6-1-2020

Integrated Framework for Data Quality and Security Evaluation on Mobile Devices

Igor Khokhlov
ixk8996@rit.edu

Integrated Framework for Data Quality and Security Evaluation on Mobile Devices

by

Igor Khokhlov

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in Computing and Information Sciences

B. Thomas Golisano College of Computing and
Information Sciences

Rochester Institute of Technology
Rochester, New York
June 1, 2020

Integrated Framework for Data Quality and Security Evaluation on Mobile Devices

by

Igor Khokhlov

Committee Approval:

We, the undersigned committee members, certify that we have advised and/or supervised the candidate on the work described in this dissertation. We further certify that we have reviewed the dissertation manuscript and approve it in partial fulfillment of the requirements of the degree of Doctor of Philosophy in Computing and Information Sciences.

Dr. Leon Reznik Dissertation Advisor	Date
Dr. Justin Cappos Dissertation Committee Member	Date
Dr. Mehdi Mirakhorli Dissertation Committee Member	Date
Dr. Alexander G. Ororbia II Dissertation Committee Member	Date
Dr. Stanisław Radziszowski Dissertation Committee Member	Date
Dr. Sergey Lyshevski Dissertation Defense Chairperson	Date

Certified by:

Dr. Pengcheng Shi Ph.D. Program Director, Computing and Information Sciences	Date

Integrated Framework for Data Quality and Security Evaluation on Mobile Devices

by
Igor Khokhlov

Submitted to the
B. Thomas Golisano College of Computing and Information Sciences Ph.D. Program in Computing and Information Sciences
in partial fulfillment of the requirements for the
Doctor of Philosophy Degree
at the Rochester Institute of Technology

Abstract

Data quality (DQ) is an important concept that is used in the design and employment of information, data management, decision making, and engineering systems with multiple applications already available for solving specific problems. Unfortunately, conventional approaches to DQ evaluation commonly do not pay enough attention or even ignore the security and privacy of the evaluated data. In this research, we develop a framework for the DQ evaluation of the sensor originated data acquired from smartphones, that incorporates security and privacy aspects into the DQ evaluation pipeline. The framework provides support for selecting the DQ metrics and implementing their calculus by integrating diverse sensor data quality and security metrics. The framework employs a knowledge graph to facilitate its adaptation in new applications development and enables knowledge accumulation. Privacy aspects evaluation is demonstrated by the detection of novel and sophisticated attacks on data privacy on the example of colluded applications attack recognition. We develop multiple calculi for DQ and security evaluation, such as a hierarchical fuzzy rules expert system, neural networks, and an algebraic function. Case studies that demonstrate the framework's performance in solving real-life tasks are presented, and the achieved results are analyzed. These case studies confirm the framework's capability of performing comprehensive DQ evaluations. The framework development resulted in producing multiple products, and tools such as datasets and Android OS applications. The datasets include the knowledge base of sensors embedded in modern mobile devices and their quality analysis, technological signals recordings of smartphones during the normal usage, and attacks on users' privacy. These datasets are made available for public use and can be used for future research in the field of data quality and security. We also released under an open-source license a set of Android OS tools that can be used for data quality and security evaluation.

Acknowledgments

First of all, I want to express my sincere appreciation to my advisor Dr. Leon Reznik, for the constant support in my Ph.D. pursuit and everyday life, for his patience, motivation, and colossal knowledge. I am extremely grateful for his guidance that helped me in my research and this dissertation work. I could not wish to have a better advisor and mentor in my Ph.D. study and life.

Besides my advisor, I would like to thank the Ph.D. program director and the rest of my dissertation committee members: Dr. Pengcheng Shi, Dr. Stanisław Radziszowski, Dr. Alexander G. Ororbia II, Dr. Justin Cappos, Dr. Mehdi Mirakhorli, and Dr. Sergey Lyshevski for their good moral, insightful comments and encouragement. I also want to thank all faculty members of the Golisano College of Information and Computer Sciences at RIT who I interacted with and who helped me to extend my knowledge. I want to express my appreciation to all students I worked with and who helped me to conduct experiments, and collect and analyze data: Michael Perez, Sahil Ajmera, Sergey Chuprov, Akshay Renavikar, Qiaoran Li, Arpit Vora, Joe Tom Job, Rohit Bhaskar, and many others. Without their collaboration, I would not be able to achieve all these results presented in this dissertation.

I dedicate this dissertation to my loving wife Anna, my parents, and my brother who supported and encouraged me throughout my study, research, and my life in general.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Data Quality Concept and Its Importance

The modern world has entered the era when we have to deal with previously unseen amounts of data. The enormous volumes of data have enabled developing data powered systems and their applications in the decision making processes. In addition, all these data are produced by sources of high diversity including crowd-sourcing. Modern smartphones, with the help of various built-in sensors [73], generate all kinds of sensor originated data, social-media users produce all sorts of non-structured data, Internet of Things (IoT) devices deliver sensor data [61], etc. For the first time in humankind's history, we might get more data than we are able to process, manage, and utilize effectively. For example, every minute YouTube users upload more than 300 hours of HD quality [89] to contribute to an already massive collection of 1,300,000,000 videos, a Boeing 737 jet engines' sensors produce ten terabytes of data every 30 minutes [111].

Various tools, data processing, and transmission equipment that are built-in into modern smartphones and other mobile devices, as well as their availability, lead us to an even more significant data generation. A broad spectrum of embedded sensors in various mobile sensor platforms and data collected using these sensors appeal to data scientists, researchers, and engineers [67]. Data collected from these sources can be used in embedded and third-party applications. For instance, a step-counter uses an accelerometer and a gyroscope, and a weather application utilizes a built-in temperature sensor, pressure sensor, and location sensors. In 2014, approximately 38% of all North-American software developers employed sensors and used sensor-originated data in their software [123]. Semiconductor Industry Association states that "Big Data exceeds the reach of commonly used hardware environments and software tools to capture, manage and process it within a tolerable elapsed time for its user population" [117]. The increasing number of crowd-sourcing applications [45, 98], mobile crowdsensing tasks [44, 74, 75, 77, 134, 146], and the Internet of Things also contribute to the unprecedented data volumes aggregation. Unfortunately, not all data generators are equal, and the quality of data produced by them may vary drastically, which significantly impacts the quality of this data processing results.

The absence of adequate sensor data quality evaluation may lead to incorrect choices and sometimes results in very undesirable consequences. For example, in October 2018, the Lion Air Flight 610 crashed and killed all 189 passengers and crew. Later, in March 2019, the same Boeing 737 Max 8 model operated by Ethiopian Airlines (Flight 302) crashed, killing 157 passengers and crew. Boeing reported that both crashes were caused because of the Maneuvering Characteristics Augmentation System (MCAS) was triggered by the data coming from a faulty sensor [1]. Crashes could have been avoided if MCAS or other onboard systems had recognized poor data quality. While Boeing attempts to release a quick fix of the issue, a system approach to an integral data quality evaluation may be a more appropriate solution. In March 2009, an Emirates Airbus passenger jet with 257 passengers aboard came perilously close to crashing at Melbourne Airport, Australia, and had the tail section severely damaged because the wrong numbers were entered into an aircraft computer [96]. It is not clear if the accident could be qualified as a malicious attack. Even though if it was a genuine mistake, it could be considered as a stress test that a jet's sensor and data acquisition system failed, resulting in a potential disaster. The examples mentioned above indicate the extreme importance of the DQ indicators in modern data-driven applications.

Data Quality (DQ) concept has been studied for more than a decade [126, 136, 137], and various definitions of DQ have been formulated. The majority of the DQ definitions include "fitness to the user needs" and/or the application context. For example, quality data can be defined as the data that fulfills user's needs or satisfies particular requirements [40, 41, 59]. Strong et al. define high-quality data as "data that is fit for use by data consumers" [126]. Unfortunately, researches commonly do not include or consider security aspects of the data source from which the data is acquired. This indifference to data security in the DQ concept motivated us to investigate whether we can improve DQ evaluation by incorporating security measures into the DQ evaluation procedures.

1.2 Research Goal and Major Research Tasks

The **primary goal of this research** is to develop a data quality and security evaluation framework that integrates security and privacy aspects into a conventional data quality evaluation pipeline. The framework includes the data quality and security metrics and methods of their integration. It also incorporates a set of tools that implement these methods taking into account the data consumers needs in the form of the application context. These tools are meant to be implemented on mobile devices.

On our way to achieving the goal, we have, first of all, to analyze and classify the existing DQ metrics, methods of their design and integration, types of DQ as well as methods of DQ evaluation. We have to accomplish the following tasks:

1. Analyze existing methods of the DQ evaluation and identify ways of their improvement.
2. Develop methods for the security and privacy aspects integration into the overall DQ evaluation pipeline.
3. Develop security and privacy evaluation methods that are suitable for integration into the

DQ evaluation framework.

4. Implement the integral DQ evaluation framework on mobile devices.

5. Validate framework in solving important real-life problems.

While we are building the framework, we have to address the following questions: What has been done in the domain of DQ evaluation? In particular:

- How is the DQ being evaluated currently?
- What needs to be improved and modified?
- What metric types and groups have been researched for DQ evaluation?
- How to choose and design DQ metrics?
- What is the role of the security and privacy aspects in the conventional DQ evaluation?
- What has been done in the area of security evaluation that is applicable in the conventional DQ evaluation pipeline?

To answer these questions, we analyze leading research on the DQ concept and techniques of its evaluation. We classify DQ types and groups of metrics that are used for the DQ evaluation. We investigate the security representation in the DQ evaluation as well as the DQ representation in the data security evaluation. We examine existing methods of the data security evaluation and approaches to its inclusion into the overall DQ evaluation pipeline. Finally, we present our review of the privacy evaluation components of the data security evaluation. We investigate sophisticated novel threats to users' privacy and existing ways of their detection.

The next task on the way to this research goal is to develop methods for the security and privacy integration into the overall DQ evaluation processes. To solve this task, we have to answer **the second research question**: How to integrate DQ evaluation into a synergetic DQ and security evaluation framework? We dissect this question into the following sub-questions: Are there any ways to connect DQ and security metrics, their integration methods, their calculus implementation, and user context? How to design the DQ evaluation framework in such a way that it can be easily adjusted and expanded for new applications? Can a knowledge graph concept be used to unite all framework components? How can we enable a knowledge accumulation in the DQ evaluation framework?

We answer these questions in chapter 3. We explore the possibility of employing a knowledge graph as a foundation of the proposed DQ evaluation framework. Since a knowledge graph does not have a fixed structure and is defined through its vocabulary, we explore existing vocabularies that are related to the DQ evaluation and develop our vocabulary. We investigate what vocabularies have been already developed and ways of their adoption for DQ and security evaluation purposes. We demonstrate how the knowledge graph may connect all framework components using the developed vocabulary. Finally, we render the knowledge accumulation through the knowledge graph extension and employment.

The next task on the way to the goal is to develop a novel security and privacy evaluation methods that will become a significant part of the framework, including metrics selection and techniques of their integration. To solve this task, we have to answer **the third research question**: How to develop flexible and extendable techniques of the data security evaluation that can be included in the framework? We divide this question into the following sub-questions:

- How to calculate each component of data security, such as data availability, integrity, and confidentiality?
- How to integrate these components into the overall, comprehensive data security score?
- How to choose data security metrics for a particular data source?
- What are the methods of data security metrics integration?
- How to evaluate users' privacy?

We answer these questions in chapter 4. To facilitate an extension of the security evaluation component and its easy adjustment, we investigate hierarchical metric design. We investigate the mutual influence of the security metrics from different security components. Since we chose Android OS smartphones as the initial data source, we develop an introductory set of security metrics for this platform. We develop various calculi for the security metrics integration ranging from conventional algebraic methods to new artificial intelligence (AI) and machine learning (ML) techniques.

We investigate methods of threats to a user's privacy detection on the example of a "colluded applications" attack. To investigate this attack, we conducted an empirical study and examined ways of this attack detection. We developed various machine-learning classifiers capable of this attack detection.

The fourth important task is an implementation of the integral DQ evaluation framework on mobile devices and its validation in various application domains. This task is solved by answering **the fourth research question**: How to develop effective and efficient framework implementation on a mobile platform, such as Android OS-based smartphones? We answer this question in the first part of chapter 5.

The final task is to validate the framework performance on important practical problems. We solve this task in the second part of chapter 5 by answering the following questions:

- How to use the developed framework in real-life tasks?
- How does the security component influence the overall security evaluation?

In order to answer these questions, we develop an expert system that implements DQ metrics integration. We develop an efficient implementation of this expert system for mobile devices based on neural networks. To demonstrate the framework employment in real applications, we present several use cases. These use cases demonstrate framework extension to new tasks through the knowledge graph vocabulary augmentation and render knowledge accumulation.

Overall in this research, the following novel and original results have been produced:

- We developed novel techniques and tools that facilitate security and privacy aspects integration into the conventional DQ evaluation pipeline, which significantly improved the DQ evaluation procedures by including the security and privacy aspects into the evaluation methods.

- We advanced methods and developed tools for the knowledge graph employment in security and privacy integration into DQ. The knowledge graph employment facilitates the framework adoption by new applications with little or no modifications.

- We produced data source security evaluation based on the device metrics that are accessible without device modifications. Our procedures refine DQ evaluation usability on standard mobile devices, such as smartphones. Their application does not require device modification and can be easily employed by untrained smartphone users.

- We validated our framework by its implementation on resource-constraint mobile devices with multiple use cases of diverse important real-life problems.

- To address the DQ evaluation expert system implementation challenge in the resource-constrained environment, we created efficient methods of the expert system realization on mobile devices with machine learning techniques. We developed techniques that facilitate converting an expert system to a neural network model that can be executed on mobile devices.

- We addressed privacy aspect in DQ evaluation by applying machine learning techniques to detect novel and sophisticated threats to a user's privacy. We conducted an empirical study that proves a novel attack detection feasibility by monitoring smartphone system resources. We developed attack detector models that can be executed on modern Android OS-based smartphones without their firmware modification.

Our research and framework development has resulted in a set of various products, including peer-reviewed publications in journals and conference proceedings, knowledge and data bases, and Android OS applications. We developed three datasets that are made available for public use and can be used in further research on data quality evaluation and security improvement. The first dataset contains data about the quality of sensors that are built-in modern mobile devices and their analysis, the second dataset includes data of smartphones' technological signals during the normal phone usage and during attacks on user privacy, and the third dataset contains data about Android smartphones, their general characteristics, sensor information, and security-related metrics. More details on these datasets are presented in section 4.4 and section 4.5. In addition, we developed and published a number of Android OS tools that are used in overall data quality estimation and Android OS smartphone security evaluation. More details on these applications are provided in section 5.10.

1.3 List of Products Developed in This Study

1.3.1 Peer-reviewed Publications

1. Khokhlov I., Reznik L., Chuprov S., "Framework for Integral Data Quality and Security Evaluation in Smartphones" in IEEE Systems Journal, DOI: 10.1109/JSYST.2020.2985343, 2020

2. Khokhlov I., Reznik L., Ajmera S., "Sensors in Mobile Devices Knowledge Base," in IEEE Sensors Letters Journal, Volume 4 , Issue 3 (pp.1-4), 2020.

3. Alrubaye H., Mkaouer M.W., Khokhlov I., Reznik L., Ouni A., Mcgoff J., "Learning to Recommend Third-Party Library Migration Opportunities at the API Level", in Applied Soft Computing Journal (2020), 106140

4. Khokhlov I., Ligade N., Reznik L., "Recurrent Neural Networks for Colluded Applications Attack Detection in Android OS Devices", IEEE World Congress on Computational Intelligence

(WCCI) 2020, Glasgow, UK, July 2020. Accepted for publication.

5. Khokhlov I., Reznik L., Lyshevski S., “Adaptive Data Fusion in Inertial Sensors and Data Quality of Sensor Networks”, 2020 IEEE 40th International Conference on Electronics and Nanotechnology (ELNANO), Kyiv, Ukraine, April 2020. DOI: 10.1109/ELNANO50318.2020

6. Khokhlov I., Reznik L., “Knowledge Graph in Data Quality Evaluation for IoT Applications”, 2020 IEEE World Forum on Internet of Things, New Orleans, Louisiana, USA, April 2020. **Received Outstanding Student Paper award**

7. Chuprov S., Marinenkov E., Viksnin I., Reznik L., Khokhlov I., “Image Processing in Autonomous Vehicle Model Positioning and Movement Control”, 2020 IEEE World Forum on Internet of Things, New Orleans, Louisiana, USA, April 2020. Accepted for publication.

8. Khokhlov I., Reznik L., “What is the Value of Data Value in Practical Security Applications”, IEEE/NDIA/INCOSE Systems Security Symposium 2020, Crystal City, Virginia, USA, April 2020. Accepted for publication.

9. Chuprov S., Viksnin I., Kim I., Reznik L., Khokhlov I., “Reputation and Trust Models with Data Quality Metrics for Improving Autonomous Vehicles Traffic Security and Safety”, IEEE/NDIA/INCOSE Systems Security Symposium 2020, Crystal City, Virginia, USA, April 2020 Accepted for publication.

10. Khokhlov I., Perez M., Reznik L., “Machine Learning in Anomaly Detection: Example of Colluded Applications Attack in Android Devices”, 18th IEEE International Conference on Machine Learning and Applications - ICMLA 2019, Boca Raton, FL, USA, December 2019 (pp. 1328-1333).

11. Khokhlov I., Reznik L., Bhaskar R., “The Machine Learning Models For Activity Recognition Applications With Wearable Sensors”, 18th IEEE International Conference on Machine Learning and Applications - ICMLA 2019, Boca Raton, FL, USA, December 2019 (pp. 387-391).

12. Khokhlov I., Perez M., Reznik L., “System Signals Monitoring And Processing For Colluded Application Attacks Detection In Android OS”, 2019 Western New York Image and Signal Processing Workshop, Rochester, NY, USA, October 2019 (pp. 1-5).

13. Khokhlov I., Pudage A., Reznik L., “Sensor Selection Optimization with Genetic Algorithms”, IEEE SENSORS 2019, Montreal, Canada, October 2019 (pp. 1-4).

14. Khokhlov I., Li Q., Reznik L., “D.I.F.E.N.S.E.: Distributed Intelligent Framework for Expendable Android Security Evaluation”, in “The 14th Annual Symposium on Information Assurance (ASIA ’19)”, Albany, NY, USA, June 2019 (pp. 18-27).

15. Khokhlov I., Jain C., Miller-Jacobson B., Heyman A., Reznik L., St.Jacques R., “MeetCI: A Computational Intelligence Software Design Automation Framework”. In IEEE World Congress on Computational Intelligence, Rio de Janeiro, Brazil, July 2018. (pp. 1499-1506). IEEE.

16. Killawala A., Khokhlov I., Reznik L., “Computational Intelligence Framework for Automatic Quiz Question Generation”. In IEEE World Congress on Computational Intelligence, Rio de Janeiro, Brazil, 2018. (pp. 76-83). IEEE.

17. Khokhlov I., Reznik L., Cappos J. and Bhaskar R., “Design of activity recognition systems with wearable sensors”. In Sensors Applications Symposium (SAS), South Korea, Seoul, 2018 IEEE (pp. 1-6). IEEE.

18. Vora A., Reznik L. Khokhlov I., “Mobile road pothole classification and reporting with data quality estimates”. In Mobile and Secure Services (MobiSecServ), 2018 Fourth International Conference on (pp. 1-6), Miami Beach, FL, March 2018 IEEE.

19. Khokhlov I., Reznik L.,“Android System Security Evaluation”. Demonstration. IEEE Consumer Communications & Networking Conference, Las-Vegas, NV, January 2018 (pp. 1-2).

20. Khokhlov I., Reznik L., Jothilingam S.B., “What Can Data Analysis Recommend on Design of Wearable Sensors?”. IEEE Consumer Communications & Networking Conference, Las-Vegas,

NV, January 2018 (pp. 1-2).

21. Khokhlov I., Reznik L., “Colluded Applications Vulnerabilities in Android Devices”. The 15th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2017), Orlando, FL, November 2017 (pp. 462-469).

22. Khokhlov I., Reznik L., Kumar A., Mookherjee A. and Dalvi R., “Data Security and Quality Evaluation Framework: Implementation Empirical Study on Android Devices.” In IEEE Information Security and Protection of Information Technologies Conference, St. Petersburg, April 2017 (pp. 161-168).

23. Khokhlov I., Reznik L., “Data Security Evaluation for Mobile Android Devices.” In IEEE Information Security and Protection of Information Technologies Conference, St. Petersburg, April 2017 (pp. 154-160).

1.3.2 Android OS Applications on Google Play

Below are Android OS applications that were developed during this research. This application are published on Google Play Store:

1. System Security Evaluation app based on a security evaluation library - https://play.google.com/store/apps/details?id=com.igorkh.trustcheck.securitycheck

2. System Security Evaluation app based on a security evaluation library with the cloud support - https://play.google.com /store/apps/ details?id= com.igorkh.trustcheck. securitycheckcloud

3. Detector of Unverified Apps - https://play.google.com/store/apps/details?id=dataqualitylab.rit.ver_app_finder

4. Smartphone Data Collection Tool - https://play.google.com/store/apps/details?id=com.dataqualitylab.collectinfo. collectinfo

5. Sensor Quality Assessment -
   https://play.google.com/store/apps/details?id=com.dataqualitylab.sensorquality
6. Road Pothole Reporter -
   https://play.google.com/store/apps/details?id=sdh.application.reportthepotholes

In addition, we developed the application that implements the NN calculus for security evaluation component. The source code of this application is made public available as well: https://drive.google.com/ file/d/ 1jwii74qSpOPMHb4EuEwTtydSiF8Bf6mW/ view?usp=sharing

1.3.3 Produced Datasets

This research also resulted in a set of datasets:

- Sensor dataset: https://drive.google.com/drive/folders/ 1vnwpp3hzmbOVf9exXVYLtN38UsT7KEIU?usp=sharing
- "Colluded applications" dataset: https://drive.google.com/drive/folders/ 1YkiiDT4NlqVuBAICGpGtYiXPMggLsSgq?usp=sharing
- Dataset that includes smartphone information for the overall DQ evaluation and security related information: https://drive.google.com/drive/folders/ 1mU_VIgYvbpm7keSldnWvvOoLFA22Y-cW?usp=sharing

Chapter 2

Research Background Review and Analysis

Typical data quality evaluation pipeline includes such steps as the selection of appropriate metrics that represent DQ and the choice of the methods of these metrics integration that fit users' needs. This chapter answers the first research question: *What has been done in the domain of data quality and security evaluation and how we can improve it?* In this chapter, we review, analyze, and classify approaches to metrics selection and design. We present their classification and ways of their organization. Also, we cover existing research in security and especially privacy evaluation. Despite some data scientists use term "information" as a product of data processing, in this research, we follow up the DQ research mainstream and use information as a synonym of data [102], unless specified otherwise. We refer to data and data source characteristics as DQ metrics. The term "DQ dimensionality", that is used below, is defined as the number of DQ metrics employed for the DQ evaluation.

2.1 Review of Existing Approaches to Metrics Design and Selection

The first question we answer in this section is "*How to choose and design DQ metrics?*". On our way to developing the DQ and security evaluation framework, we have to choose an approach to metrics' selection, the design of metrics' organization, and data object types. In this research, we employ theoretical metrics design selection. To facilitate users' needs consideration, we chose a hierarchical metrics' organization. Finally, we focus on structured sensor originated data acquired mostly from a wide sensor's range. In this subsection, we present the overview of existing methods and justification of our choices.

We classify approaches to DQ metrics design into three major groups: heuristic, theoretical, and empirical. The majority of researchers employ the heuristic approach in DQ metrics selection. This approach relies on researchers' expertise and experience in a particular domain. Unfortunately, this approach to metrics design very often results in a small number of DQ metrics, with the "accuracy" metric as one of the major or even only one metric that is used in DQ evaluation. For example, Tan et al. [128] use approach in target tracking tasks with only three metrics that are highly specific to the application. Another example can be found in Prabha et al. [103] research. They consider DQ only from the trustworthiness point of view, assuming that trustworthiness and privacy are highly correlated. The authors evaluate DQ considering only four DQ metrics that are specific to wireless sensor networks, such as identity, route, location, and data privacy. Their framework aims at improving DQ by establishing a secured route from a sensor to a sink.

In the theoretical approach, researchers consider ways on how data may become imperfect during data production or acquisition. In this approach, DQ metrics are defined as inconsistencies between the world's view that is restored from measurements and the view that is obtained through the direct observation of the real-world system.

In the empirical approach to metrics design and selection, researchers analyze data collected from

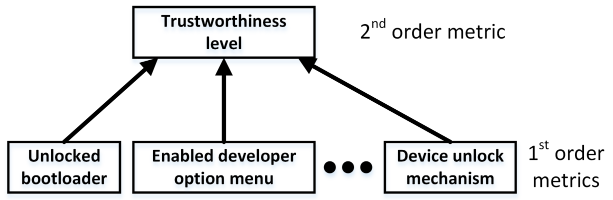


Figure 2.1: Example of metrics of first and second order

the data consumers' point of view and consider those metrics that are important to users. This approach to metrics selection may make it difficult to adapt DQ evaluation processes to new applications. In the empirical approach, metrics are often ranked based on various surveys.

In this research, we employ mostly the theoretical approach to metrics selection and design. While this approach allows taking into consideration a large number of various metrics, it creates a challenge in an efficient metrics organization.

The significant number of DQ dimensions calls for a sustainable metrics organization design. In this research, we chose a hierarchical metrics organization. Hierarchical metrics architecture is one of the approaches to improving the metrics' maintenance. In the hierarchical metrics organization, metrics may have various orders. Metrics of a higher order are based on the metrics of the lower order of the same branch (see equation 2.1 and figure 2.1).

$$M_{n,j} = f(M_{n-1,1}, ..., M_{n-1,k}) \tag{2.1}$$

where $M_{i,j}$ is a metric of order n and $M_{n-1,k}$ are metrics of lower order $n-1$.

Cai and Zhu [21] propose hierarchical DQ standard from a user's perspective: indicators -¿ DQ elements -¿ DQ dimensions. Each metric (dimension) has its elements, and each element has its quality indicators. The authors developed a two-layer data quality standard that includes five higher-order dimensions: Availability, Usability, Reliability, Relevance, and Presentation Quality. Each dimension consists of up to five data quality metrics.

An important factor in the metrics' design is the type of data that is assessed. We classify data types into three groups: structured data, semi-structured data, or unstructured data. In this research, we mostly focus on structured sensor-originated data. The data quality assessment of semi-structured and unstructured data adds additional challenges to the evaluation processes and falls out the scope of this research. However, to make the review of DQ metrics selection and design complete, we present a few approaches to metrics' design and selection further.

A huge amount of data that is freely available is unstructured or semi-structured. A significant portion of this data is collected through crowdsourcing applications. Restuccia et al. [110] presented a survey and rendered challenges in data quality assessment for citizen-science applications. This survey demonstrates that in the crowdsensing applications, the trustworthiness is the major and sometimes the only metric of the data quality.

In order to improve methods of DQ evaluation for these data types, new data parameters may be applied. For example, Immonen et al. [57] focus on DQ evaluation of the data obtained from social media sources. The authors distinguish two concepts of DQ characteristics: DQ attribute and DQ metric. DQ attribute represents a single aspect of the data, while the DQ metric measures a particular property of the DQ attribute. Authors present eleven DQ attributes, where each attribute includes several DQ metrics. Despite that this approach is similar to the one that is used by Cai and Zhu [21], Immonen et al. assign a set of properties to each metric: description, purpose, target, applicability, formula, value range, acceptable value, and rule. Also, according to the authors, each data object can have a different set of DQ attributes.

Big data phenomenon also adds new challenges to the DQ evaluation. For example, a data consumer very often acquires data from many various and heterogeneous data providers through a chain of data proxies. This indirect relation between data consumers and data providers creates obstacles on the way to DQ assessment. Two aspects influence the set of metrics that are used for the DQ evaluation: the application context and the data object type, which may result in additional layers in the DQ evaluation model. For instance, Klas et al. [80] present the DQ model with several layers and consider data quality in the context of a user's application. Their model is based on the hierarchy of such concepts as quality aspects, quality factors, and quality measures. The DQ model structure is similar to the DQ metrics design in Cai and Zhu [21] research.

2.2 Classification of Existing Metric Groups and Data Quality Types

This sections answers the following two questions:

- *What metric types and groups have been researched for DQ evaluation?*
- *How is the DQ being evaluated currently?*

In this section, we present DQ types, their metrics, and DQ evaluation methods. We also reveal insufficient attention to security aspects in the DQ evaluation processes.

2.2.1 Data Quality Metric Groups

Data scientists consider DQ from various points of view. As we mentioned earlier, the overall DQ concept is tightly connected with the data consumers' application context. However, the overall DQ concept may be dissected in DQ "sub-types". It allows us to classify DQ research into a few groups. Further, we show two major DQ sub-types: application-independent and application-dependent. Each DQ sub-type includes a corresponding set of DQ metrics. In this research, we consider two major groups of metrics: objective and subjective. Objective DQ metrics reflect those data characteristics that are independent of the application context. Subjective metrics, on the other hand, tightly related to user needs. Further, we show various approaches to the DQ metrics group classification.

At first glance, it may seem that all metrics are related to the application context. However, some of the metrics do not depend on the user's needs. For example, Hermans et al. [54] chose only four DQ metrics: accuracy, completeness, timeliness, and consistency. If we take a closer look at these metrics, we can see that consistency indeed depends on the application context. However, accuracy, completeness, and timeliness metrics have to be application-independent. Accuracy belongs to data source characteristics, timeliness is related to a communication channel, and completeness characterizes a data stream. Accuracy, for example, is defined by the data source sensors and their hardware and does not change if the user's needs changed. But the weight of accuracy metric directly related to the user's needs. The same idea applies to completeness and timeliness metrics.

Strong et al. identify quality data in the following way: "Data that is fit for use by data consumers" [126]. The authors classify DQ metrics into four categories: intrinsic DQ, accessibility DQ, contextual DQ, representational DQ. Intrinsic DQ does not depend on a user's model and includes four basic metrics: accuracy, objectivity, believability, reputation. Accessibility DQ reflects technical aspects of data accessibility and includes the following metrics: accessibility, access security. It is important to distinguish technical accessibility and accessibility from the consumer's point of view. Contextual DQ metrics are deeply linked to the application context and include relevancy, "value-added" (how much value this data object contributes to the overall decision), timeliness, completeness, and amount of data. Representational DQ metrics include the following four metrics, which also depend on a user's model: interpretability, ease of understanding, concise representation, consistent representation.

Wang et al. [137] presented a comprehensive set of 179 metrics. Authors classify all metrics into four groups that are similar to Strong et al. metric groups:

- Intrinsic DQ metrics – do not depend on application;
- Contextual DQ metrics – rely on the application context;
- Representational DQ metrics – include aspects related to the format of the data. This group

can be further divided into the following sub-groups: unstructured data, semi-structured data, and structured data;

- Accessibility DQ metrics – consider how data consumers can access data.

Pipino et al. [102] presented 15 metrics that are used in the DQ assessment. They classify metrics into objective metrics (do not depend on a task) and subjective metrics (depend on a user model).

Todoran et al. [130] extended Wang's research and use 20 metrics that are categorized into three groups: content group, source group, and presentation group. The source group includes two sub-groups: the subjective sub-group and the objective sub-group. The subjective sub-group contains metrics that depend on the application context, and, for example, includes timeliness metric.

Bar-Noy et al. [12] also present two types of DQ metrics: intrinsic and contextual. Intrinsic metrics do not depend on the application and include such metrics as freshness, correctness, precision, and security. Contextual metrics depend on the application context and include timeliness, accuracy, completeness, and the credibility of the information source.

Cai and Zhu [21] proposes five metric groups: Availability, Usability, Reliability, Relevance, and Presentation Quality. The first four groups are fundamental and inseparable from the data, while the last group is additional and improves a user's satisfaction. The availability metrics represent the level of data access convenience and include three data quality elements: accessibility, timeliness, and authorization. The usability group evaluates the level of usefulness and includes data definition/documentation, reliability, and metadata. The reliability group reflects the level of the data trustworthiness and consists of accuracy, consistency, completeness, adequacy, and auditability elements. The relevance group represents a relation between data content and a user's expectation and includes only one DQ metric - fitness. The presentation group characterizes description methods of the data and include two elements: readability and structure. The presentation group depends on the user's needs.

2.2.2 Data Quality Types

Each DQ metric group is used to evaluate a corresponding DQ sub-type, such as subjective and objective DQ. However, DQ can be classified differently. For example, Todoran et al. [130] classify DQ into two types: local DQ and global DQ. Local DQ is evaluated in the context of the data source (sensor platform) and does not depend on the application context. Global DQ presents data fitness to a data consumer's decision-making process.

Another approach to DQ type classification is suggested by Bar-Noy et al. [12] in their research on tactical military networks. The authors classified DQ into two groups: desired DQ and delivered DQ. The desired DQ is the level of DQ level that the data consumer requests. A network guarantees the desired DQ with some probability. A delivered DQ is the DQ level of the data that the user receives.

Immonen et al. present two types of DQ estimations: the DQ assessment and the DQ evaluation. The DQ assessment estimates the quality of the raw data and does not depend on the application context. The DQ evaluation estimates the quality of processed data considering the application context.

2.2.3 Review of Approaches to Data Quality Evaluation

In addition to various DQ types, different approaches to DQ evaluation have been proposed. For instance, Todoran et al. [130] distinguish an analytical and non-analytical evaluation. The analytical approach to DQ evaluation can be employed if we have complete knowledge about the data processing system and when its complexity is not high. If we do not have complete knowledge about the data processing system, the non-analytical method is used. In this case, an experimental estimation has to be done. However, it requires access to the input and output of the data processing system and knowledge about its general behavior. The necessary data can be collected, for example, by varying input data and recording output.

Immonen et al. [57] present two types of DQ attribute evaluation: quantitative and qualitative. Quantitative evaluation is a formal process that is based on a user's rules. Qualitative evaluation is based on the evaluator's expertise and knowledge about the user. Thus, both DQ estimation methods consider the application context.

Bar-Noy et al. [12] present DQ as a value that represents the instantaneous quality of the piece of data. However, data transmission conditions change in time, and the same pieces of data can have different DQ measures at the data recipient's side. In addition, the authors introduce the Operational Information Content Capacity (OICC) concept. OICC represents the amount of data that can be delivered with a specified DQ level to a data consumer through the network. Also, in-network data processing can either increase or decrease the DQ level of the transmitted data. For example, compression of a video stream can decrease the video resolution, frame rate, field of view, etc. On the other hand, due to the smaller size of the processed video, the timeliness of the delivered data may be improved. Also, in-network processing allows for DQ improvement via fusion data from various sources.

