes accessing the payload since each node in the message path could potentially receive symmetric encryption keys from the decryption of the layered object. Consequently, the identity of nodes accessing the payload remains concealed to nodes receiving the message.

In this paper, the WSN acts as a service. Authorized users construct queries and issue them to WSN gateway nodes. We refer to a query as the message composed of a head

[1]Based on the definition given by Pfitzmann and Köhntopp [21], the anonymity set is the set of subjects that might cause an action. In our protocol, the anonymity set is the set of nodes deciphering a layer of a layered object. If layer decryption reveals encryption key pairs to a node, then the node executes the computation; the action.

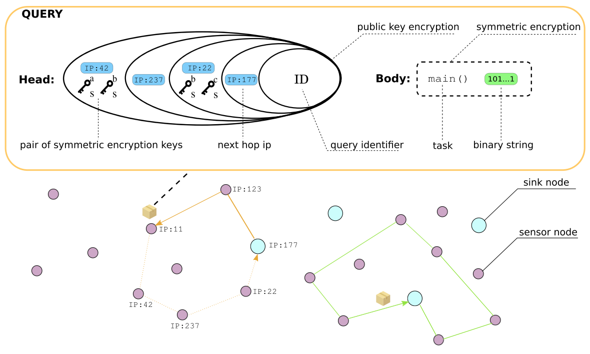


Fig. 1. Representation of the query, made of the head and the body. Notice that symmetric encryption keys are not enclosed in each layer of the query head. The figure displays the query's path forming a circuit.

consisting of the previously described layered object made of public-key encryption layers and a body consisting of the pair <*task, binary string*> as shown in Fig. 1. We refer to a task as computation instructions specifying the operations to be performed by sensor nodes. The binary string is of fixed size and stores task execution results. Each query head is constructed to lead the query over a path closing a circuit and delivering symmetric encryption key pairs to specific nodes in the query path allowing only them to access the query body content, execute the task and store results to the binary string. Other nodes in the query path not receiving symmetric encryption keys from query head decryption cannot access the query body content and only forward the query to the next-hop node. The query path closes a circuit, conveying back the results of the joint computation.

III. Related Work

Many systems were developed to preserve users' privacy while communicating on large public networks like the Internet. Several solutions originate from the work of *Chaum* [22] on mixnet. Mixnet-based schemes rely on a set of mix servers that receive encrypted messages, and after a sufficiently large amount of time, messages are re-ordered and released in batches to hide the correspondence between sender and receiver. Onion Routing [9] is a solution to preserve users' anonymity while avoiding latency introduced by mix servers. The solution relies on source routing using multiple encryption layers to route a message through a set of at least three routing servers (onion routers) to create an anonymous connection. Source routing [23] is a technique to route a message through a set of nodes by encoding path information in the message. In onion routing, path information is encrypted in each encryption layer of the message, and when the message is routed through onion routers, at each hop, a layer of encryption is removed from the message revealing the next hop. Therefore, no one of the actors involved in the communication will know the whole message path apart from the source of the message. In TOR [24] the second-generation onion routing, the message path is similarly protected by encryption layers, but the anonymous connection is established incrementally using key exchange schemes. Although the mentioned solutions effectively provide anonymous communication between two parties, they rely on the background traffic of large networks. Furthermore, the mentioned solutions protect the logical location of communicating parties (IP-address), while in WSNs, the privacy of communicating nodes can also be disclosed by observing the physical wireless communication. Despite the computational requirements and potential eavesdropping threats, the application of onion routing in WSNs has been proposed several times in the literature [25], [26], [27]. The paper [27] proposes an onion routing based mechanism for MQTT protocol communications, leveraging dynamic broker bridging to enable smart devices to subscribe and publish messages anonymously. In [26], a trust-based secure directed diffusion routing protocol for WSNs is proposed. The protocol leverages the onion routing for secure and anonymous end-to-end data transmission.

The literature review conducted by *Li et al.* [7] divides the privacy problem in WSNs into data privacy and context privacy. Data privacy is achieved if a communication protocol does not leak the collected data to external and internal adversaries. External adversaries eavesdrop on wireless communications, while internal adversaries have knowledge of some encryption keys used in the sensor network. In contrast, context privacy relates to communication traffic characteristics, as this can reveal insights over activities in the monitored environment. The paper [28] highlights the importance of concealing the sensor technology equipped on smart home devices, and the authors describe a technique for the periodic change of device identity that protects the network against external adversaries. Location privacy protection was extensively studied in event-driven WSNs using various routing strategies [14] aiming to conceal the location of data source nodes and sink nodes. The location of data source nodes can reveal insights over events detected by the WSN. Whereas keeping secret the location of sink nodes precludes the attacker from physically destroying sink nodes, which are of central importance for the correct functioning of the WSN. However, applying location privacy protection schemes in our scenario is not adequate since we aim to aggregate data from multiple nodes that may be closely located, and routing strategies designed to anonymize source and destination might disclose the region of interest.

The straightforward privacy-preserving approach to retrieve data from a region of interest in a WSN was highlighted by *Carbunar et al.* [29], and consists in gathering data from all sensor nodes in the network and then keep readings only from the sensor nodes of interest. Although effective, this approach is highly inefficient due to the multi-hop routing in WSNs requiring data to be relayed several times to reach the sink node. The efficiency of this approach was improved in [30] using compressive sensing [31] and public-key homomorphic encryption [32]. Compressive sensing is applied to transform sensor readings in a vector of coefficients which is aggregated with other node vectors along the routing paths to the sink node; therefore, requiring a low communication overhead of $O(MN)$, M the size of the vector of coefficients. Homomorphic encryption ensures privacy when aggregating vectors of coefficients.

The solution efficiently gathers data from multiple sensor nodes. However, in WSNs, neighboring nodes often share the same monitored area, and data sensed from neighbor nodes is often correlated. Therefore, data gathering schemes collecting raw data gather many duplicated data, do not utilize sensor node computing capabilities to process data, and pose a substantial communication load on the network even if the data need involves a small subset of network nodes.

In-network data aggregation [33] could effectively reduce the network's communication overhead and employ sensor node processing capabilities by aggregating multiple sensor node readings along routing paths toward the sink node. Therefore, the whole network works as a distributed processing mechanism delivering the final aggregated value to the sink node. Privacy preserving data aggregation solutions ensure data privacy against external and internal attackers [34], [35], [36]. However, proposed techniques do not consider the triggering of a data aggregation process that affects only a subset of nodes in the network without disclosing participating nodes. In building monitoring, the retrieval of aggregated data from the whole network of sensors can approximate the air quality. However, obtaining data from individual building locations is imperative to give a granular assessment.

The field of secure multi-party computation is concerned with enabling a set of parties to jointly compute an arbitrary function without disclosing their inputs [37]. A considerable amount of literature has been published on this topic, and some schemes have been implemented [38]. However, the secure multi-party computation fundamental protocols are resource demanding, and current implementations supporting arbitrary computation are not yet adequate for WSNs.

According to *Carbunar et al.* [29] a privacy-preserving query mechanism in WSNs must hide from attackers the location and identity of queried sensor nodes but also the relationship between individual queries while maintaining an adequate trade-off between privacy and efficiency. *Carbunar et al.* addressed query privacy needs with a WSN that acts as a service accessible through dedicated servers. The proposed solution hides query details from servers that provide access to the WSN. The WSN is mapped into regions and queries target aggregator nodes of individual regions. Query privacy is assured using source routing and by hiding a query constructed by the client with additional bogus queries targeting different regions of the WSN.

De Cristofaro et al. [25] proposed a privacy-preserving solution to retrieve WSN data without disclosing the identity of data source nodes to the network owner or attackers. The solution relies on source routing [23] using the onion routing to hide the query path and symmetric encryption to provide data privacy and data integrity. However, the proposed solution allows only retrieving individual sensor node readings without the possibility of in-network processing.

We state that our solution intrinsically follows the work on query privacy started by *Carbunar et al.* and *De Cristofaro et al.*. The originality of our proposed solution lies in the unique use of the onion routing technique to conceal nodes participating in the joint computation that takes place in-situ. Therefore, without aggregator nodes. Since aggregator nodes are gathering data from multiple sensor nodes, they are an appealing target for attackers and a point of failure for the network.

To the best of our knowledge, this is the first solution that allows the in-network joint computation of arbitrary functions in WSNs without aggregator nodes and keeps the participating nodes and their inputs private.

IV. Definition of the Privacy-Preserving Protocol

A. The WSN Model

Throughout the manuscript, we consider a WSN as a wireless multi-hop network consisting of two types of nodes. The majority are nodes equipped with sensing technology; we refer to these nodes as sensor nodes. The other type of node is named sink node, which purpose is to act as a gateway to external systems. The WSN relies on a routing protocol for multi-hop wireless networks, and sensor nodes in the WSN are configured at deployment with a static IP address and a public-private key pair. Sink nodes maintain a registry holding the following information of sensor nodes: IP address, public key, sensed physical quantities, and location of the building in which the sensor node is positioned (e.g., *room237*). The sink node registry can be accessed by authorized users willing to query the WSN. For simplicity, we refer to an authorized user as the user.

B. The Privacy-Preserving Communication Protocol

The network operates following an on-demand model: the user first access the sink node registry to obtain information about the WSN nodes, then constructs one or multiple queries as explained in Section IV-B.2; each query consists of a head and a body.

- **Head:** an onion-like structure made of encryption layers, the head is of fixed size L_H bytes. Layer decryption reveals the next-hop address (IP) or the next-hop address and a pair of symmetric encryption keys.
- **Body:** consists of t the task and w a fixed-size binary string used to transport the task execution results back to the user that issued the query. The query body is encrypted using symmetric encryption and is of fixed size L_B bytes.

Queries issued to the network follow the query path encoded in the query head, query processing at sensor nodes is explained in Section IV-B.3, and it branches based on the case that query head decryption reveals symmetric encryption keys. We refer to *decoy nodes* as the sensor nodes in the query path that do not receive symmetric encryption keys and do not participate in the joint computation and to *target nodes* as sensor nodes in the query path that receive symmetric encryption keys; thus, they can decipher the query body and participate in the joint computation. All queries travel a path forming a circuit that ends at the sink node that was the entry point of the query.

1) Query Preparation: The client application syncs to a sink node, downloading the registry of WSN nodes, then the user describes a data need expressing the request R_{eq}

consisting of the operation and its target. The operation is described by giving the task t, and the operation target by specifying τ, a set of one or multiple locations of the WSN.

Although the proposed solution allows conveying arbitrary computation instructions to sensor network nodes, the set of supported operations is bounded by the design of our protocol. Each query traverse several sensor nodes, and at each target node, the operation t is processed, acquiring input from sensors equipped on the sensor node and from w the binary string that carries results of t processed on the previous target node. Therefore, the solution is supporting operations that produce a partial result while feeding on the input of sensor values and the previous partial result. Between the common supporting data aggregation operations, we could list average, sum, max but also variance and standard deviation since in [39] was shown how to compute them as additive aggregation.[2] Moreover, our solution allows to pose conditions on the data to be retrieved, like the exceeding of a threshold value or past sensed events. Conditions can be posed not only on the physical quantity to retrieve but also on the state of other sensor technology equipped onto sensor nodes.

Algorithm 1 Query Path Selection

```
Require: $U$, $Q$
Ensure: $S$, $K$, $e_F$, $e_L$
  procedure RANDOM
      return a float from an uniform distribution bounded by (0,1)
  procedure PICKDECOY
      Chose randomly $s \in U \setminus (Q \cup B)$, add $s$ to $B$
      return $s$
  procedure PICKTARGET
      Chose randomly $s \in Q$, remove $s$ from $Q$, add $s$ to $B$
      return $s$
  procedure GENERATESYMKEY
      return a valid symmetric encryption key
  Compute:
  $B, S, K = \emptyset$
  $e_F$ =GENERATESYMKEY( )
  $k = null, e_L = e_F$
  $i, t = 1$
  $l = min(\|Q\|, \phi)$
  while $i \leq l$
      $t = \lceil$RANDOM( ) $* (n - 1)\rceil$
      if S[t] == null
          $S[t]$ = PICKTARGET( )
          $i++$
      end if
  end while
  $i = 1$
  while $i \leq n$
      if S[i] != null
          k = ($e_L$,GENERATESYMKEY( ))
          $e_L$ = k[2]
      else
          S[i] = PICKDECOY( )
          k = null
      end if
      $K[i] = k$
      $i++$
  end while
```

A request R_{eq} specified by the user is processed as follows: t and τ are used to generate $Q \subset U$, the set of sensor nodes that are targets of the request. U is the set of all sensor nodes in the network. The set Q is generated by selecting sensor nodes from the set U that meet the location detailed in τ and sensed physical quantities required in t. The set P is then generated with the algorithm 1 repeated until $Q = \emptyset$. Algorithm 1 will eventually empty the set Q, since every call of the function *pickTarget()* removes a target from Q and inserts it in the query path. Each run of the algorithm will generate a query definition detailed by the tuple (S, K, e_F, e_L, id).

- $S = \langle s_1, \ldots, s_n \rangle$ a list consisting of sensor nodes belonging to the set U. The list S defines the query path.
- $K = \langle k_1, \ldots, k_n \rangle$ a list of elements k_i, where $k_i = (e_a, e_b)$ if s_i is a target node, otherwise s_i is a decoy node and $k_i = null$. (k_i and s_i the i-th elements of their respective list K and S, and (e_a, e_b) a pair of not equal symmetric encryption keys.) Moreover, encryption keys in the set K are arranged as follows: if s_i is a target node and s_j is the next target node in the query path, then $k_i = (e_a, e_b)$ and $k_j = (e_b, e_c)$.
- e_F the first symmetric key. If s_i is the first target node in the query path S, then $k_i = (e_F, e_x)$.
- e_L the last symmetric key. If s_i is the last target node in the query path S, then $k_i = (e_y, e_L)$.
- id a string of bits serving as the query identifier.

By our solution design, the query path length is a fixed network parameter; therefore, each query definition generated using algorithm 1 will include n nodes in its path. Algorithm 1 iteratively generates a query definition using two `while` loops. The first loop will inserts ϕ target nodes at random positions in the query path. Uncertainty is introduced to prevent queries from having a predictable disposition of target and decoy nodes. Since the last node in the query path can identify its function of being the node that will forward the query back to the sink node, by algorithm 1 this node is always a decoy node. The second loop will fill the query path with randomly chosen decoy nodes.

Algorithm 1 also outputs e_F and e_L. e_F is the first symmetric encryption key assigned to the first target node in the query path, this key is used to apply the first encryption layer on the query body. e_L is the second symmetric encryption key assigned to the last target node in the query path, this key is used by the client to decrypt the query body, and retrieve the query result.

The key e_L is coupled to the query identifier id and the pair is mapped into π the set of recovery rules. Since a request will inquire data from a large set of nodes, and each query can include at most ϕ target nodes in its path, the typical request will be accomplished by issuing multiple queries. The set of recovery rules π holds identifiers of queries issued to accomplish one request, and is used to recover the final request result.

To summarize, the request R_{eq} is translated into P – a non empty set of tuples of cardinality $\|P\| = \left\lceil \frac{\|Q\|}{\phi} \right\rceil$, a task t, and π a set of recovery rules. For each tuple $(S, K, e_F, e_L, id) \in P$ a query is constructed as explained in Section IV-B.2.

2) *Query Construction:* In this section, we describe how the client application converts a query definition detailed by the tuple (S, K, e_F, e_L, id), and a task t into a query consisting of the *head* and the *body*. The $\varepsilon(\cdot)$ denotes the encryption operation using public-key cryptography, and the $E(\cdot)$ denotes the encryption operation using symmetric cryptography.

a) Head construction: The query head construction starts from OR_{n+1}, the innermost encryption layer, which securely delivers the query identifier back to the query issuer node. We refer to the query issuer as the sink node that dispatches the query to the network. The innermost onion layer is formed via encryption of the query identifier id and the padding p using the issuer's public key Y_{sink}. The padding p is introduced to

[2]In additive aggregation, a sensor node sums its sensed value with a received partial result, and forwards the sum to the next sensor node.

maintain the head of fixed size if the query includes fewer than $\lfloor n/2 \rfloor$ target nodes. The following equation describes how to compute the innermost onion layer.

$$OR_{n+1} = \varepsilon_{Y_{sink}}(id, p)$$

Next, the client will compute the layer OR_n. This layer is closing the circuit, forwarding the query back to the query issuer. The layer is committed to s_n the last sensor node of the list S. The layer OR_n is computed like the layer OR_i, with the sole exception of including the sink node ip address ip_{sink} as the next-hop address. Therefore, we omit explaining layer OR_n construction, and we give layer OR_i construction in the following lines.

The layer OR_i addressed to the sensor node $s_i \in S$ (i as index of the i-th element in lists S, K and index of the i-th encryption layer of the query head) is computed in two distinct ways. *A:* following equation 1 if $k_i = null$, therefore node s_i is a decoy node. Layer OR_i is computed via encryption of the next hop ip address $ip_{s_{i+1}}$, and previous onion layer OR_{i+1} using the public key Y_{s_i} belonging to the sensor node s_i. *B:* equation 2 is applied if $k_i = (e_a, e_b)$, therefore node s_i is a target node. Layer OR_i is computed via encryption of the next hop ip address $ip_{s_{i+1}}$, the two symmetric encryption keys e_a and e_b, and the previous onion layer OR_{i+1} using the public key Y_{s_i} belonging to the sensor node s_i.

$$OR_i = \varepsilon_{Y_{s_i}}(ip_{s_{i+1}}, OR_{i+1}) \tag{1}$$

$$OR_i = \varepsilon_{Y_{s_i}}(ip_{s_{i+1}}, e_a, e_b, OR_{i+1}) \tag{2}$$

The layer construction repeats until the formation of OR_1, the head's first encryption layer, which is always of size L_H bytes.

b) Body construction: The query body B includes the task t and a fixed size binary string. Since the query body must be of fixed size L_B bytes, and the size of task t can vary, additional padding p of $L_t - size(t)$ bytes must be included into the query body. (L_t the maximum allowed task size in bytes, and $size()$ the function that returns the number of bytes of the given argument) Then the query body is constructed by encrypting the binary string w, the task t, and padding p using the symmetric encryption key e_F. Query body construction can be summarized using the following equation:

$$B = E_{e_F}(w, t, p)$$

Now the query is complete: OR_1 the query head and B the query body. The query coupled with the address of the first node in the query path $s_1 \in S$ and the query identifier is forwarded to the sink node. The sink node memorizes the query identifier and issues the query to s_1.

3) Query Processing: A sensor node $s_i \in S$ (i-th node in the list S) receiving the query performs the following steps: *query decryption*, *task execution*, and *query forwarding*.

a) Query decryption: The sensor node s_i decrypts the query head OR_i using its private key X_{s_i}. Query head decryption reveals the next hop IP address $ip_{s_{i+1}}$, the next onion layer OR_{i+1}, and if s_i is a target node, head decryption also reveals the pair of symmetric encryption keys (e_a, e_b).

b) Task execution: If the sensor node s_i received symmetric encryption keys from query head decryption, then s_i is a target node and will perform the following steps. Otherwise, if s_i is a decoy node, it will skip the following steps to perform the step *query forwarding*.

The sensor node s_i decrypts B (the query body) using the first symmetric key e_a revealing: the data-carrying string w, the task t, and the padding p. The task t gets executed sourcing input from w and sensors. A task is executed at most for Δ_t milliseconds otherwise, task execution is interrupted. Task execution returns w', a binary string holding task execution results. Then s_i constructs B' the query body consisting of the data carrying string w', the task t, and the padding p all encrypted using the second symmetric encryption key e_b. Therefore, B' is constructed as follows: $B' = E_{e_b}(w', t, p)$. Since the content of B' differs from B only in the binary string, but the binary string w' is of the same size of L_w bytes as w, then query body size is maintained uniform.

c) Query forwarding: Query head is reassembled by applying the technique for onion size uniformity introduced in [9]. The query head size is maintained fixed at L_H bytes by adding λ a padding of $size(OR_i) - size(OR_{i+1})$ random bytes at the end of the onion layer OR_{i+1}. Therefore, the query head is now consisting of $OR_{i+1} + \lambda$. After Δ_q milliseconds ($\Delta_t < \Delta_q$) from receiving the query, the sensor node will randomly choose f a `float`, and will wait for other $f \cdot \Delta_q$ milliseconds before forwarding the query to the next hop. Adequate bounding values should be selected for the randomly chosen r, e.g. $0 \leq f \leq 4$. After waiting the required time, the query made of the head $OR_{i+1} + \lambda$ and the body B' (or B if node s_i is a decoy node) is forwarded to the next hop s_{i+1} at the IP address $ip_{s_{i+1}}$.

4) Result Retrieval: Each query sent to a sink node to accomplish the request R_{eq} will follow a path forming a circuit that ends at the sink node that issued the query. The sink node decrypts the query head consisting of the onion layer OR_{n+1} revealing id the query identifier. The query identifier and the query body B are then forwarded to the client that sent the query to the sink node. The client uses the query identifier to find the corresponding symmetric encryption key e_L from the recovery rules π, and the data-carrying string w is obtained from the query body decryption.