Nowadays, a huge portion of the data is gathered from sensors and sensor networks. In some cases, these sensors produce data streams. Data quality for these data streams also has to be evaluated. Klein [81] develops a framework for data quality evaluation of a sensor originated data streams. They consider data quality as a meta-data that augments the sensor data and is stored in a database. Their framework does not explicitly take into account a user's application context. Their DQ meta-data model initially consists of three DQ metrics: accuracy, confidence, and completeness. The framework evaluates data quality at a source's (sensor) side. To evaluate the DQ of a data stream, they use a sliding window technique. However, this approach does not explicitly consider user needs.

Trustworthiness is an important data aspect that should be considered in the DQ evaluation. While the trustworthiness evaluation is out of the current research's scope, there are attempts to evaluate it separately. The importance of the data trustworthiness comes from the Big Data concept evolution. The Big Data concept has evolved from three V's (volume, variety, and velocity) to the five V's: volume, variety, velocity, veracity, and value. Veracity is an important characteristic that includes the trustworthiness of the data. For example, Chen et al. [24] investigate the trust model of the sensor networks and methods to improve their trustworthiness. Authors employ historical data acquired from a sensor for a data trustworthiness evaluation. Neighbor sensors also play an important role in this task and demonstrate how data fusion helps in the data trustworthiness evaluation and may help in the DQ evaluation. On the other hand, DQ evaluation may help to improve trust models and trust evaluation [27].

2.2.4 Data Quality Concept Evolution

There are ways to detach the application context from the DQ concept. Researchers who do not directly include an application context into the DQ concept may use a novel concept of the data value (DV). DV may be perceived from various points of view. For example, Cai et al. [21] consider DV as a low-value density, where value density is inversely proportional to the overall size of the data set. It means that the greater the data set size, the less relatively valuable the data. DV also might be considered as a further evolution of the contextual data quality, which we described above. Suri et al. [127] present the quality of information and the value of information concepts, where information has the same meaning as data. Their research is devoted to tactical networks. They investigate the quality and value of data objects that are collected via these networks. The authors demonstrated that the data object increases its value if it improves the situational awareness of a data consumer or causes the data consumers to change their decision for a better outcome. All data objects are ranked in such a way that the most valuable (important) objects are consumed and processed first. Data quality and data value are two characteristics of the data that are used for the ranking.

Data quality is represented with internal and objective metrics that consider only the intrinsic characteristics of data object [91]. For example, resolution, brightness, contrast are the metrics that are included in the DQ. These metrics do not depend on the application context. Images that have been taken at night with a night-vision camera will have better values of these parameters and, thus, higher data quality than a photograph taken with a regular camera. However, during the day, the photograph from a regular camera would have better quality. Data value is represented with external and subjective metrics that rank data objects according to the utility it provides to the data consumer. This data object should support the data consumer in more effective and efficient decision-making. In this case, a data object with the same data quality may have various data values for the different data consumers. Therefore, the data value is a function of data quality and the data consumer's application context [113].

To calculate the data object's DV, we have to have a model of the data consumer. Also, we have to consider that data object that already has been consumed by the user, due to duplicate data objects' DV is zero. The history of the delivered data object should be considered as well as changes in the data object itself. In other words, we should consider in which way the new data object differs from the old one. For example, if a hostile truck moved 100 meters from the last position and in the distance of one kilometer from the soldier, this data object has a higher value than the data object about the truck that also moved 100 meters, but 10 kilometers away from the soldier. Despite the DQ of these data objects are the same, the DV of these objects are different; in other words, the consumer context influences on a ranking function.

The data value concept is often used in military-related research. During the military operation, units should make quick and right decisions in a tough or even hostile environment. Cansever [22] investigates data value evaluation for military mobile ad hoc networks. These networks have very low bandwidth, and it is important to transfer only data that are relevant in important for the data consumer.

Bisdikian et al. [14] investigate DQ and DV in sensor networks. Authors distinguish between "the ability of judging" of a data object and "the outcome of judging". The ability to judge corresponds to the DQ concept and DQ metrics that feed the judging process. The outcome of judging corresponds to the DV concept that the data object delivers to a data consumer. The authors represent the DQ and DV as a stack of layers, where the DV layer is on top of the DQ layer. They define the DQ and DV concepts in the following way:

- Data quality – "The body of tangible evidence available (i.e., the innate information properties) that can be used to make judgments about the fitness-of-use and utility of information products".
- Data value – "An assessment of the utility of an information product when used in a specific usage context".

Some researchers, for example, John Quiggin [105], extend the DV concept even further and introduce a novel "value of awareness" concept. The value of awareness is an evolution of the DV concept. The author defines data value as "the difference between the expected return to the optimal decision based on prior beliefs and the expected return based on posterior beliefs". Value of awareness is "the improvement in expected return when an agent takes all possible states of nature into account in selecting a state-contingent income vector from a choice set".

While the DV concept allows generalizing the DQ evaluation and expends the DQ concept applicability, it requires models of the data consumer. In this research, we follow up on the conventional DQ concept. In the DQ evaluation, we employ objective DQ metrics and consider the application context during the metrics integration. This approach to these metrics design and integration allows relatively effortless framework modification to include new user needs. Also, as seen from the presented works, data security and privacy are often underestimated or even ignored. Our approach to metrics design and their integration into the DQ value allows data security aspects incorporation into the DQ evaluation pipeline.

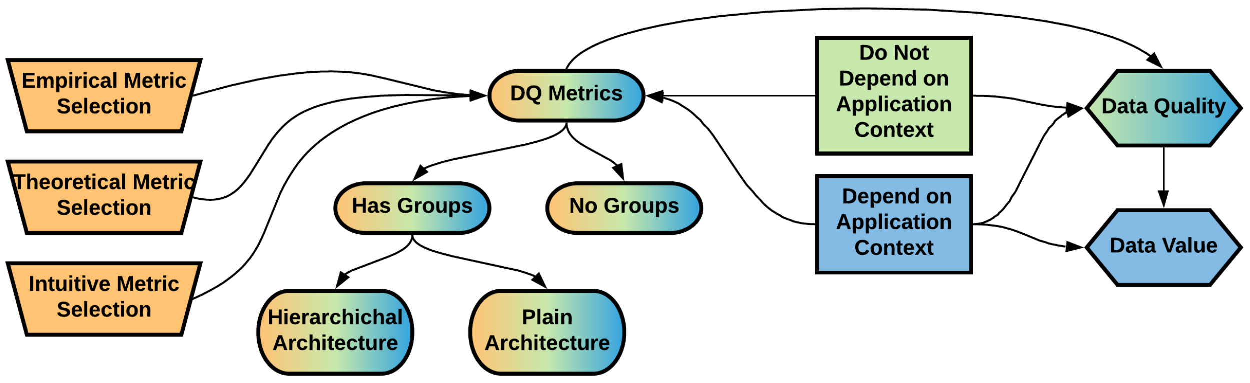

Empirical Metric Selection
Theoretical Metric Selection
Intuitive Metric Selection
DQ Metrics
Has Groups
No Groups
Hierarchichal Architecture
Plain Architecture
Do Not Depend on Application Context
Depend on Application Context
Data Quality
Data Value

Figure 2.2: Relationship between DQ concepts and DQ metrics

2.3 Review and Analysis of Existing Approaches to Security Evaluation Component in Data Quality Evaluation Pipeline

This section answers the question: "*What has been done in the area of security and privacy evaluation that is applicable in the conventional DQ evaluation pipeline?*". Data security and privacy aspects are a critical part of the overall DQ evaluation. Unfortunately, conventional DQ evaluation methods do not pay enough attention to security and privacy aspects. Researchers in the security domain very often do not consider that security and privacy can be employed in the DQ evaluation. While there are multiple studies on security and privacy evaluation, they are limited to a particular application. To develop methods for security aspect incorporation into the DQ evaluation, we have to analyze the existing approaches to the security aspect assessment.

In the security evaluation we follow the traditional security definition, which is "Security = Confidentiality + Integrity + Availability". Security evaluation integrates metrics from all three branches. The importance of each branch depends on the application context and user needs.

Data security has been studied for many years and is considered from various points of view and for various applications, for example, military [60] or medical applications. Since data security is a very complex area of research, many researchers have classified data security into categories and subcategories that may drastically vary from researcher to researcher. Moreover, some international standards have been developed that describe particular aspects of a system that are related to data security. For instance, TCSEC [84] and ITSEC [122] are used to estimate an operating system security. However, these standards describe only a small part of the data security area or focus only on one security component.

For example, Solms et al. [133] proposed a framework for an information security evaluation and developed a five-leveled information security management model. This model includes such data security aspects as hardware and software aspects, communication aspect, procedural aspect, physical aspect, personnel aspect, and environmental aspect.

Some consider security by taking into account only one aspect. For instance, Josang and Knapskog [58] consider data security from a data trust perspective. They propose a model for quantifying and reasoning about trust in IT equipment. Their model takes into consideration such parameters of an IT equipment as system reputation, security incidents, security advisory reports, developer reputation, and others. However, data trust mostly represents only the data integrity part of data security.

Neumann [95] tries to evaluate computer system security by combining concepts from both remedial and preventive approaches. Specifically, they use formal SRI Hierarchical Development Methodology and focus on an operational system security evaluation. This methodology focuses on software aspects of data security evaluation without considering hardware and communication aspects.

With the growing popularity of cloud computing over recent years, researchers started to focus on data security evaluation in cloud services. Sood [124] develops a methodology of data security assuring in cloud computing. Similarly to our approach in this research, they present data security as a combination of data confidentiality, integrity, and availability. However, unlike us, the proposed methodology does not consider the data source and starts evaluation from the moment data is stored in the cloud facilities.

Chen and Zhao [23] investigate data security and privacy (confidentiality) issues in cloud services.

Their research focuses on security and privacy protection issues in a cloud. They classify cloud computing security into four major components: software security, platform security, infrastructure security, and auditing and compliance. However, similarly to Sood, they consider data security only after the data was acquired.

In this research, we take into account those data security characteristics that may affect the overall DQ score. We combine system security, data security, and privacy evaluation approaches and integrate them into the overall DQ evaluation. We employ only those security metrics of sensor platforms that are accessible without these platforms modification. While in this research, we mostly focus on Android smartphones, the security metrics set can be extended to cover other sensor platforms as well.

2.4 Review and Analysis of Existing Approaches to Privacy Evaluation Component in Data Quality Evaluation Pipeline

In this section, we answer the following questions:

- What has been done in the field of privacy evaluation for mobile devices?
- How are addressed complex threats to user privacy and how we can improve the existing approaches?

Nowadays, privacy plays an important role not only in data science but also in our everyday life. The goal of this research is to integrate security and privacy aspects into the overall DQ evaluation processes. If approaches to security evaluation described in the previous section allow evaluation data security from the data consumers' perspective, this section presents ways on privacy evaluation from the data contributors' point of view. It is especially critical in today's era of citizen-science and crowd-sensing. If smartphone users agree to share some data (e.g., sensor data) from their devices, they have to be sure that their other private data is safe. Privacy evaluation allows the detecting of potential data leaks. In this research, we consider a modern smartphone as a sensor-rich sensor platform with a wide variety of built-in sensors. While we mostly focus on Android OS-based smartphones, the principles described in this subsection may be applied to other mobile platforms with no or little modifications.

Vendors of mobile hardware and software products attempt to address security and privacy issues with various tools developed, e.g., authentication mechanisms and anti-malware software, with many of them employing various artificial intelligence (AI) and machine learning (ML) techniques. Misuse based malware detection software and application catalogs (i.e., Google Play [49], App-Store [8], and Amazon market [4]) are called to reduce the malware threat. To protect users' privacy, users have to maintain, all the time, healthy security habits, such as keeping authentication mechanisms enabled, installing applications only from trusted sources, not blindly approving all permissions requested, etc. Unfortunately, even these measures may not be sufficient to prevent private data leakages. Malware each day becomes more sophisticated, and methods for its detection that worked yesterday may not work today. Most of the existing protection tools available for mobile devices detect and/or defend against known threats only, which makes ordinary users vulnerable to novel attacks. Alternative methodologies of anomaly detection based on analysis of data collected through monitoring of system parameters such as memory, a central processor unit (CPU), disk, and network utilization are widely used on other platforms with bigger resources available for their implementation.

Over the last decade, the research community has been slowly turning its attention to an anomaly-based attack detection on mobile devices. Schmidt et al. [116] successfully demonstrated anomaly detection cases for malware recognition in Symbian and Windows Mobile smartphones. Hu et al. [56] developed anomaly detection models based on network traffic data analysis. Burguera et al. [20] monitored 250 system calls using the GNU C library and analyzed collected data on a remote server that allowed the employment of effective but computationally expensive AI techniques, which might demonstrate high detection rates. Shabtai et al. [118] developed a behavioral malware detection framework for android devices. Their "Andromaly" framework is capable of detecting an anomaly system behavior based on real-time monitoring of various system metrics, such as CPU consumption, the number of sent packets through the Wi-Fi links, the number of running processes, and the battery level. Shabati et al. achieved very good results in detecting continuous attacks, for instance, denial of service (DoS) attacks and worm infection. Yin et al. [143] developed an anomaly detector that is based on improved self-organized maps and trained and tested it on a KDD CUP 99 data set [132] that contains captured network traffic.

In this research, as an example of the novel threats to a user's privacy, we use "colluded applications" vulnerability [68], which exploitation is currently emerging. [82]. Although the computer security community has turned their attention to this problem in 2010 [34], the existing anti-malware tools (for instance, Google Bouncer [150]) are not helpful enough in resolving this issue. In 2014, a collaborative ACiD [2] research was started to study and address this vulnerability and develop novel theoretical methods and tools to detect applications suspected in collusion and perform formal security inspection. In 2015, a modified version of this vulnerability exploitation was discovered [28]. The "Moplus" SDK library that was used in 14,112 Android applications at the time of discovery created numerous backdoors and other breaches in the smartphone security. Once the application that contains this library was installed, its exploitation allowed it to: get phone details, download and upload files to/from a device, read/send text messages, get a user's geo-location, and more. In 2016, Intel identified 21 applications that were exploiting this vulnerability [29]. These applications may execute "colluded applications" attacks to escalate privileges, bypass system limitations, and perform malicious activities. Later, in June 2016, McAfee Labs published the threats report [88] that documented 23 colluded applications. In 2017, more than twenty thousand of application pairings that may leak data were identified [135]. More applications in the wild that exploit this vulnerability were discovered in 2018 [15]. In November 2019, the security research team at Checkmarx discovered and implemented a modification of "colluded applications" attack [139, 142] that involved vulnerability of the standard "camera app" and allows to control a smartphone's camera, take photo images, record videos, and send them to a remote server. One can observe the growth of the sophisticated novel attacks on a smartphone in general and colluded applications attack in particular. Considering the current efforts of OS developers as well as smartphone vendors in improving users' privacy, it is very likely the number of such innovative and complex attacks as colluded applications will continue to grow. Therefore, millions of existing devices, as well as future devices, require adequate protection against these threats.

2.4.1 Use Case of Novel Attack on Users' Privacy

We classify "colluded application" attacks into two major categories: inter-application collusion and intra-library collusion [65]. Inter-application collusion attack exploits Android OS mechanisms that allow communication between applications without exempting malicious ones, which attempt to bypass permissions mechanisms of the Android OS intentionally. The second type of attack exploits a malicious library integration into various applications [129].

Nowadays, application developers commonly employ various libraries that allow extending application functionality without writing additional source code. Taylor et al. [129] found out that more than 57% of 30,000 tested devices have the library that may exploit intra-library collusion and access two or more additional permissions in 30.8% of cases. Over the last couple of years, researchers in the security and privacy domain paid special attention to this problem [34, 94, 109], which resulted in the broad set of tools that address it. A few types of classification can be applied to these tools: based on the reaction time, based on the OS level integration, based on who have to use a tool (an application developer, user, or vendor), and based on the analyzed application's components (see figure 2.3).

Based on their reaction time, available tools can be classified into proactive and reactive. Proactive tools try to reveal application collusion before it happens. Reactive tools detect collusion in real-time while it is happening. Proactive tools have two subcategories, which are: a static application analysis and a dynamic application analysis. Static tools analyze applications' source code and try to trace information flows in order to reveal malicious intentions. This approach is capable of detecting leakage of a very small amount of private data [11]. On the other hand, the source

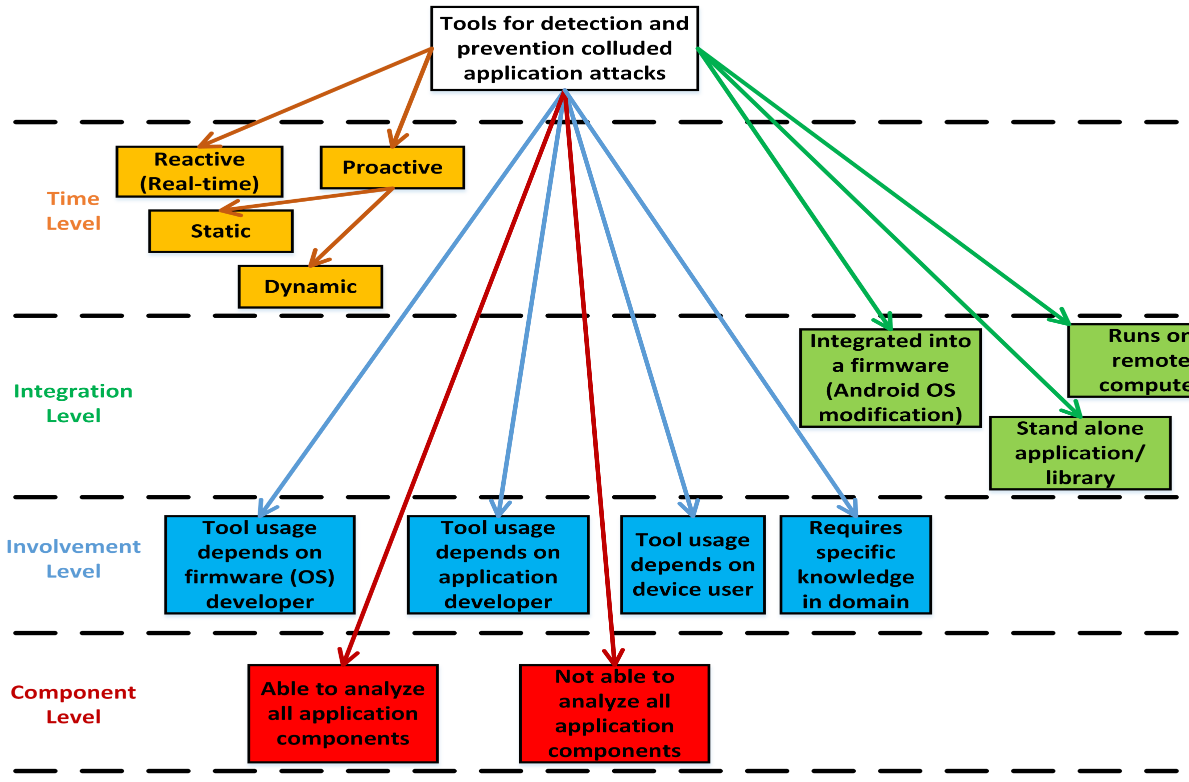

Tools for detection and prevention colluded application attacks
Time Level
Reactive (Real-time)
Proactive
Static
Dynamic
Integration Level
Integrated into a firmware (Android OS modification)
Runs on remote computer
Stand alone application/ library
Involvement Level
Tool usage depends on firmware (OS) developer
Tool usage depends on application developer
Tool usage depends on device user
Requires specific knowledge in domain
Component Level
Able to analyze all application components
Not able to analyze all application components

Figure 2.3: Existing tools for detection and prevention colluded application attacks classification

code of modern applications is obfuscated and may hide dangerous API calls. Moreover, each day, about 2,500 new applications are published in Google Play [106], and it is hardly possible to apply the static analysis of their code. Dynamic analysis is called to solve the problem with the code obfuscation. In this case, the analyzed applications are run in the controlled environment of a real or virtual device in different scenarios with the hope that malicious applications reveal their collusion. The device runs a modified version of the Android OS, which allows tracing unauthorized information flows or applications that are run in the debug mode. For example, Hay et al. [53] developed the tool for dynamic application analysis in the debug mode. However, modern malware are able to detect these controlled environments (for example, using motion sensors [46]) and hide their malicious intentions. While this approach partially solves the problem, it cannot be scaled up to the constantly increasing number of available applications. In addition, modern versions of Android OS update applications automatically by default that may result in collusion after the analysis was performed.

Reactive or real-time tools analyze only installed applications and parameters of the device, which may indicate collusion that makes them robust enough to the growing number of available applications. For example, the TaintDroid tool [39] traces sensitive information and reveals dangerous information flow. Xmandroid [19] extends the Android OS monitoring system and allows detecting the unauthorized device's resource access. However, these tools have to be integrated into the device's firmware; in other words, the Android OS has to be modified. Considering that today, more than two billion active devices are running Android OS [131], and different vendors have produced these devices, it is almost impossible to update their firmware.

Integration into a device's firmware requires OS modification and can be done only for a limited number of models due to the massive platform fragmentation. Frameworks that can be run on a standalone or a remote computer do not require a firmware modification and can perform an analysis much faster due to higher computational capabilities than mobile devices [10, 17]. However, this approach requires specific knowledge in the domain which a regular user is not expected to have.

Table 2.1 summarizes the characteristics of the most popular tools for "colluded applications" attack detection. It specifies the use of AI techniques in their design and operation. As one can see, most tools employ rules in analyzing applications source code or their behavior. Moreover, those rules are "hardcoded" in tools' code, which makes this approach sensitive to the attack variations [11]. Rules modification becomes a hard problem that would require code re-design and re-compilation. Similarly to the security evaluation approach, we are going to use only those security and privacy-related metrics that are accessible without a device modification. We propose an AI-based anomaly detection methodology capable of detecting private information leakage. This methodology employs machine learning classification models, which can be retrained and adjusted to detect novel attacks without a source code modification.

Table 2.1: Existing tools for colluded applications attack detection

Tool or author	**Time Level**	**Integ-ration Level**	**Invol-vement Level**	**AI**	**Compon-ent Level**
Asavoae et al. [10]	Static	Runs on remote computer	Requires specific knowledge	Rules	All components
SCLib [141]	Static/real-time	Integrated into application	Application developer	Rules	All components
DIALDroid [17,18]	Static	Runs on remote computer	Requires specific knowledge	Rules	All components
TrustDroid [148]	Static/real-time	Integrated into firmware	Firmware developer	Rules	Not all components
TaintDroid [39]	Real-time	Integrated into firmware	Firmware developer	Rules	All components
Xmandroid [19]	Real-time	Integrated into firmware	Firmware developer	Rules	All components
IntentDroid, [53]	Dynamic	Integrated into firmware	Requires specific knowledge	Rules	Not all components
IntelliDroid [140]	Dynamic	Runs on remote computer	Requires specific knowledge	No	Not all components

2.5 Conclusions

*Addressing the first research question posed in this chapter **"What has been done in the domain of DQ evaluation?"**, we obtained the following **major** research results, which are further explained in details:*

- *We **reviewed and classified** modern approaches to the **data quality evaluation**;*
- *We **revealed insufficient attention to the security aspect** in the conventional data quality evaluation pipeline;*
- *We **classified existing approaches to the data quality metrics design**;*
- *We **investigated privacy threats** and **classified existing approaches to the privacy protection**;*

Although the DQ concept has been studied for decades, there is no unanimously accepted DQ definition. As we have rendered in this chapter, there are multiple approaches to DQ evaluation. While some researchers present more than two DQ metric groups, our analysis proves that on a fundamental level, all conventional DQ metrics can be classified into two major groups: application-independent (intrinsic) and application-dependent (contextual) groups. Also, our analysis of the existing research in the DQ domain reveals insufficient attention to the security and privacy-related metrics.

For the metric design, we chose a theoretical approach because it allows us to include a big set of DQ metrics into the DQ evaluation processes. A big number of DQ metrics requires an efficient approach to their organization. We chose a hierarchical approach to metrics organization because it facilitates metrics set extension with no or little efforts.

Also, we classified existing approaches to the DQ concept definition. While most researchers include the application context into the DQ concept, some researchers present two types of the DQ concept: intrinsic and contextual. Some researchers move the application context to the data value concept and render the DQ concept as application-independent. Our analysis led us to choose the conventional approach to DQ evaluation that employs mostly application-independent first-order metrics. We consider the application context on the DQ metrics integration level.

Our analysis of current methods of DQ calculation revealed insufficient attention to the security and privacy aspects in the overall DQ evaluation pipeline and guided us to investigate ways of security aspects incorporation into the DQ evaluation processes. We analyzed existing methods of security evaluation and revealed that existing approaches mostly focus on a very narrow set of features. We chose to use a wide spectrum of system security metrics to perform the security evaluation and integrate it into the DQ evaluation pipeline. We are going to use only those security metrics that are accessible without a sensor-platform modification.

As an implementation platform, in this research, we focus on the Android OS based smartphones. To evaluate the user's privacy, we analyzed current threats to it and approaches to enforce privacy policies. Unfortunately, our analysis shows that modern methods of private information leakage detection and prevention are not very effective in detecting sophisticated modern attacks on the user's privacy, such as the “colluded applications” attack. We investigated existing methods of this attack detection and revealed a lack in the methods that are accessible to untrained users. In this research, we are going to develop security and privacy evaluation methods that may be used by untrained smartphone users and can be integrated into the overall DQ evaluation framework.

Chapter 3

Data Quality Evaluation Framework Design

*This chapter addresses the second major research question: "**How to integrate DQ evaluation into a synergetic DQ and security evaluation framework?**". This question can be divided into a set of sub-questions, which are answered in each section of this chapter. In this chapter, we develop and present the overall structure of the DQ evaluation framework (section 3.1). We render the overall framework structure and relationships between data source, data object, data quality, a data consumer, and application context. The developed hierarchical DQ metrics organization facilitates adding new metrics and, therefore, taking into account data user needs. Section 3.2 presents the knowledge graph concept that builds up a solid foundation for the framework development and its implementation in applications by creating a collaboration between such framework components as DQ metrics selection and metrics integration calculus. In addition, the knowledge graph employment enables knowledge accumulation and facilitates framework adoption by new applications. In section 3.3, we present our knowledge analysis of sensors that are typically built into modern mobile devices. This analysis allowed us to classify popular sensors from the quality perspective and resulted in developing sensors quality knowledge base.*

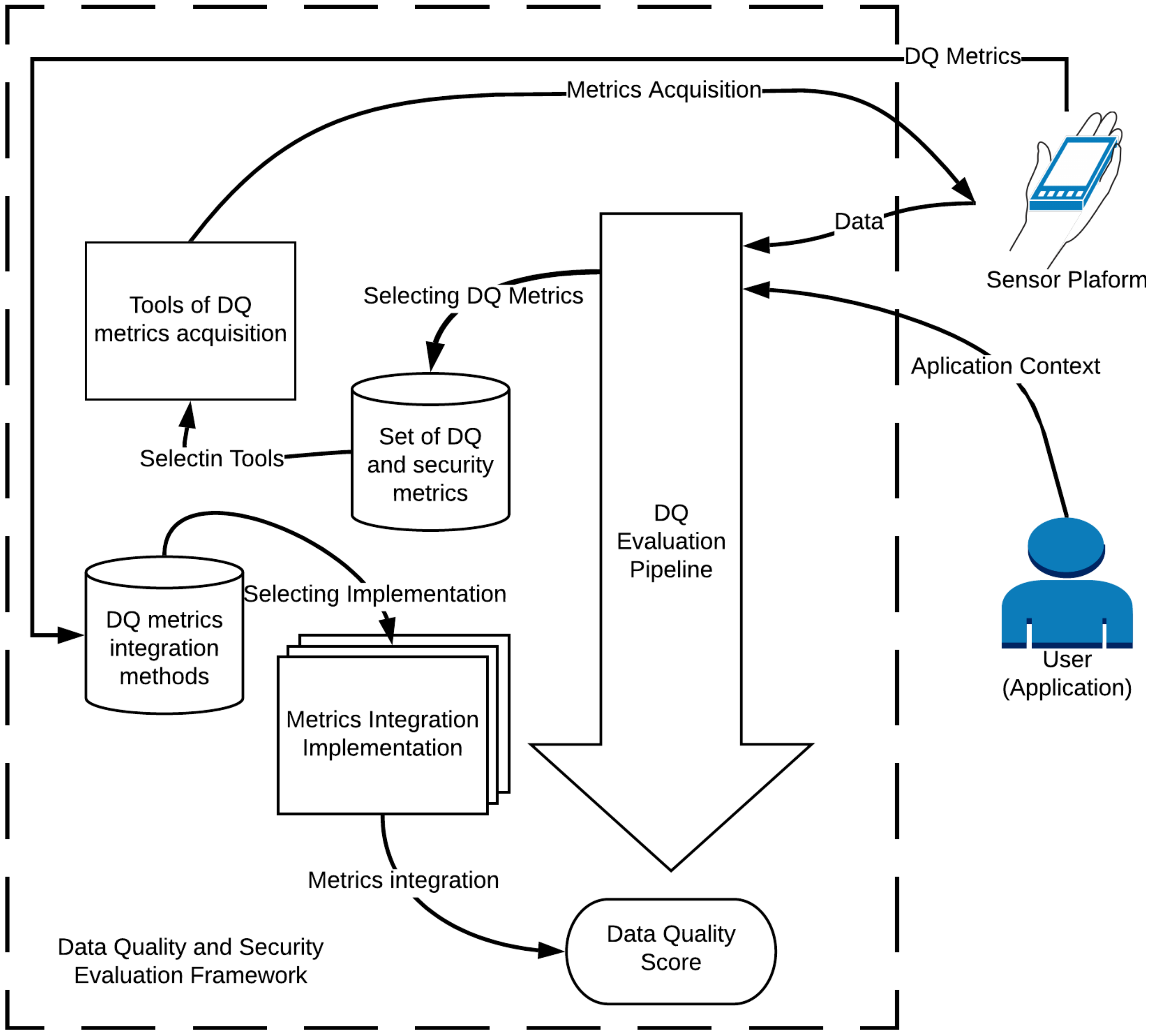


Figure 3.1: The overall structure of the integral data quality and security evaluation framework

3.1 Proposed Data Quality Evaluation Framework Architecture

In this section, we investigate the relationship between the conventional data quality evaluation and security evaluation components. This section addresses the question: "*How to design the DQ evaluation framework in such a way that it can be easily adjusted and expanded for new applications?*".

The integral framework for data quality and security evaluation includes the following components: create the list of the initial data quality and security metrics, tools for these metrics acquisition, and metrics integration calculus methods that allow consideration of an application context (see figure 3.1). As we rendered in the previous section, we developed a multilevel hierarchical structure of the metrics organization (see figure 3.2). This type of metrics organization facilitates framework modifications and adjustments in order to employ it in the new application domains.

The tree leaves of the metrics hierarchy represent the metrics of the first order. All other metrics of higher order are composed from the metrics of the previous layer (lower order metrics). This design allows us to employ the "divide and conquer" strategy, in which each branch of the hierarchy corresponds to one of the DQ components. In this approach, each DQ component can be calculated separately and then integrated into the unified overall DQ score. It also enables employing various (non-heterogeneous) calculi.

This approach to the metrics organization and DQ evaluation facilitates the DQ calculus adjustment by changing the corresponding branches. It enables the framework adaption for the new application context by including new metrics representing the data consumer's needs.

The metric hierarchical structure can be extended either horizontally or vertically. The horizontal metric hierarchy reflects adding new metric branches, groups, and types. This extension facilitates taking into consideration new applications. The vertical extension adds new metrics to the existing metric branches. This type of hierarchy extension allows taking into account new first-order metrics and calculating new higher order metrics. Adding of new low-order metrics makes our DQ calculus even more comprehensive.

Figure 3.2 presents a higher level organization of the metrics used in the framework for DQ evaluation in mobile devices and sensor networks. We classify DQ calculus or metrics?? in two major groups: Data Security and Data Correctness. Each of these groups consists of other lower level elements and represents a branch in a hierarchical structure of the DQ evaluation.

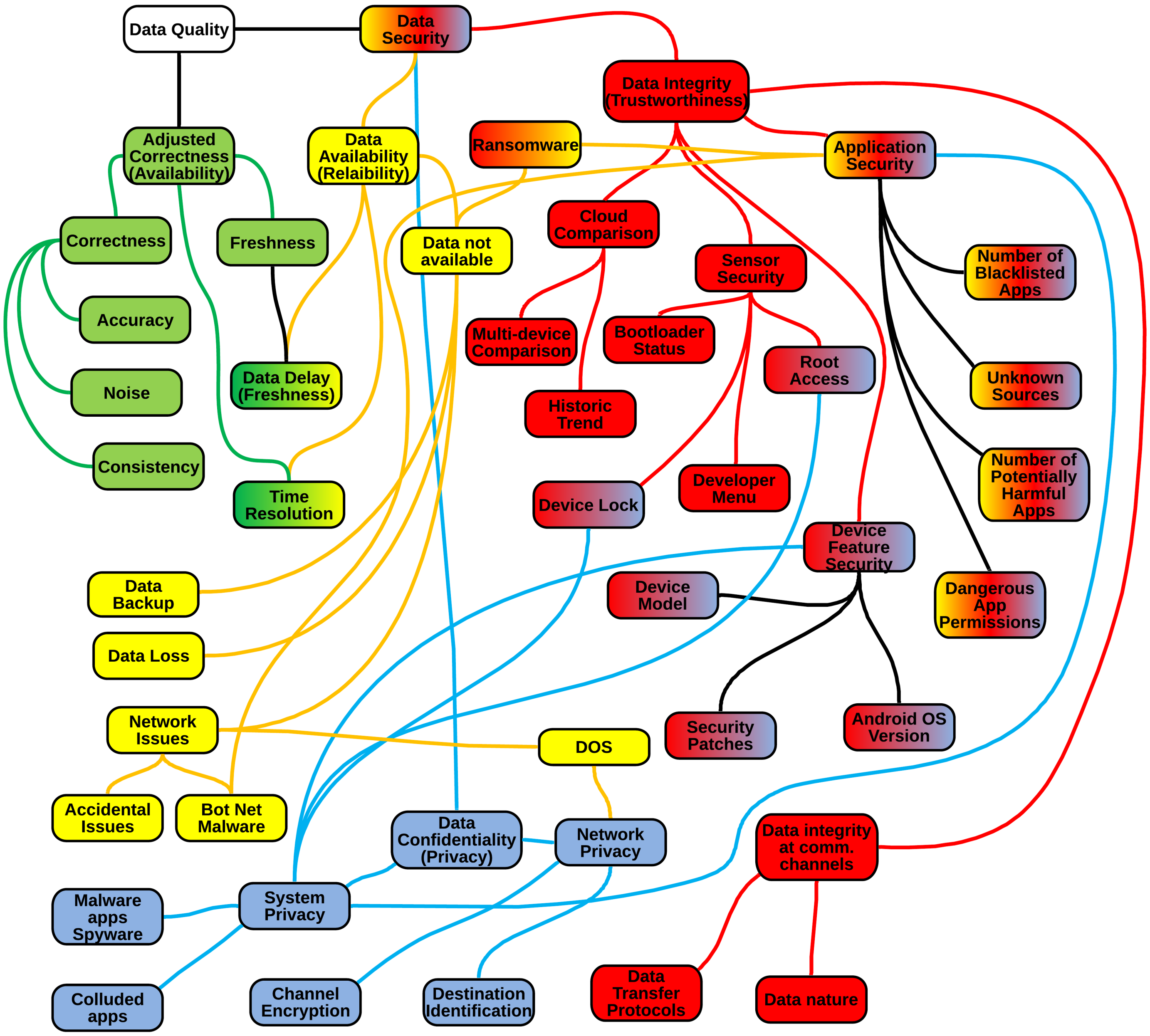


Figure 3.2: Data Quality overall hierarchical structure. Green colored metrics belong to conventional DQ group, yellow colored belong to data availability, red colored belong to data integrity, and blue colored metric belong to data confidentiality group.

Data Correctness component reflects the data source's physical parameters and characteristics, such as accuracy, noise, consistency, resolution, etc. Data Security component integrates data availability, data integrity, and data confidentiality. Data integrity, in its turn, merges data integrity of the source and data integrity of the communication channels. In addition, components of one branch (i.e., data security) may influence elements of other branches (i.e., data correctness). Data source integrity takes into account various parameters of the data source that influence data integrity in multiple ways.

Data delay (freshness) and time resolution are related to the sensor's data quality metrics, but also represents the data availability. Data delay, to some degree, reflects data freshness. However, freshness is a broader concept than data delay. Data delay describes the latency of data delivery as it deals with short time periods, while data freshness may be represented by years.

In the developed framework, we consider sensor accuracy, noise level, and consistency as basic (first-order) metrics for the data correctness calculation. Noise (e.g., the RMS noise of an accelerometer) is calculated based on the actually gathered data. The real noise value is scaled between its maximum and minimum values, which are defined by a data consumer. After scaling, the noise is represented with a dimensionless value that can take values in the range from zero (minimum noise value) to 100 points (if it is on or above the maximum allowed noise value). We apply this type of scaling to all metrics in this branch. Consistency represents the number of records that are missing. If consistency equals 100 points, all records are present. Zero points of the consistency reflects that the maximum amount (a user also defines that) of missing records is reached. Consistency is important for the decision-making process because various decision-making algorithms may tolerate various rates of missing data.

Time resolution reflects how often data can be acquired. It is similar to the "data delay" metric, but it describes "latency" within a data object if the data object represents a piece of time series data. Basically, data delay and data time resolution may be represented as subcategories of the data freshness.

While our multilevel metrics hierarchy enables framework extendability, it also has to be supported by the implementation platform. In addition, it has to be able to employ heterogeneous calculi and facilitate effortless framework extension. Also, the implementation platform has to allow taking into account an application context and choosing those metrics (and their integration calculus) that satisfies data consumer needs.

It is a very challenging task to find or develop the framework implementation that would satisfy our demanding and specific requirements. Fortunately, a knowledge graph concept satisfies all our requirements [71], which is also one of the novelties in this research. The knowledge graph allows merging all framework components into the complete solution. The absence of a fixed schema gives the required flexibility, merges such framework's components as metrics selection and their integration calculus, and allows taking into consideration the application context. The additional benefit from the knowledge graph employment is knowledge accumulation.

The knowledge accumulation works on two levels. On the first level of knowledge accumulation, the knowledge graph acts as the database. It means that each time the framework does any evaluation, this evaluation may be stored in the knowledge graph and reused later if needed.

On the second level of knowledge accumulation, the framework evolves with each new application. To modify the framework for the new application, we need to add a new application context, new relevant DQ metrics, and metrics integration methods. The nature of the knowledge graph (see section 3.2) allows creating a connection between different types of data and their entities. A new application context adds new physical and logical entities, which is a knowledge accumulation.

3.2 Developed Methodology for Knowledge Graph Employment in Data Quality Evaluation

In this section, we answer three questions:

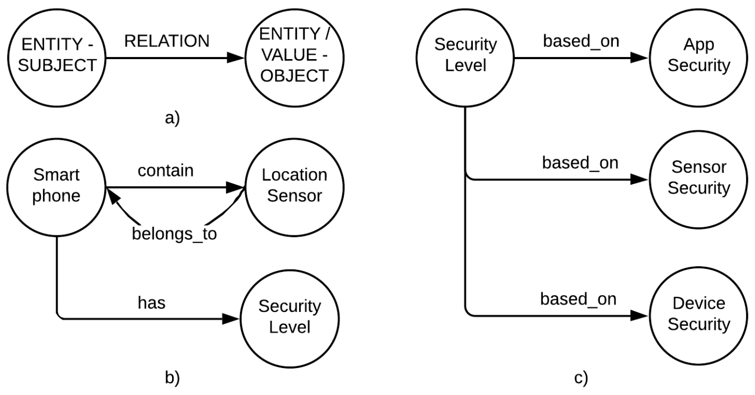


Figure 3.3: “Triplet” is the basic unit of a knowledge graph. a) Generic example. b), c) samples from the framework’s knowledge graph.

- *Are there any ways to connect DQ and security metrics, their integration methods, their calculus implementation, and user context?*
- *Can a knowledge graph concept be used to unite all framework components?*
- *How can we enable a knowledge accumulation in the DQ evaluation framework?*

This section presents the architecture of the knowledge graph and describes its role in the overall DQ evaluation. Since a knowledge graph does not have a fixed schema, its architecture can be defined by its vocabulary. The basic unit of the developed knowledge graph is a “triplet”. A triplet consists of a “subject”, a “relation”, and an “object”. “Relation” connects “subject” and “object” (see figure 3.3).

“Object” and “subject” are relative concepts, where subject points to an object. Further, we refer to “object” and “subject” as “entity”. Each entity has to have its unique ID, while all other property fields are optional. If a property points to another entity, this property is a “relation”. If a property takes a single value (can be numerical, alphabetical, or both), it is an attribute. For example, in statements “smartphone - contain - location sensor” and “smartphone - has - security level”, “smartphone” is a subject (an entity), “contain” and “has” are relations (properties), and “location sensor” and “security level” are objects (other entities/attributes). However, in the statement

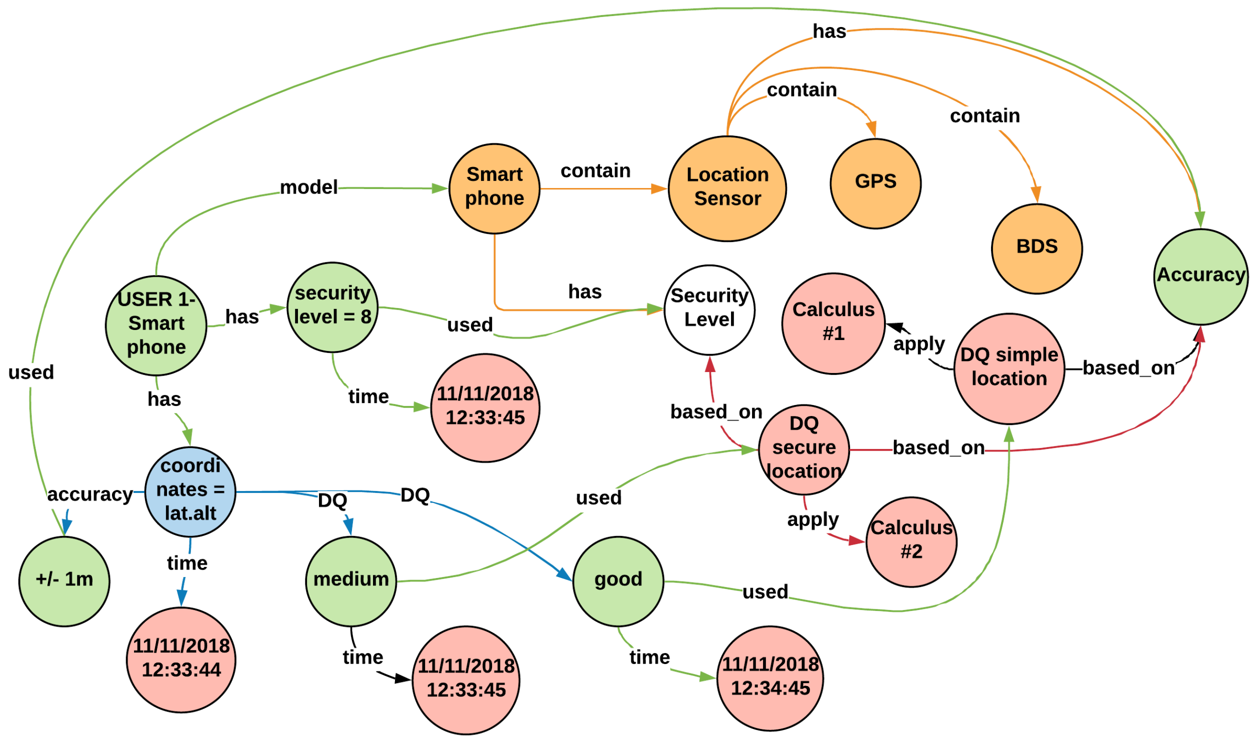


Figure 3.4: The knowledge graph concept representation as a synergy between data source and DQ evaluation. It demonstrates knowledge accumulation and the relationship between data collection DQ evaluation. Orange circles correspond to physical entities, pink circles represent logical entities, entities related to security are presented with white circles, data objects are represented with blue circles, and green circles correspond to DQ related entities.

"Location sensor belongs to the smartphone", "location sensor" is a subject, "smartphone" is an object, and "belongs_to" is a relation (see figure 3.3-c). In a statement "security level - based on - app security, sensor security, and device security", "security level" is a subject, "based on" is a relation, and "app security", "sensor security", and "device security" are objects.

The knowledge graph concept includes data source models, calculus methods, and data objects along with their data characteristics, such as DQ estimates. Figure 3.4 presents an example of a knowledge graph, where blue entities correspond to a model description of a smartphone, that may have a security level property and a location sensor. The location sensor supports "Global Positioning System" (GPS) and "BeiDou Navigation Satellite System" (BDS) and also has an accuracy property. Pink entities represent two applications that require a DQ evaluation. For example, "DQ secure location" is based on the location sensor accuracy and a smartphone security level. To integrate these metrics into a one DQ value, a "calculus #2" (for example, an expert system, a function, a weighted sum, etc.) has to be applied. "DQ simple location" task needs a DQ estimate that is based only on the location sensor accuracy and is calculated through "calculus #1". Green entities/attributes represent a particular smartphone (and some data from it), which model is described by entities of a blue color. This particular smartphone has a location data object (DO) (location coordinates), that has "time", "accuracy", and "DQ" properties (relations). "Accuracy" and "DQ" entities have "used" property that points to a corresponding logical entity that was used to create (calculate, estimate, etc.) this particular entity. For example, "DQ = Medium" has a time when it was calculated (11/11/2018 at 12:33:45) and that it was calculated for "DQ secure location", which is based on a device security level and location sensor accuracy. Since "DQ secure location" requires the security level and location accuracy, these parameters were also calculated, and corresponding entities were created (green color in figure 3.4). Each time a data object is requested, or a DQ estimate is needed, a new piece of knowledge is created and is stored in the knowledge graph in the form of a new knowledge graph entity.

The vocabulary specifies all possible property fields, their types, and their relations. Since new properties can be easily added to the vocabulary, the knowledge graph can be adopted by new applications without significant efforts. Some of the terms from our vocabulary may correspond to terms in other popular knowledge graph vocabularies, such as DBPedia [37] or Google Knowledge Graph [47]. In this case, we use the term “same_as” that allows re-utilizing other vocabularies and simplifies the migration process.

3.2.1 Knowledge Graph Vocabulary Development

While the vocabulary can be easily extended, the initial set of the vocabulary entries is presented below. Vocabulary is a collection of entities. Since DQ evaluation is very often associated with sensor data and the Internet of Things (IoT), we heavily rely on an IoT vocabulary developed by schema.org and W3 [115]. However, due to some specifics of DQ evaluation, we introduce new entity types and relations. The entity hierarchy supports the parent-child (inheritance) concept. All properties except “uid” are optional. The basic type in our knowledge graph is “Thing”. All other entities inherit properties from “Thing” and extend them with specific ones. Further, we use two basic types of entities: physical and logical. The physical entity represents a material object or a device, and the logical entity describes a concept that does not have a physical representation. Initial entities and their properties are presented in Table 3.1.

3.2.2 Demonstration of Knowledge Graph Role in Data Quality Evaluation and Knowledge Accumulation

To facilitate DQ evaluation, we developed an initial set of various calculi that can be used for DQ evaluation in a wide application range. When a user requests a new DO, our framework augments it with a DQ estimate. The requested DO (and its DQ value) form a new entity in the knowledge graph. Intermediate calculations are also stored in a form of knowledge graph entities, that allows using them in other DQ estimations and improving DQ evaluation efficiency. For example, if one user requested a DO with DQ evaluation and this DQ based on the security level of a smartphone, the security level calculated in the process is added to the knowledge graph as a

Table 3.1: Initial vocabulary of the presented knowledge graph. * - denotes mandatory fields. ** - denotes logical entity. *** - denotes physical entity.

Thing Entity	
uid*	the unique identifier of an entity/value
type	a type of an entity/attribute
parent	specifies a parent entity type. "null" or missing property indicates that the parent type is a "thing".
description	human-readable description. May be empty.
belongs_to	represent "back" relation. Points to an entity that has this entity as a property.
same_as	present a set of links on the entities from other vocabularies that contain similar entity.
time	date and time when this entity was created.
Logical Entity**	
based_on	specifies a set of other "logical" entities on which this entity is based on
has	a set of "logical" entities that this entity has
value	a set of "value" entities
apply	points on a "calculus" entity
Device Entity***	
contain	a set of "device" entities that this entity has built-in
has	a set of "logical" entities that this entity has
Sensor Entity	
support	a set of "logical" entities that specify what protocols this sensor supports
Accuracy Entity**	
Security_level Entity**	
DQ Entity**	
used	points on DQ_application entity
DQ_application Entity**	

new entity/attribute.

Figure 3.5 presents an example of DQ evaluation along with knowledge accumulation. A user requests a smartphone location for an application where the location has to be trustworthy. In this case, a "DQ secure location" entity is called, which is based on "smartphone.security level" and "smartphone.location_sensor.accuracy" entities. An "expert system" #2 (ES #2) has to be used to calculate DQ of the location DO for that particular application, The security level is based on "app security" and "device security". It is calculated using the "ES #1". Similarly, "accuracy" is based on the accuracy of "GPS" and "GLONASS" and takes the best accuracy. Once the security level, accuracy, and DQ are calculated, new entities/attributes are created (see figure 3.5 pink-colored entities). Security level got value "8" and time "01/01/2019 08:31:33", "accuracy" got value "+/- 1m" and time "01/01/2019 08:31:33", and smartphone entity got a new "last location" property. "Last location" has attributes "time" and "value" and property "DQ", that also has property/relation "used". Relation "used" points on the application, it was evaluated for. Let us

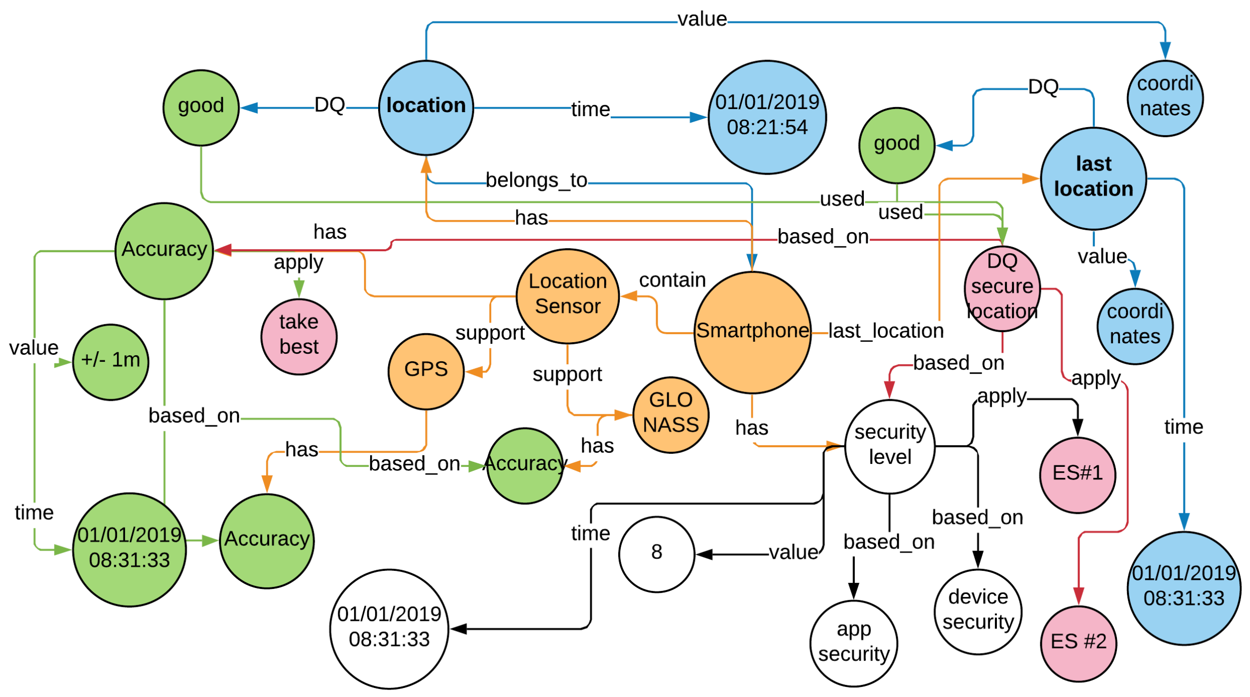


Figure 3.5: Example of DQ evaluation and knowledge accumulation. Blue and green entities represent new knowledge.

assume that there was a previous request of the secure location at “01/01/2019 08:21:54”. This request also created a new knowledge graph entities (see figure 3.5 blue-colored entities). Entity “location” has such properties/relations as “belongs_to” and “DQ”. A new location entity has been added to the smartphone entity via “has” property. New entities that have been generated and added to the knowledge graph represent knowledge accumulation.

Framework flexibility and scalability are additional benefits of the knowledge graph employment. For example, we do not need to know in advance what protocols the location sensor supports. We just need to specify that DQ evaluation for this particular task is based on location sensor accuracy and smartphone security level. Further, each instance of a smartphone entity may support various location protocols, as well as security level may be based on various specific parameters. Our framework follows the chain of relations (see figure 3.6), returns the result, and create new entities. In this particular example, we use simple relations and simple calculi, while in real life, special rules that verify data freshness have to be applied.

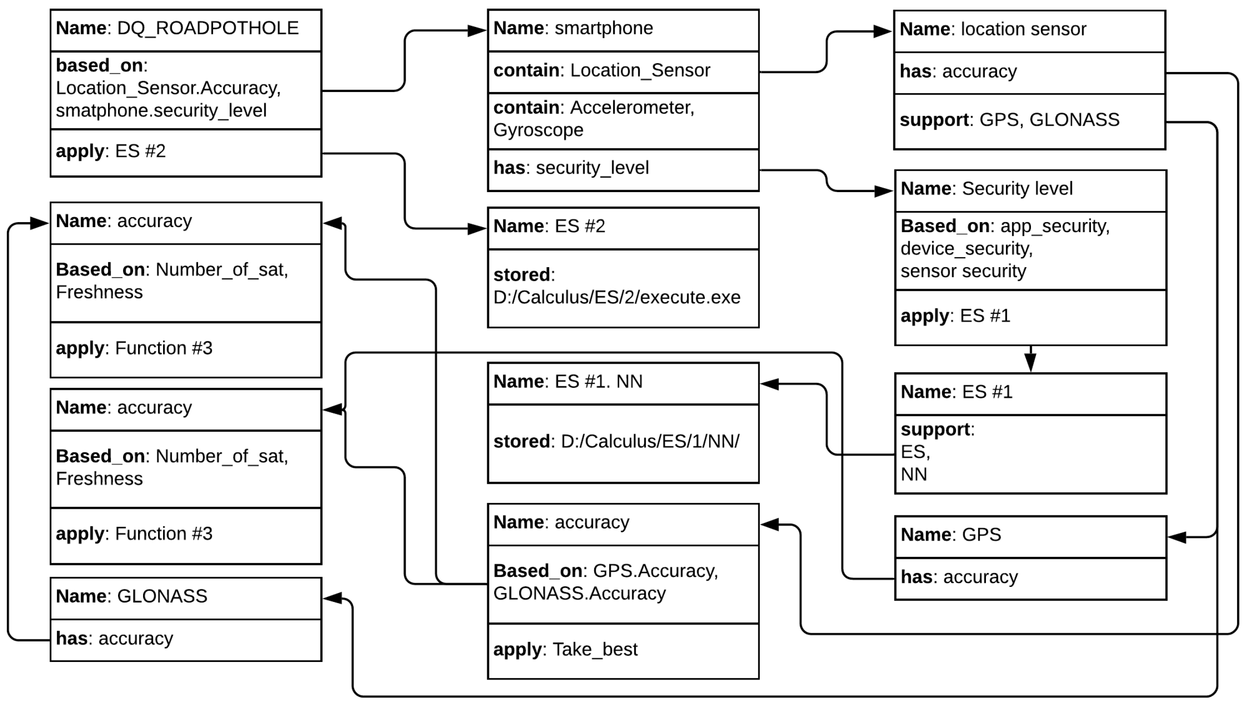


Figure 3.6: Relations chain in the knowledge graph

Because all knowledge is represented with entities and their relations, we can extend our framework to cover new applications with a little effort. Since we use a standardized approach, the learning curve is very shallow. To start using the framework, a user need to specify DQ calculi and dependencies. The user can migrate from older API to ours without significant efforts and, in return, will gain DQ estimates. The more users and the more frequently they use our framework, the more data accumulates in a knowledge graph, which makes it more generic and robust.

3.3 Created Knowledge Base of Mobile Devices' Sensors Quality

In this section, we present our research on the quality of data coming from sensors that are built into modern Android OS smartphones. Our investigation resulted in building a knowledge base that can be later utilized in various applications. While we are developing a generic knowledge base approach, previous studies and tools were highly specialized, with a very limited number of sensors and sensor types involved. A. Das et al. [31], for example, collected sensor data in real-time under different conditions by creating a web application and storing the data on a university server. Unfortunately, since web applications can access only an accelerometer and a gyroscope sensor of a mobile device, this application can be used only with two mobile sensors. K. Hilgenberg [55] develops a configurable and device-independent mobile sensor data collection framework that provides sensor readings in real-time. Z. Ma et al. [149] compared accelerometers, gyroscopes, and magnetometers based on properties of sensors such as sensitivity, noise, frequency, and range. Q. Mourcou et al. [93] evaluated mobile device sensors for clinical motion research. Shala et al. [120] compared the accuracy of mobile device sensors for indoor positioning scenarios. Unlike the aforementioned researches, we develop a service that is applicable to a wide range of applications and embedded sensors. In contrast to the researches and frameworks mentioned above, in our knowledge base, we consider multiple sensor quality metrics, integrate them into the overall quality score, classify sensors into two quality groups using machine learning techniques, and provide information about sensor quality to users through the developed Android OS application.

3.3.1 Sensor Data Collection and Processing Methodology

This knowledge base focused on the most common sensors that are embedded in modern mobile devices. To obtain data about these sensors, we searched public repositories such as GSMArena [52] that allowed us to acquire a comprehensive list of sensors and their types. Also, we investigated sensor characteristics that can be found in the manufacturers' data-sheets and online [5, 38, 42, 43, 101]. The data collection workflow is presented in figure 3.7.

We made the collected data publicly available, and we encourage the community to use it [32]. We further analyzed the original data set and identified the attributes related to the selected sensors. We normalized sensor characteristics in the range between "-1" and "1". In the case of sensor quality metrics that have a positive influence on the overall sensor quality score, "-1" represents the worst case, "0" is the average, and "1" is the best case. For those sensor quality metrics that

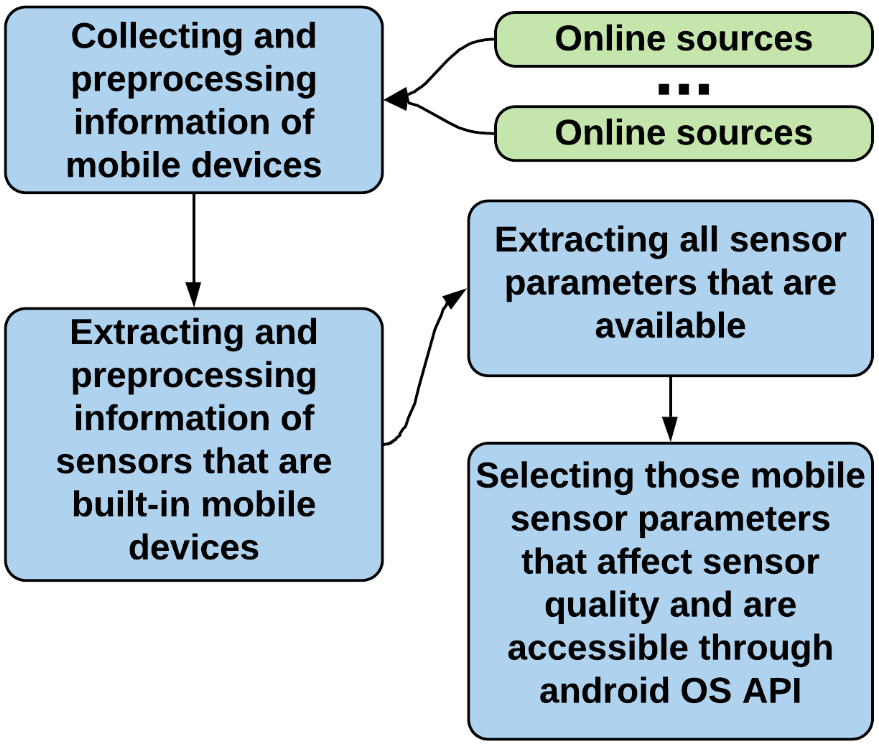

Collecting and preprocessing information of mobile devices
Online sources
...
Online sources
Extracting all sensor parameters that are available
Extracting and preprocessing information of sensors that are built-in mobile devices
Selecting those mobile sensor parameters that affect sensor quality and are accessible through android OS API

Figure 3.7: Data Collection and Processing Workflow

have a negative influence on overall sensor quality score, the scale is inverted, where “-1” represents the best case, and “1” is the worst case. This normalization facilitates further comparison of sensors and their quality evaluation.

3.3.2 Mobile Sensor Data Collection

Our data collection contains information about sensors embedded in 9443 devices, including Android OS-based smartphones, wearable devices, and tablets. This data collection includes other attributes related to mobile devices such as type, size, resolution, chipset, camera, sensors, performance. In total, we have about 58 attributes for these devices produced by over a hundred manufacturers.

The current collection contains data on 19 sensor types [97] including accelerometer, barometer, gesture, humidity [35], face id and fingerprint sensor (rear, side, under display and in front), tem-

Table 3.2: Sensors and the characteristics they were evaluated upon

Sensor Type	**Characteristics**	**Application Purpose Examples**
Accelerometer	Sensitivity, Non-linearity,Noise Density	Inclination measurement, vibration evaluation
Gyroscope	Sensitivity, Noise Density, Cross-axis Sensitivity, Non-linearity	Step counting, activity recognition
Proximity	Resolution, Range, Absolute Response	Obstacles detection
Barometer	Lowest Measurement in range, Highest Measurement in range, Absolute Accuracy, Noise(RMS)	Altitude measurement
Compass	Non-linearity, Sensitivity, Noise(RMS), Heading Accuracy, Magnetic Field Range, Resolution	Navigation

Table 3.3: Statistical analysis for the accelerometer sensor type

Stat	Sensitivity	Non-linearity	Noise Density
Min	16	0.10	75
Max	17039	2.00	800
Mean	6033.91	0.61	308.08
SD	7434.01	0.39	183.23

perature, ambient light, gyroscope, heart rate, compass, UV [7], iris scanner, SpO_2 [104], proximity, and color spectrum sensors. In this research, we selected and further analyzed the five most popular sensor types: 52 accelerometers brands, 14 gyroscopes brands, 20 proximity sensor brands, five barometers, and 16 compass sensors brands. The selected sensors and their characteristics that are chosen for further analysis are given in Table 3.2. For each of the chosen sensor types, we selected sensor parameters that can be accessed through the standard Android OS API. These parameter's values are used for the sensors' quality evaluation. The selected sensor parameters and the results of their statistical analysis are presented in Table 3.3 for accelerometer sensors, Table 3.4 for gyroscope sensors, Table 3.5 for proximity sensors, Table 3.6 for barometer, and Table 3.7 for magnetometer sensor.

Table 3.4: Statistical analysis for the gyroscope sensor type

Stat	Sensitivity	Noise Density	Cross-axis sensitivity	Non linearity
Min	33.8	0.0038	1.0	0.10
Max	131.2	0.030	2.0	0.20
Mean	114.55	0.0117	1.66	0.142
SD	24.62	0.008345	0.32025	0.036

Table 3.5: Statistical analysis for the proximity sensor type

Stat	Resolution	Range	Absolute Response
Min	8.00	50.00	100.00
Max	20.00	100.00	165.00
Mean	12.91	93.75	131.42
SD	3.43	11.023	17.54

3.3.3 Mobile Sensor Quality Analysis

In our further analysis, we employed the K-means method to classify all sensors of each type into groups based on their quality. The K-means clustering algorithm is chosen as it is relatively computationally inexpensive and, considering small quality metrics dimensionality, provides good cluster positioning [112]. This algorithm allows to partition a collection of instances into k different subsets employing a recursive technique. Since this algorithm requires choosing the initial (start) point of cluster centroids, there are several approaches to choose them. In our research, we employed the K-means++ algorithm, which uses a heuristic to find centroid seeds for k-means clustering, and take advantage of our sensor quality metrics analysis and scaling. According to Arthur and Vassilvitskii [9], K-means++ improves the efficiency and effectiveness of Lloyd's algorithm.

The workflow for quality evaluation is presented in figure 3.8. Since not all sensor parameters equally affect the overall sensor quality, in the quality evaluation calculus, weights to each of the sensor parameters can be adjusted. In this particular implementation, we employed the mean value of all metrics. One can see that metrics normalization in the range from "-1" to "1" facilitates the overall sensor quality evaluation since only weights have to be adjusted based on the metric importance. The overall quality indicators were calculated for each sensor type, and the results are

Table 3.6: Statistical analysis for the barometer sensor type

Stat	Lowest Measurement	Highest Measurement	Absolute Accuracy	Noise
Min	260.00	1100.00	0.50	0.01
Max	300.00	1260.00	2.00	1.30
Mean	284.00	1194.00	1.30	0.28
SD	21.90	85.90	0.67	0.56

Table 3.7: Statistical analysis for the compass sensor type

Stat	Non-linearity	Sensiti-vity	Noise	Heading Accuracy	Magnetic Field Range	Resolu-tion
Min	0.100	0.008	0.150	1.0	800.00	8.0
Max	2.000	1.000	1.0	2.5	7200.00	16.00
Mean	0.835	0.243	0.5437	1.833	3202.40	14.466
SD	0.4027	0.242	0.2679	0.434	2042.26	2.1561

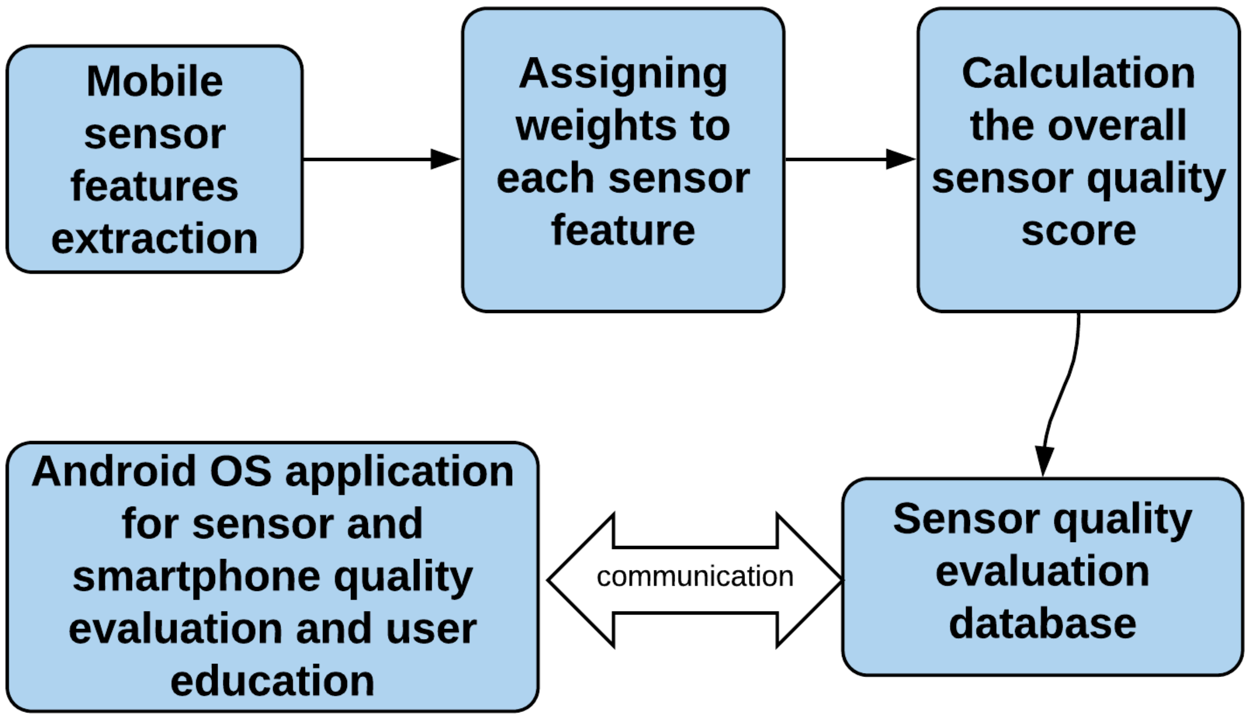

Mobile sensor features extraction
Assigning weights to each sensor feature
Calculation the overall sensor quality score
Sensor quality evaluation database
communication
Android OS application for sensor and smartphone quality evaluation and user education

Figure 3.8: Quality Score Evaluation Workflow

made available online [32] as well.