When the client gathers the feedback of all queries issued to accomplish R_{eq}, it starts the recovery process of the request result. Query results are merged following recovery rules π to obtain the end result of the request R_{eq}.

C. Arbitrary Computation

In this section, we discuss the capability of our proposed approach to perform multiple computations to accomplish an arbitrary computational job. Section IV-B described how the client application is used to construct multiple queries for retrieving an aggregated value from the WSN. In this process, all the queries issued for obtaining the desired aggregate enclose all the same task. However, the approach can be extended to enable the client application to construct several queries enclosing different tasks and using the binary string to maintain a context for the task. By properly arranging

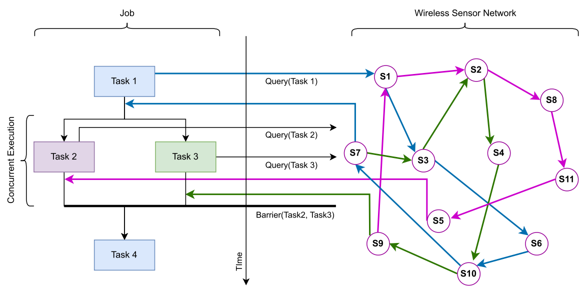


Fig. 2. Representation of a computational job divided into individual sub-tasks. Each sub-task independently executes a specific query within the WSN. The barrier synchronization manages task dependencies.

and merging the results from these queries, the client can accomplish the computational job.

To perform this, the client should first split a computational job into sub-tasks and identify those that can be executed independently and those dependent on the result of other tasks. Moreover, while splitting the computational job into sub-tasks, the client must consider the technique peculiarity of the sequential movement of the query from one node to another, which means that only the content of the query binary string and the local data of the node currently processing the query is available at each query processing step.

To manage the execution of these sub-tasks and their dependencies, the client can implement barrier synchronization. Barrier synchronization allows the client to wait for the completion of a set of queries before proceeding with the next set of queries. This ensures that any dependencies among the sub-tasks are properly managed and that the required results from previous queries are available before the subsequent queries are issued. The barrier synchronization of queries is shown in Fig. 2.

By using barrier synchronization at the client level, our approach can support the execution of complex computational jobs that involve multiple, interdependent sub-tasks. This extends the versatility of our technique, enabling the execution of more sophisticated computations across the WSN while still benefiting from privacy preservation.

D. Query Size Optimization and Query Authenticity

The technique described in this manuscript is based on queries consisting of several public-key encryption layers, and layers include encryption key material and path details. These queries are routed through a wireless multi-hop network and relayed several times. Therefore, it is important to minimize the query size to save network resources and improve the system's response time. Moreover, since queries are forwarded among sensor nodes, it is important to grant query authenticity allowing only authorized users to pose queries to sensor nodes and inhibit the injection of malicious queries. We define that a query is authentic if it was signed with an authorized private key. In the following, we describe how to optimize the query size and provide query authenticity using Elliptic Curve Cryptography (ECC) [40]. Several research [41], [42] suggest the adequacy of ECC cryptosystems for resource constrained devices, mainly due to the smaller key size compared to RSA.

To generate a digital signature we use the Edwards-curve Digital Signature Algorithm (EdDSA) [43]. The EdDSA signature of the message m consists of two values: R and s. The signature is computed using the signer's secret and public key, respectively an integer a and the point on curve $A = a \cdot G$. G is the base point on an elliptic curve defined over the finite field $\mathbb{F}_p$ of p elements with p prime. In EdDSA, R is obtained as $R = r \cdot G$, r a deterministic pseudorandom value, obtained from the hash of the last 32 bytes of a and the message to be signed. The value s is obtained from $s = r + HASH(R||A||message) \cdot a$. (We use || to denote concatenation.) The signature verifier generates two values $v_1 = s \cdot G$, and $v_2 = R + A \cdot HASH(R||A||message)$. If $v_1 == v_2$, the signature checks.

To ensure authenticity, the user constructing the query should include a signature for each target node in the query path. For a target node with address IP_s, the message to be signed consists of $m = t||IP_s$. To prevent reply attacks and ensure that the query binary representation changes with each new query, the pseudorandom generation of the value r should be derived by hashing a, m, and a session ID. The Session ID should be appropriately generated to avoid the widely known security failure that lead to the compromise of the Sony PlayStation3.

The first trick to optimize the query size is that R can be reused for multiple ECC operations, which was shown to be secure in [44], and was also applied by *De Cristofaro et al.* in [25]. We emphasize that we use R for one signature and multiple ECC Diffie Hellman [45] operations. Specifically, we adopt the Diffie Hellman key encapsulation mechanism of the Elliptic Curve Integrated Encryption Scheme (ECIES) [46]. The ECIES allows deriving a secret ς from a random integer r and a public key. The same secret can be derived by the corresponding private key and the random integer. Therefore, by sharing the random integer, two parties derive the same secret. The random integer can be securely shared via its multiplication with the base point $R = r \cdot G$. Therefore, the value R used in the EdDSA signature can be reused in the ECIES to derive a shared secret. The derived shared secret ς is used in the ECIES as a key for applying a symmetric cipher over data. Moreover, since it is enough to verify the message authenticity at target nodes, the value R can be reused for the ECIES operation at decoy nodes.

The second optimization trick relates to the delivery of symmetric encryption keys to target nodes. One symmetric encryption key can be derived by hashing the shared secret ς obtained from ECIES decryption. A similar approach was applied in the original onion routing technique [47]. The second symmetric encryption key should be delivered in query head layers since keys are chained, as shown in Fig. 1. The delivered key should correspond to the value obtained by hashing the shared secret ς obtained by the next target node.

Therefore, the query head size can be summarised with equation 3, n the query path length, and L_- the size of the subscript argument.

$$L_R + (\frac{n}{2} \cdot (L_{IP}, L_R, L_s, L_{sym})) + (\frac{n}{2} \cdot (L_{IP})) \quad (3)$$

V. Privacy-Preservation Analysis

This section examines the communication protocol to identify vulnerabilities concerning privacy preservation. The literature on WSNs generally categorizes privacy concerns into two main classifications: external and internal privacy [7], [19], [25], [35], [36]. This paper aligns with this approach and utilizes the same terminology. External privacy is threatened by actors outside of the network listening to the wireless communication, while internal privacy is threatened by trusted participating sensor nodes of the WSN.

Throughout the analysis, we assume that the network implements layer 2 security, the guarantee of in-order data delivery using TCP, sink nodes cannot be compromised, and users do not collaborate with the attacker. For simplicity, we assume that messages are not fragmented.

A. External Privacy

To examine external privacy, we visualize a foreign actor that is monitoring the network traffic by eavesdropping on wireless communications. We dub this actor the external adversary. Eavesdropping is the intercepting and reading of messages by unintended receivers. Since the majority of wireless communications use the radio frequency spectrum to broadcast signals over the air, transmitted signals can be easily intercepted using adequate receiving equipment [48]. Event though the network implements layer 2 security, we assume that the external adversary is able to differentiate a transmission transferring query information from ordinary network management traffic by observing the transmission length.

Since layer 2 security protects transmissions using encryption and query size is maintained uniform throughout the query path, the only disclosed detail of an intercepted query is the effective transferring of the query from one node to another. The node receiver of the transmission is then processing the query (as explained in Section IV-B.3) or re-transmitting it to another node (routing in multi-hop networks). However, processing the query introduces a delay, missing if it is just re-transmitted. Therefore nodes processing the query can be identified; nonetheless, the external adversary cannot differentiate decoy nodes from target nodes since the query sojourn time at both kinds of nodes depends upon a randomly chosen `float`.

We now consider an external adversary whose monitoring range covers the whole WSN; therefore, it can intercept the whole wireless traffic generated by the WSN. Hence, the adversary can track a query sourcing from the sink node and moving through the network by monitoring its transmissions. However, normally, the WSN traffic is not populated by only one query, and the randomized nature of the query path will make various queries mix at nodes on their route. Even though the adversary violates security measures of the physical layer, security at the data link layer is changing data by encryption before each transmission. Furthermore, query size is maintained uniform; therefore, it is hard for an external adversary to track how the query transit through the network since the adversary cannot distinguish between queries.

B. Internal Privacy

An attacker that owns a subset of WSN nodes is commonly referred to as an internal adversary. Nodes owned by an internal adversary are participating trusted nodes of the WSN owning cryptographic keys to decrypt messages addressed to them. Section IV-D shows how to render void query injection attacks using digital signatures. Therefore, traffic analysis is the only attack that could compromise privacy concerning assumptions in Section V and the assumption of secure cryptographic primitives.

The internal adversary can take advantage of owned nodes to analyze traffic they receive and disclose information from un-compromised nodes of the WSN. Although we assumed that the network implements layer 2 security, only individual links are secured by such a solution, and nodes intermediate to routing paths can overhear messages passing through them. In the following, we will analyze under which circumstances an internal adversary is able to gain insights over other nodes in the network and when the data privacy of a sensor node is disclosed.

To simplify the analysis, we introduce the following notation expressing the operation of sensor nodes in relation to a query: D are decoy nodes, T are target node, A are nodes owned by the adversary intermediate to the routing path of the query, AD are decoy nodes owned by the adversary, and AT are target nodes owned by the adversary. In the following, we explain the implications of a transiting query over sensor nodes.

T and D are nodes that will process the query (as explained in Section IV-B.3). On these nodes, the adversary is trying to gain information. Nodes T are target nodes for the query, and after processing at T nodes, the redirected query is entirely changed by encryption. On the other hand, after query processing at D nodes, the redirected query has query head changed by encryption, but the query body remains unchanged.

A are nodes owned by the adversary that receive the query due to routing needs in wireless multi-hop networks. Therefore, A nodes receiving the query can observe the encrypted query head and query body. Moreover, the IP header reveals the address of the previous and next node processing the query.

AD nodes are owned by the adversary and are processing the query. However, AD nodes cannot access the query body. Therefore, AD nodes disclose only the IP address of the previous and the next node processing the query.

AT nodes are owned by the adversary and are processing the query as target nodes. Therefore, nodes AT can decipher the query head layer addressed to them, revealing the next-hop IP address and a pair of symmetric encryption keys. Thus, they can decipher the query body and learn the task and the binary string that carries the partial result. The internal adversary can examine the task and disclose the function to be computed; hence the adversary can identify the value carried in the binary string. We say identify since the adversary can recognize that the value is a sum, an extreme, etc. Although the internal

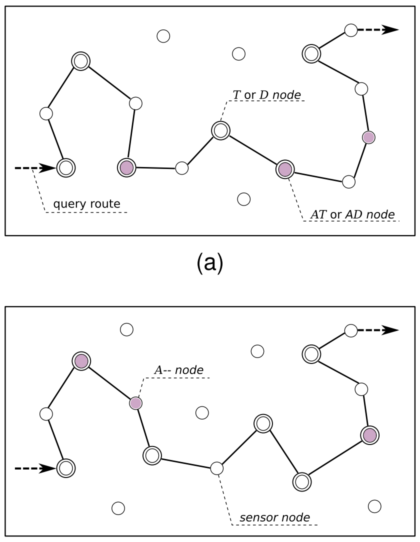


Fig. 3. (a) Route of the query that leads through one $T|D$ node confined between two nodes owned by the adversary. (b) Route of the query that leads through two or more $T|D$ nodes confined between two nodes owned by the adversary.

adversary can identify the value carried in the binary string, by owning a single node in the query path, the internal adversary cannot draw conclusions on the value or extract sensor node readings of other target nodes in the query path since it does not know which are target nodes and how many target nodes contributed to the partial result.

We now consider a query that transits through multiple nodes owned by the internal adversary. Hence the adversary can track sender and receiver IP enclosed in the IP header to partially reconstruct the query path and observe how the query body changes to gain information over other nodes in the query path. Note, we refer to the query path as the sequence of nodes on which the query is processed; on the other hand, we refer to the query route as the sequence of nodes that the query transits, including nodes in the query path and nodes that are forwarding the query due to routing needs in wireless multi-hop networks. To conduct this investigation we examine two cases where the query route leads through two nodes owned by the adversary: 1) two or more $T|D$ nodes confined between two nodes owned by the adversary; (_|_ used as exclusive OR) 2) one $T|D$ node confined between two nodes owned by the adversary.

Fig. 3 shows an example of the two cases. We consider these two cases since if the query route leads through more than two nodes owned by the adversary, the instance can be generalized to multiple of the aforementioned cases. Moreover, in Section V-B.3, we consider concerns of query exit and entry point.

1) Route of the Query That Leads Through Two or More $T|D$odes Confined Between Two Nodes Owned by the Adversary: We first look at the instance that nodes owned by the adversary are not both AT. Since the query is processing at two or more consecutive un-compromised nodes, IP information accompanying the query cannot be used to determine if both nodes owned by the adversary received the same query. Therefore, the adversary must rely solely on the query body to disclose meaningful information. Indeed, if nodes in the query path arranged between the two nodes owned by the adversary are all D, the inner encryption layer of the query body will remain unchanged. Therefore, the adversary can identify that both owned nodes received the same query and that all nodes processing the query between the owned nodes are D nodes. However, the internal adversary is not able to recognize the number of nodes in the query path between the two owned nodes since query sojourn time depends upon a randomly chosen `float`. On the other hand, if any node arranged between the two nodes owned by the adversary is a T node, then the query body is also changed by encryption, and the internal adversary cannot determine if both owned nodes are executing the same query.

We now consider that both nodes owned by the adversary are AT nodes. Then the adversary can decipher the query body at both owned nodes, and it should be possible to compare tasks and binary strings to recognize if both nodes received the same query. However, even though the adversary can recognize the value change of the binary string, he cannot identify which nodes processing the query are T nodes nor how many T nodes contributed to the value change.

2) Route of the Query That Leads Through One $T|D$ode Confined Between Two Nodes Owned by the Adversary: This disposition of nodes can be identified by the internal adversary as a transitive dependency of sender and receiver IP addresses from the IP packet header (e.g. $IP_{owned1} \rightarrow IP_i, IP_i \rightarrow IP_{owned2}$). Therefore, we assume that given the transitive relation of IP addresses, the adversary deduces that the two owned nodes are processing the same query, even if the query was changed by encryption and the randomly chosen query sojourn time does not ensure that it is the same query. Another query might be routed through $IP_i \rightarrow IP_{owned2}$ tricking the internal adversary of detecting the relation $IP_{owned1} \rightarrow IP_i, IP_i \rightarrow IP_{owned2}$.

Regardless of the aforementioned possibility of mixing queries, if the adversary identifies this particular disposition of nodes, he can examine the query received at the two owned nodes to gain insights over the node between them. We distinguish the following three cases where the adversary discloses different insights over the un-compromised node between the two owned nodes:

1) *Both nodes owned by the adversary are $A|AD$.* The adversary can compare the encrypted query body at both owned nodes to disclose if the un-compromised node is a T or D node for the received query.
2) *One node owned by the adversary is AT.* The adversary can recognize if the un-compromised node is a T or D node for the received query. Furthermore, if the un-compromised node is a T node, the adversary can observe the task to disclose the sensor technology equipped on the node.
3) *Both nodes owned by the adversary are AT.* In this case, the adversary can gain insights summarized in

the previous points. Additionally, if the un-compromised node is a T node, the adversary can compare the binary string state at the two owned nodes to extract data added by the un-compromised node.

3) Query Entry & Exit Point: We refer to the query entry point as the first sensor node in the query path and the query exit point as the last node in the query path since they both communicate with the sink node making them recognizable as such.

By our solution design, the first node in the query path can be a T node. Thus if the first and the second node in the query path are respectively T and AT, the adversary can disclose data privacy of the first node in the query path, however only if the adversary can identify that he owns the second node in the query path. Since the position of nodes in the query path can be identified only by tracking the query route from the source, this vulnerability can be exploited only if the adversary owns an A node in the query route from the sink node to the first node in the query path. Moreover, it is possible to avoid this vulnerability by setting an initial value to the data carrying string.

The query exit point or the last node in the query path is always a D node. We included this design choice since the last node in the query path can identify its position in the query from the next-hop IP address, which is the address of the sink node. If AT is the last node in the query path able to decipher the query body, the adversary could potentially infer over the number of T nodes that contributed to the partial result, since a query generally includes ϕ T nodes in the query path. However, the adversary can identify that he owns the last sensor node deciphering the query body only by tracking the query path to the sink node.

C. Data Leak Probability

The examination in Section V-B shows that under certain conditions, there is some probability that private sensor data leaks to the internal adversary. We use the term leak since the disclosure of private data depends on the randomized selection of nodes in the query path. The data leak occurs when the query path leads through three consecutive target nodes and the two outer nodes are owned by the adversary, then the adversary can compare the binary string state at the two owned nodes to disclose the data added by the inner node.

To determine the probability of a data leak during normal operation, we make the following assumptions: the selection of nodes forming the query path occurs using a uniform random generator, nodes in the query path do not repeat, and target nodes are randomly selected. Therefore in a WSN of s sensor nodes, the data leak probability depends on the query path length n, the number of target nodes in the query path ϕ, and on the number of sensor nodes owned by the adversary a. We count all the instances where selecting three target nodes, two of them are owned by the adversary; there are $n-2$ such occurrences where the adversary does not own the inner node. The data leak probability is given in equation 5, as in (4), shown at the bottom of the page.

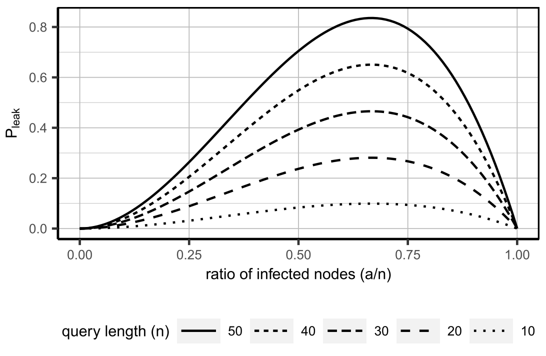


Fig. 4. The data leak probability for different query path lengths. ϕ is set to the default $\phi = n/2$.

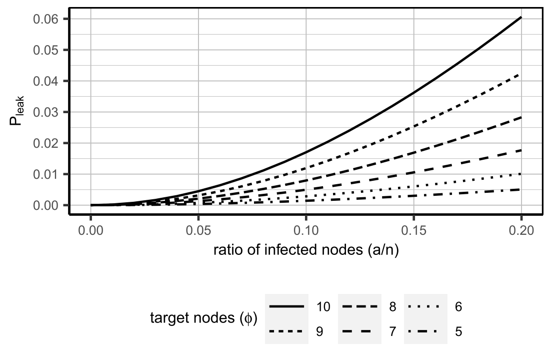


Fig. 5. The data leak probability at $n = 20$ and varying ϕ the number of target nodes in the query.

We emphasize that equation 5 has also been validated via simulation. We simulated the random query selection applying the same assumptions as for equation 5, and we counted the number of data leak occurrences. The simulation was repeated several times, and obtained values matched our analytical formula.

We plot the data leak probability distribution for different query path lengths in Fig. 4, and a zoomed version in Fig. 5 shows the decrease of the data leak probability when reducing the number of target nodes in the query path.

D. Active Attacks

The previous sections primarily focused on analyzing privacy-related attacks. However, WSNs also face a variety of additional threats that target data integrity and network availability. These types of attacks are known as active attacks, wherein adversaries actively interfere with the normal functioning of the WSN by manipulating data or disrupting

$$P_{leak} = \frac{a \cdot (n-\phi) \cdot (s-a) \cdot (n-\phi-1) \cdot (a-1) \cdot (n-\phi-2)(s-3)!}{s! \cdot n \cdot (n-1)} \tag{4}$$

network traffic. In this section, we will specifically discuss active attacks that could exploit our proposed technique to compromise the WSN while excluding attacks targeting other network layers, as they are well documented in the literature [4].

While the proposed technique offers certain security benefits, it does not fully protect against data integrity threats arising from physical environment manipulation or compromised nodes controlled by the internal adversary. Attackers may compromise data validity by interfering with sensor nodes' surroundings, such as artificially raising the temperature with a lighter. They can also undermine data integrity by altering query data using owned nodes. Specifically, target nodes processing the query have access to the query body. Although they cannot modify the task, as it is secured with query authenticity, as explained in Section IV-D, they can alter the binary string containing the aggregated data. Both scenarios can result in inaccurate data aggregation and potentially incorrect decision-making for the end user. However, such attacks on data integrity cannot specifically target individual nodes, as the proposed technique conceals the nodes processing the query. Consequently, these attacks would be inconsistent and more easily detectable.

In Section IV-D, we showed how to ensure query authenticity to prevent malicious query injection and replay attacks targeting data confidentiality. However, the technique we described only verifies query authenticity at the target nodes. Consequently, an internal adversary who knows the public keys associated with node addresses could forge a query that is only onion routed without providing access to the query body. The adversary's primary goal in doing so is to generate unnecessary traffic, leading to network congestion and disrupting the WSN's availability. An internal adversary could also execute an attack by dropping queries that are onion routed at compromised nodes. This attack is particularly effective in our proposed solution, as queries follow a fixed path and cannot bypass nodes that will onion-route them. As a result, the query would fail to reach its final destination, disrupting the data retrieval process and undermining the network's functionality.