Table 3.8: Sensor clustering based on sensor quality metrics. Acc.- accelerometer, Gyro. - gyroscope, Prox. - proximity, Bar. - barometer, Comp. - compass.

Sensor Type	Acc.	Gyro.	Prox.	Bar.	Comp.
Total sensor models	52	14	19	5	16
Number of clusters	2	2	2	2	2
Cluster 1 (Good) (# of sensors)	17	2	6	2	11
Cluster 2 (Bad) (# of sensors)	35	12	13	3	5

Table 3.9: Device classification based on sensors quality. Acc.- accelerometer, Gyro. - gyroscope, Prox. - proximity, Bar. - barometer, Comp. - compass.

Sensor type	Acc.	Gyro.	Prox.	Bar.	Comp.
Cluster 0 (No data)	449	142	315	15	128
Cluster 1 (Good)	119	89	34	2	88
Cluster 2 (Bad)	343	24	18	20	79

Table 3.8 presents the results of the clustering of the accelerometers, gyroscopes, proximity sensors, magnetometers, and barometers based on their sensitivity, non-linearity, and noise values. Since in the developed knowledge base each sensor type has its own table, we do not include sensor type in the table. Table 3.9 presents the results of our device classification based on their embedded sensors' quality. All devices were clustered based on one of the five described sensors; for example, there are 119 devices with an accelerometer sensor of good quality and 343 with a poor quality accelerometer.

3.3.4 Knowledge Base Employment

The created knowledge base can be used in a variety of domains, for example, in the crowd-sensing, in which it will allow choosing and fusing the best sensor sources for particular application design. Another example is a multicriteria optimization of sensor systems and networks. One of the traditional ways to improve the overall data quality in SCADA and other data collection systems is fusing numerous data streams that are originated from multiple networked sensor platforms. Various methods of sensor networks optimization have been proposed [13, 99, 114]. These methods could be improved if sensor quality is evaluated, and these estimates are utilized in the sensor network design and optimization with the goal to improve the overall quality of the collected sensor data [67]. In the section 5.10.1, we present a tool that implements this knowledge base on Android OS smartphones.

3.4 Conclusions

In response to the major research question posed in this chapter ***"How to integrate DQ evaluation into a synergetic DQ and security evaluation framework?"****, we achieved the following* ***major*** *research results, which are further described in details:*

- *We studied the variety of data quality and security metrics and developed a* ***multilevel hierarchical knowledge*** *structure that facilitates the framework development and adaptation to new applications;*
- *We investigated* ***the knowledge graph concept*** *and developed the methodology of its employment as the novel implementation platform of our DQ evaluation framework;*
- *We demonstrated an application of the developed novel methodology and applied it* ***in the case DQ evaluation of the sensor originated data*** *in crowd-sensing applications.*
- *We formulated* ***an initial knowledge graph vocabulary****, which facilitates the framework usage in crowd-sensing tasks for evaluation sensor originated data.*
- *We investigated and analyzed sensors incorporated in the modern mobile devices, produced the comprehensive knowledge base and* ***made it available for community use****.*

We developed a novel systematic pipeline for the integral DQ and security evaluation framework. We also formed the comprehensive, multilevel hierarchical structure of the DQ and security metrics. This structure allows framework extension to new applications and inclusion of new metrics without severe framework modification. The framework integrates metrics from the two major metrics group: data correctness and data security. We also developed the approach to incorporating the security and privacy aspects into the conventional DQ evaluation pipeline.

The proposed framework includes several vital components: metrics selection, metrics integration calculus, and application context consideration. We formed and examined a novel idea of a knowledge graph employment to unite all these components in the framework. As we rendered in this section, the knowledge graph is a very effective way of merging all the framework's components. The knowledge graph employment is also the answer to the question of how to incorporate security and privacy aspects into the conventional DQ evaluation pipeline.

We harnessed the adaptable schema of the knowledge graph to support the overall framework flexibility. We can adapt the framework to new user needs by augmenting the knowledge graph's vocabulary with new entities that are relevant to the application context. In addition, due to the knowledge graph employment, we achieved the utilization of the versatile calculi for the DQ evaluation. In the future, we can automate the selection of the metrics integration calculus depending on the application context.

We also achieved knowledge accumulation on two levels. On the first level of knowledge accumulation, we employed the knowledge graph to store data objects. Saving data objects along with their DQ scores in the knowledge graph enables knowledge accumulation and improves the overall framework efficiency. The second level is reached when the framework is used in new application domains. In this case, we extend the knowledge graph vocabulary that later can be used in similar applications.

In addition, to provide the sensor comparison and recommendations on the sensor applications based on their quality, we created and made available to the community the knowledge base of a wide range of sensors embedded in various mobile devices and their characteristics. This research on sensor quality facilitates closer collaboration between sensor systems and network developers and sensor users as well as mobile device owners. The sensor analysis reveals that while the quality characteristics of embedded sensors are lower than the specialized individually calibrated instruments in professional use, they may vary significantly. Based on the analysis, all sensors were classified into good and poor, and the integral quality indicator was calculated for each device.

The presented framework is a “skeleton” that later can be easily expanded and modified for various new applications.

Chapter 4

Developed Techniques for Incorporating Security and Privacy into Data Quality Evaluation Framework

In this chapter, we address the problem of ***"How to develop effective and efficient security and privacy evaluation methods targeted for their implementation on resource-constrained mobile devices and incorporate them into the overall DQ evaluation pipeline?".*** *This research question is subdivided into several questions that are answered in each section of this chapter. Section 4.1 presents the developed calculus for evaluating each security component, such as data integrity, data privacy, and data availability. In sections 4.2 - 4.5, we demonstrate an application of the developed methods in the case of the novel violations of the security and privacy in the mobile devices. We present a novel attack on user privacy, describe it and derive a formalized model of this threat, investigate attack models and scenarios, and conduct an empirical study of this attack and its detection with the help of DQ metrics. We describe an application of machine* *learning techniques and artificial neural networks models to develop attack classifiers based on DQ technological signals in mobile devices.*

4.1 Data Security Evaluation Components Formalization

In our systematic approach to DQ evaluation, the data security component plays an essential role in the overall DQ evaluation [26, 27, 69, 72, 76, 78]. This section presents a detailed description of the structure of the data security evaluation component, we developed, such as the metrics and their hierarchy, the metrics integration calculus, and its implementation.

The proposed data security model has a hierarchical structure that is composed by the data availability, data integrity, and data confidentiality. The data integrity component includes data integrity of the source and data integrity of the communication channels. Employment of the knowledge graph allows investigating the interrelationship between components of different branches (i.e., integrity and confidentiality branches), which makes the developed model more complete.

4.1.1 Data Integrity Metrics

"Data source integrity" takes into account various parameters of the data source (i.e., sensor platform) that influence the overall data integrity in multiple ways [63, 70]. Table 4.1 presents calculus formulae that are used for data security calculation.

Since our implementation focuses on Android OS-based devices, "data source integrity" incorporates such hierarchy branches as application security, a device feature security, and a sensor integrity. Sensor integrity takes into account those smartphone parameters that may influence the trustworthiness of the data that comes from a smartphone's sensors. This component includes such parameters as bootloader status, "root" status, developer menu status, and device lock status along with the type of the locking method (e.g., password, graphical pattern, fingerprint, facial identification, etc.). Unlocked bootloader status endangers data integrity by allowing “rooting” a device or installing a modified firmware. The formula 4.1 (see table 4.1) calculates the probability that a device will have an unlocked bootloader.

Root access represents an even more significant threat to data integrity. An application that has super-user privileges (e.g., “root”) can modify any data on a smartphone. For example, data from such motion sensors as an accelerometer and a gyroscope can be modified through editing the corresponding files in the “/dev/input/” directory. This directory is accessible only by a user that has super-user privileges. The formula 4.2 (see table 4.1) calculates the probability that a device will have “root”. This formula reflects the “unlocked bootloader” requirement for a smartphone “root”. In other words, to root a smartphone, its user has to unlock the bootloader at first.

“Developer options” menu can be used to temper the position sensor data, for example, to spoof location. In addition, the “developer options menu” has to be enabled to turn on Android Debug Bridge (ADB), which allows spoofing data from such sensors as touch screen [92]. The formula 4.3 (see table 4.1) calculates the probability that a device has an enabled developer options menu.

“Device lock” (i.e., screen lock) prevents unauthorized access to the device. An unauthorized user may modify or spoof data collected by a smartphone’s sensors without legitimate user awareness. An intruder may gain access to the unprotected device, install malicious malware, spyware, or manually edit and fake data. Moreover, these actions may go unnoticed, and the device will stay compromised for a more extended period of time. The formula 4.4 (see table 4.1) calculates the probability that a device does not have a screen lock mechanism enabled.

Formula 4.5 (see table 4.1) integrates together all metrics in this “sensor integrity” sub-branch.

“Devices feature security” (see figure 3.2) incorporates those parameters of a device that are related to the firmware and hardware. The latest versions of Android OS likely have fewer known vulnerabilities than older versions, and it is important to keep the devices’ OS version updated. Some vendors, for example, Google and Samsung, periodically release patches that fix discovered vulnerabilities. The latest security patch that is installed on a device makes a device more secure and increases the overall security level. The device model, along with an OS version and installed security patches, can provide information about device security in terms of discovered vulnerabilities. To calculate these scores, formulas 4.6, 4.7, and 4.8 are used (see table 4.1). Formula 4.9 (see table 4.1) integrates together metrics in this sub-branch and produces "firmware score" of the device.

The "application security" branch represents threats to the data integrity and data privacy brought by installed applications. This branch is also contributing to the data confidentiality evaluation as well as to the data availability evaluation. "Application security" integrates metrics that are related to threats that are coming from the installed applications. These metrics include allowing application installation from unknown sources, a number of the blacklisted applications, a number of potentially harmful applications, and the dangerous application permission score, which is calculated through the permission analysis of all applications installed on the smartphone.

Allowing the application installation from unknown sources increases the risks of malware installation, such as spyware, ransomware, botnet-ware, dedicated software that modifies sensor data, etc. These malware pieces have different goals and have to be taken into account in different branches of security evaluation. Malware that alters or fakes sensor data influences on "data integrity" branch. To calculate the "unknown sources" score, the formula 4.10 is used (see table 4.1).

Sometimes, an application may be included in the so-called blacklist based on various reasons. For example, a developer of an application might have a bad reputation; the application is over-privileged, or other reasons. An over-privileged application is potentially dangerous since it has already been granted a lot of permissions, which are not used yet. It may be unclear why it needs these many permissions and how they will be used in the future. Later, this application can be updated, and all these granted permissions may be used with malicious intentions. To calculate the "black listed app score", formula 4.11 is used (see table 4.1).

While the "potentially harmful applications" score has a meaning similar to the "number of blacklisted application", it is determined differently. To decide if an application is potentially harmful or not, the safety-net library from Google, which is a part of Google Play Protect, is used [6]. Potentially harmful applications are those applications that may harm your device or do something undesirable with the data on your device. This library scans [50] all installed application and is able to detect such application type as:

- Backdoors: Apps that let hackers control your device, giving them an unauthorized access to your data.
- Billing fraud: Apps that charge you in an intentionally misleading way, like premium SMS scams or call scams.
- Spyware: Apps that collect personal information from your device without consent.
- Hostile Downloads: Apps that download harmful programs, often through bundling with another program.
- Trojan Apps: Apps that appear benign (e.g., a game that claims only to be a game) but actually perform undesirable actions.

Formula 4.12 calculates the "potentially dangerous score" of the device.

The "dangerous application permissions" metric (see figure 3.2) may employ machine learning techniques to analyze the permission distribution on the device and evaluate the application security level.

All these metrics are integrated into the "application security score" using an expert system and fuzzy rules (see formula 4.13 in table 4.1) and are submitted to the cloud service. This cloud service keeps track of the device integrity score. Analytics on the cloud side include building a trend of the integrity score change and score comparison with other similar devices. The cloud employment allows analytics extension over time without a significant modification on the client-side. The mobile client may have thousands of instances, and their modification may introduce a challenge, while the cloud service has only one instance and totally under a developer's control. The expert system takes these cloud service analytics as the input, which creates a recurrent architecture of the data integrity evaluation. Expert system employment allows handling partial metrics availability and evaluating data integrity even when only a few parameters are available.

Data integrity of the communication channels takes into account such parameters as a type of communication channel encryption, data transfer protocols, and data nature. The type of encryption describes how long it may take to break encryption and modify data. If it is possible to modify data without decryption, what are the costs of breaking encryption in the computational and economic sense? Data transfer protocols indicate if they have a mechanism to prevent unauthorized data modification and how difficult it is to do that. This parameter, to some degree, is related to the channel encryption since some encryption types have data integrity protection mechanisms. Data nature (or origin) may influence data integrity as well. Sometimes it is pointless to modify or fake some data objects due to the nature of the data. Formula 4.14 (see table 4.1) calculates data integrity based on the metrics of this sub-branch.

4.1.2 Data Confidentiality Metrics

Data privacy estimates the level of data leakage threat and consists of two branches: system privacy and network privacy. System privacy evaluation takes into account issues related to the malware presence and partially is derived from the "application security" branch of the "data integrity evaluation" component. System privacy considers such threats as spy-ware and the "colluded applications" attack, which is a more sophisticated type of a spy-ware. "Colluded applications" attack detection involves various techniques that are available on a particular piece of hardware. In addition, system privacy may take into consideration the probabilities of installing malware based on system parameters received from the "data integrity" component.

Formula 4.16 (see table 4.1) computes the system privacy of a device. $P_{Mal.GPlay}$ - is the probability that a user installed a malicious application from Google Play Store. This probability may be calculated as a ratio of detected malicious applications on the Google Play Store to all applications in Google Play. For example, in September 2018, 145 malicious apps were detected in Google Play [36] among 2600000 total applications [125]. In this case, the probability of getting malicious application from Google Play equals to $P_{Mal.GPlay} = 145/2600000 = 5.58 \times 10^{-5}$

$P_{Mal.Unkn.src}$ is the probability that a user installed a malicious application from an unknown (unverified) source. This probability may be calculated as a ratio of the number of all detected malware to the number of all Android OS applications. In addition, we have to consider that application installation from unknown sources requires the corresponding system setting.

$P_{Col.app}$ is the probability that a user has installed one of the applications that are capable of collusion. This probability can be calculated as a ratio of the number of known colluded apps to the number of all applications. Since the "application collusion" requires two or more applications, this ratio has to be squared.

In addition, the system privacy is also affected by other security mechanisms that prevent an unauthorized access to the device. Root access and an absence of the device lock may negatively affect the system privacy of a device. Root access may lead to cases when a software that has superuser privileges accesses private data and transmits it to the third parties. It is similar to spyware, but a malware with superuser privileges may be much more harmful. The disabled device lock may simply lead to a case, when an unauthorized user gains a physical access to a device and may access all data they want, such as photo-image gallery, location history, text messages, passwords (if they are stored in non-encrypted form), etc. Aforementioned parameters are important to system security evaluation.

The network privacy considers two major parameters such as a channel encryption type and the probability of correct identification of data endpoints (see formula 4.17 in table 4.1). In some cases, it is crucial to keep data endpoints hidden. For example, TOR (The Onion Router) network may hide data endpoints. However, due to a website fingerprinting, a user's destination may be revealed with some degree of confidence [121]. VPN services may conceal data points, as well. The importance of this metric is profoundly affected by the channel encryption type. If an attacker may find out data endpoints from the traffic due to a weak transmission channel encryption, there is no point in performing the fingerprinting attack. For example, HTTPS traffic is encrypted, but data endpoints are open to anyone who sniffs the traffic.

4.1.3 Data Availability Metrics

Data availability is another important branch in data security evaluation. It also reflects the reliability of a sensor platform. It considers the unavailability of the data due to various reasons, mostly due to a malicious attack. This branch includes such metrics as a possibility that data is not available, data delivery delay, and measurement's time resolution.

Data may not be available because of various reasons, for instance, because of data was lost, because of the ransomware, or because the sensor platform was infected with a specific malware that transforms this sensor platform into a part of a botnet. Converting device into a part of a botnet may lead to inaccessibility of a device's resources or significant latency. Ransomware is a type of malware that encrypts data on the device and requires a ransom to decrypt it back. Some decryption keys may be obtained by reverse engineering and generated without paying a ransom; in some cases, the ransom has to be paid; and sometimes, the key just does not exist or communication with extortionists is lost, which leads to a very low probability of successful data decryption.

Data backups also influence "data availability". In the case of ransomware and simple data loss, data may be recovered from the backup, and the weight of the ransomware factor will be much smaller.

However, data backup cannot resolve network-related issues. This metric is partially influenced by the network confidentiality of the data confidentiality branch as well. If a network is broken, the data availability level goes down; however, the confidentiality level of the network may go up. This simple example demonstrates that we keep the conventional relationship between data availability and data confidentiality: if data is not available, its confidentiality is very high since nobody can access it. “Network issues” metric is affected by three parameters: denial of service (DOS) attack, accidental network issues, and botnet malware.

DOS attacks may partially affect or completely stop the functionality of the network. Moreover, the “network issues” metric affects network confidentiality through this parameter.

Another parameter that affects the data availability from a network perspective is a “Bot Net Malware” parameter. This type of malware converts a sensor platform into a part of a botnet that may perform various malicious actions. The most popular goals of a botnet are: distributed DOS (DDOS) attacks and mining crypto-currencies. Participating in a DDOS attack may reduce network bandwidth for the actual data objects flow and decrease its data availability. Participating in a distributed crypto-currency mining may consume too many computational resources of a sensor platform and make the sensor-platforms inaccessible. It is also partially affected by the “application security” of the data integrity branch. Since it is a malware, application security has a direct impact on this aspect. This parameter can be estimated as a rare event probability as well, since converting a mobile into a botnet part requires the concurrence of multiple events, such as an installation of the particular type of malware, an absence of anti-malware software, third-party interest in performing DDOS or mining crypto-currency, etc.

Accidental network issues are not directly related to data security but heavily affect it; that is why we include it into the data availability. These issues include all problems with a network that has a non-malicious origin, such as a power absence in transmitting equipment due to natural disasters (earthquake, flood, etc.), time of network congestion due to some social event, etc. While some of these events may be predicted to some degree (social event), others may have spontaneous character and can be assessed only from the rare event probability perspective.

Formula 4.18 (see table 4.1) represents data unavailability as a simple sum of other probabilities due to other probabilities, much less than one. Otherwise, it has to be divided by three.

Since, data security is an integration of data confidentiality, integrity, and availability, formula 4.19 (see table 4.1) represents their integration through logical functions of an expert system.

Table 4.1: Data Security Metric Formulas. $\underset{L}{\wedge}$ is metrics integration through an expert system

Metric and its description	**Formula**
Probability that device has unlocked bootloader. NumOfUnlckd - number of all devices in an organization that have unlocked bootloader, and NumOfAllUsers - a number of all users in the organization	$P(btldrUnlckd) = \frac{NumOfUnlckd}{NumOfAllUsers} \quad (4.1)$
Probability that device has root access. NumOfRoot - number of all devices in an organization that have a root access	$P(root) = P(btldrUnlckd) \times \frac{NumOfRoot}{NumOfAllUsers} \quad (4.2)$

Continued on next page

Table 4.1 – *Continued from previous page*

Metric and its description	**Formula**
Probability that device has enabled developer options menu. NumOfdevOptn - number of all devices in an organization that have an enabled developer option menu, SmplKoef - simplicity coefficient, which reflects relative easiness of enabling a developer option menu. This coefficient depends on the average technical experience of organization members	$$P(devOptn) = \frac{NumOfdevOptn}{NumOfAllUsers} \times SmplCoef \quad (4.3)$$

Continued on next page

Table 4.1 – *Continued from previous page*

Metric and its description	**Formula**
Probability that device has no screen lock. NumOfscrnLck - number of all devices in an organization that have no a screen lock, incnCoef - inconvenience coefficient, reflects the inconvenience of using a screen lock. For example, screen pattern is more convenient than password, and fingerprint sensor is more convenient than pattern. This coefficient indicates the likelihood that a user will decide to turn of the screen lock	$$P(scrnLck) = \frac{NumOfscrnLck}{NumOfAllUsers} \times incnCoef \quad (4.4)$$
Sensor security score	$$\begin{gathered} SensorSecurity = 4 - P(btldrUnlckd) - \\ - P(root) - P(devOptn) - P(scrnLck) \end{gathered} \quad (4.5)$$

Continued on next page

Table 4.1 – *Continued from previous page*

Metric and its description	**Formula**
OS version score	$$VerScore = \begin{cases} 0 \quad \text{if } CurVer > VerThreshold \\ MaxVerScore- \\ \qquad - MaxVer - CurVer \\ \qquad \text{if } CurVer \leq VerThreshold \end{cases} \tag{4.6}$$
Security patch score	$$PatchScore = \begin{cases} 0 \quad \text{if } CurVer > VerThreshold \\ MaxPatchScore- \\ \qquad - MaxVer - CurVer \\ \qquad \text{if } CurVer \leq VerThreshold \end{cases} \tag{4.7}$$
Device model score	$$ModelScore = \begin{cases} 0 \quad \text{if } CurVer > VerThreshold \\ MaxVerScore- \\ \qquad - MaxVer - CurVer \\ \qquad \text{if } CurVer \leq VerThreshold \end{cases} \tag{4.8}$$
Overall firmware score	$$\begin{aligned} FirmwareSecurity = VerScore+ \\ + PatchScore + ModelScore \end{aligned} \tag{4.9}$$

Continued on next page

Table 4.1 – *Continued from previous page*

Metric and its description	**Formula**
App unknown sources score. NumOfUnkn - number of all devices in an organization that allows unknown application sources	$UnknSrcScore = \begin{cases} 0 & \text{if UnknSrc is ON} \\ 1 - P(UnknSrc) & \\ & \text{if UnknSrc is OFF} \end{cases} \quad (4.10)$ $P(UnknSrc) = \frac{NumOfUnkn}{NumOfAllUsers}$
Black listed app score	$BlkLstScore = \begin{cases} 0 & \text{if Number ¿ Threshold} \\ MaxScore- & \\ \quad -(Num - Threshold) & \\ \quad \text{if Number} \leq \text{Threshold} & \end{cases} \quad (4.11)$
Dangerous permission utilization score	$PDanScore = \begin{cases} 0 & \text{if Number ¿ Threshold} \\ MaxScore - (Number- & \\ \quad - Threshold) & \\ \quad \text{if Number} \leq \text{Threshold} & \end{cases} \quad (4.12)$
Application security score	$AppSecScore = UnknSrcScore \underset{L}{\wedge} BlkLstScore \underset{L}{\wedge} \underset{L}{\wedge} PDanScore \underset{L}{\wedge} PermScore \quad (4.13)$

Continued on next page

Table 4.1 – *Continued from previous page*

Metric and its description	**Formula**
Data integrity score	$DataIntegrity = CloudScore \underset{L}{\wedge} SensorSecurity \underset{L}{\wedge} \underset{L}{\wedge} AppSecScore \underset{L}{\wedge} DevSecurity \underset{L}{\wedge} \underset{L}{\wedge} CommChannel$ (4.14)
Spyware installation probabilities. $P_{Mal.GPlay}$ - is the probability that a user installed a malicious application from Google Play Store. $P_{Mal.Unkn.src}$ - is the probability that a user installed a malicious application from an unknown (unverified) source. $P_{Col.app}$ - is the probability that a user installed one of the application that is capable of collusion.	$P_{Mal.GPlay} = \frac{N_{mal.GPlay}}{N_{All.GPlay}}$ $P_{Mal.Unkn.src} = \frac{N_{mal}}{N_{All}} \times P(UnknSrc)$ $P_{Col.app} = \frac{N_{col}}{N_{All}}^2$ (4.15)
System privacy score	$SystemPrivacyScore = 3 - P_{Mal.GPlay} - - P_{Mal.Unkn.src} - P_{Col.app}$ (4.16)

Continued on next page

Table 4.1 – *Continued from previous page*

Metric and its description	**Formula**
Network privacy score. P_{Dest} - represents a probability of successfully identification of a website that visits a user and $EncryptLevel$ - represents a level of communication channel encryption	$NetworkPrivacyScore = EncryptLevel - P_{Dest} \quad (4.17)$
Data unavailability probability	$P_{Ransom} = \frac{NumOfRansom}{AllUsers}$ $P_{DataLoss} = \frac{NumOfLoss}{AllData}$ $P_{NetPrbm} = (\frac{NumOfDOS + NumOfAcdnt}{2 \times AllTime} + + \frac{BotNet}{AllUsers})/2$ $P_{DataNA} = \begin{cases} 0 & \text{if Data Backed Up} \\ P_{Ransom} + P_{DataLoss} + P_{NetPrbm} & \text{if NOT Data Backed Up} \end{cases} \quad (4.18)$
Data security score	$DataSecurity = Confidentiality \wedge_{L} Integrity \wedge_{L} \wedge_{L} Availability \quad (4.19)$

4.2 Privacy Evaluation Components Formalization

In the following sections, we present a novel, sophisticated attack on users' privacy and its influence on sensor platforms. We formalize the attack, analyze its scenarios, derive attack model, and investigate ways of its detection. We render the place of this attack detector in the systematic security evaluation pipeline.

As was described in section 2.4, "colluded applications" attack threats users' privacy and may decrease the "confidentiality component" score of a mobile phone's (sensor platform) security. However, this is a sophisticated and complex attack that requires significant efforts from a piece of malware and its developer (i.e. "colluded applications" attack involves two or more malicious applications). Hence, executing this attack in an environment where data can be leaked using more straightforward approaches is meaningless. Therefore, we perform a lightweight yet efficient overall smartphone security and privacy evaluation. A high-security score achieved indicates that users did everything possible to protect their privacy, and the attack detector can be started.

The "colluded applications" attack detector is incorporated into a complex pipeline (see figure 4.1) that includes the following stages:

1. As the first stage, we have to perform the preliminary periodic analysis that includes system security evaluation with AI-based calculus that examines the device and system security, collects a number of metrics, calculates the security score, and provides a system owner with the recommendations on what needs to be implemented. We will not proceed to the next stages until these recommendations are implemented.

2. On this step, we will also conduct a misuse based attack detection with available tools.

3. We will start system parameter monitoring only on the systems with a reasonably high-security level with preventive and simple detection mechanisms already implemented.

4. Anomaly detection event will trigger post-detection analysis, which will be performed in real-

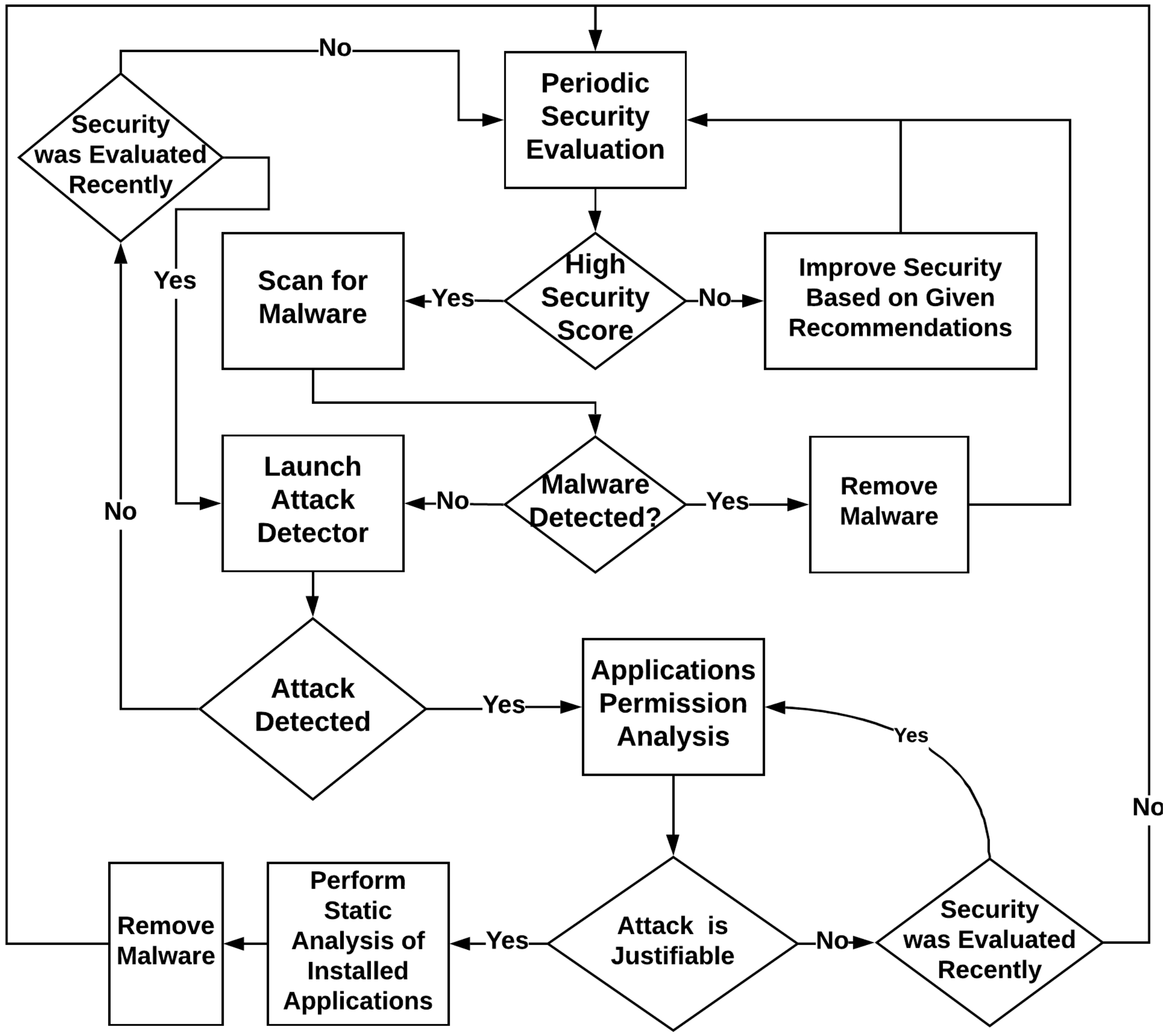

Periodic Security Evaluation
Security was Evaluated Recently
No
Yes
High Security Score
Scan for Malware
Improve Security Based on Given Recommendations
Malware Detected?
Launch Attack Detector
Remove Malware
Attack Detected
Applications Permission Analysis
Attack is Justifiable
Perform Static Analysis of Installed Applications
Remove Malware
Security was Evaluated Recently
Yes
No

Figure 4.1: The overall methodology of security and privacy improvement of a smartphone

time as well and will investigate specific details of the involved applications and system parts in order to confirm an attack case.

For the initial security evaluation, we integrate such security-related metrics as application installation from unknown sources, Android OS version, screen lock, enabled “Developer options menu”, root access, bootloader state (locked/unlocked), and presence of potentially harmful applications. To obtain these parameters, we developed a special Android OS library and application that is based on this library (see details in chapter 5). The overall security score is calculated using the following formula:

$$\begin{aligned} SecurityScore = M_{SL} + M_{V} + M_{US} + M_{PH} + M_{DO} + \\ + M_{BI} + M_{CT} \times M_{BI} \end{aligned} \tag{4.20}$$

Table 5.1 provides more details about formula 4.20.

4.2.1 "Colluded Applications" Model Formalization

Since we use "colluded applications" attack as an example of novel attacks, we need to properly describe and explain this threat to a user's privacy. This section contains a formalized description of the attack, its model, exploitation scenarios, and developed tools that try to address this particular attack type.

4.2.2 "Colluded Applications" Attack Formalized Description

Let the applications A and B belong to a set of installed applications S. A has the permission set P_A and B has the permission set P_B. P_A consists of the normal permission subset P_{NA} and the dangerous permission subset P_{DA}. P_B consists of the normal permission subset P_{NB} subset and the dangerous permission subset P_{DB} (see figure 4.2). P_{DA} and P_{DB} represent subsets of dangerous permissions DP. P_{DA} and P_{DB} are not equal. A generates data object D. To generate D a permission p_D is required and to leak D to third parties a permission p_L is required. A transmits D to B ($t_A(B, D_{p_D}, background)$) while A and B are in the background. If P_{DA} includes p_D and does not include p_L, P_{DB} does not include p_D but includes p_L, then A and B are colluded applications. Definition of colluded applications can be written with the following statements:

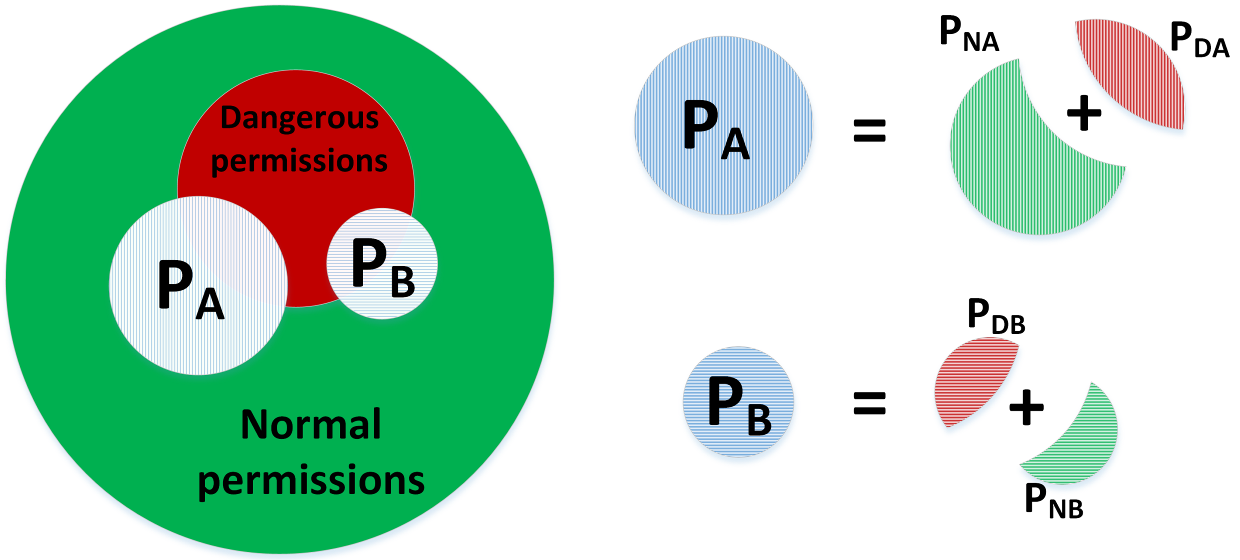


Figure 4.2: Application A and B permission sets

$(A, B \in S) \wedge (P_{DA}, P_{DB} \subset DP) \wedge P_{DA} \neq P_{DB} \wedge (p_D \in P_{DA}) \wedge (p_D \notin P_{DB}) \wedge (p_L \in P_{DB}) \wedge (p_L \notin P_{DA})\wedge$
$\wedge t_A(B, D_{p_D}, background) \rightarrow A$ and B are colluded.

It is important to notice, that $P_{DA} \neq P_{DB}$. In the case $P_{DA} = P_{DB}$ there is no sense for application collusion. Applying this rule before further analysis may reduce the search space of analyzed applications. In addition, this definition is true for Intra-Library Collusion (ILC) attack [129] as well, since instances of an embedded library are part of host applications.

4.2.3 "Colluded Applications" Attack Model

We assume that the goal of the attack is the same malicious actions as for single malware applications and include information theft, money theft, service misuse, sabotage, denial of service, ransom, etc.

An attacker is an agent that tries to get a user's private data without a user's permission. The

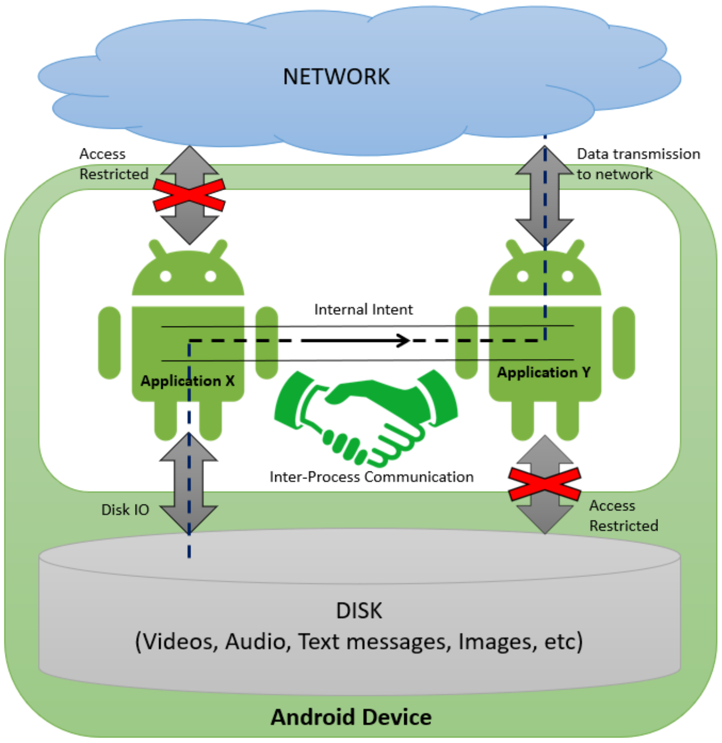


Figure 4.3: Attack model data flow

motivation of the attack is to obtain a user's private data without the user's awareness. The attacker exploits "colluded applications" vulnerability that may allow an attacker to obtain a user's private data with bypassing permission requests.

The attack is an action that results in unauthorized data flow from the device's source to the attacker's destination through colluded applications without a user's permission (see figure 4.3).

Further, we present four use cases that support our attack model and its formalized description. Use case #4 is implemented in scenarios that are presented in section 4.4.1.

4.2.4 Use Case 1. No Collusion. Applications Cannot Leak Data.

Let the application A be a camera app. To be able to take a photo (to generate a data object D), it requires and obtains a camera permission p_D, where p_D belongs to the dangerous permission group ($p_D \in DP$). The set of the dangerous permissions P_{DA} of the app A does not include any permission that allows photo image leaking to the third parties ($p_L \notin P_{DA}$). To allow photo editing, the application A transmits a photo image D to the application B, which is a photo editing software. The permission set P_B of the application B does not include the permission p_D ($p_D \notin P_B$) and B obtains photo image D without the camera permission p_D. However, the permission set P_B also does not include permission that allows photo image leaking ($p_L \notin P_B$). Since application B cannot leak data object D and communication between A and B is not malicious, applications A and B are not "colluded applications".

4.2.5 Use Case 2. No Collusion. Legitimate Use of Inter-application Communication.

Let the application A be a shopping list app. It is able to search for products in the remote database with the Internet permission p_L. However, p_L is the only permission that A has. Application B is a bar-code scanner. Application B has an open API that allows using the app's functionality by other apps. In order to scan bar-codes, B has only one permission, which is camera permission p_D. p_D allows generating a data object D, which is a photo image. The application A can request from B to scan a bar-code to simplify the process of products adding to the shopping list. The application B returns the raw image D and processed bar-code in the form of text to the A through "intent" objects. The application A uses only text and discards the image. However, at some point, A possesses the data-object D for which it does not have the permission. However, since B explicitly allows using its functionality and A discards D as soon as possible, A and B are not "colluded applications".

4.2.6 Use Case 3. No Collusion. Application Collaboration Does Not Compromise Privacy.

Let the application A be a shopping list app. It is able to search for products in the remote database with Internet permission p_L. A has a functionality of bar-code scanning through a phone's camera. To scan bar-code, A has camera permission p_D, which allows generating data object D (photo image). A QR-code scanning application B that can process various types of bar-codes and QR-codes, and provides open API that can be used by other apps is also installed on the phone. When A requests to scan a bar-code, B returns the raw image D and a processed bar-code. A uses only text and discards the image. Since A already has the permission p_D that allows generating D, there is no reason for A and B to be considered as "colluded applications".

4.2.7 Use Case 4. "Colluded Applications" Attack.

Let application A be a camera app. In order to take a photo (to generate a data object D), it requires and gets the camera permission p_D. The set of the dangerous permissions P_{DA} of the app A does not include any permission that allows photo image leaking to the third parties ($p_L \notin P_{DA}$). Also, on the phone, a weather forecast application B is installed. The application B has the Internet permission p_L, which potentially allows information leakage. When A detects that B has been installed on the phone, it sends to B data object D, and B is able to receive D and send it to the remote server through the Internet. Since B possesses D without the appropriate permission and leaks it to the third parties, A and B are "colluded applications".

4.3 Other Approaches to the "Colluded Applications" Attack Detection Feasibility

Various AI and ML techniques have been investigated for this attack detection. In our preliminary empirical study, we employed multiple artificial intelligence techniques such as various types of decision trees and random forest. Unfortunately, classifiers that are based on these AI techniques performed relatively poorly, even despite the collected technological data had minimum background noise. The achieved accuracy is less than 90% on clean data without the background noise.

Another approach to the attack detection that has been reported in the literature is the application of Hidden Markov Models (HMM). An HMM is a doubly stochastic process with an underlying not observable directly that can only be seen through another set of stochastic processes that produce the sequence of observed symbols [107]. HMM-based attack detectors may show a relatively good performance. Still, they require more time to detect an attack and give a higher error rate in comparison to conventional misuse detection techniques [25]. This delay makes the development of a real-time attack detector difficult. One of the ways to decrease the error rate of an HMM in intrusion detection systems (IDS) is by building an ensemble of HMMs [25, 30]. However, this improvement in accuracy increases the overall model complexity.

The attack detection task, in its essence, represents a sub-class of the classification problems. While some types of HMM may outperform IDS based on such techniques as K-NN and SVM [87], it is hard for HMM-based IDS to compete with systems that are based on neural networks, especially recurrent neural networks [51] in classifier's performance [85, 90, 100]. Classification accuracy of HMMs can be improved, for example, by combining HMM with neural networks. However, this integration increases the model complexity as well and does not allow utilizing dedicated NN chips such as visual core in modern smartphones [48]. Another limitation in the HMM employment is that HMMs make the Markovian assumption. In other words, the model assumes that the current state depends only on the previous state, which may not be the case in the data collected during cyber-attacks. These HMM limitations make neural networks more preferable for applications in modern smartphones [145]. In addition, the non-Linearity of NN-based, and RNN-based models, particularly, make them more expressive and more adaptive at learning from real-world data [3, 62, 79]. The recurrent neural networks, in general, gives higher accuracy and overall performance in such tasks as intrusion and attack detection [144, 147].

4.4 Attack Detection Using Memory Consumption and CPU Utilization

In our attack detector architecture initially, we employed two types of technological signals [64, 66] to feed up into a classifier: random access memory (RAM) consumption and CPU utilization rate. In this study, 45 different attack scenarios were examined. Memory and CPU usage were logged in each of the scenarios. The attacks were conducted on an Android emulator for the Nexus 6 device. The device ran Android version 6.0 (API 23) and used a 1.5 GB of RAM. The scenarios consisted of different data types transmission such as contact list, SMS, audio, image, and video files of various sizes. Multiple combinations of block sizes were used. The individual attacks were run for longer and shorter time periods. In order to obtain the comparison base, some scenarios recorded the application usage under a "non-colluded" (no attack/normal) situation. In addition, we ran a legitimate user process such as Internet Browser during the attacks. While this process added additional "noise" to the memory consumption and CPU usage, it imitates a normal user's activity on the device. Finally, some overlapping attack scenarios were run. We tried multiple statistical analysis techniques, such as exploratory data analysis and principal component analysis, to study a potential influence of the background processes on the attack recognition. This analysis confirms that collected data has a lot of noise (see figure 4.4 for an audio scenario) that reflects real device operational patterns.

All collected data were organized into the "colluded applications" attack dataset, which was made

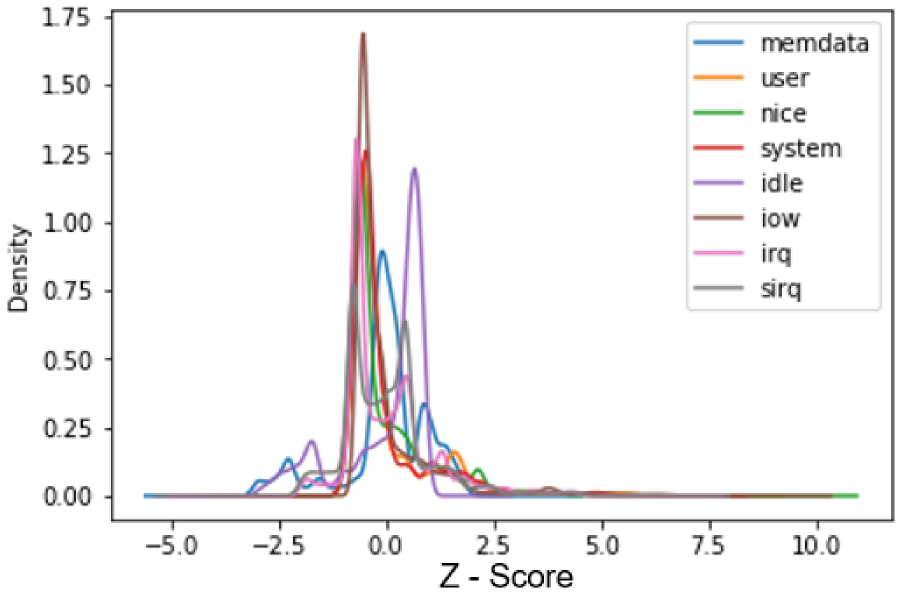


Figure 4.4: Nexus5x: kernel density estimation for an audio scenario.

available for public use. This dataset can be used in future research of detection and mitigation “colluded applications” attacks.

4.4.1 Attack Scenarios Description

Five attack scenarios were selected for further analysis: contacts list data transmission, text messages transmission, photo image transmission, audio data transmission, and video data transmission. These scenarios were implemented on seven smartphones: Google Nexus 5X, Google Nexus 6P, OnePlus 3, OnePlus 5T, OnePlus 6, OnePlus 6T, Google Pixel 2XL. The collected data resulted in a second dataset that is also made available.

Scenario 1. Contacts data. The contact list was transferred from the source application to the sink application with batches of 20 contacts and a delay between batches with ten milliseconds. The attack lasts approximately 500 milliseconds for 540 contacts. Attack duration varies from phone to phone and also depends on a number of contacts.

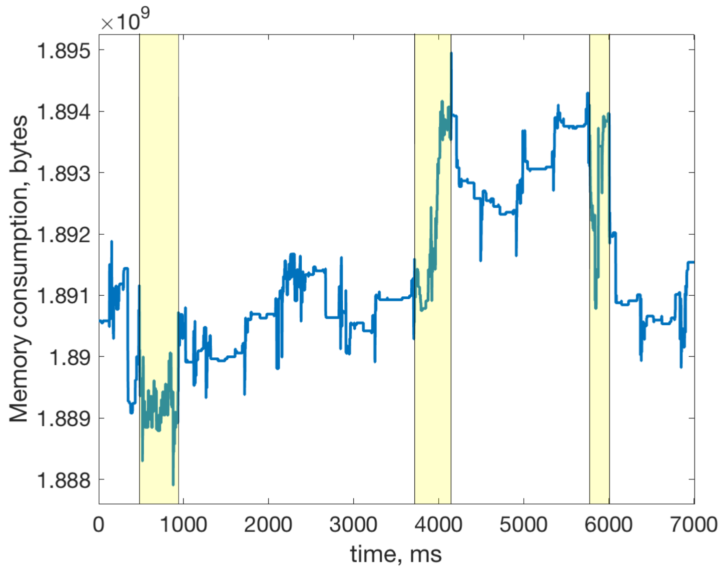


Figure 4.5: Memory consumption during the “attack” and “no attack” periods for contact data scenario. Attack periods are marked with yellow.

Audio data. In this scenario, a 13MB audio file was transmitted with 100KB chunks with a 15ms delay between chunks. The attack duration is approximately 1.5 seconds.

Image data. In this scenario, a 10MB file was transferred between applications in chunks of 50 KB and a 5ms delay between chunks. The duration of the attacks is approximately 1.3 seconds.

4.4.2 Attack Results and Their Analysis

Contacts data.

Figure 4.5 shows the memory usage in bytes during the periods of no attack and attack, which was sending ten contact details at a time. As the amount of data is small, the usage pattern does not show a significant increase (only about four megabytes over an attack). Due to the data size is so small, the CPU usage does not show any visible patterns (see figure 4.6).

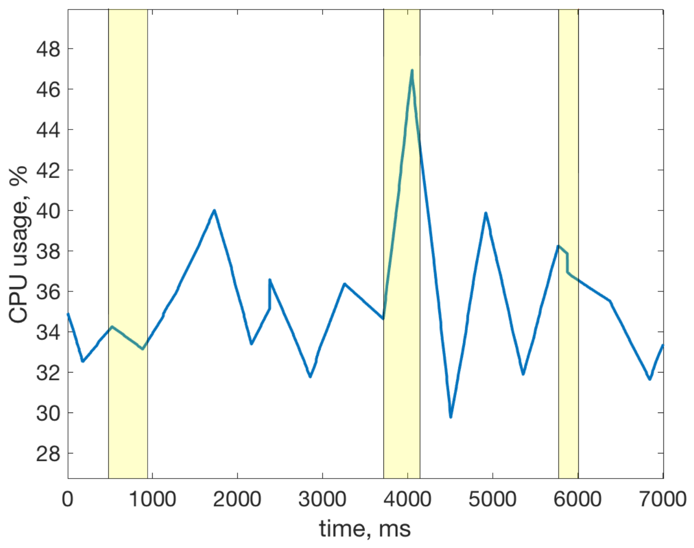


Figure 4.6: CPU load during the "attack" and "no attack" periods for contact data scenario. Attack periods are marked with yellow.

Image and audio data.

Figure 4.7 shows the memory consumption with clearly visible patterns. There is a rise of about 20 MB of memory for an attack. Also, after each attack, one can see memory usage stabilizes. In some cases, there could be a drop in memory usage during the attack periods. This memory usage drop happens due to the system's garbage collector reclaims the memory that is occupied by "intent" objects that were used during the attack.

Figure 4.8 demonstrates the increase in CPU usage during the attack. The CPU usage rises as high as 15 percent as compared to the no attack period. There are a low number of data points in the CPU usage as compared to memory because of the execution time for the CPU load recording services. A similar CPU and memory usage patterns were observed in the case of Audio data as well (see figure 4.9 and figure 4.10.

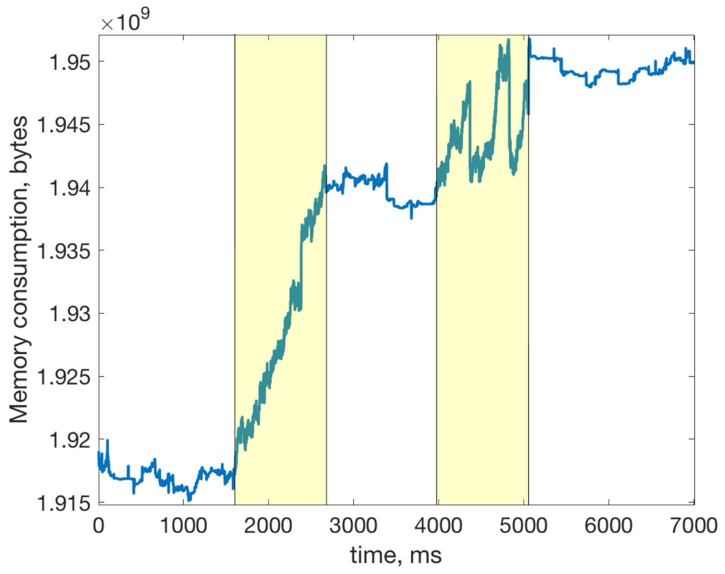


Figure 4.7: Memory consumption during the “attack” and “no attack” periods for image data. Attack periods are marked with yellow.

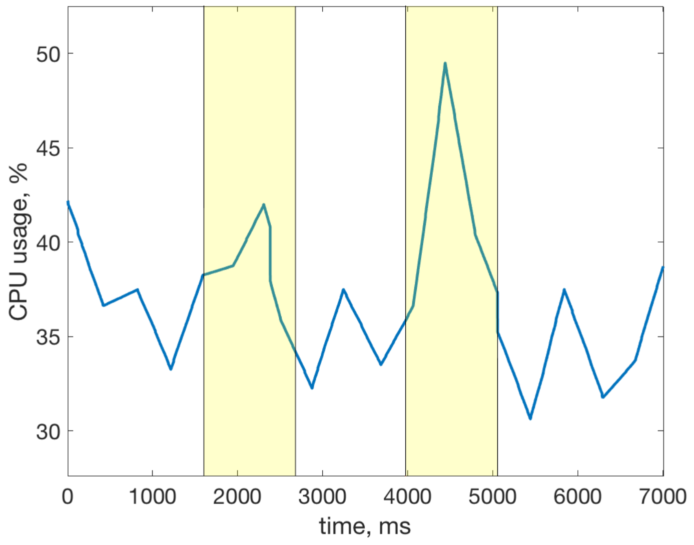


Figure 4.8: CPU load during the “attack” and “no attack” periods for image data scenario. Attack periods are marked with yellow.

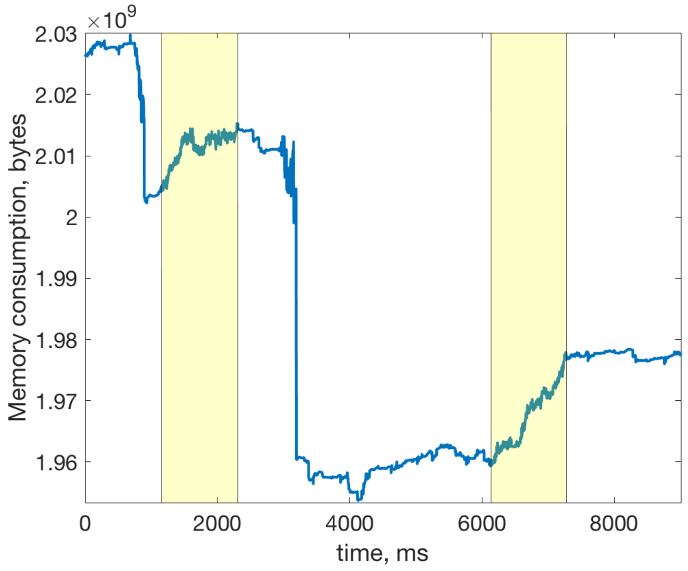


Figure 4.9: Memory consumption during the “attack” and “no attack” periods for audio data. Attack periods are marked with yellow.

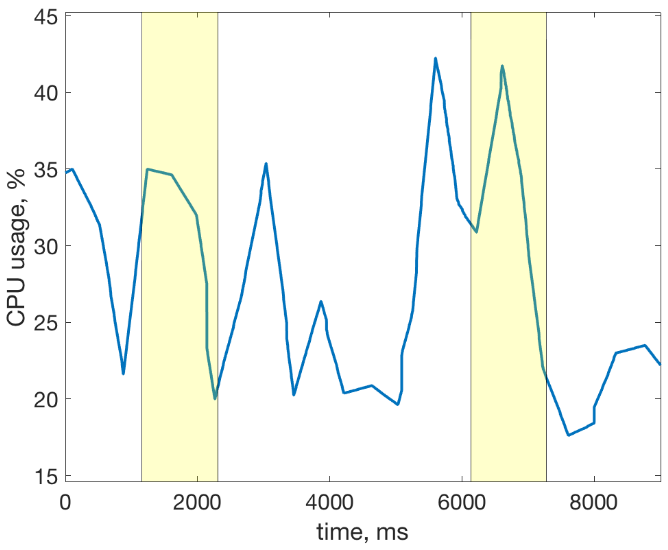


Figure 4.10: CPU load during the “attack” and “no attack” periods for audio data scenario. Attack periods are marked with yellow.

4.5 Attack Detection Using Memory Consumption and CPU Clock Speed

Unfortunately, starting Android OS version 8.0, the system does not allow acquiring the CPU utilization. This modification in the OS has been done to improve the privacy of the system but prevents our anomaly detection algorithm from proper functioning. This limitation can be overcome if our processes have superuser (root) privileges, but it contradicts our idea that the framework can be employed on a smartphone with a stock, unmodified firmware.

To overcome this new limitation, we substituted CPU utilization with CPU cores clock speeds (a.k.a. CPU frequency), which can be monitored through a standard Android API.

Memory consumption and CPU cores' clock speed were recorded every 20 ms through the standard Android OS API calls. While memory consumption can be recorded with a much higher resolution, the CPU cores' clock speed time resolution is limited to one request per 20 ms. On the other hand, as conducted empirical study proved, this interval between technological signals recording does not significantly affect the battery life of the smartphone. The study participants reported that they did not notice any changes in battery life during the experiments. The data were recorded for eight hours.

All this collected data was added to the "colluded applications" attack dataset that is mentioned section 4.4, can be used in future research on investigating and developing ML techniques. The data includes the following technological signals: the attack scenario, timestamps when the attack started and stopped, the record's timestamp, the clock speed of all available CPU cores, total RAM, free RAM, and used RAM. Overall, 4194 attacks were executed, and 402796 measurements of technological signals have been recorded. Examples of the recorded data are shown in figure 4.11 and figure 4.12.

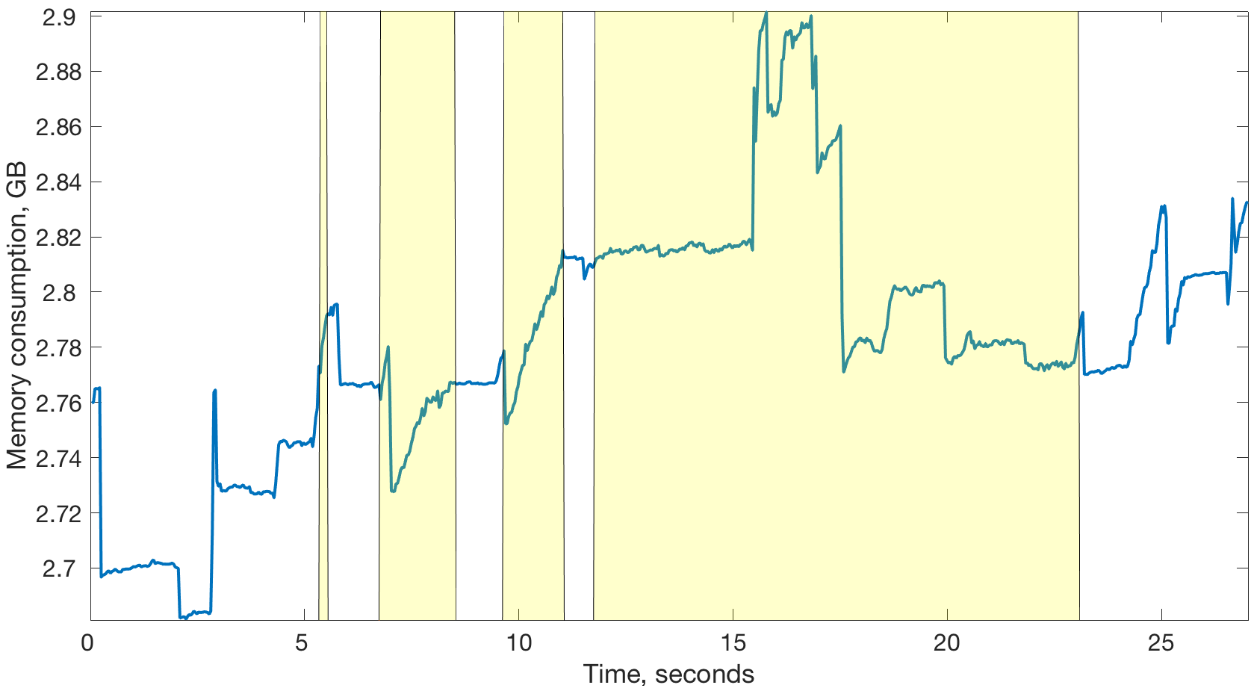


Figure 4.11: Memory consumption during the "attack" and "no attack" periods. Attack periods are marked with yellow. In the picture, attacks are presented in the following order (left to right): audio, video, video, SMS.

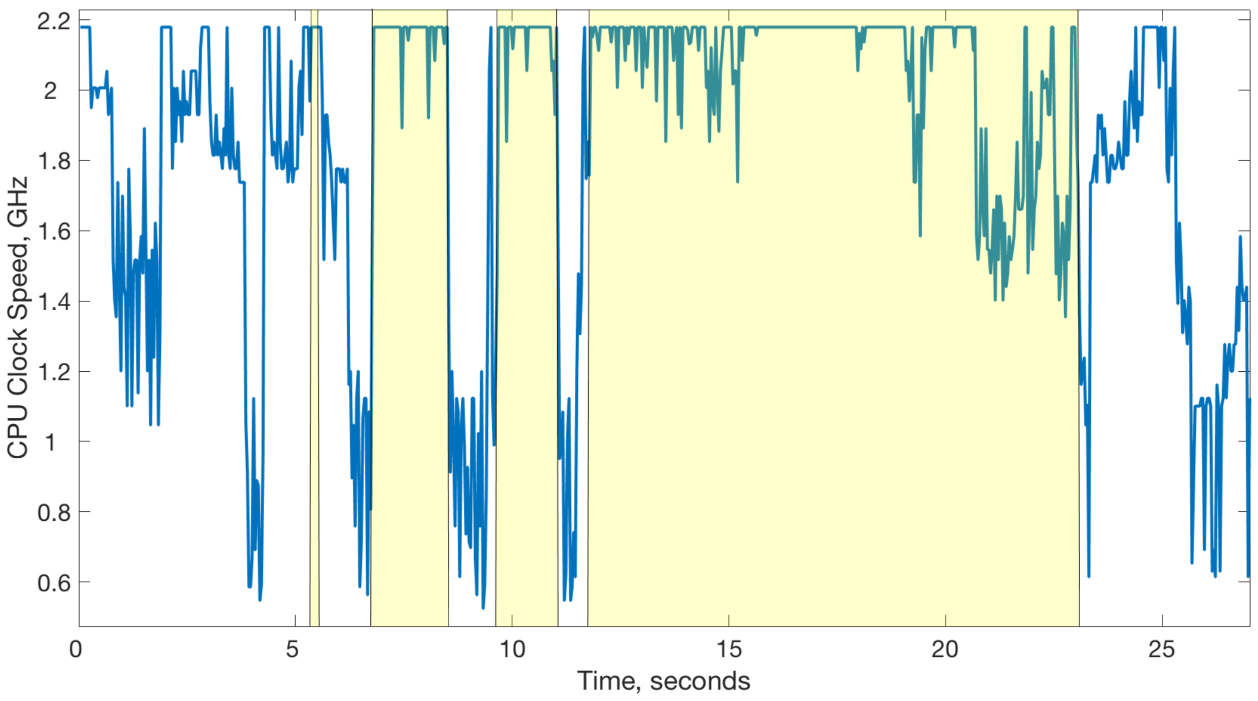


Figure 4.12: CPU clock speed (frequency) during the "attack" and "no attack" periods. Attack periods are marked with yellow. In the picture, attacks are presented in the following order (left to right): audio, video, video, SMS.

4.6 Conclusions

Addressing the primary research question posed in this chapter ***"How to develop effective and efficient security and privacy evaluation methods targeted for their implementation on resource-constrained mobile devices and incorporate them into the overall DQ evaluation pipeline?"****, we obtained the following* ***major*** *research results, which are further explained in details:*

- *We investigated modern Android OS based smartphones and developed* ***multilayered hierarchical security-related metrics structure*** *that includes three major branches.*
- *We* ***developed the initial calculus for metrics integration*** *from each of the security branches.*
- *We* ***investigated current threats*** *to user privacy and* ***revealed the lack of tools for detection novel and sophisticated attacks****.*
- *We* ***developed a systematic security evaluation pipeline*** *that considers smartphones' resource-constraint nature and* ***includes a comprehensive set of security and privacy evaluation approaches****.*
- *We* ***formalized "colluded applications" attack and developed its model****.*
- *We* ***conducted an empirical study of this attack****, and* ***created a dataset*** *that can be used for developing machine learning-based attack detectors.*
- *We* ***investigated and analyzed various artificial intelligence techniques for the detection attacks*** *on users' privacy.*

Existing approaches to DQ evaluation very often ignore the security aspect or do not pay proper attention to it. In the developed DQ and security evaluation framework, security plays a crucial role. In this section, we developed flexible and extendable techniques for data security and privacy evaluation. We developed a hierarchical structure of security-related metrics that includes three major branches: integrity, confidentiality, and availability. We produced and presented calculus for each of the security components. We investigated and analyzed what sensor platform characteristics and properties may influence on data security privacy, and, thus, on overall DQ level. Based on our analysis, we developed an initial set of security and privacy-related metrics for sensor platforms that run the Android OS. While the presented set of metrics is mostly related to Android OS-based smartphones, it can be extended to cover other sensor platforms as well.

We investigated current threats to data privacy and revealed the inability of modern anti-malware software to detect novel and sophisticated attacks on data privacy in modern smartphones. We examined in detail colluded applications attack, which is one of the new sophisticated attacks on data privacy that is currently emerging. After studying various inter-application communication use cases and analyzing possible attack scenarios, we formulated the attack's definition and developed the attack model. In order to build an attack detector (see section 5.2 for details), we conducted an empirical study.

Before the detection of more sophisticated attacks on data privacy, more straightforward threats to data privacy have to be addressed. We developed a systematic pipeline that includes initial steps of the data security evaluation, which have to be performed before colluded application attack detector starts. Our two-step approach to detection of attacks on data privacy allows effective and efficient data privacy evaluation implementation on the resource constraint devices.

We investigated the employment feasibility of various artificial intelligence techniques for attack detection. Our study covers such techniques as various decision trees, hidden Markov-chain models, and neural networks. Our analysis revealed that decision trees have the worst performance in this attack detection task. Also, we revealed that while hidden Markov-chain models may have high attack detection rates, they are outperformed by recurrent neural networks, especially long-short term memory models. In addition, neural network models may take advantage of the Android OS neural network API and dedicated chips, such as "visual core". Employing neural network API allows running NN models in a very efficient way and preserves devices' battery.

We implemented the developed scenarios on an Android OS-based smartphone to investigate and analyze attacks influence on the sensor-platform. During the attacks, overall memory consumption, the CPU usage of the system's and user's processes, and changes in CPU cores clock speed were recorded. Logged data is organized into a comprehensive dataset and is made available for public use and future research. Four scenarios that represent restricted data transmission of various types such as contacts, audio, images, and video were selected for further research and were implemented on seven real smartphone models. The second empirical study resulted in a second dataset that contains detailed time-series data of overall memory consumption, CPU usage, and process-by-process data during the attacks. The obtained dataset is used for attack detector development described in chapter 5.

Chapter 5

Framework Implementation and Validation on Use Cases

*This chapter addresses the fourth major research question: "**How to develop effective and efficient framework implementation on a platform of mobile devices, such as Android OS-based smartphones?**". This question can be divided into a set of sub-questions, which are answered in each section of this chapter. In this chapter, we develop and present novel techniques and tools that enable security and privacy aspects integration into a conventional data quality evaluation pipeline. Section 5.1 presents the implementation of the initial data quality and security metrics acquisition. We present our implementation of the detection attacks on users' privacy in sections 5.2 - 5.4. Section 5.5 presents the implementation of data quality and security metrics integration methods. Implementation of the cloud component is presented in section 5.6.1. In section 5.6, we present metrics integration implementation targeted at the resource-constrained devices. Finally, we validate the developed framework on a set of diverse use cases presented in sections 5.7 - 5.9.*

5.1 Framework Components Implementation

In the following sections, we present our implementation of the developed framework. This section presents our implementation of the initial metrics acquisition.

To implement the framework, we employed a number of artificial intelligence techniques. Initially, the metrics integration calculus has been implemented through the fuzzy rules of the expert system (ES). Employment of the ES improves the framework scalability, simplifies its design and maintenance, and enables its modification. Unfortunately, ES systems are computationally expensive, that presents a particular problem for resource-constrained mobile devices. To overcome this limitation, we substituted developed ES modules with the neural network models that approximate input/output hyper-surfaces produced by ES modules. Developed NN models are computationally cheaper and faster than ES systems and may take advantage of the NN dedicated chips that are built into modern smartphones and other mobile devices [48]. We present details of our framework implementation below.