While the described attacks can potentially impact the network's functioning, they share similarities with threats targeting other network layers. As such, these attacks can be mitigated by implementing intrusion detection systems [49].

E. Discussion of Privacy-Preservation Analyses

In Section V-A, we provide evidence that the solution withstands eavesdropping attacks since queries follow all the same circuit-like patterns while paths are randomized, transmitted queries are indistinguishable by encryption and uniform query size, and queries are mixing while transiting the WSN. Furthermore, external actors observing the wireless communication cannot disclose the nodes' target of the query since query forwarding time is decoupled from query execution.

In Section V-B we showed that the proposed protocol preserves query privacy by constraining information of the query path. Nodes receiving the query that can decipher the query body can learn about the task. However, without knowledge of the identities of other nodes involved in the joint computation, the adversary cannot infer information other than those revealed by the task.

Further analysis has shown that an attacker owning a portion of network nodes could possibly disclose insights about non-compromised sensor nodes and even threaten data privacy in the WSN. However, assuming secure cryptographic primitives, the adversary can increase the odds of privacy disclosures only by increasing the number of owned nodes in the network. Taking control over a sensor node is generally hard since it requires physical access to the node [50]. Moreover, due to the randomized selection of nodes forming the query path, the adversary cannot increase the data leak probability for a specific node in the network. In Section V-C, we expressed the data leak probability, and it was shown that it depends on the query path length, the number of target nodes in the query, and the number of nodes owned by the adversary. Therefore, the data leak probability can be adjusted based on the application's security requirements. Additionally, by applying the technique for query size optimization described in Section IV-D, adding a decoy node to the query path increases the query size only by the size of storing the node address; therefore, higher privacy requirements will result in a small decrease in efficiency.

In Section IV-D we explained how to use digital signatures to grant query authenticity. Here we want to emphasize that the described technique does not invalidate the security analyses in Section V. The signatures in the query head are all diverse since each signature is generated by coupling the task and a target node address. Therefore, digital signatures do not uniquely identify the query. Moreover, users could share signing keys if digital signatures are used only to verify query authenticity.

The privacy preservation analyses in Section V concern an attacker attacking the network. However, as discussed by *Carbunar et al.* [29] and *De Cristofaro et al.* [25] there are some circumstances where the WSN user wants to protect its privacy against the network owner. The solution of *De Cristofaro et al.* achieves a very high level of privacy; nonetheless, the technique allows only retrieving individual sensor node readings without the possibility of in-network processing. In the case of our technique, we incur the same issues as discussed by *Carbunar et al.*, requiring the user to generate multiple bogus queries to obfuscate the one of interest. The problem is extensively studied by *Carbunar et al.* in [29], taking into account spatial and temporal privacy.

We emphasize that protecting the user's privacy against the network owner is out of scope for this manuscript since here we focused on describing a technique for the joint computation of arbitrary functions on WSN nodes; moreover, in our solution, this problem could be addressed using code obfuscation techniques [51] to obfuscate the task and the carried data.

VI. Simulation Results

This section presents results of the privacy preserving communication protocol simulated using the NS3 [20], [52]. We examine the protocol scalability in a simulated WSN by

TABLE I
SUMMARY OF SIMULATION PARAMETERS AND CORRESPONDING VALUES

Param.	Value	Description
s	$50, 100, 200, 300, 400$	network size - number of sensor nodes
q	$5, 10, 15, , 20..., 50$	query path length
$topology$	GT, RDT	network topology grid / random disc
bs	$128, 256, 512, ..., 8192$	query body size (bytes)
$data\ rate$	$12\ Mbps$	transmission data rate at IEEE 802.11n

observing how querying is affected by the following independent variables: query path length (number of nodes on which the query will be processed), network size (number of nodes in the WSN), network topology, and query body size.

In order to quantify the response of independent variable adjustments, we meter the Round-Trip-Time (RTT) of queries. We define the RTT of a query, as the elapsed time between the issuing of the query from the sink node and the return of the issued query to the issuer node. Therefore, the RTT includes the sojourn time of the query at sensor nodes in the query path. However, Section IV-B.3 describes that the query sojourn time depends upon the application-specific parameter Δ_q and a randomly chosen `float` f. Therefore, we decided not measure the query randomized sojourn time since it introduces delays that are not dependent on the network. Moreover, by the law of large numbers [53], if choosing the value of f from a uniform random distribution bounded by the interval [0, F], the average of the results obtained from a large number of trials will converge to the expected value of f. Therefore, the average cumulative sojourn time for a specific query path length n at an adequate Δ_q and F can be estimated using the following equation:

$$sojourn\_time = n \cdot \Delta_q \cdot \frac{F}{2} \tag{5}$$

A. Experimental Setup

The simulated WSN consists of one sink node and s sensor nodes. We consider two network topologies: the *grid topology* (GT) and the *random disc topology* (RDT). In the former, sensor nodes are deployed according to a grid structure; each sensor node is equidistant from the closest sensor nodes in cardinal directions. We set the distance between sensor nodes to $a = 60$ meters so that a sensor node is in the communication range of at most eight sensor nodes. In the latter, sensor nodes are randomly deployed on a disc-shaped plane of radius r_p. The radius r_p is obtained from $r_p = \sqrt{A/\pi}$, A being the sum of circular area's covered by s sensor nodes at radius $r_s = 35$ meters. Therefore, the average sensor node density of the network is maintained fixed at diverse s. Since in RDT sensor nodes are casually deployed on the target area, some sensor nodes could form independent network segments not connected to the network segment of which the sink node is a member. Therefore some sensor nodes might never be queried. In both network topologies, the sink node is deployed in the center of the WSN. We chose values of parameters $a = 60m$ and $r_s = 35m$, since networks of different topologies will have a near equal average node density: GT: $\frac{1}{60^2} = 0,277 * 10^{-3} nodes/m^2$, and RDT: $\frac{1}{35^2\pi} = 0,26*10^{-3} nodes/m^2$. Key simulation parameters are given in Table I.

The simulation implements the IEEE 802.11n standard for local wireless networks, operating in the 2.4 GHz band at the data rate of 12Mbps. The maximum segment size is set to the NS3 default 536 bytes. Each node in the WSN has installed the IP stack, and messages are transmitted over the TCP protocol.

However, the TCP was designed to function over low-error wired networks where the packet loss is usually the outcome of a network congestion [54]. Several studies are suggesting that the use of TCP in wireless multi-hop networks results in low throughput since packet loss due to transmission error and route discovery is handled using congestion avoidance and control [54], [55], [56]. Route discovery is performed by the routing protocol when searching for a route from sender to receiver. It is possible that discovering a route may take more time than the TCP retransmission timeout (RTO) [55]. The RTO is an internal timer of the TCP used to determine when a segment needs to be retransmitted. If the RTO elapses before receiving the acknowledgment of segment delivery, the segment is retransmitted, the RTO is increased using exponential backoff, and the TCP is adjusted for congestion. The minimum RTO value in the simulation is set to the default, 1 second. To avoid complications due to route discovery, we decided to use the Optimized Link State Routing Protocol (OLSR) [57], a proactive routing protocol so that routes are immediately available when needed. In proactive routing protocols, routes between each pair of nodes are determined at the network start-up and maintained with periodic updates.

Queries are constructed from the sink node by randomly selecting n nodes to include in the query path, n being the query path length. Query construction occurs using the technique for query size optimization ensuring query authenticity as explained in Section IV-D by using the elliptic curve Curve25519 [58] with key length 256-bit and the symmetric cipher Advanced Encryption Standard (AES) [59] at key length 128-bit. Therefore, recalling Section IV-D, R, s, and sym sizes are respectively 32 bytes, 32 bytes, and 16 bytes. To further reduce the query size, we use network addresses of 16-bits. The query body should be a multiple of the AES block size, and its size is adapted based on the experiment requirements.

Queries are issued from the sink node sequentially; after a query returns back to the sink node, the following query is issued. Nodes in the query path performing query processing maintain the uniform query size by adding padding. If the query does not reach the next-hop node in 100-seconds, the query is aborted, and a new query of equal parameters is issued from the sink node.

B. Experiment 1 – Remote Procedure Call

In this experiment, we consider the minimal size of the query body since this will give a better overview of how the network properties affect the RTT and since the query body size mainly depends on the computation instructions conveyed using the protocol.

The minimal query body size applies to the Remote Procedure Call (RPC) settings, where nodes of the WSN have encoded a set of functions, and the task t carried in the query

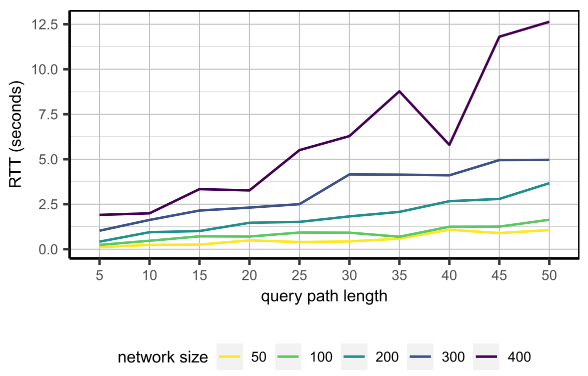


Fig. 6. The chart shows the average query RTT for the GT WSN at varying of the query path length and network size. Data from the experiment 1 in Section VI-B.

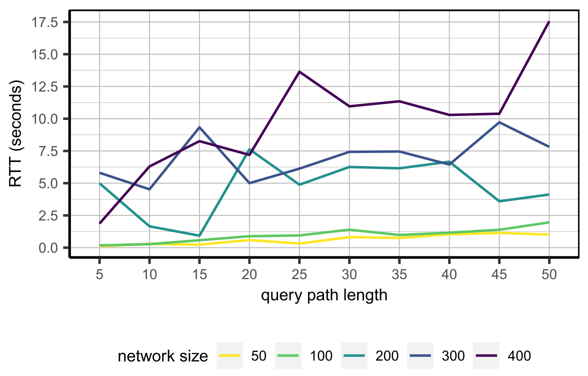


Fig. 7. The chart shows the average query RTT for the RDT WSN at varying of the query path length and network size. Data from the experiment 1 in Section VI-B.

body specifies which function to compute over sensor readings and w the data carrying string. We set the task t and the binary string w to 16 bytes.

A set of simulations was run for both GT and RDT at $s = \{50, 100, 200, 300, 400\}$. Each run executing 30 queries for each value of $n = \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$, the query path length. The obtained data is presented in Fig. 6, and Fig. 7. In the RDT simulations, the network segment, including the sink node, was connected to approximately 90% of all nodes. We observed some aborted queries in the RDT at the network size of 300 and 400 nodes. However, the events were all observed right after the simulation started; therefore, probably due to the OLSR routing requiring more time to converge. In the GT no aborted queries were detected.

C. Experiment 2 – Arbitrary Computation

In this experiment, we examine how the RTT is affected by the size of the query body. The large size of the query body is distinctive to an implementation of the communication protocol that delivers the function to be computed on sensor nodes within the query.

A set of simulations was run for the GT WSN at $s = \{50, 100, 200, 300, 400\}$. Each run executing 30 queries for the

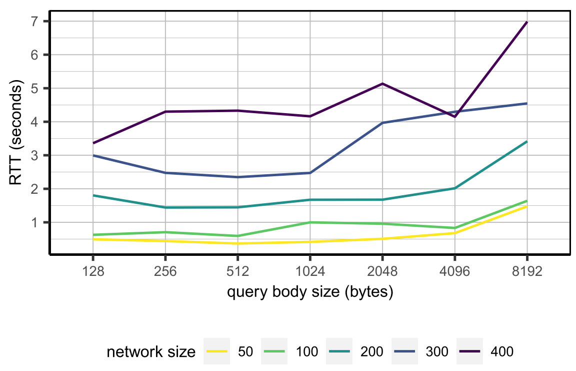


Fig. 8. The chart shows the average query RTT for the GT WSN with query path length set to $n = 20$, varying the query body size and network size. Data from the experiment 2 in Section VI-C.

query body size $bs = \{128, 256, 512, 1024, 2048, 4096, 8192\}$ bytes, at the query path length $n = 20$. The obtained data is presented in Fig. 8.

D. Discussion of Simulation Results

From Fig. 6, it is apparent that the query RTT is highly affected by the network size and the query path length. Furthermore, at larger network sizes the query path length has a higher effect on the query RTT. The high query RTT at large network sizes is due to the randomized selection of the query path that cause queries to cross nodes that may be very distant. Therefore, in a large network it should be practical to constrain the random selection of nodes to a network region that includes the nodes of interest but it is small enough to not incur in high query RTT.

The difference in query RTT between the GT and RDT can be observed in Fig. 6 and Fig. 7. In particular, it is possible to notice that the average query RTT at RDT is higher than at GT. Furthermore, the RDT data has a higher variance. Besides high variance of collected data, we also report a non-normal distribution of RTT measurements. A possible explanation for the non-normal distribution of RTT observation could be intrinsic to the TCP protocol and the RTO timer.

Experiment 2 considers the query RTT of queries with large query body sizes. Interestingly, in Fig. 8 can be seen that at selected parameters, the query body size does affect the average RTT of queries; however the increase is noticeable only for larger query body sizes (larger than 1024 bytes).

VII. Limitations

Although our proposed privacy-preserving protocol presents several advantages for WSNs, it also exhibits certain limitations that we discuss in this section.

A limitation related to our proposed technique that warrants further discussion is the use of layered messages constructed using public key encryption. Public key cryptography is well known for being more computationally intensive than symmetric key cryptography. However, public key encryption provides benefits that symmetric key encryption cannot, such as non-repudiation and no need for pre-shared keys. As a result, many studies apply public key encryption also in WSN settings [25], [60], [61]. The survey [62] provides an overview of public key cryptographic primitives adequate for WSNs, testing them on WSN devices. The suitability of 32-bit ARM-based microcontrollers for applications with significant cryptographic requirements is emphasized in [63] and [64], while [65] demonstrates the implementation of end-to-end integrity protection using Elliptic Curve Digital Signature Algorithm (ECDSA), further validating the efficacy of the 32-bit ARM architecture.

With regards to our protocol, it should be noted that even though messages consist of several public key encryption layers, the message creation process is performed outside the WSN on client applications. Consequently, the high overhead of applying multiple public key encryption layers does not load the WSN nodes. Upon receiving a query, WSN nodes perform either one or, at most, two public key operations. A single operation occurs if the query is only onion-routed through the node, while two operations take place when the node accesses the query body content and checks its authenticity. Therefore, our approach does not require many more public key operations than other techniques proposed for WSNs. Moreover, the computational load of decrypting query layers is uniformly distributed throughout the WSN.

Another limitation associated with the use of onion routing is the inability to change query routes after their creation. The underlying routing protocol can adapt to node failures by dynamically adjusting the routing path. For example, if a query is created to be processed at nodes A and C and must travel from A to C through a third node B, the route can be dynamically adjusted if node B fails, routing the query through an alternative path linking A and C. However, this cannot resolve issues arising from the fixed nature of onion routing. In cases where node C fails, the query will never reach its destination because path information can only be obtained by sequentially removing the encryption layers. Node C is the only node capable of removing the encryption layer addressed to it. If node C fails, no other node can remove that layer. In such situations, the network should adapt by flagging the failed sensor node or removing it from the registry held by sink nodes, making it ineligible for queries.

As observed in Section VI, our approach introduces a significant latency when processing a query, which impacts the real-time performance of the network. Furthermore, longer query path lengths result in longer query RTT. A potential solution to mitigate this issue is to send multiple smaller queries that can collectively obtain the same result, achieving reduced overall latency through parallel execution rather than sending a single query through a longer path. This approach allows for improved query response times while maintaining the desired privacy-preserving features.

It is important to note that the discussed limitations do not undermine the overall effectiveness of our privacy-preserving protocol in the context of WSNs. However, they provide insight into the trade-offs and challenges that need to be considered when implementing and deploying our proposed solution.

VIII. Conclusion and Further Work

This paper proposes a technique that enables WSN nodes to jointly compute a function in a privacy-preserving manner. We compare our proposal to the related work in the field, showing that, to the best of our knowledge, this is the first scheme that allows in-network joint computation of arbitrary functions in WSNs without aggregator nodes and without disclosing the nodes participating in the computation and their private inputs. We show that the scheme mitigates traffic analysis attacks; thereby making it adequate for indoor monitoring. As future work; we plan to implement and test our proposal in several buildings to collect and in-situ analyze air quality data. Additional future work includes investigating further adaptation of the proposed technique to IoT environments, the technique will be considered for training and evaluation of machine learning models and the federated learning paradigm could offer a valuable avenue for exploration. Moreover, the application of code obfuscation techniques or homomorphic encryption to the tasks conveyed to sensor nodes, would considerably enhance privacy preservation, warranting further exploration into the potential trade-offs of such an approach.

References

[1] E. F. Nakamura, A. A. F. Loureiro, and A. C. Frery, "Information fusion for wireless sensor networks: Methods, models, and classifications," *ACM Comput. Surv.*, vol. 39, no. 3, p. 9, Sep. 2007.

[2] F. Jabeen and S. Nawaz, "In-network wireless sensor network query processors: State of the art, challenges and future directions," *Inf. Fusion*, vol. 25, pp. 1–15, Sep. 2015.

[3] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 56–69, Jul. 2006.

[4] I. Tomic and J. A. McCann, "A survey of potential security issues in existing wireless sensor network protocols," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1910–1923, Dec. 2017.

[5] Y.-H. Lin, S.-Y. Chang, and H.-M. Sun, "CDAMA: Concealed data aggregation scheme for multiple applications in wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 7, pp. 1471–1483, Jul. 2013.

[6] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "PDA: Privacy-preserving data aggregation in wireless sensor networks," in *Proc. IEEE INFOCOM 26th Int. Conf. Comput. Commun.*, 2007, pp. 2045–2053.

[7] N. Li, N. Zhang, S. K. Das, and B. Thuraisingham, "Privacy preservation in wireless sensor networks: A state-of-the-art survey," *Ad Hoc Netw.*, vol. 7, no. 8, pp. 1501–1514, Nov. 2009.

[8] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Cryptography from anonymity," in *Proc. 47th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, 2006, pp. 239–248.

[9] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, "Anonymous connections and onion routing," in *Proc. IEEE Symp. Secur. Privacy*, May 1997, pp. 44–54.

[10] M. Mrissa et al., "Extending BIM for air quality monitoring," in *Construction Materials for a Sustainable Future*, vol. 1. Ljubljana, Slovenia: Slovenian National Building and Civil Engineering Institute, 2020, pp. 244–250.

[11] D. Clements-Croome, "Sustainable intelligent buildings for people: A review," *Intell. Buildings Int.*, vol. 3, no. 2, pp. 67–86, 2011.

[12] Y. Kang, L. Aye, T. D. Ngo, and J. Zhou, "Performance evaluation of low-cost air quality sensors: A review," *Sci. Total Environ.*, vol. 818, Apr. 2022, Art. no. 151769. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0048969721068455

[13] J. Xu, G. Yang, Z. Chen, and Q. Wang, "A survey on the privacy-preserving data aggregation in wireless sensor networks," *China Commun.*, vol. 12, no. 5, pp. 162–180, May 2015.