5.1.1 Initial Security Metrics Acquisition

We have developed an Android OS library, and a specialized application that is based on this library, that retrieves security and privacy-related parameters, which are used as the inputs to the expert system. The developed library provides an API that can be used by other researchers and software developers and is available at Google Play Store. The developed software provides recommendations on how to improve the smartphone security level and gives an explanation of each parameter to a user. In addition, it allows opening a “simulation” screen where users can experiment with these parameters and see how they influence overall security. Table 5.1 provides the library’s API and figure 5.1 presents the architecture of the developed software.

Table 5.1: System's parameters that are gathered for the initial security evaluation

Metric	Symbol	Values
Screen lock	M_{SL}	1 - Pattern, PIN or password; 0 - otherwise
Android OS version	M_V	2 - The latest version, 1 - previous version; 0 - otherwise
Unknown sources	M_{US}	1 - Unknown sources disabled; 0 - otherwise
Potentially harmful applications	M_{PH}	0 - Installed at least one potentially harmful application; 1 - otherwise
Developer's menu	M_{DO}	1 - Developer option menu disabled; 0 - otherwise
Basic integrity test	M_{BI}	1 - System passed basic integrity test; 0 - otherwise
Android compatibility test	M_{CT}	1 - System passed Android compatibility test; 0 - otherwise

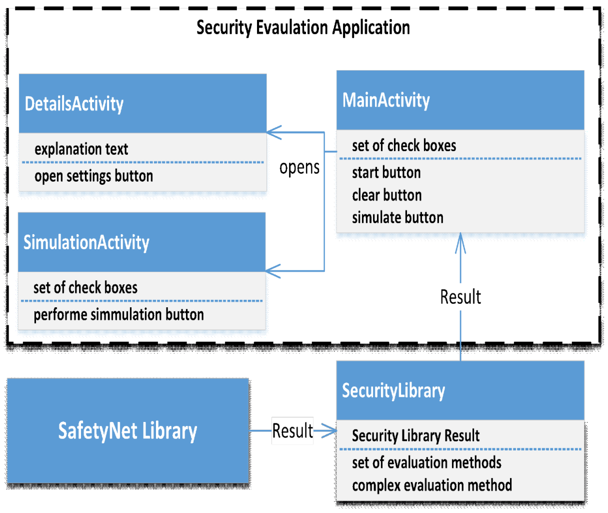


Figure 5.1: Architecture of the software for initial security evaluation

5.2 Attacks Against Privacy Detection Implementation

In this section, we present our implementation of the developed technique aimed at reducing false-positive alerts for the detector of novel attacks on users' privacy.

To implement the anomaly detection, we use an example of the "colluded applications" attack described in section 4.2. The "colluded applications" attack is a threat to user privacy because it may leak private data from a smartphone. We completed an empirical study and collected data from multiple phones during the attack and during the normal smartphone operation (see section 4.2 for more details). To conduct these experiments, we developed two applications. The Source application has the data and the necessary permissions to receive private data. And the other one is the Sink application, which requests and receives data from the Source and could violate privacy protection by leaking this data to a remote host. Apart from the above-mentioned applications, a third application was developed to log memory and CPU usage.

Unfortunately, some legitimate applications may produce patterns in the system resources utilization signals,which are similar to malicious applications, therefore we need to develop a technique to distinguish them. An application permission analysis is called to differentiate legitimate applications from malicious ones and with the goal to reduce the false-positive rate. In the case of "colluded applications", this analysis allows revealing applications that may benefit from the collusion. The architecture of the developed tool is presented in figure 5.2. We have built an expert system whose knowledge base is constructed by analyzing the permissions available in an Android application Manifest file, which is a vital part of all Android OS applications. The built system is capable of suspecting collusion between pairs of applications.

This tool scans the user-installed applications by ignoring the pre-installed ones (system applications). From the list of user-installed applications, application pairs are formed from which permission pairs are extracted (one from each of the application pairs). These permission pairs are then passed to the rule-based engine, which performs forward chaining and generates a boolean

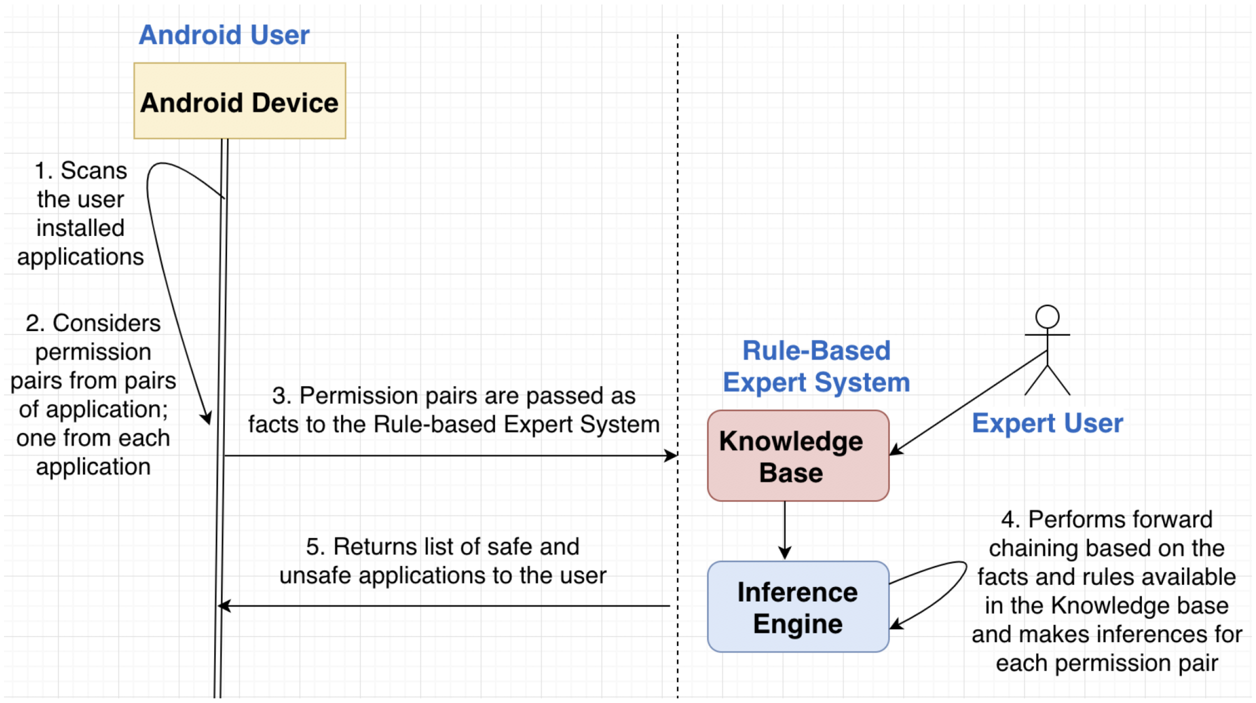

Android User
Android Device
1. Scans the user installed applications
2. Considers permission pairs from pairs of application; one from each application
3. Permission pairs are passed as facts to the Rule-based Expert System
5. Returns list of safe and unsafe applications to the user
Rule-Based Expert System
Knowledge Base
Inference Engine
Expert User
4. Performs forward chaining based on the facts and rules available in the Knowledge base and makes inferences for each permission pair

Figure 5.2: Schematic representation of suspecting malicious collusion between Android applications using a rule-based expert system

value. If this boolean value is set, it means that the application pairs may benefit from collusion. On the other hand, if the boolean value is “false”, the collaboration of application pairs do not endanger a user’s privacy. In the case when attack detector warned about the attack and permission analysis shows that the system contains applications that may benefit from the collusion, a more detailed analysis of the suspected applications is required.

5.3 Attacks Against Privacy Detection With Intelligent Techniques Using Memory Consumption and CPU Utilization

This section presents our attack detector implementation that is based on machine learning techniques such as feed-forward neural network and recurrent neural network. The attack detector relies on such technological signals as overall memory consumption and CPU load.

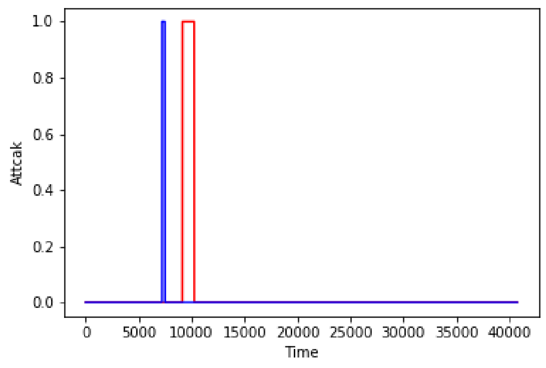
To develop an attack detector, we investigated various machine learning techniques such as feed-forward neural networks (FFNN), simple recurrent neural networks (RNN), long short-term memory (LSTM), and bidirectional LSTM. We completed the empirical study employing two approaches to the attack detector design: a generic model and individual model. The generic model is trained on the collected data combined from all smartphones that participate in the study. The individual model is based on the data from a particular phone. Since all collected data are the time-series data, a sliding window approach was used to train and test the developed attack detectors. The data time resolution is one millisecond, and the window size of 32 measurements results in a 32-millisecond delay in the attack detection.

5.3.1 Attack Detector Generic Model for Multiple Smartphones

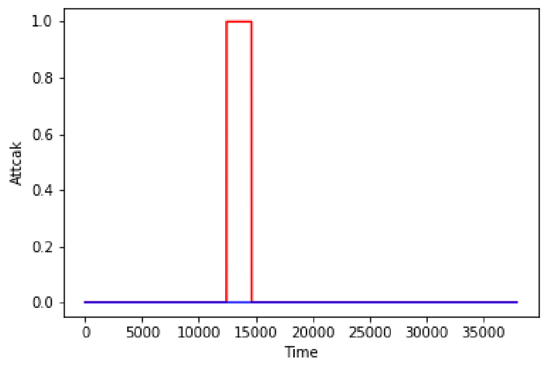
FFNN is the artificial neural network that does not have any feedback loop and has a lightweight architecture as compared to RNN. The architecture consists of an input layer, multiple hidden layers, and an output layer with a tuned set of hyper-parameters. Figure 5.3a shows an example of the actual data against the model classification that demonstrates this model failures in the attack recognition.

A simple RNN model consists of a single input layer, multiple hidden layers with the feedback loop, and an output layer. In our empirical study, it works better in terms of false positives rates as compared to FFNN, but the performance is still not sufficient enough. Figure 5.3b shows an example of the model classification.

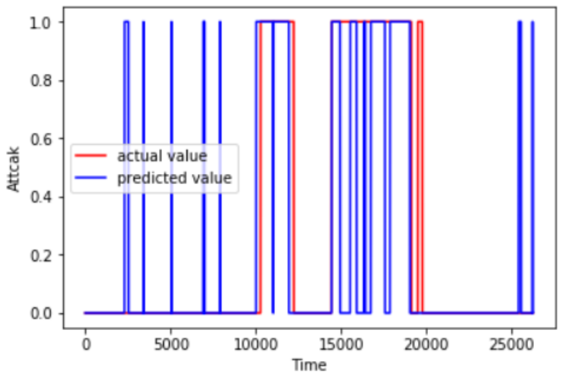
LSTM is a special type of RNN that has memory cells to store information in memory for a longer period, which also overcomes the standard RNN problem of the vanishing gradient. LSTM has input, output, and a forget gate making it a robust architecture to deliver better results, but this performance comes at the cost of its higher complexity. In this research, the LSTM model consists of a single input, hidden, and an output layer. It demonstrates a better performance than the RNN and FFNN models, as observed in figure 5.3c.



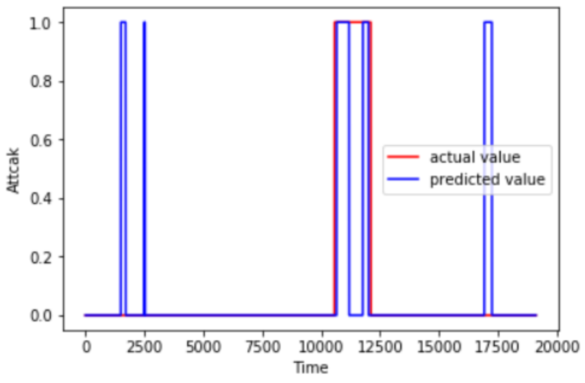
(a) FFNN



(b) RNN



(c) LSTM



(d) BiLSTM

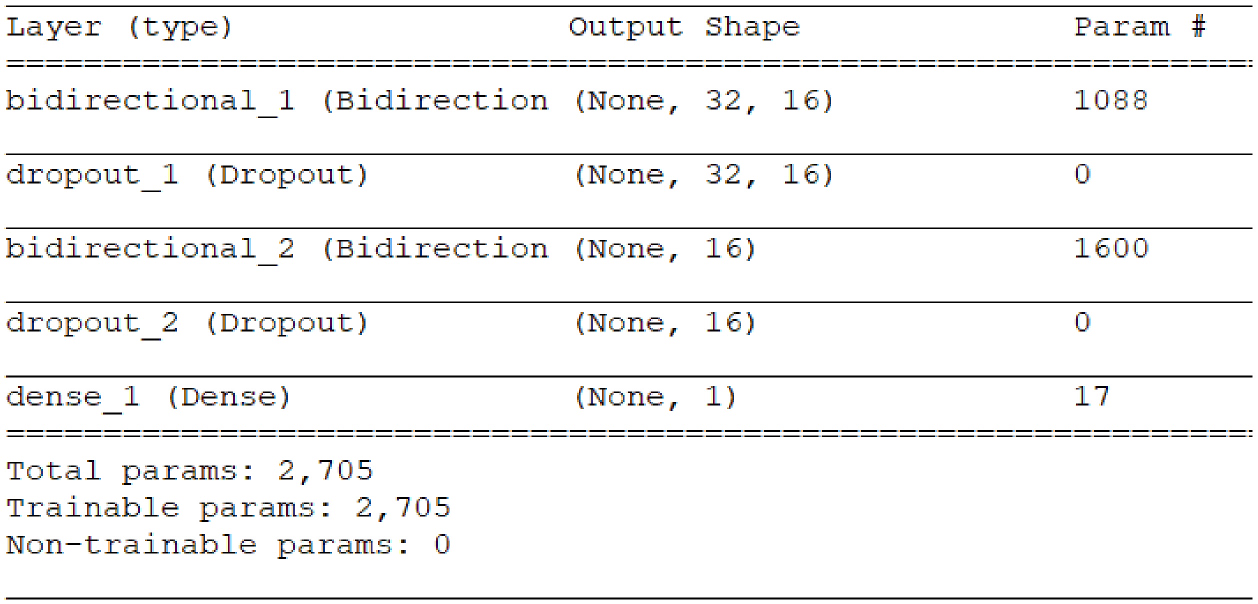
Figure 5.3: Models performance. Actual (red color) versus classified (blue color).

A bidirectional LSTM (BiLSTM) is a variant of LSTM wherein we add a “bidirectional” layer so the model can learn the patterns in the data in both directions. It has a more complex and robust architecture as it stores information from the past and future classifications, which results in better performance. Figure 5.3d shows the BiLSTM model classification vs actual data.

This BiLSTM model demonstrates a better performance than others that we observed previously, with higher precision and recall rates resulting in high recognition accuracy. Training accuracy for the BiLSTM model is 92%, with a loss of 0.0895 and validation accuracy of 87.68% having considered 32 history records. Figure 5.4 shows the summary of a BiLSTM model parameter values obtained at the end of the training. Table 5.2 shows the performance of the developed generic BiLSTM model. The overall accuracy for the BiLSTM model is appealing for a generic

Table 5.2: Test results for BiLSTM model

Device	**Scenario**	**Accuracy**	**Precision**	**Recall**
Nexus 5X	Overall	0.941	1.0	0.42
OnePlus 5T	Overall	0.891	0.52	0.09
OnePlus 5x	Contacts	0.985	0.0	0.0



```
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_1 (Bidirection (None, 32, 16)            1088
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 16)            0
_________________________________________________________________
bidirectional_2 (Bidirection (None, 16)                1600
_________________________________________________________________
dropout_2 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 17
=================================================================
Total params: 2,705
Trainable params: 2,705
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.4: Summary of a Bidirectional LSTM model.

model, but the precision and recall rates are relatively low. This low performance motivated us to transition from a generic model to an individual model training based on the data collected from a particular smartphone.

5.3.2 Attack Detector Individual Model for a Particular Smartphone

In this section, we verify our hypothesis that the attack detector may perform better on a smartphone if it is trained on the data from the same smartphone. Since BiLSTM gives the best results, we use this architecture in the individual training model as well. In addition, to more accurately reflect the performance of the attack detector, we use the true positive rate (TPR) and false-positive rate (FPR) concepts.

$$True\,Positive\,Rate = \frac{Correctly\,Identified\,Attacks}{Total\,Attacks} \tag{5.1}$$

A true-positive rate is the ability of a model to correctly identify an attack, i.e., how sensitive the model is in detecting an attack. The higher the true-positive rate, the more sensitive the model is, which maximizes the chances of attack detection. However, this high sensitivity comes at a cost as it also increases false alarms.

$$False\,Positive\,Rate = \frac{Wrongly\,Identified\,Attacks}{Total\,NoAttacks} \tag{5.2}$$

A false-positive is the error that happens when the model detects an attack while there is no attack. This error rate must be minimized to reduce the chances of "False Alarm" and prevent a further reaction to a non-existing attack. Similarly, the cost of FPR reduction is TPR decreasing.

In this research, we aim at increasing the attack detection effectiveness. We employed the following rules for the model performance evaluation:

1. We calculate the TPR based on a number of attacks detected against the overall actual attacks.
2. We introduce a time delay (delta parameter) in attack detection decision. Table 5.4 and 5.5 prove that adding a small delay can significantly improve the model overall performance.
3. We also take into account the edge case where an attack is detected after an actual attack starts, and the detected attack ends after the actual attack ends. Considering the edge cases, we finally calculate the false-positive rate based on wrongly identified samples in the data.

We developed models for two smartphones: OnePlus 6T and Google Pixel 2XL. For OnePlus 6T, we used data that was collected for generic model training. In addition, to make data more diverse, we included a scenario with video data transmission. The results are presented in table 5.3 for

Table 5.3: Performance of the model trained on the data from OnePlus 6T

Confidence	**TPR**	**FPR**
0.99	100	2.9
0.95	100	3.6
0.90	100	4.1

various confidence levels of the attack detector. A confidence level is a threshold of the attack detector's decision confidence. A high confidence level decreases FPR but also decreases TPR. One can observe, in the case of OnePlus 6T smartphone, we achieved high TPR with low FPR.

To confirm the high performance of the attack detector that is trained on the data of the particular phone, we used a similar approach for Google Pixel 2XL. We collected data for three scenarios from Google Pixel 2XL. Each scenario contains 50 attack instances.

In the first performed experiment, the attack detection model uses only overall memory consumption as its inputs. The model was not able to converge well and hence failed to detect attacks. This proves that only overall memory consumption use does not provide enough information for reliable attack detection.

In the second experiment, the attack detection model uses memory consumption, CPU utilization, and system interrupts as its inputs. Our study reveals that bigger volumes of data, such as image and video, can be detected with high TPR and low FPR. Table 5.4 shows the performance evaluation of the BiLSTM model on unseen data of audio scenario for Pixel-2XL Android smartphone. The audio scenario test data consists of overall "Memory", all "CPU" usage parameters, and system interrupts during the 15 attacks. We can see the trade-off between the TPR and FPR. Similarly, table 5.5 shows the performance evaluation of the architecture on unseen image scenario data for the same device. The image scenario test data consists of overall "Memory consumption", all "CPU" usage parameters, and system interrupts during 12 attacks.

Table 5.4: Pixel 2XL: Performance evaluation for audio scenario

Confidence	**TPR**	**FPR**	**FPR W/ Delta**
0.99	0.9333	0.1169	0.1049
0.95	0.9333	0.1637	0.1488
0.90	0.9333	0.2062	0.1798
0.85	1.0	0.2238	0.1899
0.80	1.0	0.2418	0.1985

Table 5.5: Pixel 2XL: Performance evaluation for image scenario

Confidence	**TPR**	**FPR**	**FPR W/ Delta**
0.99	0.9167	0.1637	0.1538
0.95	0.9167	0.2196	0.1970
0.90	0.9167	0.2582	0.2250
0.85	1.0	0.2777	0.2442
0.80	1.0	0.2875	0.2539

Table 5.6: Results for FFNN and LSTM models (green shade means the best result for the window size). Model uses as inputs memory consumption and CPU cores clock speed.

Window Size	55 Points			11 Points			5 Points			3 Points			1 Points		
	FFNN		LSTM	FFNN		LSTM	FFNN		LSTM	FFNN		LSTM	FFNN		LSTM
	In Window	Point-by-point		In Window	Point-by-point		In Window	Point-by-point		In Window	Point-by-point		In Window	Point-by-point	
FP	56	6354	1755	232	3141	2178	576	3249	2359	1242	3960	2697	3763	3763	3468
FN	36	4898	1138	77	1960	907	146	1133	1390	246	1046	1539	2488	2488	3004
TP	1081	56180	59926	5431	59151	60201	12071	59985	59724	20102	60071	59577	58631	58631	58114
TN	291	13127	17686	1583	16307	17263	3319	16192	17082	5263	15481	16744	15678	15678	15973
Accuracy, %	**93.72%**	**86.03%**	**96.41%**	**95.78%**	**93.67%**	**96.17%**	**95.52%**	**94.56%**	**95.35%**	**94.46%**	**93.79%**	**94.74%**	**92.24%**	**92.24%**	**91.97%**
Precision	0.95	0.90	0.97	0.96	0.95	0.97	0.95	0.95	0.96	0.94	0.94	0.96	0.94	0.94	0.94
Recall	0.97	0.92	0.98	0.99	0.97	0.99	0.99	0.98	0.98	0.99	0.98	0.97	0.96	0.96	0.95
F1-score	**0.96**	**0.91**	**0.98**	**0.97**	**0.96**	**0.98**	**0.97**	**0.96**	**0.97**	**0.96**	**0.96**	**0.97**	**0.95**	**0.95**	**0.95**

5.4 Attacks Against Privacy Detection With Intelligent Techniques Using Memory Consumption and CPU Frequency

In this section, we present the attack detector implementation based on two machine learning techniques: feed-forward neural network and long-short term memory. These implementations of attack detector use overall memory consumption and CPU cores clock speed for attack detection.

Unfortunately, as we described in section 4.5, starting Android OS version 8.0 and above, standard

Table 5.7: Results for FFNN and LSTM models (green shade means the best result for the window size). Model uses as inputs CPU cores clock speed only.

Window Size	11 Points			5 Points			3 Points			1 Points		
	FFNN		LSTM	FFNN		LSTM	FFNN		LSTM	FFNN		LSTM
	In Window	Point-by-point		In Window	Point-by-point		In Window	Point-by-point		In Window	Point-by-point	
FP	286	3792	2129	726	4007	2614	1222	3917	2766	3824	3824	3853
FN	166	2996	897	107	946	1147	235	1030	1286	2286	2286	1997
TP	5342	58115	60211	12110	60172	59967	20113	60087	59830	58833	58833	59121
TN	1529	15649	17312	3169	15434	16827	5283	15524	16675	15617	15617	15588
Accuracy, %	**93.83%**	**91.57%**	**96.24%**	**94.83%**	**93.85%**	**95.33%**	**94.57%**	**93.86%**	**94.97%**	**92.42%**	**92.42%**	**92.74%**
Precision	0.95	0.94	0.97	0.94	0.94	0.96	0.94	0.94	0.96	0.94	0.94	0.94
Recall	0.97	0.95	0.99	0.99	0.98	0.98	0.99	0.98	0.98	0.96	0.96	0.97
F1-score	**0.96**	**0.94**	**0.98**	**0.97**	**0.96**	**0.97**	**0.97**	**0.96**	**0.97**	**0.95**	**0.95**	**0.95**

Android API does not allow accessing CPU utilization data. That restriction directed us to use CPU cores clock speeds instead. To develop attack detectors, we chose two machine learning architectures: FFNN and LSTM. As we rendered in the previous section, LSTM proved to be an efficient solution for anomaly detection based on memory consumption and CPU utilization. At the same time, the FFNN model is more straightforward and computationally cheaper. While this empirical study involves data from one smartphone, in the future, this research can be extended to creating the generic model based on the collected data combined from various smartphone models.

As one can observe in figures 4.11 and 4.12, in the case of CPU frequency, the data is relatively noisy and has to be filtered. We concatenated every ten records and replaced their values with the mean value of each technological signal, which reduced our dataset to 80559 records. All these records were divided into test, validation, and test data sets with the 60/20/20 proportion. Since these technological signals represent time series, it is crucial to keep the sequence of the data records.

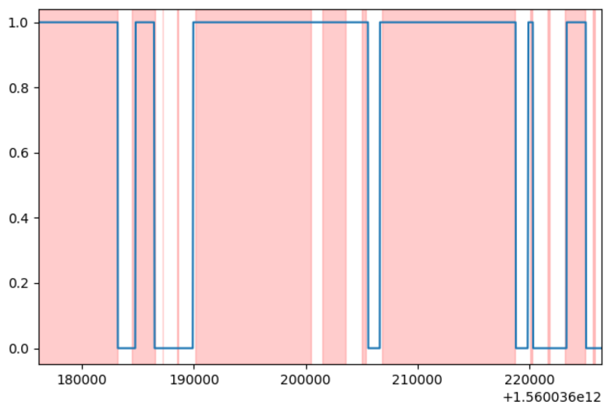
The data time resolution is 20 milliseconds, but after filtering, the time resolution becomes 200 ms. Since all collected data are the time-series data, a sliding window approach was used to train and test the developed attack detectors. We tried five various window sizes: one, three, five, eleven, and fifty-five record points. The window size of one filtered record equals to 0.2 seconds. The window shift is equal to the window size. Models were designed to take as an input memory consumption and CPU cores' clock speeds (see table 5.6 for the results), CPU cores' clock speed only (see table 5.7 for the results), and memory consumption only. Models that rely on memory consumption only could not effectively identify attacks, which confirms our previous findings, and were excluded from the further analysis. In addition, to compare FFNN model performance with the LSTM model, we employed the so-called "point-by-point" approach to data points classification. In the "window" mode, if the model classifies the window that contains at least one attack instance as an attack, we count it as a true positive. In the "point-by-point" mode, after model classified the data window, the assigned label is applied to all data points within this window, and only after that, we compare results point-by-point with round truth values.

5.4.1 Application of Feed Forward Neural Network Model

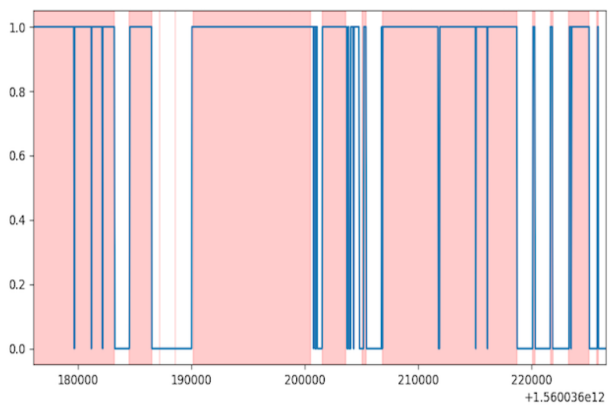
FFNN is the artificial neural network which has no feedback loop and has a lightweight architecture as compared to LSTM. The architecture consists of an input layer, multiple hidden layers, and an output layer with a tuned set of hyper-parameters. Figure 5.5a shows an example of the actual data vs. the model classification that demonstrates this model performance in attack detection task.

We used the following architecture of the FFNN model:

- Number of neurons in the input layer: $windows\_size \times number\_of\_features$;
- Windows_size: 1,3, 5, 11, or 55 record points;
- A hidden layer with 40 neurons;
- An output layer with 2 neurons.
- Hidden layer activation function: $tanh$
- Output layer activation function: $softmax$
- The loss function: the mean of the categorical cross-entropy.



(a) FFNN model performance.



(b) LSTM model performance.

Figure 5.5: Actual (red shade) versus detected (blue line).

The FFNN model achieves high accuracy with a low false-positive rate and false-negative rate. The overall results of the FFNN model are presented in table 5.6 and table 5.7. As one can observe in figure 5.7, in the mode of complete window classification, the best accuracy is achieved with a window size of 11 record points. However, in the "point-by-point" mode, as window size increases, the model accuracy decreases. This decrease in accuracy happens due to the model classifies the whole window, and all records within this window are labeled according to the window label. However, within the classified window, the attack might not occupy the whole window, or there might be several attacks. Since later in the "point-by-point" mode, we compare classified labels with ground truth labels, it results in increasing of false-positive and false-negative rates as window size increases.

Both model that is trained on CPU frequency only and model that is trained on "memory and CPU frequency" achieved similar accuracy for windows of one and three records. However, for the window sizes of five and eleven records, the model that is based on memory usage and CPU clock speed has higher accuracy.

5.4.2 Application of Recurrent Neural Network Model

As we mentioned earlier, long short-term memory is a special type of recurrent neural network (RNN) having memory cells that store information in memory for a longer period, which also overcomes the standard RNN problem of the vanishing gradient. LSTM has input, output, and a forget gate making it a robust architecture to deliver better results, but this performance comes at the cost of its higher complexity. In this research, we used the following LSTM model architecture:

- An input layer: “number of timestamps in sequence” × “number of features”;
- An LSTM layer with 100 neurons
- A dense layer with 100 neurons
- A dropout layer
- An output layer with 2 neurons
- Recurrent activation function: $hard\_sigmoid$,
- Kernel initializer: $glorot\_uniform$
- Recurrent initializer: $orthogonal$
- Bias initializer: $zeros$
- The dense layer activation function: ReLU
- The dropout rate in the dropout layer is 0.5
- The output layer activation function: $softmax$
- The loss function: the mean of the categorical cross-entropy

Table 5.6 presents results of the LSTM model. Figure 5.7 demonstrates that LSTM model accuracy increases as window size increases. Comparing models based on the type of inputs, one can observe

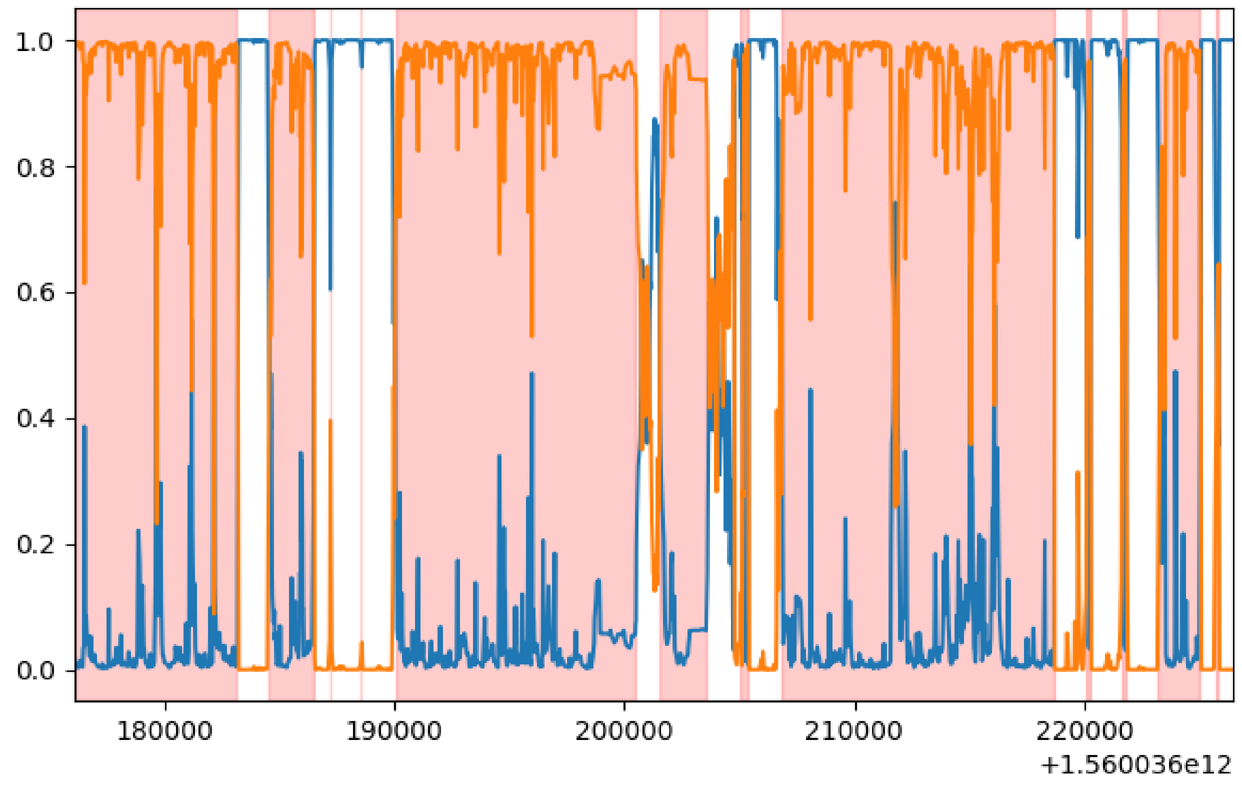


Figure 5.6: LSTM model performance. Actual attack - red shade, blue lines are the confidence levels for no-attacks, and the yellow lines are the confidence level for attacks.

that the model that takes only CPU clock speed performs similarly to the model that takes as inputs both memory usage and CPU clock speed. This can happen due to memory usage data being extremely noisy.

5.4.3 Feed Forward and Recurrent Neural Network Models Performance Comparison

Both FFNN and LSTM models achieved a high performance in anomaly detection. With the size of the sliding window equal to five record points, the FFNN model slightly (0.4%) outperforms the LSTM model. With bigger window sizes (11 and 55 record points), LSTM clearly performs better. The advantage of the FFNN model is in its simplicity.

While the LSTM model demonstrates a comparable performance to the FFNN model (see figure 5.5b), it has much higher decision granularity. Figure 5.6 renders the LSTM model confidence during the attack and no attack periods. Although the model uses as inputs a number of record points equal to the window size, it classifies the last point in the window that results in better classification granularity in comparison to the FFNN model. The FFNN model classifies the whole window. FFNN in the point-by-point mode has the same classification granularity as the LSTM model, but the FFNN model performance in this mode is much worse.

In terms of input types, the FFNN model that relies on both memory usage and CPU clock speed performs better than the model that takes only CPU clock speed. However, using memory consumption in the LSTM model did not show any significant difference except for increasing the model complexity.

The LSTM model achieves a higher performance in accuracy (see figure 5.7) and precision (see figure 5.8), and demonstrates similar to FFNN model recall (see figure 5.9) and F1-Score (see figure 5.10). On the other hand, the LSTM model complexity is much higher than the FFNN models. Figure 5.11 demonstrates the training time for FFNN and LSTM models that take as inputs both CPU clock speed and RAM consumption.

The results of these anomaly detectors can be used as inputs to the expert system modules responsible for the security evaluation.

5.5 Expert System Employment in Framework Implementation

In this section, we present our implementation of data quality and security metrics integration through fuzzy rules expert systems.

After all metrics of the first order have been acquired, they are integrated into the overall data quality score. We chose an expert system with fuzzy rules to implement the developed calculus.

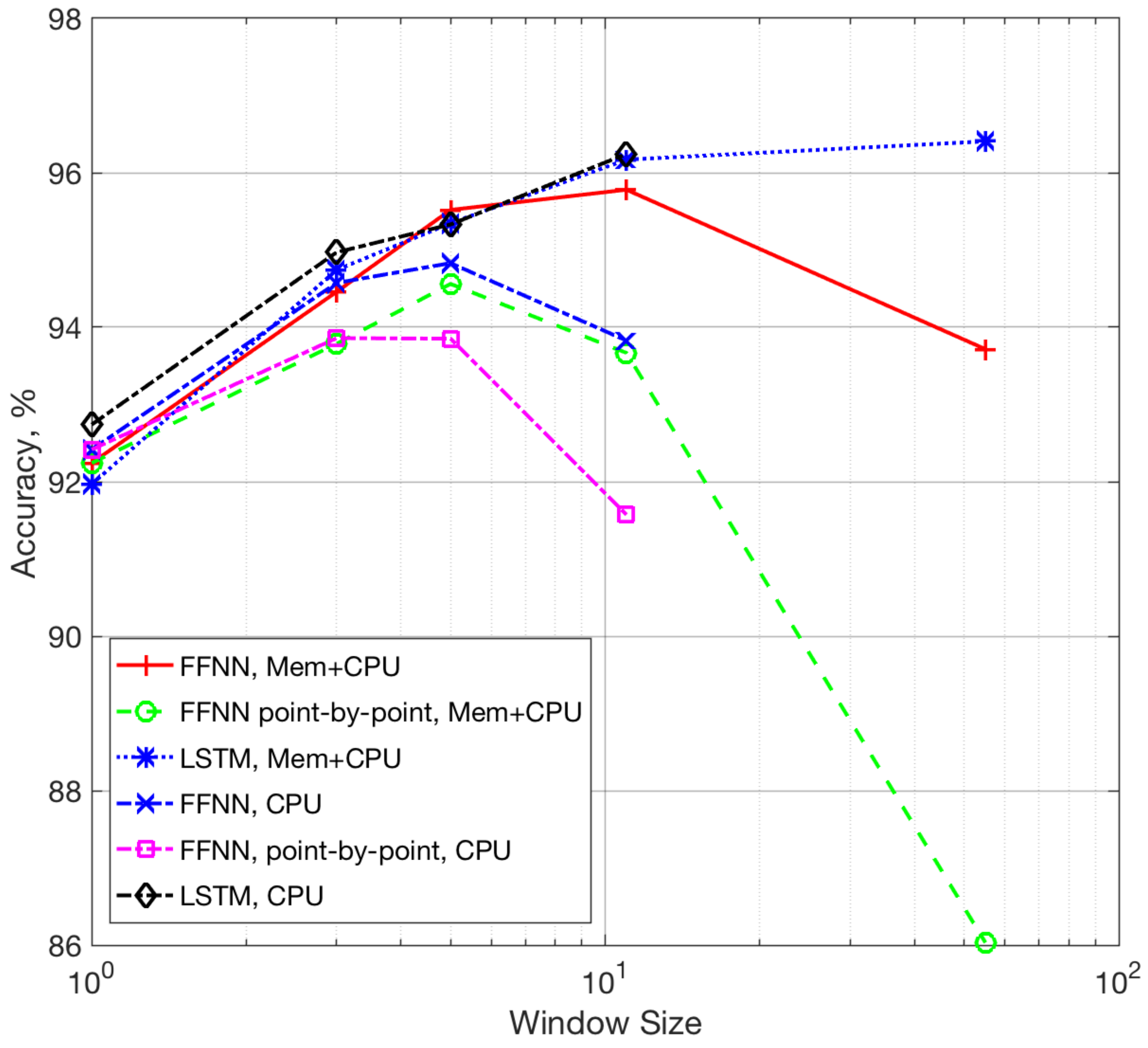


Figure 5.7: Model accuracy vs. window size

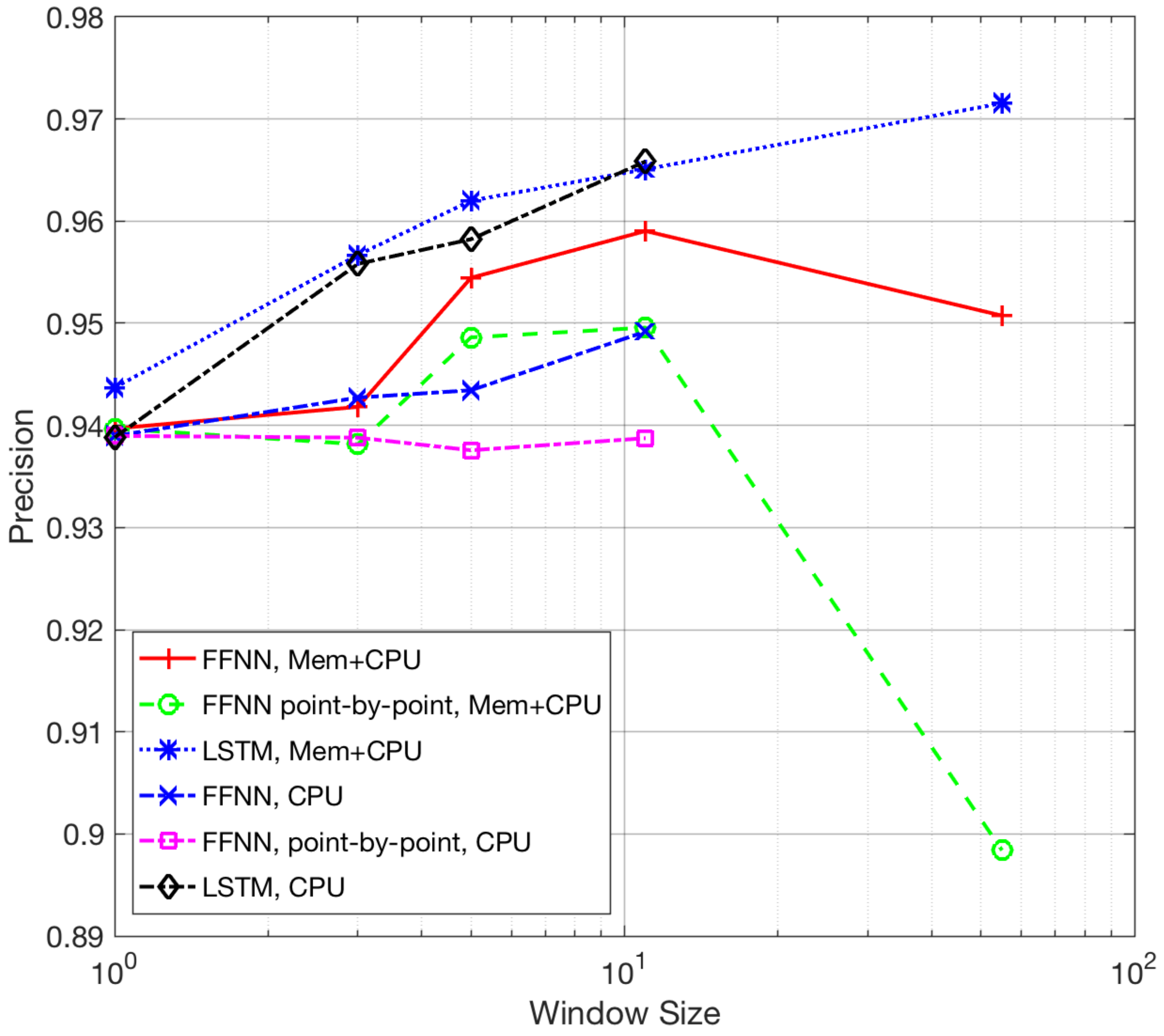


Figure 5.8: Model precision vs. window size

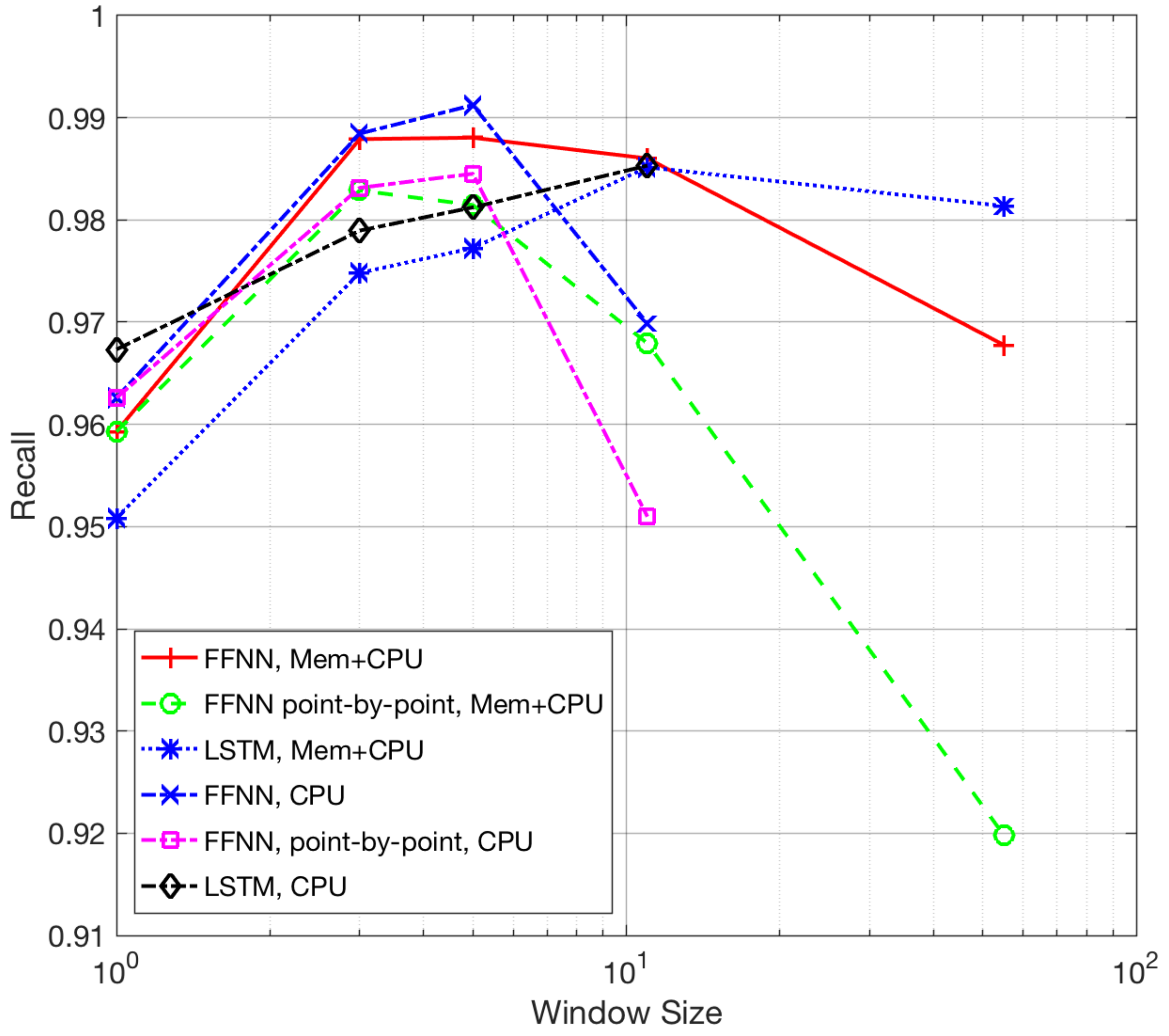


Figure 5.9: Model recall vs. window size

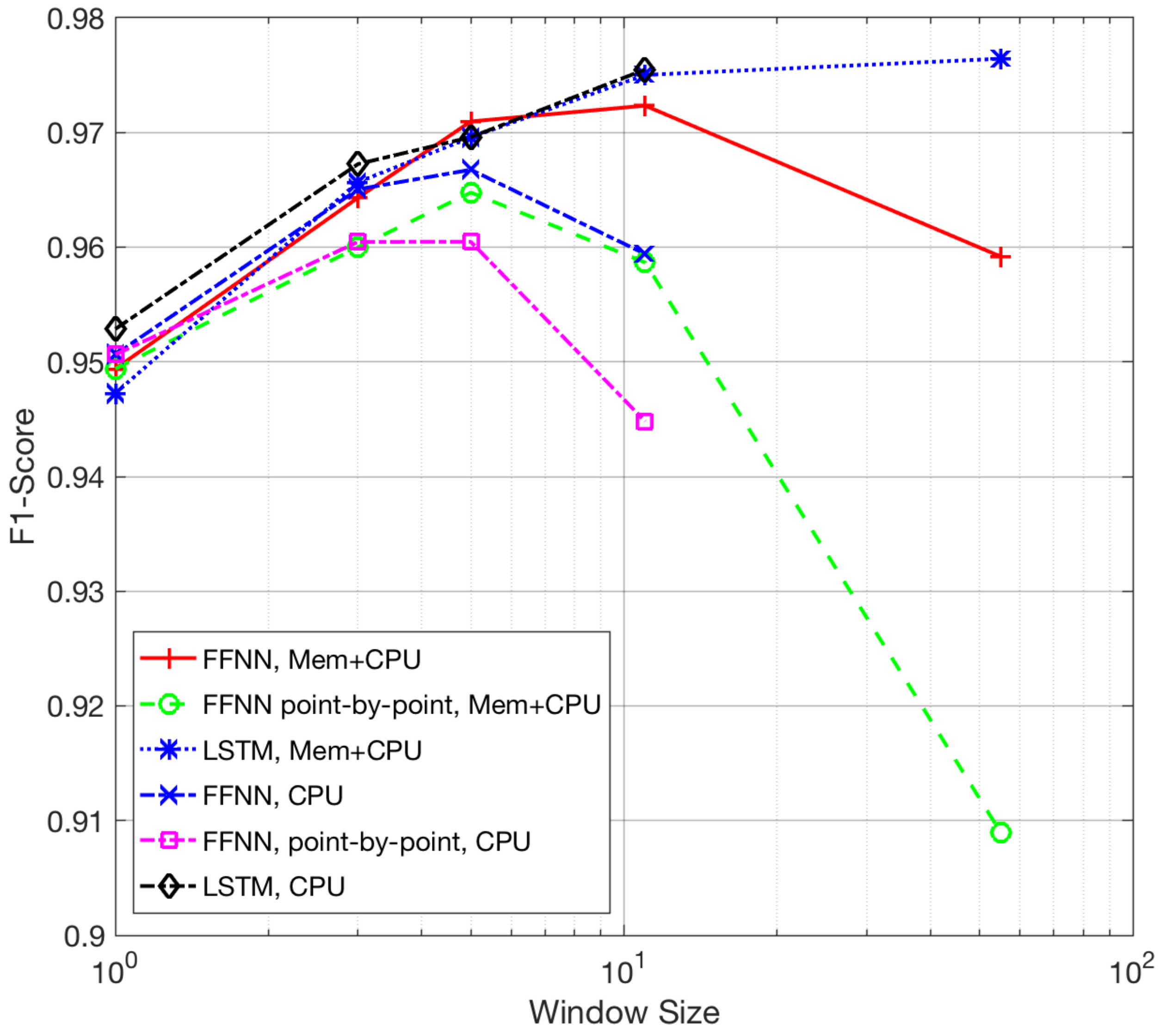


Figure 5.10: Model F1-Score vs. window size

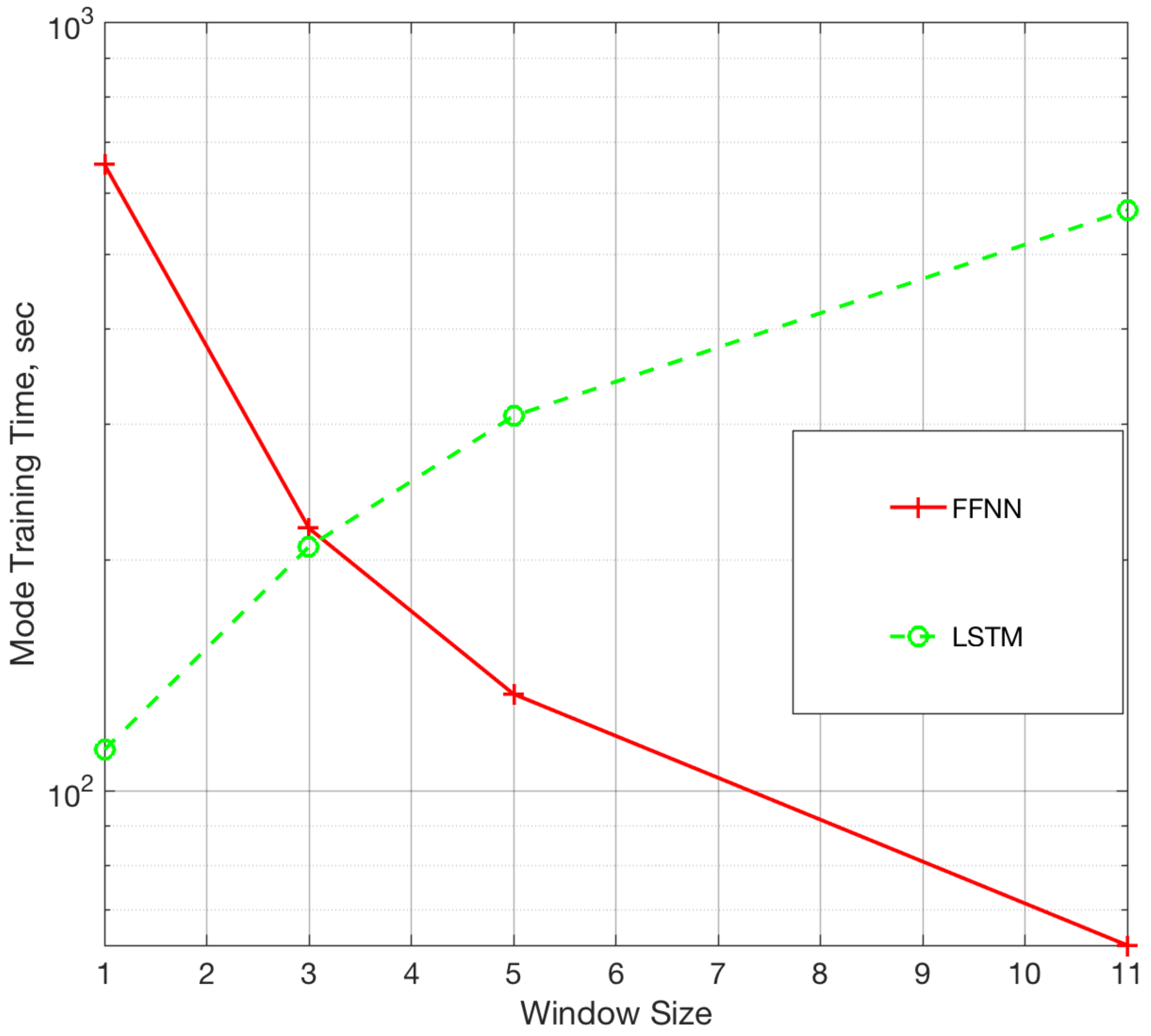


Figure 5.11: Model training time vs. window size

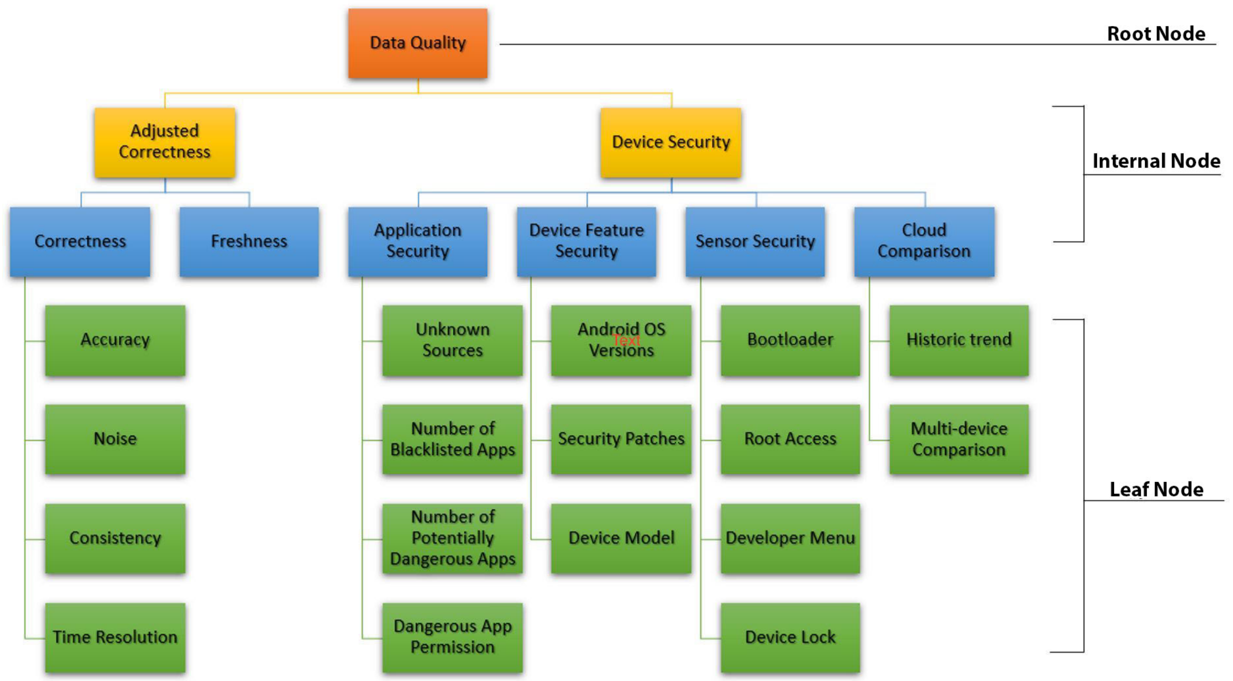


Figure 5.12: The metrics hierarchy tree for data quality and security evaluation expert system.

Fuzzy rules allow dealing with vague inputs such as "old/fresh" Android OS version, or "good/bad" sensor accuracy. For example, for the security patches, when do they become old? Maybe one day? One week? Or even six months? Assume the "expiration date" of the security patches is one week. Is it fair to say that a security patch released eight days ago is "old" and the one that is released seven days ago is "fresh"? Fuzzy logic can handle such ambiguous definitions. This subsection further provides a bottom-up view of the expert system implementation by stepping through the development process, starting from the first-order metrics up to the overall data quality score (see figure 5.12).

Let us focus our attention on the "Device Feature Security". Three metrics within this higher-order metric are "Android OS Versions", "Security Patches", and "Device Model". Among the 15 Version of Android OS, the majority of the users have Version "Marshmallow" (API 23) or above. Version "Lollipop" (API 22) is not only discontinued within the device manufactures, but also rarely available on the second-hand market. Therefore, two member-functions, "older" and "newer", were chosen to represent the Android OS Versions, as shown in figure 5.13.

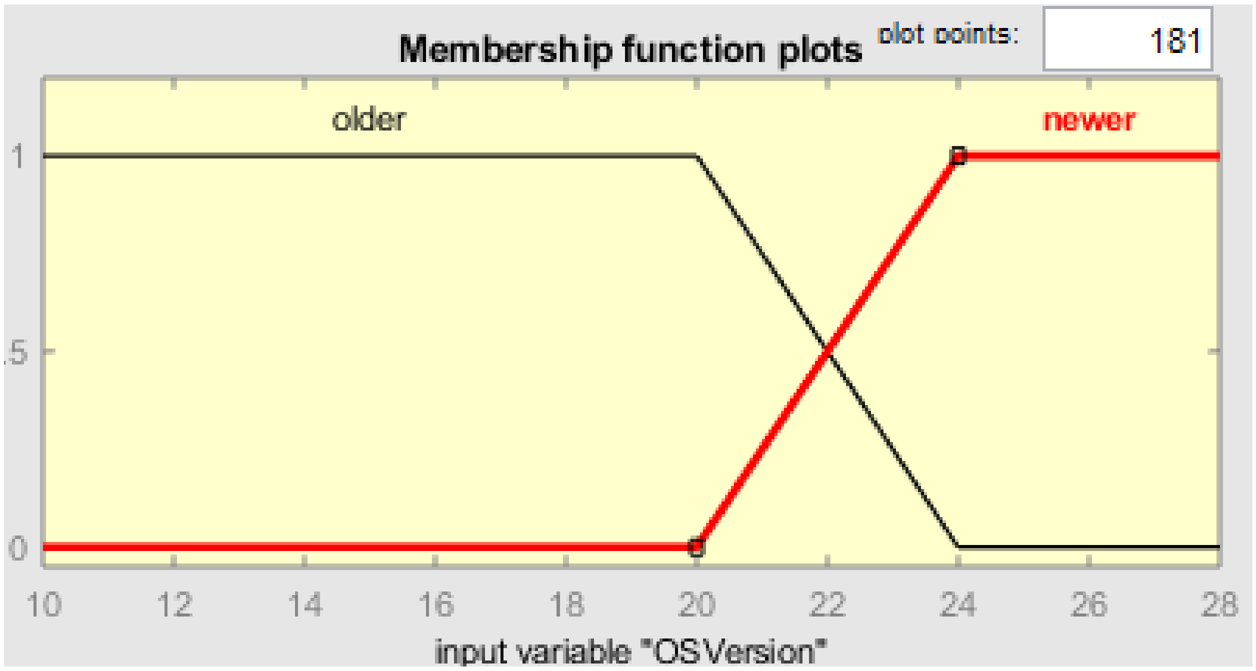


Figure 5.13: Membership functions for device feature security input - Android OS version.

The output from this module is split into three categories – low, medium, and high, as shown in Figure 5.14. More fuzzy rules of the "Device Security Feature" ES module are given below:

- IF (OSVersion is older) AND (SecurityPatch is outdated) AND (DeviceModel is older) $THEN$ (DeviceFeatureSecurityScore is low)
- IF (OSVersion is older) AND (SecurityPatch is outdated) AND (DeviceModel is newer) $THEN$ (DeviceFeatureSecurityScore is medium)
- IF (OSVersion is older) AND (SecurityPatch is upToDate) AND (DeviceModel is older) $THEN$ (DeviceFeatureSecurityScore is low)

Human experts formulate the rules based on their knowledge. Conflicting rules are resolved with weight assignments. The metrics, such as the Android OS version, security patches age, and the device model parameters, are gathered from the smartphone and passed into the fuzzy inference

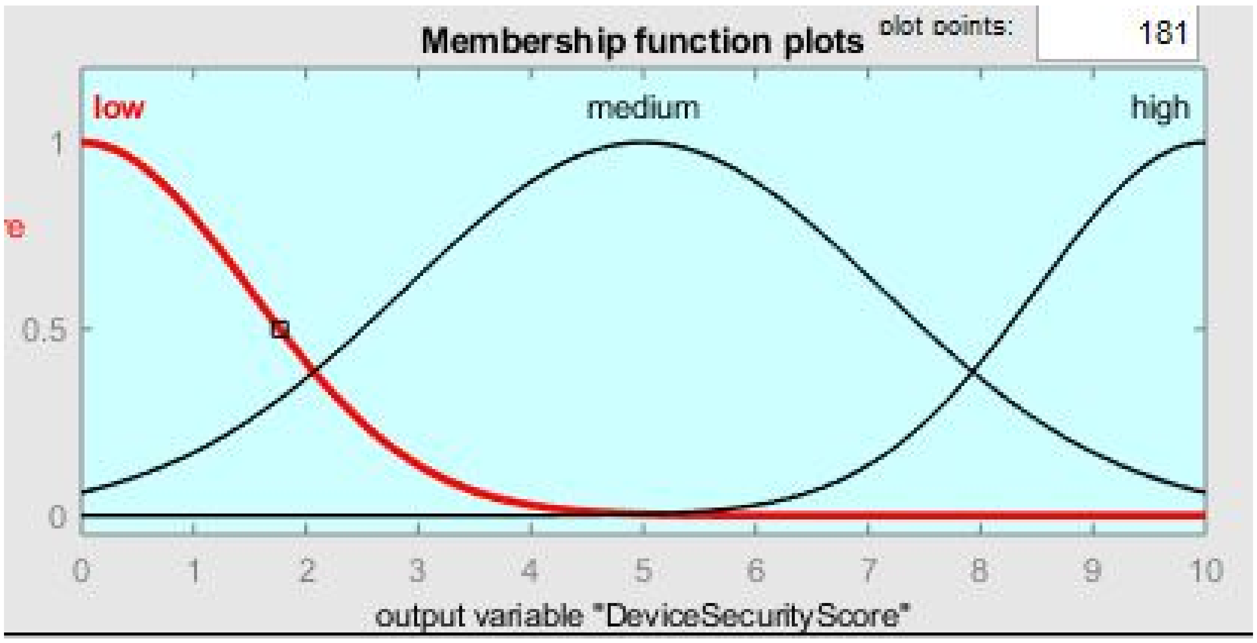


Figure 5.14: Membership functions for device feature security output.

engine (FIE) described above.

5.6 Distributed Architecture for the Framework Implementation

In this section, we present the developed expert system (see details in section 5.5) implementation for the resource-constraint devices, such as Android OS smartphones.

5.6.1 Cloud Component Implementation

This section renders the implementation of the cloud component that performs additional analysis of smartphones' security metrics, which are later used as the input for the overall security evaluation component.

The state of any device may change over some time. How do we adapt to it? If we have prior knowledge of the device's former state, then it can be a boon. By combining the prior knowledge

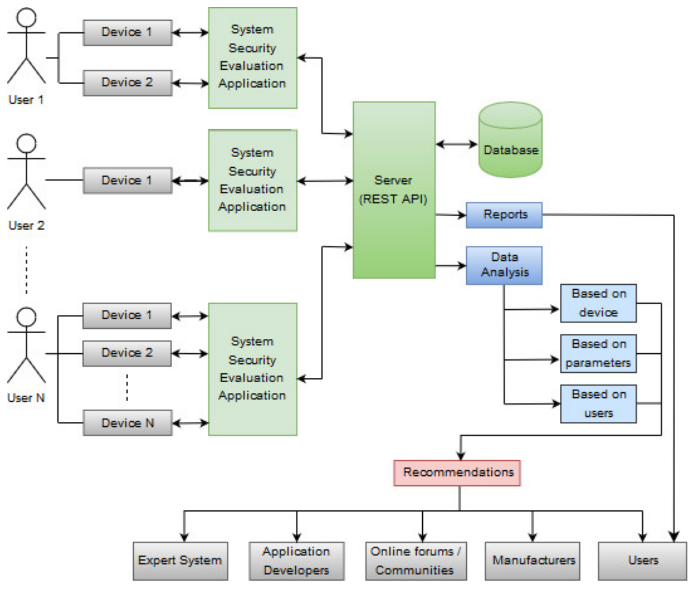


Figure 5.15: The cloud module

of the device's previous security states, along with the system that is responsible for evaluating the possible existing and new threats, we can develop a more efficient security evaluation system.

The device's previous security evaluation score can be stored on the device itself, but it is not beneficial to do so. Recording data on the device consumes some memory and is not recommended. In the case where the application experiences failure, there exists the possibility of relevant information being lost. Analyzing results obtained after evaluating the security parameters for a given device, would be tedious as it would involve direct communication between the devices. This would result in the need for requesting the user's permission to use the data stored on their device when attempting to assist a different user. Large amounts of data would be frequently communicated over the network unnecessarily.

Alternatively, if the results are stored in a single place for all users, it provides a unique opportunity to use and organize the user's device efficiently. The server application communicates with all the devices running the "System Security Evaluation" application. Each device has its instance of the Android OS application mentioned above. A user having multiple devices can have these devices tracked in one location. Figure 5.15 presents the flow of the data generated on various devices and the analysis that is performed. The data can then be used to help the user to make the device more secure. The server application contains information such as the score details, the user information, and the comparison score. This information can help to analyze the data from different aspects and helps to gain more insights. The server application can also generate detailed history reports of the score changes.

This data can be analyzed in different ways:

- based on the device type, e.g., how many devices from a specific manufacturer never allow installing applications from unknown sources;
- based on the device parameters, e.g., what is the percentage of the devices that never enable screen-lock mechanisms;
- based on the users, e.g., how many users have more than one device.

This information analysis can then be given to users to help them maintain and improve the security of their devices. Similarly, this information can be provided to the manufacturers to foster the development of more secure devices. Lastly, developers would also find this analysis useful to develop secure applications considering the vital aspects of a given device. The expert system can also benefit from the input of this information. Whenever the android application is about to create a new entry on the server, the user's permission would be requested. For keeping track of different devices against a given user, the device's IMEI number is recorded in the database (only with a user's consent).

Figure 5.16 demonstrates that a web application can also use cloud service. All the points discussed above are also valid for this web application. This web application can give the user an additional interface to access the relevant information. A user having multiple registered devices can see all the information from those devices from a single place. Otherwise, the user would have to manually go through the application to access the information pertaining to every device. We further discuss the details of the API discussed.

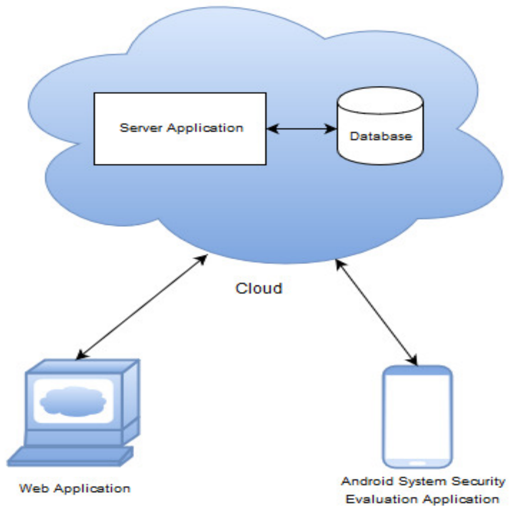


Figure 5.16: Cloud Benefactors

API Design

There exist various architectural patterns of developing application programming interfaces, but two of them are the most popular: Simple Object Access Protocol (SOAP) and Representation State Transfer (REST). Based on the recommendations [83], we chose REST as it has better throughput and faster response time than SOAP. SOAP involves a lot of overhead, which is a huge drawback [83]. The REST properties, such as simplicity, usability, and scalability, made it suitable for the back-end system designed in this research [86].

API is designed in order to permit its usage on different platforms. We followed the Richardson Maturity Model (RMM) [108] and implemented a REST API of level 3, which allowed us to provide services that would take hypermedia as an engine of the application state. Also, the Model-View-Controller pattern design results in a loosely coupled API [119].

Outer-facing contract. The outer facing contract for the consumer of the API is a specific part of the API design. It consists of a Uniform Resource Identifier (URI), HTTP methods, and an optional payload. The payload can have data in a format such as JSON or XML.

Designing the URI. For designing the URIs, appropriate resource names (nouns) should be used, rather than the actions that can be performed on those resources.

Unique ID. Global Unique Identifier (GUID) usage is preferable rather than using the auto-numbered database field, for having a unique identifier as it remains the same even if their database is changed. This is not true in the case of auto-numbered unique identifiers.

Status Codes. A REST API should be configured with appropriate status codes, as they provide a lot of valuable information regarding the communication taking place between the API and its consumer.

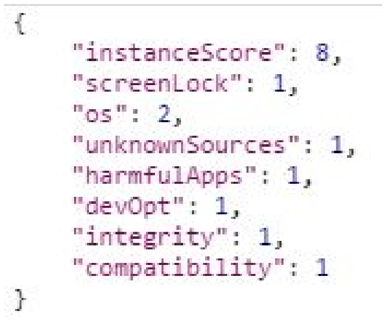
Media Types. Requests and responses can be encoded in various standard formats. A customized media type can also be designed and used for passing customized messages as a part of its communication.

Security score comparison. The security score comparison is one of the most vital functions of the back-end system. A new score is generated after evaluating the device and is then received by the API. At this point, the score is compared with the latest scores from the other devices. As a response to the message containing the score, the API also sends a score comparison result in the form “x/y”, as an indication that the given device’s score is the X_{th} best among Y devices. Along with the score, the values of the system scoring parameters are also saved. The scores from different devices can be used to analyze and gather more valuable information. Figure 5.17 and figure 5.18 present examples of a JSON request and response when a request has been made to post a score of a user’s device.

Faults logging. Faults occurring in any part of the API during its execution should be properly logged, and preferably in a single file, allowing the errors to be easily traced.

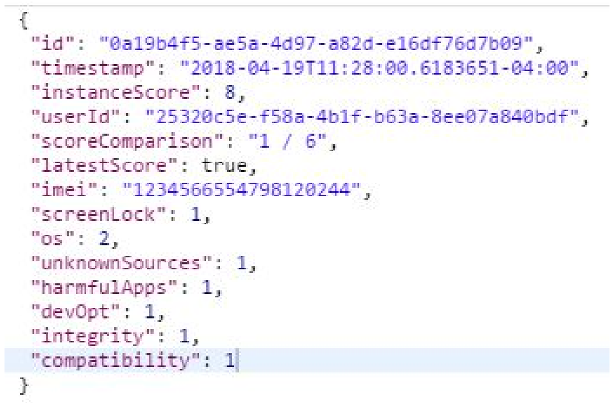
Caching. To decrease the network bandwidth and the number of requests being processed, caching can be implemented.

Protecting the API. The API can be protected by rate-limiting, i.e., limiting the number of