[14] J. Jiang, G. Han, H. Wang, and M. Guizani, "A survey on location privacy protection in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 125, pp. 93–114, Jan. 2019.
[15] A. Maddumabandara, H. Leung, and M. Liu, "Experimental evaluation of indoor localization using wireless sensor networks," *IEEE Sensors J.*, vol. 15, no. 9, pp. 5228–5237, Sep. 2015.
[16] B. Andò, S. Baglio, and C. O. Lombardo, "RESIMA: An assistive paradigm to support weak people in indoor environments," *IEEE Trans. Instrum. Meas.*, vol. 63, no. 11, pp. 2522–2528, Nov. 2014.
[17] P. Zhou, G. Huang, L. Zhang, and K.-F. Tsang, "Wireless sensor network based monitoring system for a large-scale indoor space: Data process and supply air allocation optimization," *Energy Buildings*, vol. 103, pp. 365–374, Sep. 2015.
[18] J. Molka-Danielsen, P. Engelseth, and H. Wang, "Large scale integration of wireless sensor network technologies for air quality monitoring at a logistics shipping base," *J. Ind. Inf. Integr.*, vol. 10, pp. 20–28, Jun. 2018.
[19] J.-F. Raymond, "Traffic analysis: Protocols, attacks, design issues, and open problems," in *Designing Privacy Enhancing Technologies*. Cham, Switzerland: Springer, 2001, pp. 10–29.
[20] Nsnam. (Oct. 1, 2021). *Ns-3, a Discrete-Event Network Simulator for Internet Systems–Version 3.32*. [Online]. Available: https://www.nsnam.org/
[21] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management," Version v0.34, Aug. 2010. [Online]. Available: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf
[22] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, Feb. 1981.
[23] C. A. Sunshine, "Source routing in computer networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 7, no. 1, pp. 29–33, Jan. 1977.
[24] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Naval Res. Lab, Washington, DC, USA, Tech. Rep. 0704-0188, 2004.
[25] E. D. Cristofaro, X. Ding, and G. Tsudik, "Privacy-preserving querying in sensor networks," in *Proc. 18th Int. Conf. Comput. Commun. Netw.*, Aug. 2009, pp. 1–6.
[26] X. Yu, F. Li, T. Li, N. Wu, H. Wang, and H. Zhou, "Trust-based secure directed diffusion routing protocol in WSN," *J. Ambient Intell. Humanized Comput.*, vol. 13, pp. 1405–1417, Nov. 2022.
[27] Y. Protskaya and L. Veltri, "Broker bridging mechanism for providing anonymity in MQTT," in *Proc. 10th Int. Conf. Netw. Future (NoF)*, Oct. 2019, pp. 110–113.
[28] Y. Alshboul, A. A. R. Bsoul, M. AL Zamil, and S. Samarah, "Cybersecurity of smart home systems: Sensor identity protection," *J. Netw. Syst. Manage.*, vol. 29, no. 3, pp. 1–27, Jul. 2021.
[29] B. Carbunar, Y. Yu, W. Shi, M. Pearce, and V. Vasudevan, "Query privacy in wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 6, no. 2, pp. 1–34, Feb. 2010.
[30] K. Xie et al., "An efficient privacy-preserving compressive data gathering scheme in WSNs," *Inf. Sci.*, vol. 390, pp. 82–94, Jun. 2017.
[31] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.
[32] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 1999, pp. 223–238.
[33] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. 6th Annu. Int. Conf. Mobile Comput. Netw.*, Aug. 2000, pp. 56–67.
[34] M. Conti, L. Zhang, S. Roy, R. Di Pietro, S. Jajodia, and L. V. Mancini, "Privacy-preserving robust data aggregation in wireless sensor networks," *Secur. Commun. Netw.*, vol. 2, no. 2, pp. 195–213, Mar. 2009.
[35] L. Zhang, H. Zhang, M. Conti, R. Di Pietro, S. Jajodia, and L. V. Mancini, "Preserving privacy against external and internal threats in WSN data aggregation," *Telecommun. Syst.*, vol. 52, no. 4, pp. 2163–2176, Apr. 2013.
[36] R. Bista and J.-W. Chang, "Privacy-preserving data aggregation protocols for wireless sensor networks: A survey," *Sensors*, vol. 10, no. 5, pp. 4577–4601, May 2010.
[37] C. Zhao et al., "Secure multi-party computation: Theory, practice and applications," *Inf. Sci.*, vol. 476, pp. 357–372, Feb. 2019.
[38] Y. Lindell, "Secure multiparty computation," *Commun. ACM*, vol. 64, no. 1, pp. 86–96, 2020.
[39] C. Castelluccia, E. Mykletun, and G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks," in *Proc. 2nd Annu. Int. Conf. Mobile Ubiquitous Syst., Netw. Services*, 2005, pp. 109–117.
[40] V. S. Miller, "Use of elliptic curves in cryptography," in *Proc. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 1985, pp. 417–426.
[41] C. A. Lara-Nino, A. Diaz-Perez, and M. Morales-Sandoval, "Elliptic curve lightweight cryptography: A survey," *IEEE Access*, vol. 6, pp. 72514–72550, 2018.
[42] K. Maletsky, "RSA vs. ECC comparison for embedded systems," Microchip Technol. Inc., Chandler, AZ, USA, Tech. Rep. DS00003442A, 2020.
[43] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *J. Cryptograph. Eng.*, vol. 2, no. 2, pp. 77–89, Sep. 2012.
[44] M. Bellare, A. Boldyreva, and J. Staddon, "Randomness re-use in multi-recipient encryption schemeas," in *Proc. Int. Workshop Public Key Cryptogr.* Cham, Switzerland: Springer, 2003, pp. 85–99.
[45] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Secure Communications and Asymmetric Cryptosystems*. Evanston, IL, USA: Routledge, 2019, pp. 143–180.
[46] D. Galindo, S. Martin, and J. L. Villar, "Evaluating elliptic curve based KEMs in the light of pairings," Cryptol. ePrint Arch., Paper 2004/084, 2004. [Online]. Available: https://eprint.iacr.org/2004/084
[47] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 4, pp. 482–494, May 1998.
[48] B. Wu, J. Chen, J. Wu, and M. Cardei, "A survey of attacks and countermeasures in mobile ad hoc networks," in *Wireless Network Security*. Cham, Switzerland: Springer, 2007, pp. 103–135.
[49] B. B. Zarpelão, R. S Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017.
[50] I. Butun, P. Osterberg, and H. Song, "Security of the Internet of Things: Vulnerabilities, attacks, and countermeasures," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 616–644, 1st Quart., 2020.
[51] B. Barak et al., "On the (IM) possibility of obfuscating programs," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2001, pp. 1–18.
[52] N. Hrovatin, A. Tošić, and J. Vičič, "PPWSim: Privacy preserving wireless sensor network simulator," *SoftwareX*, vol. 18, Jun. 2022, Art. no. 101067.
[53] P. L. Hsu and H. Robbins, "Complete convergence and the law of large numbers," *Proc. Nat. Acad. Sci. USA*, vol. 33, no. 2, pp. 25–31, Feb. 1947.
[54] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wireless networks," *IEEE Pers. Commun.*, vol. 8, no. 1, pp. 34–39, 1st Quart., 2001.
[55] J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 7, pp. 1300–1315, Jul. 2001.
[56] C. Gomez, A. Arcia-Moret, and J. Crowcroft, "TCP in the Internet of Things: From ostracism to prominence," *IEEE Internet Comput.*, vol. 22, no. 1, pp. 29–41, Jan. 2018.
[57] T. Clausen et al., *Optimized Link State Routing Protocol (OLSR)*, document RFC 3626, INRIA, 2003.
[58] D. J. Bernstein, "Curve25519: New Diffie–Hellman speed records," in *Proc. Int. Workshop Public Key Cryptogr.* Cham, Switzerland: Springer, 2006, pp. 207–228.
[59] J. Daemen, "AES proposal: Rijndael," 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:17885291
[60] Y.-T. Tsou, C.-S. Lu, and S.-Y. Kuo, "SER: Secure and efficient retrieval for anonymous range query in wireless sensor networks," *Comput. Commun.*, vol. 108, pp. 1–16, Aug. 2017.
[61] C.-M. Chen, Y.-H. Lin, Y.-C. Lin, and H.-M. Sun, "RCDA: Recoverable concealed data aggregation for data integrity in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 4, pp. 727–734, Apr. 2012.
[62] K.-A. Shim, "A survey of public-key cryptographic primitives in wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 577–601, 1st Quart., 2016.
[63] M. Sethi, J. Arkko, A. Keranen, and H. Back, "Practical considerations and implementation experiences in securing smart object networks," Internet Eng. Task Force (IETF), Fremont, CA, USA, Tech. Rep. RFC 8387, 2018.

[64] L. E. Kane, J. J. Chen, R. Thomas, V. Liu, and M. Mckague, "Security and performance in IoT: A balancing act," *IEEE Access*, vol. 8, pp. 121969–121986, 2020.

[65] J. Bauer, R. C. Staudemeyer, H. C. Pöhls, and A. Fragkiadakis, "ECDSA on things: IoT integrity protection in practise," in *Proc. Int. Conf. Inf. Commun. Secur.* Singapore: Springer, Dec. 2016, pp. 3–17.

Niki Hrovatin was born in Trieste, Italy, in 1994. He received the B.S. and M.S. degrees in computer science from the University of Primorska, Koper, Slovenia, in 2020, where he is currently pursuing the Ph.D. degree. He is also a Teaching Assistant with the University of Primorska and a Research Assistant with the InnoRenew CoE, Izola, Slovenia. His current research interests include sensor networks, machine learning, distributed systems, and blockchain.

Aleksandar Tošić is currently an Associate Professor with the University of Primorska, Koper, and a Researcher with the InnoRenew CoE. His main research interests include distributed systems, privacy and security, sensors, and distributed ledger technologies.

Michael Mrissa received the Ph.D. and HDR (accreditation to supervise research) degrees from Université Claude Bernard Lyon 1, France, in 2007 and 2014, respectively. He is currently a Full Professor in computer science with the University of Primorska and a Researcher with the InnoRenew CoE, Slovenia. He has published over 100 peer-reviewed papers in international conferences and journals and has been involved in numerous national and international projects. His research interests include privacy and security in distributed systems, sensor networks, and the Web of Things.

Jernej Vičič is currently an Associate Professor with the University of Primorska, Slovenia. He is also the Head of the Distributed Ledger Technologies and Language Technologies Laboratory. His research interests are mostly linked to the title of the laboratory.

2.5 Paper 5

Title: Privacy-Preserving Data Mining on Blockchain-Based WSNs
Authors: Niki Hrovatin, Aleksandar Tošić, Michael Mrissa, Branko Kavšek
Year: 2022
Journal: MDPI Applied Sciences
DOI: `https://doi.org/10.3390/app12115646`
Link: `https://www.mdpi.com/2076-3417/12/11/5646`

applied sciences

Article

Privacy-Preserving Data Mining on Blockchain-Based WSNs

Niki Hrovatin [1,2,†], Aleksandar Tošić [1,2,†], Michael Mrissa [1,2,†] and Branko Kavšek [1,3,*,†]

1 Faculty of Mathematics, Natural Sciences and Information Technologies, University of Primorska, Glagoljaška 8, 6000 Koper, Slovenia; niki.hrovatin@famnit.upr.si (N.H.); aleksandar.tosic@upr.si (A.T.); michael.mrissa@innorenew.eu (M.M.)
2 InnoRenew CoE, Livade 6a, 6310 Izola, Slovenia
3 Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
* Correspondence: branko.kavsek@upr.si
† These authors contributed equally to this work.

Abstract: Currently, the computational power present in the sensors forming a wireless sensor network (WSN) allows for implementing most of the data processing and analysis directly on the sensors in a decentralized way. This shift in paradigm introduces a shift in the privacy and security problems that need to be addressed. While a decentralized implementation avoids the single point of failure problem that typically applies to centralized approaches, it is subject to other threats, such as external monitoring, and new challenges, such as the complexity of providing decentralized implementations for data mining algorithms. In this paper, we present a solution for privacy-aware distributed data mining on wireless sensor networks. Our solution uses a permissioned blockchain to avoid a single point of failure in the system. Contracts are used to construct an onion-like structure encompassing the Hoeffding trees and a route. The onion-routed query conceals the network identity of the sensors from external adversaries, and obfuscates the actual computation to hide it from internally compromised nodes. We validate our solution on a use case related to an air quality-monitoring sensor network. We compare the quality of our model against traditional models to support the feasibility and viability of the solution.

Keywords: WSN; air quality; privacy; blockchain; data mining

check for updates

Citation: Hrovatin, N.; Tošić, A.; Mrissa, M.; Kavšek, B. Privacy-Preserving Data Mining on Blockchain-Based WSNs. *Appl. Sci.* **2022**, *12*, 5646. https://doi.org/10.3390/app12115646

Academic Editor: Federico Divina

Received: 25 April 2022
Accepted: 31 May 2022
Published: 2 June 2022

1. Introduction

Currently, wireless sensor networks (WSNs) are widely used in various application domains (e.g., healthcare, supply chains, or agriculture). They support collecting and storing data about the state of monitored objects (e.g., patients, products, or crops), enabling further analysis that benefits stakeholders (e.g., better health, traceability, increased productivity). As typical WSNs rely on cloud infrastructures to handle the large amount of collected data, they are subject to diverse issues related to maintenance, scalability, and vulnerability to security threats. Most cloud-related issues originate from its centralized design, naturally opening the way to decentralized systems as an alternative to handle data. In addition, the recent development of embedded technology available in sensors is a strong incentive for the design of decentralized solutions. Present-day smart sensors have such strong computational capacities that they can handle demanding tasks, including hosting an entire operating system and the full web protocol stack, including a web server, or they are directly connected to one. Therefore, they become part of the web, supporting the vision of a worldwide sensor web [1]. Such a context makes the decentralized operation of the data collection and management processes very appealing.

Indeed, decentralized systems have seen quite a strong development since the release of Bitcoin, a practical blockchain implementation that provides decentralized trust, leading to the development of various other implementations. While, for centralized WSNs, security attacks target specific weak points of the system, in decentralized WSNs, attacks consist in observing the flow of messages that go through the network, and in identifying the role of each node to better target further attacks. Additionally, the data mining tasks performed in centralized databases, usually in the cloud, need some adjustments to be realized in a decentralized WSN.

Therefore, there is a major need to (1) rethink data mining tasks so that they better fit the decentralized underlying system where they should be executed, and (2) protect the privacy of these tasks with moderate additional computing requirements. Several trends of work exist in this direction; however, most are very computationally intensive, or show high communication costs.

In this paper, we present a solution for privacy-preserving DM (PPDM) that relies on the blockchain to provide trust, and on onion routing (OR) to preserve the privacy of the DM tasks that are executed in WSNs.

In particular, our contribution features smart contracts to provide for role-based access control and secure DM execution , so that nodes can trust each other. Our solution, instead of sharing data between nodes, is to share the partially constructed models so that each sensor node only has access to the partial model that is the result of the computation of the previous sensor nodes.

The remainder of this paper is structured as follows: In Section 2, we overview the most relevant work on PPDM, OR, and blockchain, and highlight the research gap which our work covers. In Section 3, we present our architecture and detail how its different components are articulated. In Section 4, we detail the methodology followed to enable its operation over a use case related to indoor air quality monitoring. In Section 5, we present the results obtained and compare them to the closest work in the literature. Finally, we summarize our main contributions and present some guidelines for future work in Section 6.

2. Related Work

In this section, the related work relevant to our research is presented, and knowledge gaps are identified. Section 2.1 gives an overview of the state of the art in privacy-preserving data mining in WSNs, Section 2.2 presents the recent research on the use of onion routing in WSNs, and Section 2.3 lists the relevant applications of the blockchain on WSNs.

2.1. Privacy-Preserving Data Mining in WSNs

Conventionally, DM on WSN data is realized in a centralized processing point external to the WSN. While it is simple to implement, such an approach raises privacy concerns due to the required relocation of sensor node data to the external system through the WSN, a multi-hop wireless network operated by resource-constrained devices. There are many studies addressing privacy concerns in WSNs [2–4], several of them indicating that, in addition to data privacy, contextual privacy must also be considered.

A considerable amount of literature has been published on privacy-preserving data aggregation since it is particularly suited to WSNs' characteristics [5,6]. The concept is based on aggregating data as it travels through the multi-hop network towards the sink node, and on applying privacy-preservation techniques during aggregation points to prevent the disclosure of the data of individual sensor nodes. Although the data aggregate does not disclose individual sensor privacy, aggregating the data may also reduce the knowledge obtainable using DM.

Local differential privacy (LDP) is well established in DM applications that require the privacy of individual data sources [7]. Data is encoded or perturbed at the origin or intermediate point, and then sent to the consumer. Therefore, LDP techniques also leverage the trade-off between privacy and data utility. The authors in [8] discuss the application of LDP in decentralized IoT networks. An LDP IoT framework is presented in [9]; however, LDP is applied only at edge servers.

Homomorphic encryption was proposed for preserving privacy, both in DM [10] and WSN applications [6]. Privacy is assured since computations are performed on cryptographically secured data without decryption. However, the technique is computationally intensive [11,12], and reasonable solutions for WSNs are designed only to compute addition and multiplication on encrypted data.

Secure multi-party computation (SMC) [13] allows multiple parties to jointly compute a function over their inputs while keeping those inputs private. Many studies [10,14] use the SMC technique to preserve the privacy of individual data sources during DM operations. However, traditional solutions relying on SMC require high computation and communication costs; thus, they are not convenient for application in WSNs. Therefore, SMC in WSNs is limited to low-complexity tasks, such as data aggregation [15].

Data mining algorithms that are performed over a WSN are inherently distributed, which has shifted the focus from traditional data mining (DM) approaches to the paradigm of distributed data mining (DDM) in recent years. The authors of [16] provide a relatively recent survey of state-of-the-art DDM techniques, focusing also on privacy-preserving distributed data mining (PPDDM). On the other hand, [17] introduces the notion of decentralized spatial data mining (DSDM), where individual sensor-enabled computing nodes possess only local knowledge about their immediate neighborhood, but derive global knowledge through local collaboration and information exchange. This is especially relevant for our research, where we aim at monitoring indoor air quality over a WSN.

2.2. Onion Routing in WSNs

Onion routing has a long tradition in enabling anonymous communication in public networks, first using layered objects to establish anonymous connections [18], and now being implemented in the TOR network to construct anonymous connections one hop at a time using a key-agreement protocol [19].

Although combined applications in the context of PPDM are rare [20], onion routing remains a privileged technique to protect from external monitoring attacks, as is proven by existing work on the topic [21].

De Cristofaro et al. [22] used a layered object to retrieve data from a WSN without disclosing the identity of the queried sensor node. Although the technique preserves privacy effectively, it only allows for data retrieval from individual nodes. One important work related to query privacy over WSNs is the work of Carbunar et al. [23], which proposes a similar solution, although with different underlying architectural choices (i.e., no region-based servers and a different trust model). The solution works by using source routing to privately route a declarative query to the aggregator nodes of a WSN. However, aggregator nodes are a point of failure for the network since they collect data from the sensor nodes in their proximity.

In general, we noted that most related work adopts a strong separation between data collection/querying and data mining. This highlights the originality of our work, where the data remains on the sensors, with a direct decentralized implementation of the data mining process that involves using onion routing for the purpose of protecting the tasks together with the processed data.

2.3. Blockchain Applications in WSNs

The unique properties that blockchain networks have, and how they can benefit IoT and WSNs, has been studied extensively. Early research focused on using a blockchain network as an external component of systems that are mainly used as databases for key storage and WSN management [24,25]. Blockchain networks have been used in combination with WSNs for data security, sensor node authentication [26], removing single points of failure in WSNs [27], and secure data accumulation [28].

Recent advances in blockchain protocols have enabled the design of light clients, which are nodes participating in the network without needing to maintain the entire chain. This enables clients to run on sensor nodes with low computing power and storage requirements. Coupled with smart contract platforms, new applications, such as malicious node detection [29] and anomaly detection [30], have surfaced.

Although we use blockchain for access control, which is a typical way to ensure trust between actors in a decentralized setup, the originality of our work lies in the use of smart blockchain contracts to ensure the authenticity of the onion messages, which is a novel approach to blockchain when combined with onion routing, as in this context.

2.4. Main Contributions

1. We develop a fully decentralized PPDDM framework for building incremental models where data never leaves the sensor, further improving data privacy and security;
2. By using onion routing, we are able to obfuscate computations between participating sensors and external adversaries;
3. We remove any SPOF by implementing RBAC and key storage using blockchain;
4. We show that the PPDDM approach achieves comparable accuracy to traditional DM approaches for the air quality monitoring use case.

3. Architecture

Our architecture is built over a set of sensor nodes that have the computing power of a Raspberry Pi, and is connected to an IAQ sensor that collects temperature (in Celsius), relative humidity, dew point, absolute humidity, CO_2, VOC index, and luminance.

Each sensor node gathers local data from its attached sensor, and at the same time, is a blockchain node. The proposed architecture requires a blockchain supporting smart contract functionality. The term “smart contract” was first defined in [31] as a computerized transaction protocol that executes the term of a contract. Today, the term “smart contract” is adopted for software that is recorded on the blockchain and that is executed by nodes maintaining the blockchain as part of transaction processing. Ethereum [32] was the first blockchain to implement Turing-complete smart contracts. This was achieved by introducing the Ethereum Virtual Machine (EVM), the gas concept, and the account-based transaction model. In addition to allowing the encoding of arbitrary state transition functions, the smart contract functionality implemented in the Ethereum network allows the smart contract to maintain a state; therefore, more complex applications can be built.

The OpenEthereum private network could serve our system as a secure, decentralized, and immutable smart contract platform. The permissioned Ethereum network operates the proof of authority (PoA) [33] consensus; therefore, the network is maintained by a selected set of validator nodes. Other nodes do not contribute to the state transition processing; therefore, these nodes can run an Ethereum light client [34].

An illustration of the system architecture is given in Figure 1. The use of onion routing for PPDDM is described in Section 3.1. The operation of the proposed architecture and a smart contract description is given in Section 3.2.

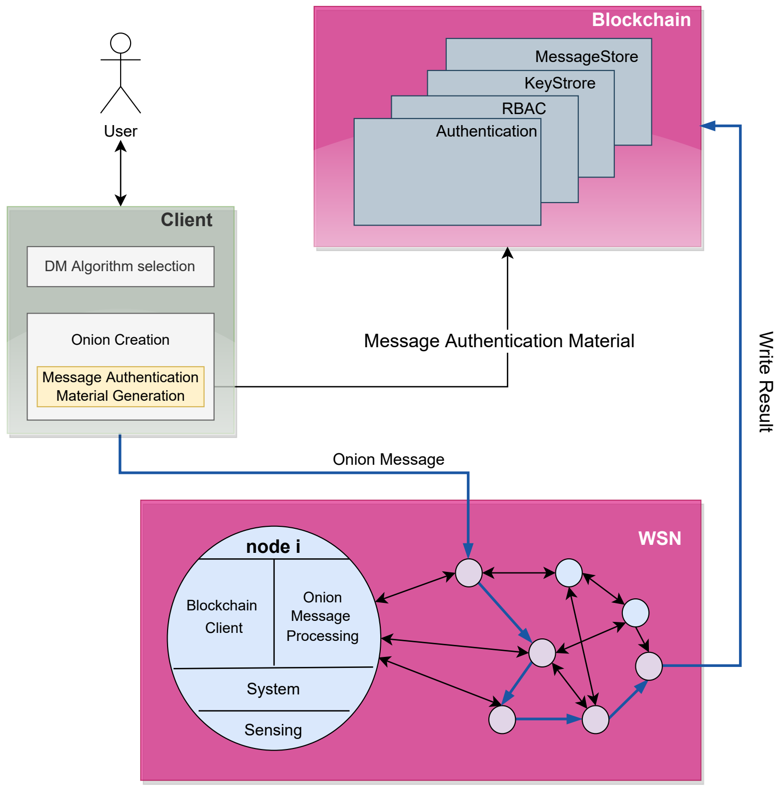


Figure 1. The figure displays the architecture of the proposed system and the interaction between the user, the WSN, and the blockchain.

3.1. Onion Routing for Privacy-Preserving Decentralized Data Mining

This section describes how to use the onion routing technique for privacy-preserving decentralized data mining. The onion routing technique builds upon messages consisting of encryption layers ciphered using public-key cryptography. Each layer of a message contains the necessary encryption key material and the address of the node that possesses the secret key, which will decipher the next layer. Therefore, the layered object is deciphered following the path given at message construction and delivers encryption key material to in-path systems. The encryption key material is used to establish a connection between the layered object origin and the last receiver. The connection is anonymous since none of the systems involved in the communication will know the whole connection route, apart from the layered object origin.

To achieve privacy-preservation for DDM, we rely on a new use of onion routing that was first described in [35]. Similar to onion routing, this technique uses messages consisting of several encryption layers, and each layer includes path details. However, the encryption key material is not included in each layer of the layered object, and it is not used to establish an anonymous connection. Instead, the encryption key material gives access to the payload accompanying the layered object. Therefore, only specific nodes in the message path are able to access the message payload, and their identity is concealed among the set of nodes routing the layered object.

Our system uses this technique to convey a DM model to WSN nodes. Figure 2 illustrates a message in our system. The DM model is enclosed in the payload accompanying the layered object. Nodes receiving encryption key material from the layered object perform model processing on their local data. Therefore, model processing is realized on WSN nodes without a central processing point. The technique ensures data privacy since the private sensed data of nodes does not leave the origin node. The generated communication traffic is uniform; nodes performing model processing are concealed among the set of nodes routing the message, the message path and sojourn time are randomized, the message size is uniform, and the messages are indistinguishable by encryption and are mixed when routed in the network. Therefore, external actors eavesdropping on a wireless communication cannot learn information from the network traffic.

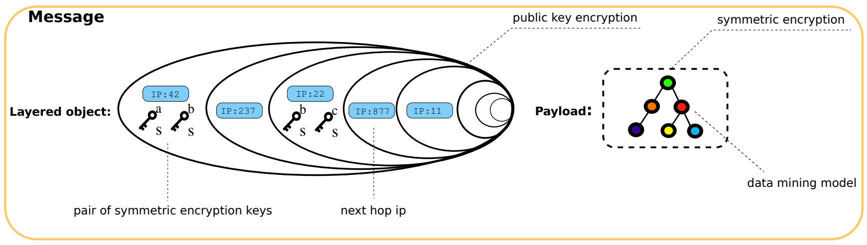


Figure 2. A message of the privacy-preserving protocol described in [35], adapted for DM model processing. The layers of the layered object, containing pairs of symmetric encryption keys, include distinct keys. The first key is used for payload decryption, and the second key for payload encryption after DM model processing. Please note that encryption key pairs are chained through layers of the layered object.

3.2. Role-Based Access Control and Secure Data Mining with Smart Contract

The smart contract enabling our architecture supports the following functionalities:

- *Authentication:* allows sensor node authentication and stores the sensor node public key in the *KeyStore*;
- *Role-Based Access Control (RBAC):* verifies user permissions. The smart contract contains the roles assigned to the public keys of authorized users. Roles enforce the set of DM algorithms that a user can execute on nodes of the WSN;
- *Query Submission:* authenticates the user with RBAC and writes the message authentication material in the *MessageStore*;
- *Message Authentication:* a view function called by sensor nodes to verify the message origin and permissions using the message authentication material;
- *Result Submission:* allows authenticated sensor nodes to submit the message result and remove the corresponding message authentication material from the *MessageStore*.

The outlined architecture allows for the management of user roles with RBAC functionalities. A user entrusted with a specific role is allowed to construct a message made of the layered object and the payload, as is described in Section 3.1. The payload content is constrained by the user role, as defined by the RBAC.

The user interrogates the *KeyStore* to learn sensor node addresses for message construction. After message construction, the user calls the *query submission* functionality providing the message authentication material. The message authentication material consists of a Merkle proof [36] constructed from the layered object layers, and the signature of the Merkle proof produced using the user's private key. The user then feeds the constructed message to the WSN.

Each sensor node receiving a message performs the following tasks. The layer of the layered object that is dedicated to this specific node is deciphered, and the message is authenticated by calling the *Message Authentication* function. The message authentication is performed by verifying the hash of the layered object's content against the Merkle proof in the *MessageStore*. The user's role and permission for DM model execution are also verified; this is conducted using the signature attached to the Merkle proof.

The penultimate layer of the layered object holds a flag signaling its position. The sensor node detecting this flag calls the *Result Submission* function and stores the message result. For simplicity, we assume that the message result is stored on the blockchain; however, the result could be stored in a separate data store, submitting on-chain only the message authentication material, the hash of the result, and the resource locator.

The user that issued the request tracks the chain for results associated with the message authentication material of the query submission. The obtained message result consists of the last layer of the layered object and the accompanying payload enclosing the DM model. Next, the user decrypts the layered object with its private key, revealing the symmetric encryption key that gives access to the payload content.

Please note that problems related to Byzantine faults are out of the scope of our work, so we assume in this paper that the nodes collect the data properly, and execute exactly the tasks assigned to them.

4. Methodology

This section presents the methodology behind the experimental evaluation of our proposed PPDDM method in detail. Section 4.1 gives an outline of the dataset that was used in the experiments—the GAMS IAQ dataset [37]. In Section 4.2, the "traditional" data mining algorithms that were used for comparison against our PPDDM are presented. Finally, Section 4.3 provides all the details about our PPDDM method and describes how the experimental evaluation was performed.

4.1. Indoor Air Quality Dataset

In order to validate the proposed distributed data mining approach, we look at the GAMS indoor air quality (IAQ) dataset, which is publicly available in [37]. This dataset was used for experimenting and benchmarking in several research works [38–40]. In this work, we use the GAMS dataset to compare the distributed data mining approach with traditional centralized data mining.

As reported in Table 1, the GAMS dataset includes observations from six IAQ indicators. Observations were collected from 21 November 2016 to 28 March 2017 on a minute basis. From further dataset inspection, we identified that, for benchmarking, it could be interesting to estimate the CO_2 level from T, RH, and VOC.

Table 1. The IAQ indicators of the GAMS dataset.

Feature	Sample Value	Unit
Carbon dioxide (CO_2)	708.0	ppm
Relative humidity (RH)	40.09	%
Particulate matter 2.5 micron ($PM_{2.5}$)	10.2	$\mu g/m^3$
Particulate matter 10 micron (PM_{10})	9.0	$\mu g/m^3$
Temperature (T)	20.83	Celsius
Volatile organic compounds (VOC)	0.093	ppm

The data were pre-processed by averaging observations on a six-minute interval. Table 2 summarizes the statistics of the pre-processed GAMS dataset.

Table 2. Summary of the statistics of the pre-processed GAMS dataset.

	CO_2	T	RH	VOC
mean	712.1	23.00	28.40	0.1168
std	405.61	2.08	5.46	0.086
min	370.2	18.04	22.05	0.062
25%	431.70	21.43	34.70	0.0635
50%	494.0	22.89	38.30	0.0740
75%	900.1	24.73	42.05	0.1405
max	2603.0	27.95	72.09	0.8702

All measured features include 29,420 observations.

4.2. Traditional Data Mining

Here, by "traditional" data mining, we mean machine learning (ML) algorithms run in a centralized fashion on data that is available in memory. The data used to learn and evaluate the "traditional" ML algorithms were the same pre-processed GAMS IAQ data with summary statistics that are presented in Table 2.

For the ML algorithms selected in this phase of the experimental evaluation, we used the WEKA ML framework [41]—an open-source data mining (DM) framework developed in the University of Waikato in New Zealand, containing over 200 implementations of various ML and DM algorithms.

Since all the attributes that describe our GAMS dataset are of numeric type, it was natural to select regression algorithms for the comparisons. Three well-known regression algorithms were selected, namely, linear regression, model trees (the M5′ algorithm [42,43]), and random forest [44]. The linear regression algorithm was selected as a baseline, because it is the most well-known and studied algorithm; the M5′ model trees algorithm was selected because it can produce more sophisticated, yet still well-explainable models, and the Random Forest was selected with the hope of producing a highly accurate result.

All three ML algorithms were run with their default parameters on the first 90% of the GAMS IAQ dataset to learn a model, which was then evaluated on the remaining 10% of the data. This rather unintuitive evaluation protocol was used (instead of the standard k-fold cross validation) because of the time-series nature of the GAMS dataset.

4.3. Distributed Data Mining

The distributed data mining setting described in Section 3 characterizes a network of nodes that store the last historical data about locally sensed IAQ features. A DM algorithm is then privately conveyed from node to node and executed in situ. Therefore, the DM model cannot access the global data—only the data provided by the node on which the DM model is executing is accessible to the model. Moreover, conveying large DM models from node to node in a WSN is resource-demanding and significantly affects the response times of the proposed architecture. Therefore, it is necessary to send the least amount of data.

To validate the distributed data mining approach, we first partition the GAMS IAQ dataset to simulate the described context according to Section 4.3.1. Section 4.3.3 gives the selected DDM algorithm for comparison with traditional centralized DM and outlines the DDM algorithm processing on the partitioned data. Section 4.3.4 details the experiment design for assessing the message propagation time at different DM model sizes.

4.3.1. Data Partitioning

The GAMS pre-processed dataset was partitioned into ten subsets to resemble a WSN of 10 closely located nodes. As shown in Figure 3, we applied round-robin partitioning. Therefore, each subset holds data from the entire monitored period, and data is chronologically ordered. As can be read from Figure 3, each observation summarizes a 1-hour interval of a sensor node.

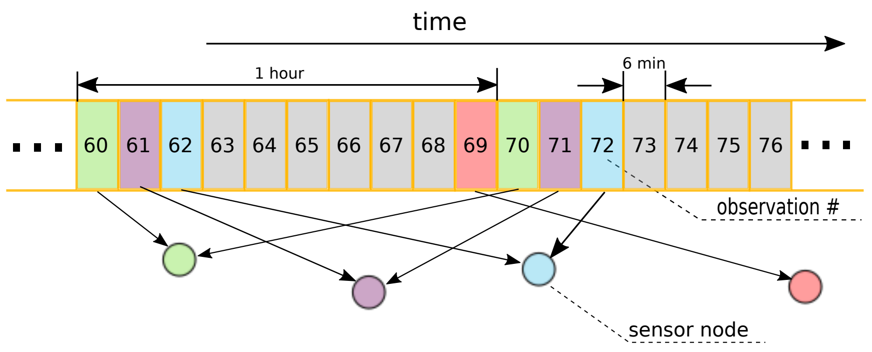


Figure 3. Data partitioning according to the round-robin method.

4.3.2. Design of the Experiment

Each sensor node collects its own data (in our simulation, there is actually a round-robin data assignment scheme that simulates sensor node data collection, as is depicted in Figure 3). The sensor node data is then used to "augment" the Hoeffding tree received from the previous sensor node, producing an updated Hoeffding tree that is then passed to the next sensor node. The whole procedure is depicted in Figure 4 and has already been proved effective in [45].

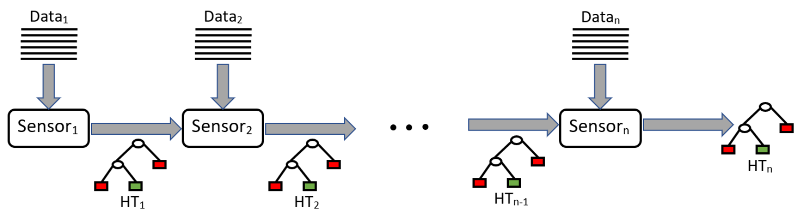


Figure 4. Incremental learning with Hoeffding trees.

As can be observed from Figure 4, the initial Hoeffding tree (HT_1) is first learned in the batch from the data provided by the first sensor node—$Sensor_1$ ($Data_1$). This tree is then propagated through the WSN, and in each subsequent WSN node ($Sensor_k$), an updated Hoeffding tree is generated (HT_k) from the previous Hoeffding tree (HT_{k-1}) and the data from the current sensor node—$Sensor_k$ ($Data_k$). The final Hoeffding tree is then represented by the last tree in this chain—HT_n.

Such a partial Hoeffding tree's (HT_k's) propagation through the WSN (instead of the actual sensor data being propagated) is most suitable for preserving privacy, and also diminishes the load on the WSN.

4.3.3. Distributed Data Mining Algorithm

Due to the context of distributed data, where only limited local data is available for processing, we cannot run the same algorithms as in Section 4.2. Instead, we look at the Hoeffding tree [46], a very fast decision tree algorithm proposed for stream data. The algorithm builds a decision tree as data arrives, without the need to reuse instances from already-processed data. This is achieved by maintaining the statistics needed for splitting attributes in each leaf node and determining when a split should occur as data arrives, using the Hoeffding's bound [47]. The Hoeffding probability bound enables us to compute ε, which can be used as a confidence interval for the estimation of the split.

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta))}{2N}}$$

R is the range of the random variable r. By performing N-independent observations of the variable r, and computing their mean $\bar{r}$, the Hoeffding bound enables us to state with confidence $1-\delta$ that $\bar{r}$ is within ε distance from the true mean of the variable r.

In [46], it was shown that the Hoeffding tree output is asymptotically nearly identical to that of a traditional batch learner in infinitely many examples.

We use the Python implementation provided in the scikit-multiflow [48] library, specifically, the regression variant named *HoeffdingTreeRegressor*. Development of the scikit-multiflow library stopped in the year 2020, when the developers of scikit-multiflow and Creme [49], another Python library for data stream mining, decided to merge their efforts and release a new library named River [50]. Even though the state of the art of data stream mining is currently implemented in the River library, we decided to employ the *HoeffdingTreeRegressor* implemented in the scikit-multiflow since this implementation allowed us to control the tree size in bytes. The *HoeffdingTreeRegressor* algorithm at the base of the implementation in River is identical to the one in scikit-multiflow. The algorithm is based on the Fast and Incremental Model Trees (FIMT-DD) method described in [51], without concept drift adaptation.

The pseudo-code is given in Algorithm 1. From the code, it is possible to notice that the algorithm starts with an empty tree that keeps statistics at the leaves from arriving data. Each time k values arrive at a leaf, the algorithm finds the best split for each attribute and ranks the attributes according to the standard deviation reduction (SDR) measure [51]. Therefore, attributes are ranked such that the standard deviation is maximally reduced, if splitting by the highest ranked attribute, by considering the ratio of the SDR values between any of the attributes and the highest-ranked attribute as the random variable r with range $R \in [0,1]$. If the inequality $\bar{r}+\varepsilon < 1$ holds, then the highest-ranked attribute is the best split over the whole distribution for that portion of data.

Algorithm 1: Pseudo-code of the *HoeffdingTreeRegressor*.

```
1  Begin with an empty leaf (root)
2  repeat
3      Read the next example;
4      Traverse the example to a leaf;
5      Update statistics in the leaf;
6      if k examples were read then
7          Find best split per attribute;
8          Rank attributes using the SDR measure;
9          if splitting criterion is satisfied then
10             Make a split on the best attribute;
11             make two new branches leading to (empty) leaves;
12         end
13     end
14 until End of the stream;
```

We ran the ML algorithm with the sample average leaf predictor for the following maximal model sizes: $M = \{5k, 10k, 25k, 50k, 100k, 250k, 500k, 1M, 2.5M, 5M\}$ (bytes). In the provided code snippet (Listing 1), the `MAX_TREE_SIZE` parameter on line 8 corresponds to one of the values of M. The algorithm was run on the partitioned pre-processed GAMS IAQ dataset. To closely simulate the architecture described in Section 3, we design the experiment as if model processing occurs once a day, processing 24 h of data from each node following the round-robin approach. Therefore, the *HoeffdingTreeRegressor* processes 24 h of data from subset #0, then moves to data from subset #1, and so on. After processing at subset #9, the model moves to the next-day data from subset #0 and repeats until it reaches the end of data.

Listing 1. Hoeffding Tree training and evaluation.

```
1  from skmultiflow.trees import HoeffdingTreeRegressor
2  data = pd.read_csv("gams_aq/gams_preprocessed.csv")
3  X = data[['temperature', 'humidity', 'voc']]
4  Y = data['co2']
5  y_actual = []
6  y_predicted = []
7
8  model = HoeffdingTreeRegressor(max_byte_size=MAX_TREE_SIZE, leaf_prediction="mean")
9
10 for i in range(len(Y)):
11 prediction = model.predict(X[i])
12 y_predicted.append(prediction)
13 y_actual.append(Y[i])
14 model.partial_fit(X[i], Y[i])
```

We say that the *HoeffdingTreeRegressor* is processing data because when a data instance is provided to the algorithm, it first estimates the response variable value and then uses the observed value for model learning. These operations are shown in the provided code snippet (Listing 1) on lines 11 and 14, respectively. To hold consistency with Section 4.2, in Section 5, we report the model evaluation results on the last 10% of the data.

4.3.4. Message Propagation Time

The system we are presenting makes use of the onion routing technique described in Section 3.1 to convey a DM model through the nodes of a WSN. Therefore, messages are routed through several nodes of a wireless multi-hop network and relayed multiple times, introducing substantial latency before receiving the result. To assess the response time of the proposed architecture operating over a large WSN, we look at the publicly released NS3-based simulator described in [52]. NS3 [53] is a discrete-event network simulator for internet systems, and it is the most-used network simulator by the IoT research community [54]. The simulator described in [52] is designed to simulate the technique described in Section 3.1 over a WSN that varies in node number, topology, routing, etc. Furthermore, it allows for the manipulation of message structures by defining nodes in the message path (corresponding to layers of the layered object), detailing the payload content, and other settings such as maintaining message uniformity.

We set up a simulation that constructs a WSN of 200 nodes. Nodes are deployed at random locations on a disc-shaped plane with a radius of 300 m. We selected the deployment scheme affected by randomness to model the usually non-uniform node density of WSNs. Nodes have a wireless communication range of approximately 30 m, operating based on the IEEE 802.11n standard at 2.4 GHz and with a data rate of 13 Mbps. The maximum transmission unit and maximum segment size are set to the ns3 default value, namely, 2296 bytes and 536 bytes, respectively.

Messages are transmitted over the TCP protocol, multi-hop routing is conducted using the Optimized Link State Routing Protocol (OLSR) [55], and cryptography is completed with the Libsodium library [56].

The simulator is set up to issue 20 messages for each value of M, with M being the size in bytes of the model carried in the message payload. The layered object included in the message is constructed to lead the message through 20 randomly selected nodes. Our previous work [35] stated that for privacy requirements, half the nodes in the message path are only performing routing operations, without access to the payload content (decoy nodes). Therefore, this setup allows for estimations of the propagation time of messages constructed to perform model processing at 10 nodes. Message size is maintained as uniform after the onion routing operation, with padding when necessary. Figure 5 shows the simulated WSN and an onion message routed through 20 sensor nodes.

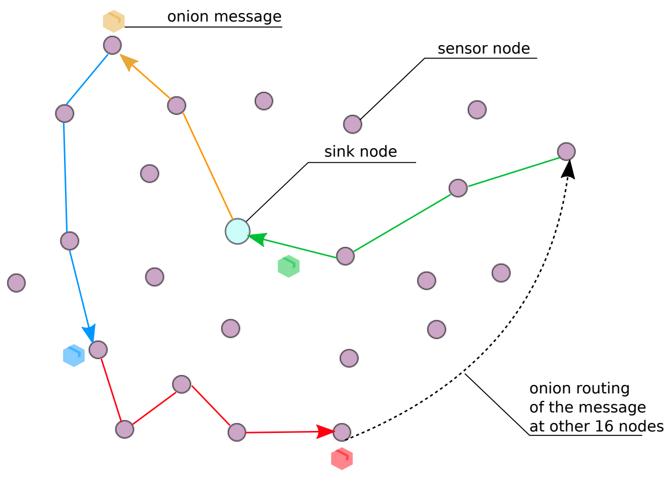


Figure 5. Example of the disc-shaped WSN with random node placement generated in the simulator [52]. The sink node is deployed in the center of the network, issuing onion messages sequentially. Each onion message is onion-routed at 20 nodes and ends its path at the sink node. In the figure, we indicate an onion routing operation with the change in color of the packet.

5. Results

5.1. Distributed Data Mining Approach

To validate our approach, we analyze the accuracy of Hoeffding trees in predicting CO_2, depending on the set tree size. Figure 6 shows the accuracy of predicted CO_2 compared to observed values from sensors, depending on the tree size. The last 10% of the chronologically ordered data was used in the evaluation phase. We observe that the overall accuracy of Hoeffding trees improves with the increase in tree size. As discussed in Section 4.3.3, Hoeffding trees store attribute statistics in leaves; therefore, as new leaves are formed due to split decisions, the model size increases, and also the prediction accuracy should increase.

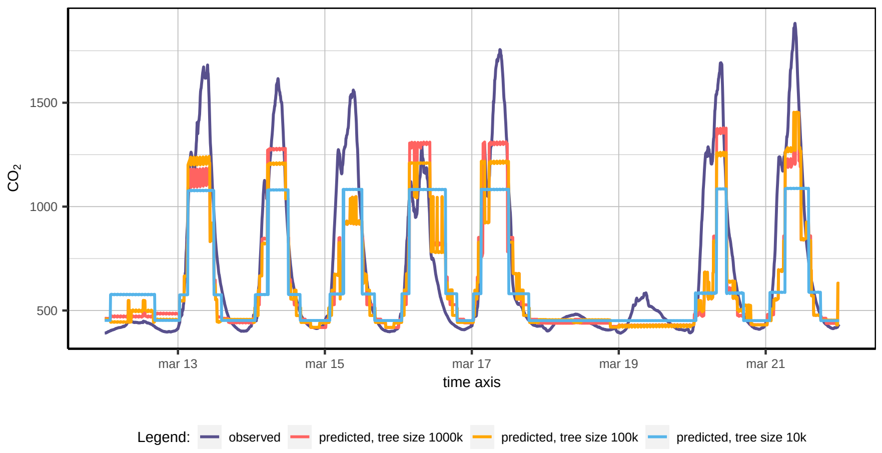


Figure 6. Comparison between predicted CO_2 using Hoeffding trees and observed for different tree sizes.

While training Hoeffding trees, we make continuous predictions in order to monitor the progressive improvements in accuracy through time. Figure 7 illustrates four different tree sizes and their respective prediction accuracies. The box plots show the aggregated weekly data points for clarity. We observe that, in all cases, the variance and mean decrease with time. Moreover, we observe that a small tree size (5 Kb) is worse overall when compared to the other sizes, while increasing the size of the tree beyond 100 Kb has no marginal effect on the accuracy.

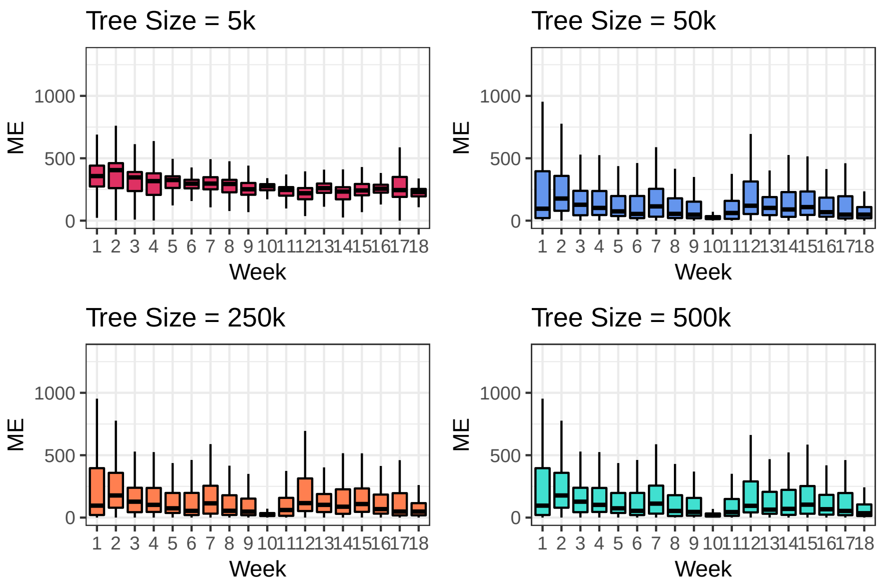


Figure 7. Overall prediction accuracy of different-sized Hoeffding trees while learning.

Figure 8 shows the comparison between the Hoeffding tree and the traditional DM algorithms. We compared the absolute error between the predicted and observed values on a daily basis. All models were trained using the first 90% of the data, and evaluated using the last 10%. We observe that the prediction error of the Hoeffding tree is comparable with traditional algorithms throughout all the reported days in Figure 8. Additionally, we provide standard model quality metrics in Table 3, obtained from a model evaluation on the last 10% of the data. From Figure 8, it is possible to notice that the Hoeffding tree model outperforms the random forest and M5′ models. This can be also read in Table 3. From Table 3, it appears that the linear regression model is the best predictor, achieving the highest values for the RMSE, RRSE, and COR metrics. Although the Hoeffding tree model did not achieve the best results, the quality metrics reported in Table 3 and Figure 8 confirm that the DDM approach using the Hoeffding tree achieves comparable accuracy to traditional DM approaches. Furthermore, our DDM approach can be compared to the research in [40], where researchers developed an inference model that was also tested on the same GAMS IAQ dataset [37]. They report a coefficient of determination of 0.6742 when predicting the CO_2 value from other predictors, whereas deriving the coefficient of determination from Table 3 for the Hoeffding tree results in a value of 0.7109.

Table 3. Statistical comparison of the prediction accuracy of different models. MAE—mean absolute error, RMSE—root mean squared error, RRSE—root relative squared error, and COR—correlation coefficient.

Algorithm	MAE	RMSE	RRSE	COR
Random forest	154.8616	249.2222	61.5162	0.803
M5′	148.5085	232.2563	57.3284	0.8323
Linear regression	157.7538	212.1302	52.3606	0.8717
Hoeffding tree	135.3154	230.9903	57.0625	0.8432

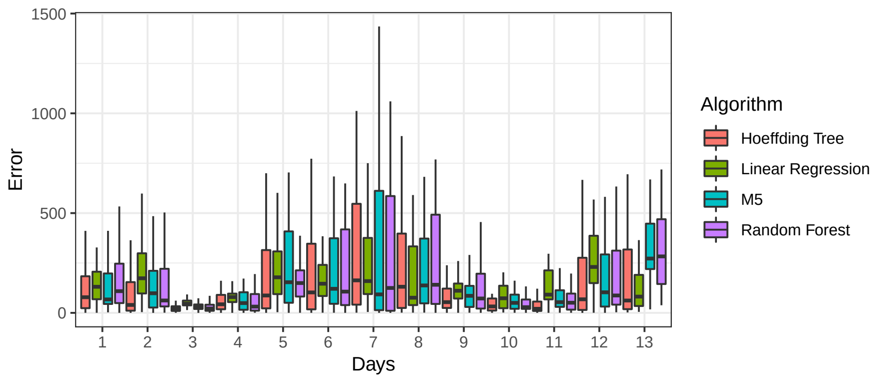


Figure 8. Absolute error comparison between the Hoeffding tree and traditional algorithms. The box plots aggregate the daily absolute error per DM model.

5.2. Validation of the Privacy-Preserving Architecture

The described privacy-preserving architecture builds upon the Ethereum blockchain running the PoA consensus and the privacy-preserving communication protocol described in [35]. Since the use of the blockchain in our architecture does not diverge from the typical use, we focus on evaluating the privacy-preserving communication protocol for distributed data mining. Moreover, the adoption of blockchain technology for resource-constrained devices was discussed in several other research works; specifically, we point out [57], which provides performance analyses of the Ethereum blockchain running the PoA consensus in IoT networks.

The privacy-preserving communication protocol uses the onion routing technique, which builds on messages having several layers of public-key cryptography; public-key cryptography is known for being computationally intensive. However, in the proposed architecture, multiple public-key cryptography operations are required only at the point of message creation; therefore, this is conducted by the user on the client, which is supposed to be running on a typical consumer machine. Nodes of the WSN at the onion message receipt perform only one public-key cryptography operation. To assess the time required for a node of our system to decipher a message, we coded a test script that measures the required time for ciphering several bytes of data using the same cipher as the simulator [52]. The cipher is based on the elliptic curve Curve2551 [58], and is implemented in the Libsodium library [56] under the name *Sealed Box*. By our measurments, encrypting 1 kB and 16 kB of data using the *Sealed Box* implementation on a Raspberry Pi4 takes, respectively, 603 and 735 microseconds. Therefore, the processing time required for cryptographic operations is low compared to the DM model processing and the message propagation time. On the other hand, DM model processing depends on the amount of data to be processed at each node and on the computational complexity of the applied DM model. In the proposed architecture, we describe the use of the DM model known as Hoeffding tree, which is known to be memory and computationally efficient [59].

To show the feasibility of the onion-routed approach, we study the impact of message propagation through the WSN based on different tree sizes. The experiment was designed to simulate the worst-case scenario with random node deployment, inducing a high packet drop rate. Figure 9 shows the average propagation time of an onion message relative to the size of the Hoeffding tree. We observe a low impact of larger trees on encryption, decryption, and training on total propagation times.

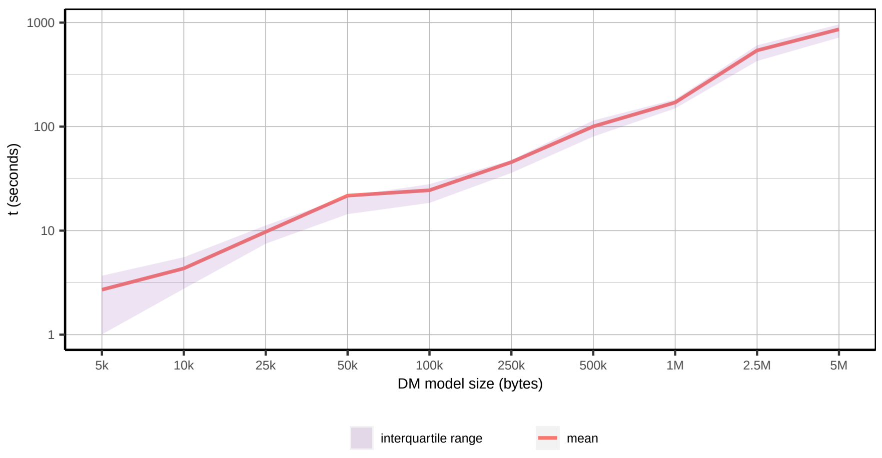


Figure 9. The average propagation time of onion messages through 20 WSN nodes, conveying a DM model of size M. To provide a better overview of the collected data, we report the log transformation of the y axis.

6. Conclusions

In this paper, we describe an innovative combined use of blockchain and onion routing to realize privacy-preserving decentralized data mining over wireless sensor networks. The advantage of such an approach is to avoid problems such as single points of failure and dependencies on remote infrastructures, while providing qualitatively similar results to those of traditional data mining techniques.

In effect, our solution relies on smart contracts and onion routing to address further problems that arise from the decentralized nature of the solution. Onion routing protects the decentralized data mining process from external monitoring, and smart contracts introduce a way to trust the computation being realized over the different sensors that form the network. We rely on a public key infrastructure to provide for data integrity and confidentiality.

Future work includes exploring advanced methodologies for data processing and data storage to guarantee properties such as fault tolerance or resilience against network partition. Pushing the boundaries of data mining on the edge of the network opens doors for smart, autonomous, embedded systems to support human activities, while reducing the required environmental impact.

Author Contributions: Conceptualization, N.H. and A.T.; Data curation, N.H., A.T. and B.K.; Formal analysis, M.M.; Funding acquisition, M.M.; Investigation, N.H., A.T., M.M. and B.K.; Methodology, N.H., A.T. and B.K.; Project administration, A.T. and M.M.; Resources, M.M. and B.K.; Software, N.H.; Supervision, A.T. and M.M.; Validation, N.H., A.T. and B.K.; Visualization, N.H. and A.T.; Writing—original draft, N.H., A.T., M.M. and B.K.; Writing—review & editing, N.H., A.T., M.M. and B.K. All authors have read and agreed to the published version of the manuscript.

Funding: The authors gratefully acknowledge the European Commission for funding the InnoRenew project (Grant Agreement #739574) under the Horizon2020 Widespread-Teaming program and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Regional

Development Fund). They also acknowledge the Slovenian Research Agency ARRS for funding the project J2-2504.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Gibbons, P.; Karp, B.; Ke, Y.; Nath, S.; Seshan, S. IrisNet: An architecture for a worldwide sensor Web. *IEEE Pervasive Comput.* **2003**, *2*, 22–33. [CrossRef]
2. Li, N.; Zhang, N.; Das, S.K.; Thuraisingham, B. Privacy preservation in wireless sensor networks: A state-of-the-art survey. *Ad Hoc Netw.* **2009**, *7*, 1501–1514. [CrossRef]
3. Tomić, I.; McCann, J.A. A survey of potential security issues in existing wireless sensor network protocols. *IEEE Internet Things J.* **2017**, *4*, 1910–1923. [CrossRef]
4. Jiang, J.; Han, G.; Wang, H.; Guizani, M. A survey on location privacy protection in wireless sensor networks. *J. Netw. Comput. Appl.* **2019**, *125*, 93–114. [CrossRef]
5. Bista, R.; Chang, J.W. Privacy-preserving data aggregation protocols for wireless sensor networks: A survey. *Sensors* **2010**, *10*, 4577–4601. [CrossRef] [PubMed]
6. Xu, J.; Yang, G.; Chen, Z.; Wang, Q. A survey on the privacy-preserving data aggregation in wireless sensor networks. *China Commun.* **2015**, *12*, 162–180. [CrossRef]
7. Kasiviswanathan, S.P.; Lee, H.K.; Nissim, K.; Raskhodnikova, S.; Smith, A. What can we learn privately? *SIAM J. Comput.* **2011**, *40*, 793–826. [CrossRef]
8. Sun, M.; Tay, W.P. On the relationship between inference and data privacy in decentralized IoT networks. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 852–866. [CrossRef]
9. Xu, C.; Ren, J.; Zhang, D.; Zhang, Y. Distilling at the edge: A local differential privacy obfuscation framework for IoT data analytics. *IEEE Commun. Mag.* **2018**, *56*, 20–25. [CrossRef]
10. Mendes, R.; Vilela, J.P. Privacy-preserving data mining: Methods, metrics, and applications. *IEEE Access* **2017**, *5*, 10562–10582. [CrossRef]
11. Othman, S.B.; Bahattab, A.A.; Trad, A.; Youssef, H. Confidentiality and integrity for data aggregation in WSN using homomorphic encryption. *Wirel. Pers. Commun.* **2015**, *80*, 867–889. [CrossRef]
12. Hayouni, H.; Hamdi, M.; Kim, T.H. A survey on encryption schemes in wireless sensor networks. In Proceedings of the 2014 7th International Conference on Advanced Software Engineering and Its Applications, Haikou, China, 20–23 December 2014; pp. 39–43.
13. Zhao, C.; Zhao, S.; Zhao, M.; Chen, Z.; Gao, C.Z.; Li, H.; Tan, Y.a. Secure multi-party computation: Theory, practice and applications. *Inf. Sci.* **2019**, *476*, 357–372. [CrossRef]
14. Cock, M.d.; Dowsley, R.; Nascimento, A.C.; Newman, S.C. Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security, Denver, CO, USA, 16 October 2015; pp. 3–14.
15. Jung, T.; Mao, X.; Li, X.Y.; Tang, S.J.; Gong, W.; Zhang, L. Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation. In Proceedings of the 2013 Proceedings IEEE INFOCOM, Turin, Italy, 14–19 April 2013; pp. 2634–2642.
16. Gan, W.; Lin, J.C.W.; Chao, H.C.; Zhan, J. Data mining in distributed environment: A survey. *WIREs Data Min. Knowl. Discov.* **2017**, *7*, e1216. [CrossRef]
17. Laube, P.; Duckham, M. Decentralized Spatial Data Mining for Geosensor Networks. In *Geographic Data Mining and Knowledge Discovery*; Routledge: London, UK, 2008. [CrossRef]
18. Reed, M.G.; Syverson, P.F.; Goldschlag, D.M. Anonymous connections and onion routing. *IEEE J. Sel. Areas Commun.* **1998**, *16*, 482–494. [CrossRef]
19. Dingledine, R.; Mathewson, N.; Syverson, P. *Tor: The Second-Generation Onion Router*; Technical report; Naval Research Lab.: Washington, DC, USA, 2004.
20. El Mougy, A.; Sameh, S. Preserving Privacy in Wireless Sensor Networks using Onion Routing. In Proceedings of the 2018 International Symposium on Networks, Computers and Communications (ISNCC), Rome, Italy, 19–21 June 2018; pp. 1–6. [CrossRef]
21. Ravi, N.; Jeyanthi, P. Secure Inter Hop Verification with Onion Protocol Implementation for Reliable Routing In Wireless Networks. *Int. J. Eng. Technol.* **2016**, *8*, 183–190.
22. De Cristofaro, E.; Ding, X.; Tsudik, G. Privacy-preserving querying in sensor networks. In Proceedings of the 2009 18th International Conference on Computer Communications and Networks, San Francisco, CA, USA, 3–6 August 2009; pp. 1–6.

23. Carbunar, B.; Yu, Y.; Shi, W.; Pearce, M.; Vasudevan, V. Query Privacy in Wireless Sensor Networks. *ACM Trans. Sen. Netw.* **2010**, *6*, 1–34. [CrossRef]
24. Feng, L.; Zhang, H.; Lou, L.; Chen, Y. A blockchain-based collocation storage architecture for data security process platform of WSN. In Proceedings of the 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD)), Nanjing, China, 9–11 May 2018; pp. 75–80.
25. Roman, R.; Alcaraz, C.; Lopez, J.; Sklavos, N. Key management systems for sensor networks in the context of the Internet of Things. *Comput. Electr. Eng.* **2011**, *37*, 147–159. [CrossRef]
26. Moinet, A.; Darties, B.; Baril, J.L. Blockchain based trust & authentication for decentralized sensor networks. *arXiv* **2017**, arXiv:1706.01730.
27. Casado-Vara, R.; de la Prieta, F.; Prieto, J.; Corchado, J.M. Blockchain framework for IoT data quality via edge computing. In Proceedings of the 1st Workshop on Blockchain-Enabled Networked Sensor Systems, Shenzhen China, 4 November 2018; pp. 19–24.
28. Islam, A.; Al Amin, A.; Shin, S.Y. FBI: A Federated Learning-Based Blockchain-Embedded Data Accumulation Scheme Using Drones for Internet of Things. *IEEE Wirel. Commun. Lett.* **2022**, 11, 972–976. [CrossRef]
29. Shahid, A.R.; Pissinou, N.; Staier, C.; Kwan, R. Sensor-chain: A lightweight scalable blockchain framework for internet of things. In Proceedings of the 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Atlanta, GA, USA, 14–17 July 2019; pp. 1154–1161.
30. HUANG, H.; Xiaoxiao, W.; Gangqiang, L. Anomaly detection and location of malicious node for IoT based on smart contract in blockchain network. *Chin. J. Internet Things* **2020**, *4*, 58–69.
31. Szabo, N. Formalizing and securing relationships on public networks. *First Monday* **1997**, *2* . [CrossRef]
32. Buterin, V. Ethereum white paper. *GitHub Repos.* **2013**, *1*, 22–23.
33. Ethereum Clique Proof-of-Authority Consensus Protocol. Available online: https://github.com/ethereum/EIPs/issues/225 (accessed on 25 May 2022).
34. Ethereum Light Client Protocol. Available online: https://eth.wiki/concepts/light-client-protocol (accessed on 25 May 2022).
35. Hrovatin, N.; Tošić, A.; Mrissa, M.; Vičič, J. A General Purpose Data and Query Privacy Preserving Protocol for Wireless Sensor Networks. *arXiv* **2021**, arXiv:2111.14994.
36. Merkle, R.C. A digital signature based on a conventional encryption function. In *Conference on the Theory and Application of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1987; pp. 369–378.
37. GAMS Indoor Air Quality Dataset. Available online: https://github.com/twairball/gams-dataset (accessed on 24 April 2022).
38. Ristanović, M.; Miloš, M. Experimental Protocol for Assessing the Relation Between Indoor Air Quality and Living Unit Conditions in AAL. *Teh. Vjesn.* **2022**, *29*, 536–541.
39. Wu, Z.; Ma, C.; Shi, X.; Wu, L.; Dong, Y.; Stojmenovic, M. Imputing missing indoor air quality data with inverse mapping generative adversarial network. *Build. Environ.* **2022**, *215*, 108896. [CrossRef]
40. Saini, J.; Dutta, M.; Marques, G. ADFIST: Adaptive Dynamic Fuzzy Inference System Tree Driven by Optimized Knowledge Base for Indoor Air Quality Assessment. *Sensors* **2022**, *22*, 1008. [CrossRef]
41. Witten, I.H.; Frank, E.; Hall, M.A.; Pal, C.J. *Online Appendix for "Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques"*, 4th ed.; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2016.
42. Quinlan, J.R. *Learning With Continuous Classes*; World Scientific: Singapore, 1992; pp. 343–348.
43. Wang, Y.; Witten, I.H. Inducing Model Trees for Continuous Classes. In *Poster Papers of the 9th European Conference on Machine Learning*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 128–137.
44. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
45. Mrissa, M.; Tošić, A.; Hrovatin, N.; Aslam, S.; Dávid, B.; Hajdu, L.; Krész, M.; Brodnik, A.; Kavšek, B. Privacy-Aware and Secure Decentralized Air Quality Monitoring. *Appl. Sci.* **2022**, *12*, 2147. [CrossRef]
46. Domingos, P.; Hulten, G. Mining high-speed data streams. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, USA, 20–23 August 2000; pp. 71–80.
47. Hoeffding, W. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*; Springer: Berlin/Heidelberg, Germany, 1994; pp. 409–426.
48. Montiel, J.; Read, J.; Bifet, A.; Abdessalem, T. Scikit-Multiflow: A Multi-output Streaming Framework. *J. Mach. Learn. Res.* **2018**, *19*, 1–5.
49. Halford, M.; Bolmier, G.; Sourty, R.; Vaysse, R.; Zouitine, A. Creme, a Python Library for Online Machine Learning. 2019. Available online: https://pypi.org/project/creme/ (accessed on 8 March 2022).
50. Montiel, J.; Halford, M.; Mastelini, S.M.; Bolmier, G.; Sourty, R.; Vaysse, R.; Zouitine, A.; Gomes, H.M.; Read, J.; Abdessalem, T.; et al. River: Machine Learning for Streaming Data in Python. 2021. Availableonline:https://www.jmlr.org/papers/volume22/20-1380/20-1380.pdf (accessed on 17 March 2022).
51. Ikonomovska, E.; Gama, J.; Džeroski, S. Learning model trees from evolving data streams. *Data Min. Knowl. Discov.* **2011**, *23*, 128–168. [CrossRef]
52. Hrovatin, N.; Tošić, A.; Vičič, J. PPWSim: Privacy preserving wireless sensor network simulator. *SoftwareX* **2022**, *18*, 101067. [CrossRef]

53. nsnam. ns-3, a Discrete-Event Network Simulator for Internet Systems—Version 3.32. 1 October 2021. Available online: https://www.nsnam.org/ (accessed on 11 February 2022).
54. Chernyshev, M.; Baig, Z.; Bello, O.; Zeadally, S. Internet of things (iot): Research, simulators, and testbeds. *IEEE Internet Things J.* **2017**, *5*, 1637–1647. [CrossRef]
55. Clausen, T.; Jacquet, P.; Adjih, C.; Laouiti, A.; Minet, P.; Muhlethaler, P.; Qayyum, A.; Viennot, L. Optimized Link State Routing Protocol (OLSR). Rfc, INRIA. 2003. Available online: https://hal.inria.fr/file/index/docid/471712/filename/5145.pdf (accessed on 23 April 2022).
56. Libsodium The Sodium Crypto Library. Available online: https://libsodium.gitbook.io/doc/ (accessed on 28 May 2021).
57. Alrubei, S.M.; Ball, E.A.; Rigelsford, J.M.; Willis, C.A. Latency and performance analyses of real-world wireless IoT-blockchain application. *IEEE Sens. J.* **2020**, *20*, 7372–7383. [CrossRef]
58. Bernstein, D.J. Curve25519: New Diffie-Hellman speed records. In *International Workshop on Public Key Cryptography*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 207–228. [CrossRef]
59. Ikonomovska, E.; Gama, J.; Zenko, B.; Dzeroski, S. Speeding-Up Hoeffding-Based Regression Trees with Options. ICML. 2011. Available online: https://openreview.net/forum?id=ByVpXjZ_WS (accessed on 17 April 2022).

Chapter 3

Conclusions

Sensor networks, a product of advances in wireless communication, embedded electronics, and sensing technology, are ubiquitous in modern life. They are at the core of numerous applications from environmental monitoring to industrial automation, enhancing our interaction with the environment. However, the traditional centralized sensor network architecture presents several limitations, notably around communication overhead, performance, SPOFs, security, privacy, and data management, amongst others. This has catalyzed the shift towards decentralized architectures and the rise of edge computing in sensor networks. However, many challenges still lie ahead, particularly concerning implementing versatile, efficient, and secure distributed computing frameworks, as well as ensuring data privacy.

In this dissertation, we delve into privacy-preserving, in-network data processing in sensor networks, emphasizing in-situ data processing and leveraging the advantages of edge computing and decentralized architectures to mitigate limitations of the traditional centralized architecture.

3.1 In-Network Convolution in Grid Shaped Sensor Networks

CNNs are powerful machine learning algorithms, achieving high inference quality; however, their resource-intensive nature makes them unsuitable for running on single nodes in a sensor network, resulting in such networks often adopting a centralized architecture. In response to this challenge, our research into distributing the computational load of

CNN's convolutional layers among sensor nodes in grid-shaped networks uncovers a promising solution. By leveraging the inherent structure of CNN kernels and coordinating nodes locally, we show how to perform in-network CNN-layer processing. In [11] we describe a multi-layer architecture where several convolutional layers can be sequentially processed on sensor network nodes. Notably, this approach does not present SPOFs, and data processing occurs close to the monitored activity, eliminating the need for raw data to traverse the network.

Employing the ns-3 simulator, we assess the viability of the researched approach. Our findings suggest that while communication overhead is a concern, it becomes particularly challenging with large convolution kernels and low bit-rate links. For example, with a 7 x 7 kernel size, a 5% packet loss was observed at a data-rate of 250kbps, and a striking 87% packet loss at 125kbps. Simulations were conducted considering two network topologies: the grid topology and the diagonal grid topology, with the latter incorporating additional diagonal links. Notably, the diagonal grid topology achieved a 44% lower message propagation time compared to its counterpart. We have made the raw simulation data publicly available at the Zenodo repository [1].

Within this study, we collected a large set of simulated fall events using a grid shaped sensor network embedded in a flooring surface. Our published results [18] accurately details the methodology employed for data collection, and we have made this valuable dataset publicly available. This data stands as a significant contribution, offering a foundation for future research focused on distributing the processing load of CNNs over grid-shaped sensor networks. Moreover, the fall simulation data provides insights into fall patterns and the dynamics of individual reactions during fall event. Additionally, it lays the groundwork for the development of an intelligent flooring system adept at detecting fall events with high accuracy.

Looking forward, we suggest steering future research towards integrating the convolutional layer processing approach with existing or emerging solutions for distributing the inference load of a fully connected deep neural network among sensor network nodes. Such integration promises to enable complete CNN inference directly within the sensor network, thus eliminating the need for external processing systems. This development

[1]Zenodo repository: `https://doi.org/10.5281/zenodo.5141721`

could significantly enhance privacy and efficiency of the sensor network while reducing the inference response time.

3.2 A General Purpose Privacy Preserving Protocol for Sensor Networks

Despite the maturity of sensor networks, a significant gap persists in the development of a decentralized computing and information retrieval framework that assures privacy and enables generalized computation. This research seeks to address this gap, proposing a novel layered cryptographic approach inspired by Onion Routing [7]. This approach signifies a paradigm shift in privacy-preserving computation by prioritizing the movement of computation over sensitive data. By enabling in-situ processing on nodes without the need to transfer sensitive data, this approach lays the foundation for robust, decentralized, and privacy-preserving in-network data analysis frameworks.

In this thesis, we have articulated and formalized the layered cryptographic approach, transforming it into a privacy-preserving communication protocol for sensor networks. Drawing from this advancement, the principal contributions made within this thesis regarding our privacy-preserving communication protocol are as follows:

- In [10], we have formally defined the privacy-preserving communication protocol that enables sensor network nodes to jointly compute a function in a privacy-preserving manner. We compare our proposal to the related work in the field, showing that, this is the first scheme that allows in-network joint computation of arbitrary functions in sensor networks without aggregator nodes and without disclosing the nodes participating in the computation and their private inputs.
- In [10], we have analytically demonstrated that our communication protocol is secure against traffic analysis attacks. This makes it particularly suitable for scenarios such as indoor monitoring, where data privacy is paramount. Notably, our solution exhibits resilience against both eavesdropping and attacks based on node tampering. For example, when an attacker controls 10% of the network nodes, the probability of a query revealing a private data point of a sensor

node to the attacker is 1/60. By adjusting protocol parameters, the probability of data leakage can be further mitigated.

- Utilizing the ns-3 simulator, we crafted a simulation for the privacy-preserving communication protocol, allowing it to operate within simulated sensor networks of varying sizes, topologies, and operational parameters. Through empirical experimentation, we found that the solution scales effectively. However, for larger sensor networks (400+ nodes), it is advisable to limit the random node selection for queries to a specific network region. This region should encompass the nodes of interest but remain compact enough to not incur in high message propagation times. Furthermore, we noted that the network topology has only a marginal impact on message propagation time. Both the description and architecture of the simulation are presented in [12], and we have made the developed simulation publicly available [2].

- In [9], we have shown how to apply the privacy-preserving communication protocol to realize privacy-preserving decentralized data mining over sensor networks. We have simulated the training of a machine learning model using our proposed communication protocol and compared its inference results with those of various models trained using the traditional centralized approach. We conduct empirical experimentation through a regression task using indoor air quality data, evaluating the models based on several metrics: mean absolute error, root mean squared error, root relative squared error, and correlation coefficient. Our findings indicate that our distributed machine learning model training approach yields results comparable to those obtained with the traditional centralized method.

- In [9], we have shown the viability of a decentralized, privacy-preserving, SPOF-free, and secure in-network data analysis framework. Harnessing the combined strengths of blockchain technology and onion routing, our approach offers enhanced privacy and security, transcending the limitations of centralized systems. The efficacy of the decentralized framework was further validated for data mining tasks through simulations using the ns-3 simulator. Our findings indicate that the framework can operate effectively, even

[2]Github repository: `https://github.com/ElsevierSoftwareX/SOFTX-D-21-00231`

with large machine learning model sizes, with successful testing conducted for models up to 5MB in size.

In the near future, our primary focus will be to take our proposal from theoretical constructs to real-world applications. Specifically, we have charted out plans to implement and test our innovative protocol in various buildings, aiming to collect and analyze air quality data directly on-site. In the course of this development, particular emphasis will be placed on leveraging our technique for the training and evaluation of machine learning models. Within this context, the federated learning paradigm appears to be a promising direction, potentially offering synergies with our decentralized and privacy-centric approach. Furthermore, in our quest for enhanced privacy, we are intrigued by the prospects of integrating advanced cryptographic methods. Specifically, the integration of code obfuscation techniques and homomorphic encryption into tasks performed by sensor nodes can be a game-changer in terms of privacy preservation. This, of course, will necessitate a thorough investigation into the feasibility and potential trade-offs of these sophisticated methods, ensuring that the balance between privacy and performance is maintained.

Alongside, there is growing interest in extending the reach and applicability of our technique within the expansive landscape of the Internet of Things (IoT). We envision the creation of a decentralized resource marketplace, where each individual IoT device, equipped with varying levels of computational power, data storage, sensory and actuators, has the potential to actively participate in this new ecosystem. Whether it is offering stored information, contributing processing capabilities, or sharing sensory data, these devices can be active contributors rather than mere passive entities. By introducing an incentive mechanism, which aligns with the trend of rewarding resource contributions, users can gain for the roles their devices play, thus fostering a vibrant, user-centric ecosystem. In this envisioned model, data and processing power are not just utilities; they are commodities, and the power to harness and trade them rests in the hands of the user. The general-purpose privacy-preserving protocol developed in this thesis could serve as the foundational layer for utilizing IoT devices, while blockchain technology can instill trust among nodes and introduce the necessary incentive mechanism.

Harnessing a decentralized resource marketplace not only optimizes the utilization of users' IoT infrastructure, thereby fostering significant cost savings and rewarding the user, but also unveils a spectrum of applications. It can substitute sensing infrastructures in smart cities and environmental monitoring scenarios, provide a readily available foundation for diverse research initiatives, and much more, marking a pivotal step towards a more interconnected and efficient future.

In conclusion, this thesis has been a rigorous exploration into the realms of sensor networks, addressing both their potential and current limitations. By leveraging novel paradigms such as edge computing, decentralization, and emphasizing the imperative of privacy, we have laid a foundation that future research can build upon. The methodologies and solutions we have developed, particularly our layered cryptographic approach and in-network processing strategies, are not just theoretical constructs; they present tangible avenues for the next wave of advancements in real-world applications. We have demonstrated that with careful design and innovation, sensor networks can be more efficient, decentralized, and privacy-centric. As we look forward, the immediate next steps are clear: transitioning from simulations to real-world applications, refining our approaches based on practical findings, and ensuring our solutions remain relevant and adaptable in the ever-evolving technological landscape.

Bibliography

[1] S. Abbasian Dehkordi, K. Farajzadeh, J. Rezazadeh, R. Farahbakhsh, K. Sandrasegaran, and M. Abbasian Dehkordi. A survey on data aggregation techniques in iot sensor networks. *Wireless Networks*, 26(2):1243–1263, Feb 2020.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.

[3] M. Alloghani, M. M. Alani, D. Al-Jumeily, T. Baker, J. Mustafina, A. Hussain, and A. J. Aljaaf. A systematic review on the status and progress of homomorphic encryption technologies. *Journal of Information Security and Applications*, 48:102362, 2019.

[4] E. De Coninck, T. Verbelen, B. Vankeirsbilck, S. Bohez, P. Simoens, P. Demeester, and B. Dhoedt. Distributed neural networks for internet of things: The big-little approach. In B. Mandler, J. Marquez-Barja, M. E. Mitre Campista, D. Cagáňová, H. Chaouchi, S. Zeadally, M. Badra, S. Giordano, M. Fazio, A. Somov, and R.-L. Vieriu, editors, *Internet of Things. IoT Infrastructures*, pages 484–492, Cham, 2016. Springer International Publishing.

[5] R. Du, S. Magnusson, and C. Fischione. The internet of things as a deep neural network. *IEEE Communications Magazine*, 58(9):20–25, 2020.

[6] A. Giridhar and P. Kumar. Toward a theory of in-network computation in wireless sensor networks. *IEEE Communications Magazine*, 44(4):98–107, 2006.

[7] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding routing information. In *International workshop on information hiding*, pages

137–150. Springer, 1996.

[8] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 14(14):527, 2008.

[9] N. Hrovatin, A. Tošić, M. Mrissa, and B. Kavšek. Privacy-preserving data mining on blockchain-based wsns. *Applied Sciences*, 12(11):5646, 2022.

[10] N. Hrovatin, A. Tošić, M. Mrissa, and J. Vičič. A general purpose data and query privacy preserving protocol for wireless sensor networks. *IEEE Transactions on Information Forensics and Security*, 2023.

[11] N. Hrovatin, A. Tošić, and J. Vičič. In-network convolution in grid shaped sensor networks. *Journal of Web Engineering*, pages 75–96, 2022.

[12] N. Hrovatin, A. Tošić, and J. Vičič. Ppwsim: Privacy preserving wireless sensor network simulator. *SoftwareX*, 18:101067, 2022.

[13] F. Jabeen and S. Nawaz. In-network wireless sensor network query processors: State of the art, challenges and future directions. *Information Fusion*, 25:1–15, 2015.

[14] N. Jarah. Deep learning in wireless sensor network. *Journal of Al-Qadisiyah for computer science and mathematics*, 13(1):Comp Page 11 –, Feb. 2021.

[15] C. P. Mayer. Security and privacy challenges in the internet of things. *Electronic Communications of the EASST*, 17, 2009.

[16] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[17] I. Tomić and J. A. McCann. A survey of potential security issues in existing wireless sensor network protocols. *IEEE Internet of Things Journal*, 4(6):1910–1923, 2017.

[18] A. Tošić, N. Hrovatin, and J. Vičič. Data about fall events and ordinary daily activities from a sensorized smart floor. *Data in brief*, 37:107253, 2021.

[19] A. Tošić, N. Hrovatin, and J. Vičič. A wsn framework for privacy aware indoor location. *Applied Sciences*, 12(6):3204, 2022.

[20] M. S. Yousefpoor, E. Yousefpoor, H. Barati, A. Barati, A. Movaghar, and M. Hosseinzadeh. Secure data aggregation methods and countermeasures against various attacks in wireless sensor networks: A comprehensive review. *Journal of Network and Computer Applications*, 190:103118, 2021.

[21] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y. an Tan. Secure multi-party computation: Theory, practice and applications. *Information Sciences*, 476:357–372, 2019.

Povzetek v slovenskem jeziku

Senzorska omrežja predstavljajo neločljiv del sedanje tehnološke scene. Ključni napredki v brezžični komunikaciji, vgrajeni elektroniki in tehnologiji senzorjev so omogočili razvoj cenovno dostopnih, energijsko učinkovitih in miniaturnih naprav, ki so sposobne zaznavanja, obdelave in prenosa podatkov. V literaturi so takšne majhne, energijsko varčne naprave znane kot senzorska vozlišča.

Senzorska vozlišča razporejena v velikem številu in ki si delijo skupni (ponavadi brezžični, včasih pa tudi žični) komunikacijski medij, tvorijo senzorsko omrežje [2]. V tradicionalni arhitekturi senzorskih omrežij vozlišča opravljajo meritve ali zaznavajo dogodke. Nato zbrane podatke pošljejo do zbiralnega vozlišča (ang. sink node), ki služi kot centralna točka za zbiranje podatkov in običajno podatke posreduje zunanjim sistemom za trajno shranjevanje, obdelavo in analizo.

Danes se senzorska omrežja uporabljajo v širokem naboru aplikacij, od spremljanja okolja in zdravstvenega varstva do industrijske avtomatizacije, senzorska omrežja ne samo zbirajo in analizirajo podatke, temveč tudi revolucionirajo naše dojemanje in interakcijo z okoljem.

Kljub mnogim prednostim, ki jih senzorska omrežja prinašajo obstajajo tudi določene omejitve, ki jih je treba upoštevati [2]. Med glavnimi omejitvami so omejena moč in zmogljivosti obdelave senzorskih vozlišč, ki lahko omeji količino in kompleksnost zbranih, obdelanih in prenesenih podatkov. Poleg tega so senzorska omrežja občutljiva na različne vrste napak, kot so okvare vozlišč, delitve omrežja in prekinitve komunikacije, kar lahko privede do izgube podatkov, zamud in zmanjšane zanesljivosti. Druga pomembna omejitev je možnost kršitve varnosti in zasebnosti, saj senzorska omrežja pogosto zbirajo občutljive informacije in so izpostavljena različnim vrstam napadov [17].

Zgoraj omenjene omejitve so še posebej izrazite v tradicionalni centralizirani arhitekturi senzorskih omrežij, kjer vozlišča zbirajo in prenašajo surove podatke v oddaljen sistem izven senzorskega omrežja za obdelavo in analizo. Tako trenutne implementacije senzorskih omrežij prehajajo k decentralizirani arhitekturi, kjer senzorsko omrežje zaznava, obdeluje in shranjuje podatke ter odgovarja z informacijami namesto s podatki ob povpraševanju [13, 6].

Ta nov decentraliziran pristop spodbujajo napredki v računalniški moči in učinkovitosti mikroelektronike ter pojav robnega računanja [16]. Robno računanje predstavlja distribuiran računalniški model, ki vključuje izvajanje računalniških nalog čim bližje viru podatkov, na primer na "robu" omrežja. V kontekstu senzorskih omrežij obdelava na robu pomeni lokalna obdelavo podatkov na samih senzorskih vozliščih ali na bližnjih napravah. Z izkoriščanjem geografske bližine podatkov obdelava na robu prinaša številne prednosti senzorskim omrežjem.

Prvič, z izvajanjem obdelave podatkov in analize bližje izvoru podatkov se znatno zmanjša količina podatkov, ki jih je treba prenesti čez omrežje. To je še posebej pomembno v primeru omejenih virov, saj poveča učinkovitost omrežja in zmanjša zastoje, kar vodi v bolj učinkovit in zanesljiv sistem. Poleg tega manjše prenašanje podatkov rezultira v nižjo latenco, kar izboljša celotno delovanje in odzivnost sistema.

Drugič, z distribucijo računskih nalog čez več vozlišč, obdelava na robu lahko prepreči problem centralne točke okvare (ang Single Point Of Failure – SPOF) v omrežju. Če ena naprava ali vozlišče odpove, se lahko obdelovalne naloge preusmerijo na drugo napravo ali vozlišče, kar zagotavlja, da omrežje še naprej deluje, s čimer se izboljša splošna odpornost na okvare in zanesljivost sistema.

Tretjič, z lokalno obdelavo podatkov se lahko zagotovi, da občutljive informacije ostanejo lokalizirane in se ne prenašajo čez omrežje, kar poveča varnost in zasebnost uporabnikov.

S uporabo računanja na robu se v senzorskih omrežjih soočamo z različnimi izzivi. Med najbolj izrazite spadajo: odločanje med lokalno in oddaljeno obdelavo podatkov, uravnoteženje obremenitve omrežja, optimizacija komunikacije in obdelave podatkov ter zagotavljanje zanesljivosti in varnosti v distribuiranem okolju [16].

Kljub prednostim robnega računanja za senzorska omrežja so trenutne implementacije ogrodij porazdeljenega računanja omejene. Ni univerzalnega ogrodja za porazdeljeno računanje, ki bi podpiral poljubno računanje in bil varen, učinkovit ter primeren za senzorska omrežja. Obstajajo sicer ogrodja, optimizirana za določene primere uporabe, kot so agregacija podatkov [1], procesor poizvedb [13] ali strojno učenje [5], vendar so ta ogrodja prilagojena specifičnim nalogam in niso primerna za druge tipe izračunov. Poleg tega mnoga ogrodja nimajo vgrajenih potrebnih varnostnih mehanizmov za zaščito občutljivih podatkov v senzorskih omrežjih, medtem ko so ostale rešitve preveč zahtevne za delovanje v senzorskih omrežjih. Posledično obstaja potreba po raziskavah novih rešitev porazdeljenega računanja, ki bi odpravile te omejitve.

Pri razvoju naprednega ogrodja za porazdeljeno obdelavo podatkov v senzorskih omrežjih je ključno postaviti pozornost na varovanje zasebnosti. Zasebnost v senzorskih omrežjih predstavlja velik izziv, saj zaznane okoljske značilnosti odražajo aktivnosti ljudi v nadzorovanem okolju. To pomeni, da so zbrani podatki občutljive narave in zahtevajo ustrezne varnostne ukrepe [15]. Čeprav lahko koncept robnega računanja in selitev računskih nalog čim bližje izvoru podatkov pripomore k zmanjšanju potencialnih varnostnih pomanjkljivosti, je zasebnost še vedno odprt problem, saj morajo nekatere informacije vedno potovati do končnih točk omrežja [17]. Poleg tega lahko izvajanje računskih nalog čim bližje izvoru podatkov na vozliščih senzorskega omrežja ogrozi zasebnost podatkov. Natančneje, v aplikacijah, ki temeljijo na računskih nalogah, ki potrebujejo podatke iz več vozlišč za izračun rezultatov, vozlišča obdelujejo podatke iz drugih vozlišč in zahtevajo dešifriranje pred obdelavo podatkov; zato vozlišča, ki izvajajo izračun, imajo dostop do podatkov iz drugih vozlišč. Pomen te težave je še posebej poudarjen v senzorskih omrežij, kjer so vozlišča običajno po namestitvi nenadzorovana, zaradi česar so izpostavljena možnim manipulacijam s strani zlonamernih akterjev. V skladu s tem bi lahko napadalec manipuliral z majhnim številom senzorskih vozlišč, da bi pridobil podatke iz veliko večjega nabora vozlišč.

V kontekstu senzorskih omrežij je bil ta problem obravnavan z različnimi pristopi. Veliko študij se osredotoča na računsko nalogo agregacije podatkov znotraj omrežja [20]. Čeprav so predlagane rešitve učinkovite in zagotavljajo zasebnost, so računske naloge večinoma omejene na osnovne agregacijske funkcije, kar pomeni, da so te rešitve težko prilagodljive za druge računske naloge. Druge študije se naslanjajo na raziskave Bizantinskih odpornih sistemov ter predstavljajo tehnike, ki temeljijo na homomorfni enkripciji [3] ali varni večstranski obdelavi [21]. Vendar so temeljni protokoli zahtevni z vidika virov in trenutne implementacije, ki podpirajo poljubno računanje, niso primerne za uporabo v senzorskih omrežjih.

Kljub številnim napredkom v kriptografiji ostaja zasebnost ključna skrb v senzorskih omrežjih, zlasti v zvezi z aplikacijami, ki uporabljajo koncept robnega računanja. Obstoječe rešitve ne zagotavljajo učinkovitega, praktičnega in vsestranskega pristopa, ki bi omogočal obdelavo podatkov znotraj omrežja ob ohranjanju zasebnosti. Zato obstaja pomembna raziskovalna vrzel, ki zahteva nadaljnje preučevanje.

Glavni cilj te doktorske naloge je raziskovanje decentraliziranih pristopov za ohranjanje zasebnosti pri obdelavi podatkov znotraj senzorskih omrežij. Raziskava je usmerjena k ogrodju, ki podpira strojno učenje in omogoča, da se modeli strojnega učenja učijo in izvajajo na vozliščih omrežja, medtem ko ohranja zasebnost občutljivih podatkov pred prisluškovanjem in napadi, ki temeljijo na manipulaciji vozlišč.

Poleg tega, tradicionalna centralizirana senzorska omrežja pogosto temeljijo na agregacijskih vozliščih, ta model uvaja ranljivosti. Uporaba agregacijskih vozlišč ustvari SPOF tako z tehničnega kot z vidika zasebnosti. Čeprav je takšna arhitektura skalabilna in učinkovita, je lahko ranljiva za varnostne kršitve ali napake v omrežju, kar lahko privede do pomembnih izgub podatkov in nedelovanja sistema. V tem zaključnem delu se osredotočamo na decentraliziran pristop brez agregacijskih vozlišč, kjer vsa senzorska vozlišča imajo enako vlogo in se obnašajo enako, s čimer se odpravlja vsak SPOF.

V tem zaključnem delu predstavljamo dva inovativna pristopa za porazdeljeno računanje v senzorskih omrežjih, s katerima želimo izkoristiti prednosti robnega računanja in skupne računalniške moči omrežja. Oba pristopa temeljita na decentralizirani arhitekturi, ki omogoča, da senzorska omrežja obvladujejo računsko zahtevne aplikacije, kot je strojno učenje, brez zanašanja na zunanje sisteme. To omogoča obdelavo podatkov bliže njihovemu izvoru, kar zmanjšuje izpostavljenost podatkov. Da bi ocenili prednosti in slabosti obeh metod ter njihovo sposobnost zagotavljanja učinkovite in varne obdelave podatkov, smo izvedli analitične študije in simulacije z uporabo simulatorja ns-3 [8].

Porazdeljeno računanje konvolucije v senzorskem omrežju

Računska obremenitev konvolucijske nevronske mreže (KNM) je lahko zelo visoka, kar onemogoča obdelavo sklepanja na enem vozlišču senzorskega omrežja. Poleg tega uporaba modela KNM potrebuje podatke iz večih senzorskih vozlišč za sklepanje. To privede do koncentracije občutljivih podatkov na centralni točki obdelave, kar lahko ogrozi zasebnost sodelujočih vozlišč. Zato se soočamo s ključnim izzivom: kako razporediti računsko obremenitev KNM med vozlišči za obdelavo v realnem času.

Čeprav obstaja že nekaj rešitev [4, 14] za porazdeljeno obdelavo običajnih umetnih nevronskih mrež na vozliščih senzorskih omrežij, so raziskave na porazdeljeni obdelavi konvolucijskih plasti zelo omejene. KNM so priznane po svoji učinkovitosti pri aplikacijah, ki delujejo s podatki strukturiranimi v obliki mreže, kot sta analiza slik in videa. Kljub temu pa potencial uporabe KNM v kontekstu mrežno postavljenih senzorskih omrežij še ni bil v celoti izkoriščen

V okviru te disertacije smo zasnovali rešitev brez SPOF za porazdeljeno obdelavo konvolucijske plasti KNM čez vozlišča mrežno postavljenega senzorskega omrežja. Naš pristop izkorišča strukturo jeder KNM in lokalno koordinacijo vozlišč za izvajanje sklepanja čim bližje izvoru podatkov. To zmanjšuje potrebo prenosa občutljivih podatkov čez omrežje. Uporabnost te rešitve je široka, saj ima potencial izboljšati tako zasebnost kot tudi učinkovitost v katerikoli aplikaciji mrežno postavljenega senzorskega omrežja, ki lahko koristi KNM, kot so sledenje aktivnosti, analiza hoje, zaznavanje padcev ali različne vrste avtomatizacijskih sistemov.

Splošni protokol za varovanje zasebnosti v senzorskih omrežjih

Zasebnost v senzorskih omrežjih je ključnega pomena, saj so zbrani podatki občutljive narave. Tradicionalne tehnike šifriranja lahko podatke varujejo med prenosom, vendar pa aplikacije, ki obdelujejo podatke iz več vozlišč, še vedno dostopajo do dešifriranih podatkov. Doslej so bile raziskave predvsem usmerjene v zagotavljanje zasebnosti med obdelavo podatkov znotraj omrežja s uporabo tehnike transformacije podatkov preko kriptografije ali drugimi tehnikami zamaskiranja. Ta pristop omogoča več vozliščem delovanje na transformiranih podatkih brez možnosti pridobivanja zasebnih podatkov iz drugih sodelujočih vozlišč. Kljub temu da je ta pristop privedel do razvoja številnih rešitev, v rešitvah, ki so praktične v senzorskih omrežjih, tehnika zakrivanja podatkov omejuje nabor operacij, ki so možne na transformiranih podatkih.

Da bi izboljšali zaščito zasebnosti v senzorskih omrežjih, predlagamo rešitev, ki prenaša računanje namesto občutljivih podatkov z uporabo slojne kriptografije. Naš pristop prenaša navodila za izračun in delne rezultate za obdelavo neposredno na vozliščih senzorskega omrežja, s čimer odpravlja potrebo po prenosu občutljivih podatkov in povečuje zaščito zasebnosti pred prisluškovanjem in manipulacijo vozlišč. Dodatno naš pristop omogoča izvajanje poljubnih izračunov na vozliščih senzorskega omrežja, in tako omogoči strojno učenja. Naša raziskovalna smer vodi k ustvarjanju decentraliziranega ogrodja za analizo podatkov v senzorskih omrežjih, ki je brez SPOF, in ki zagotavlja zasebnost.

Raziskovalni cilji in hipoteze

Splošni raziskovalni cilj te zaključne naloge je preučiti in razviti nove decentralizirane rešitve, ki omogočajo porazdelitev računalniških nalog med vozlišči senzorskih omrežij. S tem želimo izkoristiti prednosti robnega računalništva in skupne procesorske moči omrežja ter omogočiti obdelavo zahtevnih aplikacij, kot je strojno učenje, pri čemer zmanjšamo izpostavljenost občutljivih podatkov in povečamo zasebnost.

RC-1 in H-1

RC-1: Je mogoče izkoristiti mrežno postavitev senzorskega omrežja za porazdelitev računske obremenitve konvolucijskih plasti KNM med vozlišči senzorskega omrežja?

H-1: S uporabo strukture jeder KNM in lokalne koordinacije vozlišč je mogoče konvolucijske plasti KNM obdelovati na porazdelen način čez vozlišča mrežno postavljenega senzorskega omrežja.

Obravnavano v naslednjih prispevkih:

- Data about fall events and ordinary daily activities from a sensorized smart floor
- In-Network Convolution in Grid Shaped Sensor Networks

RC-2 in H-2

RC-2: Ali je mogoče v senzorskih omrežjih izboljšati varovanje zasebnosti z anonimno premestitvijo računskih nalog, namesto prenosa občutljivih podatkov?

H-2: Z izkoriščanjem slojne kriptografije se lahko omogoči anonimno obdelavo podatkov neposredno na vozliščih senzorskega omrežja, kar odpravi potrebo po prenosu občutljivih podatkov in s tem povečuje zaščito zasebnosti pred prisluškovanjem in manipulacijo vozlišč.

Obravnavano v naslednjih prispevkih:

- A General Purpose Data and Query Privacy Preserving Protocol for Wireless Sensor Networks
- PPWSim: Privacy preserving wireless sensor network simulator

RC-3 in H-3

RC-3: Ali je mogoče s strojnim učenjem neposredno na senzorskih vozliščih in z premikom modela čez omrežje ob hkratnem ohranjanju zasebnosti doseči primerljive rezultate s tradicionalnimi centraliziranimi pristopi strojnega učenja?

H-3: Z uporabo strojnega učenja neposredno na senzorskih vozliščih in z premikom modela čez senzorsko omrežje je mogoče ohraniti zasebnost in hkrati doseči rezultate, ki so primerljivi s tistimi, pridobljenimi s tradicionalnimi centraliziranimi pristopi strojnega učenja.

Obravnavano v prispevku: Privacy-Preserving Data Mining on Blockchain-Based WSNs

RC-4 in H-4

RC-4: Ali je mogoče izdelati decentralizirano ogrodje za analizo podatkov znotraj senzorskih omrežij, ki je brez SPOF, varno in varuje zasebnost?

H-4: Z uporabo slojne kriptografije za anonimno prenašanje računskih nalog in izkoriščanje tehnologije blockchain je mogoče izdelati decentralizirano ogrodje za analizo podatkov znotraj senzorskih omrežij, ki je brez SPOF, varno in varuje zasebnost.

Obravnavano v prispevku: Privacy-Preserving Data Mining on Blockchain-Based WSNs

Metode

Porazdeljeno računanje konvolucije v senzorskem omrežju

V študiji predlagamo rešitev brez SPOF za porazdeljeno obdelavo konvolucijske plasti KNM čez vozlišča mrežno postavljenega senzorskega omrežja. Naš pristop izkorišča strukturo jeder KNM in lokalno koordinacijo vozlišč za izvajanje sklepanja čim bližje izvoru podatkov. S tem pristopom zmanjšujemo potrebo po prenosu občutljivih podatkov čez omrežje. Dodatno naša rešitev omogoča obdelavo več zaporednih konvolucijskih plast in je kompatibilna tako z 2D kot tudi 3D konvolucijo. V okviru naše študije smo uporabili senzorsko omrežje v obliki mreže, vgrajeno v tla (pametna tla), da bi zbrali nabor simuliranih padcev. Metodologijo zbiranja podatkov in opis sistema smo podrobno predstavili v objavljenem članku 1, medtem ko smo zbrane podatke javno objavili. Za testiranje naše predlagane rešitve smo uporabili simulator ns-3. Izvedli smo nabor testov z uporabo dveh različnih topologij omrežja, pri čemer smo spreminjali omrežne parametre in velikosti konvolucijskih jeder. Testiranje je bilo osredotočeno predvsem na validacijo uporabe naše rešitve v kontekstu pametnih tal. Predlagana rešitev in rezultati simulacije so objavljeni v članku 2.

Splošni protokol za varovanje zasebnosti v senzorskih omrežjih

V raziskavi predlagamo novo paradigmo porazdeljene obdelave podatkov v senzorskih omrežjih, ki daje prednost varovanju zasebnosti. V nasprotju z obstoječimi pristopi, kjer se občutljivi podatki pogosto prenašajo iz vozlišča v vozlišče, naš pristop temelji na prenosu navodil za izračun in delnih rezultatov ter izvedbo teh na vozlišči senzorskega omrežja. Da bi zagotovili zasebnost, so ta navodila in rezultati zakriti s tehniko slojne kriptografije, ki je podobna mehanizmu Onion Routinga, metodologiji, ki omogoča anonimno komunikacijo v omrežju [7]."

Našo metodo smo implementirali v simulacijsko okolje ns-3. Pri izdelavi simulacijskega ogrodja smo poseben poudarek dali enostavnemu spreminjanju ključnih omrežnih parametrov in razširjivosti. To ogrodje smo zaradi njegove uporabnosti in prilagodljivosti javno objavili. Podrobnejši opis ogrodja in naših izkušenj z njim smo predstavili v članku 3 te disertacije.

V članku 4 smo formalizirali rešitev kot komunikacijski protokol ki ohranja zasebnost za senzorska omrežja. Preko podrobnega pregleda področja smo izpostavili obstoječe pomanjkljivosti in poudarili pomembnost naše rešitve. V prispevku smo analitično dokazali, da rešitev ohranja zasebnost pred prisluškovanjem in napadi, ki slonijo na manipulaciji vozlišč. Ter s uporabo simulacijskega ogrodja izdelanega v članku 3 smo empirično testirali rešitev z več simulacij ter ocenili skalabilnost rešitve.

V članku 5 smo se posvetili ključni prednosti naše rešitve katera je izvajanje poljubnih izračunov na vozliščih senzorskega omrežja. V tem članku smo predstavili različico protokola, ki omogoči decentralizirano strojno učenje. Predstavili smo različico protokola, ki podpira decentralizirano strojno učenje. Validacija effektivnosti tega protokola je bila izvedena s simulacijo decentraliziranega učenja na zbirki podatkov o kakovosti zraka. Naučene modele smo nato primerjali s tistimi, pridobljenimi s tradicionalnim centraliziranim pristopom. Dodatno smo v članku 5 prikazali, da je možno razviti decentralizirano ogrodje za analizo podatkov v senzorskih omrežjih, ki je brez SPOF, varno in zagotavlja zasebnost. Za validacijo predlagane rešitve smo uporabili simulacijsko ogrodje, ki smo ga izdelali v članku 3, in ga preizkusili pri procesu podatkovnega rudarjenja.

Zaključek

Senzorska omrežja so postala vsepovsod prisotna zaradi napredka v brezžični komunikaciji, vgrajeni elektroniki in senzorski tehnologiji. Kljub temu tradicionalna centralizirana arhitektura senzorskih omrežij prinaša številne izzive, zlasti na področjih energije, procesorskih zmogljivosti, zasebnosti in varnosti. To je privedlo do prehoda k decentraliziranim arhitekturam in uporabe računalništva na robu v senzorskih omrežjih. V disertaciji smo se poglobili v zasebnost in obdelavi podatkov znotraj senzorskih omrežij, s poudarkom na obdelavi podatkov na mestu in izkoriščanju prednosti računalništva na robu.

Naša raziskava o porazdelitviji računske obremenitve konvolucijskih plasti KNM med vozlišči v senzorskih omrežjih je prinesla obetavno rešitev. Z uporabo simulacij na osnovi ns-3 simulatorja smo ugotovili, da komunikacijski zahtevnost rešitve predstavljaja izziv le v ekstremu kombinacije velikih konvolucijskih jeder in povezav z nizko prenosno kapaciteto. V tej študiji smo zbrali tudi velik nabor simuliranih padcev z uporabo mrežno oblikovanega senzorskega omrežja. Naši rezultati predstavljajo dragocen prispevek, ki služi kot temelj za nadaljnje raziskave.

V doktorski disertaciji smo razvili nov pristop na osnovi večplastne kriptografije za distribuirano obdelavo podatkov z ohranjanjem zasebnosti v senzorskih omrežjih. Omogočili smo obdelavo na vozliščih, ne da bi prenašali občutljive podatke, s čimer smo odprli pot robustnim, decentraliziranim in zasebnim ogrodjem za analizo podatkov. Na podlagi tega napredka so glavni prispevki te disertacije v zvezi z našim komunikacijskim protokolom, ki ohranja zasebnost, naslednji:

- Formalno smo definirali komunikacijski protokol, ki ohranja zasebnost, in omogoča vozliščem v senzorskih omrežjih skupno računanje brez razkrivanja sodelujočih vozlišč in njihovih zasebnih vnosov. Naš predlog smo primerjali z obstoječimi deli in pokazali, da gre za prvo shemo, ki omogoča tovrstno skupno izvajanje funkcij brez agregacijskih vozlišč v senzorskih omrežjih.
- Analitično smo dokazali, da je naš komunikacijski protokol varen pred napadi analize prometa. Naša rešitev je odporna tako na prisluškovanje kot na napade, ki temeljijo na manipulaciji vozlišč.
- Z uporabo simulatorja ns-3 smo izdelali simulacijsko ogrodje za protokol, ki omogoča izvajati protokol v simuliranih senzorskih omrežjih različnih velikosti in topologij. S pomočjo simulacij smo analizirali skalabilnost protokola in ugotovili smo, da rešitev učinkovito deluje tudi v velikih omrežjih.
- Pokazali smo, kako uporabiti protokol za ohranjanje zasebnosti pri realizaciji decentraliziranega podatkovnega rudarjenja v senzorskih omrežjih. Simulirali smo učenje modela podatkovnega rudarjenja preko našega protokola in primerjali rezultate z modeli, učeni s tradicionalnim centraliziranim pristopom. Ugotovitve kažejo, da naš pristop doseže primerljive rezultate s tradicionalnim pristopom.
- Pokazali smo, da je možno ustvariti decentralizirano ogrodje za analizo podatkov v senzorskem omrežju, brez SPOF, ki je varno in ohranja zasebnost. Predstavljeno ogrodje sloni na integraciji blockchain tehnologije z našim komunikacijskim protokolom, ki ohranja zasebnost. Validirali smo našo rešitev pri procesu podatkovnega rudarjenja z uporabo simulatorja ns-3. Simulacije so pokazale, da naša predlagana arhitektura lahko učinkovito deluje tudi pri velikih modelih strojnega učenja.

V prihodnosti nameravamo naše teoretične koncepte prenesti v praktične aplikacije. Naš glavni cilj je implementacija in testiranje našega protokola v različnih zgradbah za zbiranje in analizo podatkov o kakovosti zraka. Prav tako nas zanima razširitev naše tehnike v okviru interneta stvari z možnostjo integracije naprednih kriptografskih metod, kot so tehnike zameglitve kode in homomorfno šifriranje, kar bi pripomoglo k še bolj efektivnem varovanju zasebnosti.

Declaration

I declare that this doctoral dissertation does not contain any materials previously published or written by another person except where due reference is made in the text.

Niki Hrovatin