```
{
    "instanceScore": 8,
    "screenLock": 1,
    "os": 2,
    "unknownSources": 1,
    "harmfulApps": 1,
    "devOpt": 1,
    "integrity": 1,
    "compatibility": 1
}
```

Figure 5.17: JSON request for posting a new score



```
{
 "id": "0a19b4f5-ae5a-4d97-a82d-e16df76d7b09",
 "timestamp": "2018-04-19T11:28:00.6183651-04:00",
 "instanceScore": 8,
 "userId": "25320c5e-f58a-4b1f-b63a-8ee07a840bdf",
 "scoreComparison": "1 / 6",
 "latestScore": true,
 "imei": "1234566554798120244",
 "screenLock": 1,
 "os": 2,
 "unknownSources": 1,
 "harmfulApps": 1,
 "devOpt": 1,
 "integrity": 1,
 "compatibility": 1
}
```

Figure 5.18: JSON response for posting a new score

requests received from a specific IP within a fixed time span.

5.6.2 Expert System Implementation on a Mobile Platform with Neural Networks

Neural Network Architecture

To overcome the limitation of the expert systems implementation on mobile devices, we produce a neural network hierarchical architecture that follows up the ES structure. Each NN model approximates an input/output hyper-surface of a corresponding ES module. ES implementation consists of eight modules (see Figure 5.12) that correspond to application security, device feature security, sensor security, cloud comparison, correctness, adjusted correctness, device security, and

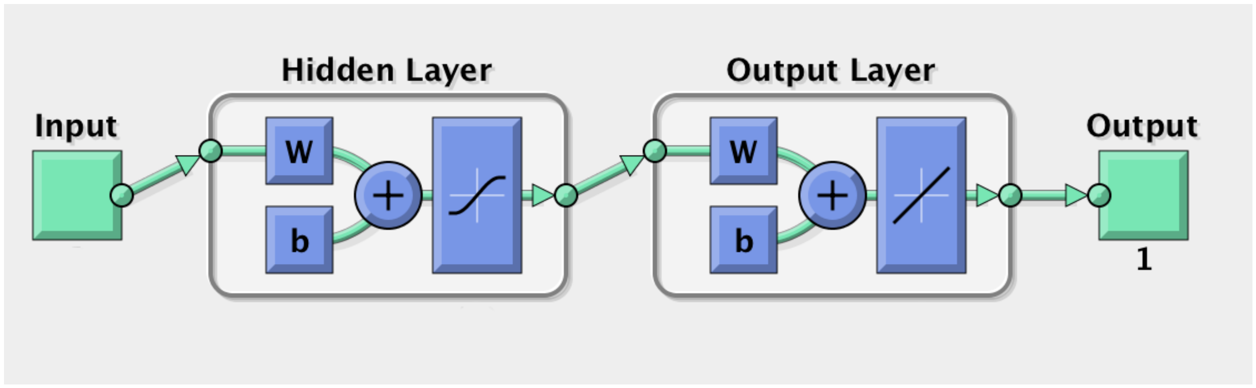


Figure 5.19: Data Quality dependency on adjusted correctness and device security. Neural network model.

data quality. The neural network implementation of the framework has a very similar structure with seven NN models: application security, device feature security, cloud comparison, correctness, adjusted correctness, device security, and data quality. Since sensor security ES module inputs are boolean (true/false), there are only 16 possible combinations of them. This module is substituted with a simple lookup table function. The advantage of creating separate NN models that directly substitute ES modules instead of creating an NN module that approximates the overall ES system is flexibility. In the case when rules have changed, only the corresponding model has to be retrained and replaced.

To approximate ES modules, a multilayer perceptron NN architecture is used. Each module has one hidden layer. The number of neurons in the input layer equals to the number of inputs of the corresponding ES module. The number of neurons in a hidden layer was chosen through performing the empirical study and varies for each model in the range from ten to 50 neurons. The output layer contains only one neuron. The generic architecture of used models is presented in Figure 5.19. All seven models were developed in MatLab using a neural network toolkit. The models were trained using the Bayesian regularization algorithm.

Table 5.8: Parameters of the NN models

NN Module	Number of neurons in hidden layer	Size of the training dataset (records)	Testing Mean Squared Error
App security	50	10249	0.024
Device feature security	30	2247	0.0015
Cloud security	10	161	0.0027
Correctness	50	10249	0.0082
Device security	50	10249	0.0048
Adjusted Correctness	50	7141	0.0013
Data Quality	50	7141	0.00083

Neural Network Training

To produce a training dataset, we generated an extensive number of possible combinations (permutations) of an input to an ES module. To do this, we iterated each input with a reasonable interval. For example, for the "app security" module, there are four inputs, each can take any real number between "0" and "1". We iterated each of the inputs with delta equals "0.1". This gives 11 values for each input. However, the number of all possible input combinations equals $11^4 = 14641$. Then, 70% of these inputs (randomly) were taken to train the model, 15% were taken for the cross-validation, and 15% were taken for testing. Thus, in the aforementioned examples, 10249 records were used for training, 2196 were used for validation, and 2196 were used for testing. Table 5.8 presents parameters of each model. It is worth noticing that in the column "Size of the training dataset (records)" is the actual number of records used for training (70% of the overall generated dataset).

Neural Network Models Validation and Implementation

To validate the developed models, we performed unit testing and then integration testing of the whole system. We generated 10000 system inputs and fed them to the developed expert system and NN models. One of the possible issues of substituting each ES module with the NN model is the error accumulation. However, our system gives only 1.5% of the mean squared error, which is reasonably good for the ES as human expert errors may have higher rates.

After all NN models were trained and tested, they have been implemented in JAVA functions. Since these functions mostly consist of matrix multiplication, they can be easily executed in the Android OS environment. An Android application has been developed that calculates the overall DQ score using trained NN models and demonstrates the framework functionality. We performed a test with 1000 DQ evaluations to demonstrate the performance benefits of NN models employment instead of ES modules. Android smartphone Pixel 2XL with mobile chipset "Qualcomm Snapdragon 835", 4GB of RAM, and Android v.9.0 using NN models finished evaluation in 0.2 seconds, while a desktop computer with CPU Intel Core-i5 (I5-6500), 32GB of RAM, and MacOS v.10.12.6 using ES finished DQ evaluation in 64.8 seconds. Unfortunately, ES and NN have to be compared on different platforms due to a lack of ES implementation for the mobile platform. However, the mobile platform computationally weaker than the desktop platform, yet NN DQ implementation works 324 times faster.

5.7 Framework Validation Use Case One: Data Quality and Security Evaluation for Sensor Originated Data

This section presents a generic example of DQ and security evaluation of sensor originated data that is collected from a smartphone in a crowd-sourced application.

A company MedResML is developing an application that can detect specific patterns in a human walk that may indicate severe health problems. Building up a model requires an enormous amount of data collected from an accelerometer, a gyroscope, a step counter, and a magnetometer during a person's walk. MedResML decided to employ a crowd-sensing approach and has distributed an Android OS application that collects data from volunteers. To improve the quality of the collected data, MedResML used conventional DQ evaluation techniques.

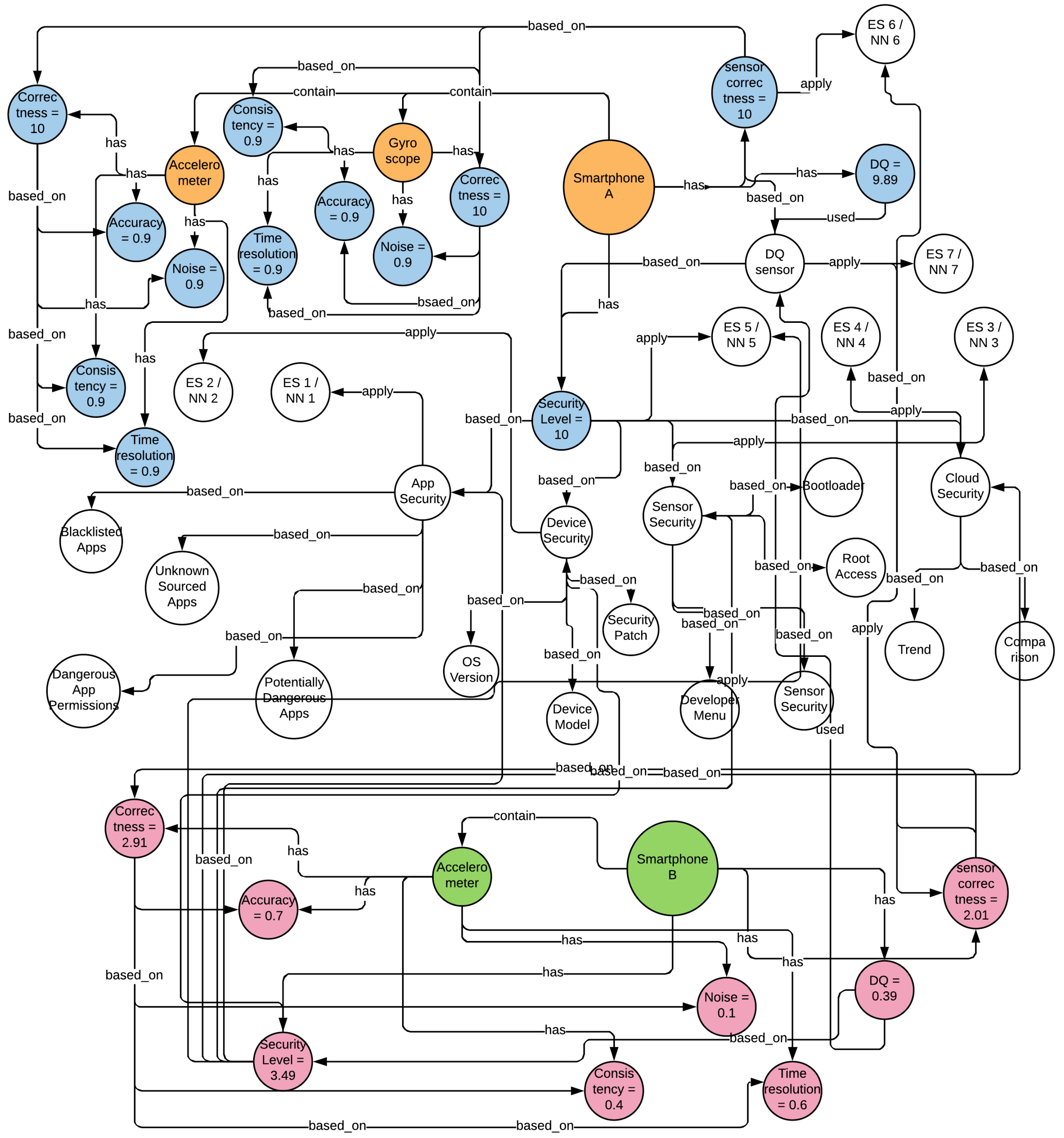
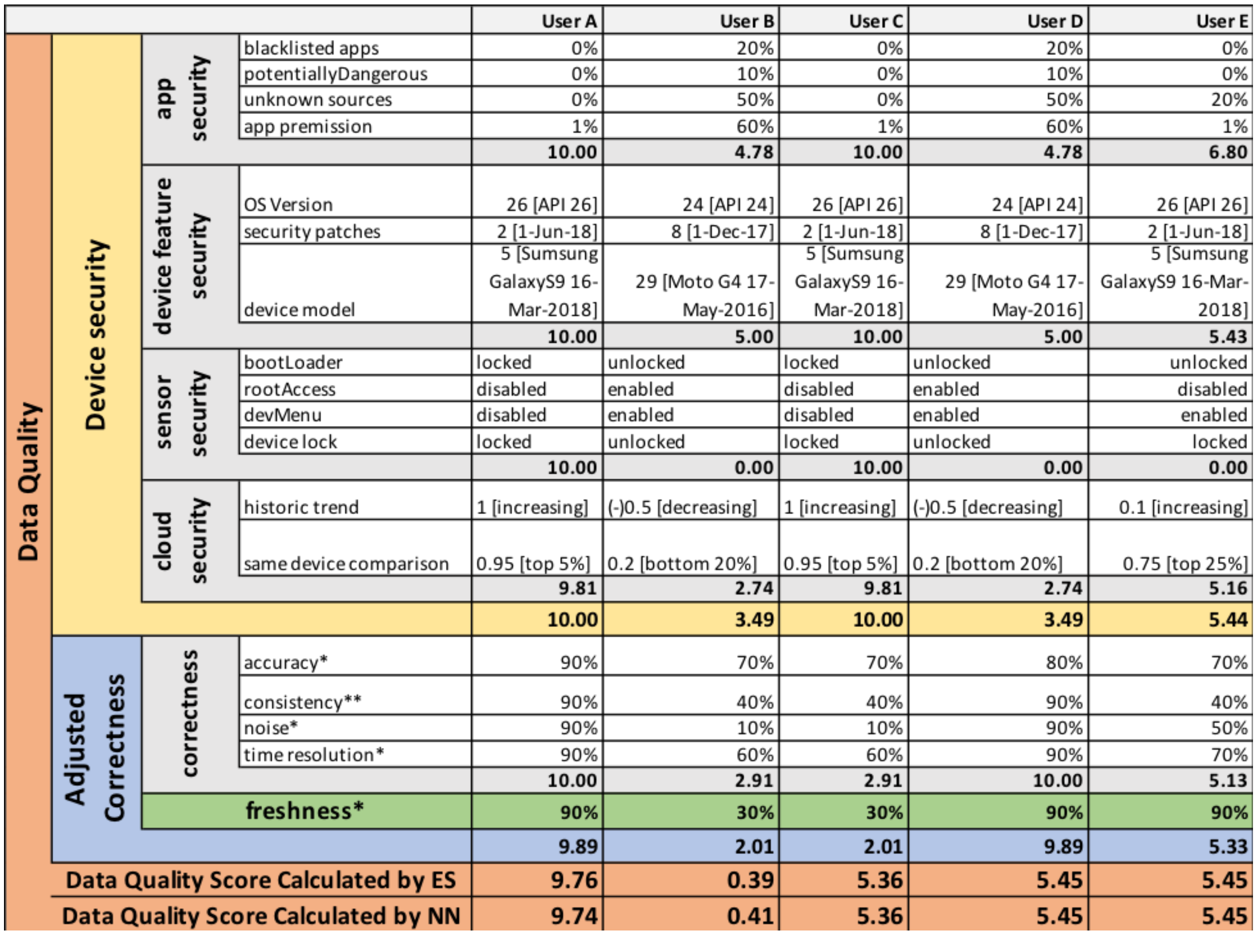


Figure 5.20: Employing the knowledge graph in data quality evaluation. Sub-graphs for user A and B (see table 5.9) are shown.

To collect data, the participated volunteers use their own smartphones. Unfortunately, some of these volunteers are not very responsible smartphone users and have a low-security level of their

Table 5.9: Use Cases Summary



				User A	User B	User C	User D	User E
Data Quality	Device security	app security	blacklisted apps	0%	20%	0%	20%	0%
			potentiallyDangerous	0%	10%	0%	10%	0%
			unknown sources	0%	50%	0%	50%	20%
			app premission	1%	60%	1%	60%	1%
				10.00	**4.78**	**10.00**	**4.78**	**6.80**
		device feature security	OS Version	26 [API 26]	24 [API 24]	26 [API 26]	24 [API 24]	26 [API 26]
			security patches	2 [1-Jun-18]	8 [1-Dec-17]	2 [1-Jun-18]	8 [1-Dec-17]	2 [1-Jun-18]
			device model	5 [Sumsung GalaxyS9 16-Mar-2018]	29 [Moto G4 17-May-2016]	5 [Sumsung GalaxyS9 16-Mar-2018]	29 [Moto G4 17-May-2016]	5 [Sumsung GalaxyS9 16-Mar-2018]
				10.00	**5.00**	**10.00**	**5.00**	**5.43**
		sensor security	bootLoader	locked	unlocked	locked	unlocked	unlocked
			rootAccess	disabled	enabled	disabled	enabled	disabled
			devMenu	disabled	enabled	disabled	enabled	enabled
			device lock	locked	unlocked	locked	unlocked	locked
				10.00	**0.00**	**10.00**	**0.00**	**0.00**
		cloud security	historic trend	1 [increasing]	(-)0.5 [decreasing]	1 [increasing]	(-)0.5 [decreasing]	0.1 [increasing]
			same device comparison	0.95 [top 5%]	0.2 [bottom 20%]	0.95 [top 5%]	0.2 [bottom 20%]	0.75 [top 25%]
				9.81	**2.74**	**9.81**	**2.74**	**5.16**
				10.00	**3.49**	**10.00**	**3.49**	**5.44**
	Adjusted Correctness	correctness	accuracy*	90%	70%	70%	80%	70%
			consistency**	90%	40%	40%	90%	40%
			noise*	90%	10%	10%	90%	50%
			time resolution*	90%	60%	60%	90%	70%
				10.00	**2.91**	**2.91**	**10.00**	**5.13**
		freshness*		**90%**	**30%**	**30%**	**90%**	**90%**
				9.89	**2.01**	**2.01**	**9.89**	**5.33**
	Data Quality Score Calculated by ES			**9.76**	**0.39**	**5.36**	**5.45**	**5.45**
	Data Quality Score Calculated by NN			**9.74**	**0.41**	**5.36**	**5.45**	**5.45**

* - *All metrics such as accuracy, noise, time resolution are scaled and take values from 0% to 100%. 0% denotes the less desirable value and 100% corresponds to the most desirable value. In the parenthesis are absolute values.*
** - *Consistency is scaled between 0% and 100%. In the parenthesis are absolute values (how many records were missed).*

devices, which may lead to malicious data alternation. Conventional methods of DQ evaluation, which do not consider the source trustworthiness and security, may assign high-quality indicators to data acquired from these devices.

The sensor data evaluation alone does not provide sufficient information to identify data integrity violations. Thus, the company decides to include a device security evaluation into conventional DQ estimation procedures. In this case, the application that gathers sensor data also acquires information from the framework's module, which is responsible for the device security evaluation. Table 5.9 lists examples of parameters of five devices from individual volunteers. Furthermore, we present use cases that demonstrate security evaluation influences on DQS evaluation, and vice versa.

5.7.1 Knowledge Graph Vocabulary in Sensor Originated Data Quality Evaluation

In this example, we extend our vocabulary with a set of logical and physical entities such as "smartphone", "correctness", "accuracy", "noise", "consistency", and "time resolution". "Smartphone" is a physical entity that extends the "Device" entity, and represents a smartphone of a user that participates in a crowd-sourcing application. "Accuracy" and "noise" reflects similar physical values of a sensor.

The noise value is scaled between the maximum and minimum values, which are defined by the data consumer. After scaling, noise is represented with a dimensionless value between zero (minimum noise value) and 100 points (maximum noise value).

"Consistency" represents the number of records that are missing. If consistency is 100 points, all records are present. Zero points of the consistency mean that the maximum value (a user also defines that) of missing records is reached. "Time resolution" reflects how often data can be acquired. It is also scaled between zero and 100 points. "Correctness" entity integrates noise, consistency, time resolution, and sensor accuracy (that depends on a sensor model) and represents sensor DQ.

5.7.2 Calculus Implementation in Sensor Originated Data Quality Evaluation

In this example, two major aspects are integrated into the overall DQ score: sensor DQ and smartphone security. Security evaluation calculus is presented in section 5.1 in details. This subsection presents sensor DQ evaluation calculus.

A smartphone may provide the raw data from sensors such as an accelerometer, a gyroscope, a GPS, a magnetometer, a gravity sensor, an illumination sensor, a proximity sensor, etc. Each "sensor" entity has technical characteristics of the sensor hardware that participate in DQ evaluation. In this use case, two types of calculus implementation were used: based on expert system and based on a neural network that approximates the expert system input-output surface.

We developed an expert system module for a sensor correctness evaluation that takes as inputs sensor accuracy, noise, consistency, and time resolution. Scaling these parameters between "0" and "100" allows us to use developed calculus for various types of sensors. In order to implement this calculus on the resource constraint mobile sensor-platform such as Android OS-based smartphone, we substituted the developed expert system with the neural network models. The process is identical to that one is described in section 5.6.

Figure 5.21, figure 5.22, and figure 5.23 show the input/output surfaces of the developed expert system modules.

5.7.3 Knowledge Graph Employment in Sensor Originated Data Quality Evaluation

The knowledge graph acts as a database due to each new data object (a set of measurements), and its DQ estimates are stored in the knowledge graph as new entities. The developed calculi are presented in a knowledge graph as logical "calculus" entities. Figure 5.20 presents a simplified sub-graph structure of the knowledge graph. One can observe that a hierarchical structure of a security evaluation is also implemented in the knowledge graph. In addition, DQ estimation may be easily modified by changing corresponding entities in the knowledge graph.

5.7.4 Study of Sensor Originated Data Quality Influence on Integrated DQ

This case study involves user A and user C. As stated in table 5.9, both user A and user C have excellent device security, both devices have the following characteristics:

- Application security - no suspicious applications, no unknown source applications, and relatively strict permission controls.
- Device feature security – Android OS Oreo (API 26), security patch installed two months ago, and running on Samsung Galaxy S9, which was a device release on June 1st, 2018, about five months ago.
- Sensor security – bootloader is locked, root access is disabled, developer menu is disabled, and has a pin code locking mechanism.
- Cloud security – the device evaluation history has an upward trend.

Naturally, both user A and user C achieve a high score of 10 on device security evaluation. We now have a secure environment that can foster the production of high-quality data. Since both users A and user C have highly secure devices, does that mean the data quality is high for both devices? Looking back at the lower portion of the table, a clear distinction in the correctness of the sensors between the first and second device stands out.

User A has relatively good accuracy, consistency, noise, and time resolution, which are +/-28mg, 20, 1.3mg, and 2.9ms, respectively. This grants user A's device a score of 10/10 for sensor correctness. Moreover, this sensor measurement data is collected not long ago, as shown in the metric of freshness that adjusts the correctness score slightly based on the purpose of this evaluation, as shown in figure 5.22. In this case, we placed the medium importance to the freshness of the data, resulting in the adjusted correctness score of 9.89/10. Lastly, we combine user A's device security evaluation, "10", and the adjusted correctness score, "9.89/10", and calculated the overall data quality score of "9.76/10".

On the other hand, user C's device sensors performed poorly with bad accuracy, consistency, noise, and time resolution, which are +/-44mg, 120, 4.7mg, 8.6ms, respectively. This gives this sensor correctness a score of 2.91/10. In addition, this data was collected more than a month ago, lowering their adjusted correctness score to 2.01/10. Coupled with the device security score, 10/10, user C's overall data quality score is now 5.36/10. Considering device security alone may lead the researchers to overlook important details that could tamper with the data quality. It is confirmed with user A and user C, which could be found sitting on the peak and the right middle slope of the data quality surface, respectively, as shown in figure 5.21.

5.7.5 Investigation of Device Security Influence on Integrated DQ

As stated in table 5.9, both user A and user D achieved scores of 10/10 on sensor correctness, for having sensors that are accurate, consistent, with low-noise and good time resolution. As opposed to case study 1, this case focuses on the adjusted sensor correctness, which is also the most widely used area of evaluating data quality on a smartphone. However, we must not forget the influence imposed by the data source environment on the data quality itself. In this case, the data source environment is the device security.

Comparing against user A, user D has the opposite setup, with an assumption of having 50 apps installed:

- App security – 10/20 apps are blacklisted, five apps are potentially dangerous, 25 apps are downloaded from unknown sources, and more than half of the apps are at risk of abusing dangerous permission.
- Device feature security – mediocre Android OS Nougat (API 24), installed security patch eight months ago and running on Moto G4 smartphone, which was released about 29 months ago.
- Sensor security – bootloader is unlocked, root access is enabled, developer menu is enabled,

and a device lock present.

- Cloud security – the device evaluation history has a downward trend.

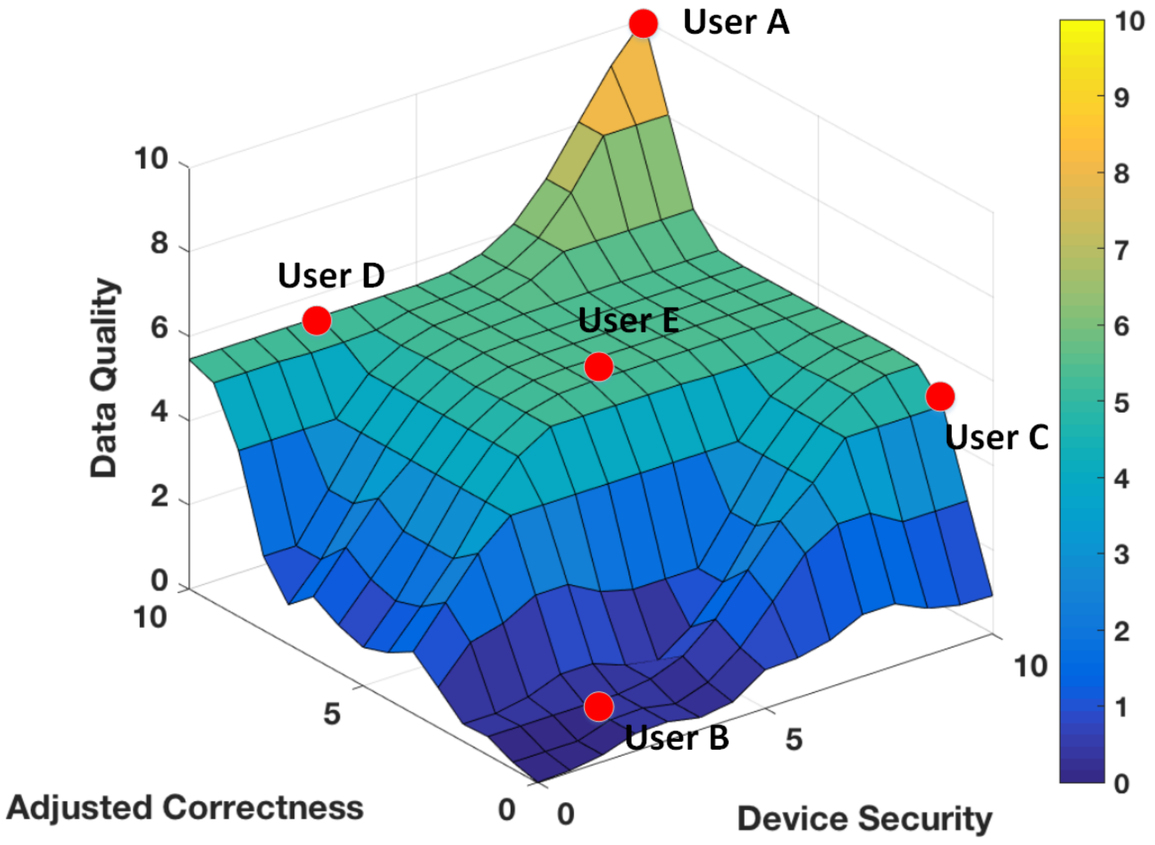


Figure 5.21: Data Quality dependency on adjusted correctness and device security.

The device security score for user D is 3.49/10, a good deal lower than user A's. It may not be apparent at first, why user D's sensor data seems so good considering the state of the device. However, as mentioned earlier, when device security is compromised, tampering the sensor data becomes a trivial task. For example, when bootloader and root access are enabled, legitimate users/apps can obtain superuser privileges, which may perform the modification of virtually everything on and about the device, such as faking good sensor data. The proposed framework considers this and combines both the device security and sensor correctness for the overall data quality evaluation -

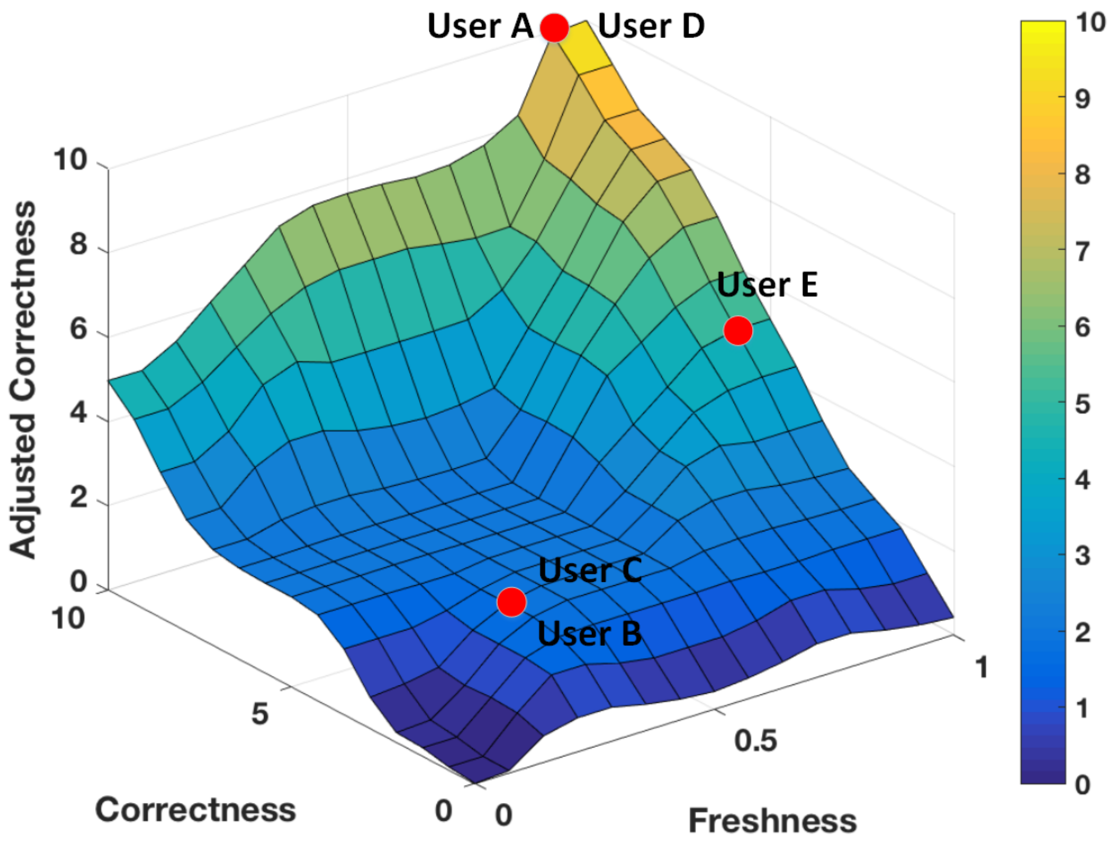


Figure 5.22: Adjusted correctness dependency on data correctness and freshness.

resulting in a score of 5.45/10 for user D. As shown in figure 5.21, user A sits at the peak of the data quality surface, while user D lays at the middle left slope, which leaves more to be desired for the device security feature.

5.8 Framework Validation Use Case Two: Data Quality Evaluation in the Road Pothole Reporting System

In this example, we consider a design of the system that receives reports from citizens about road potholes, calculates, and assigns the DQ evaluation to each report. DQ estimates are used

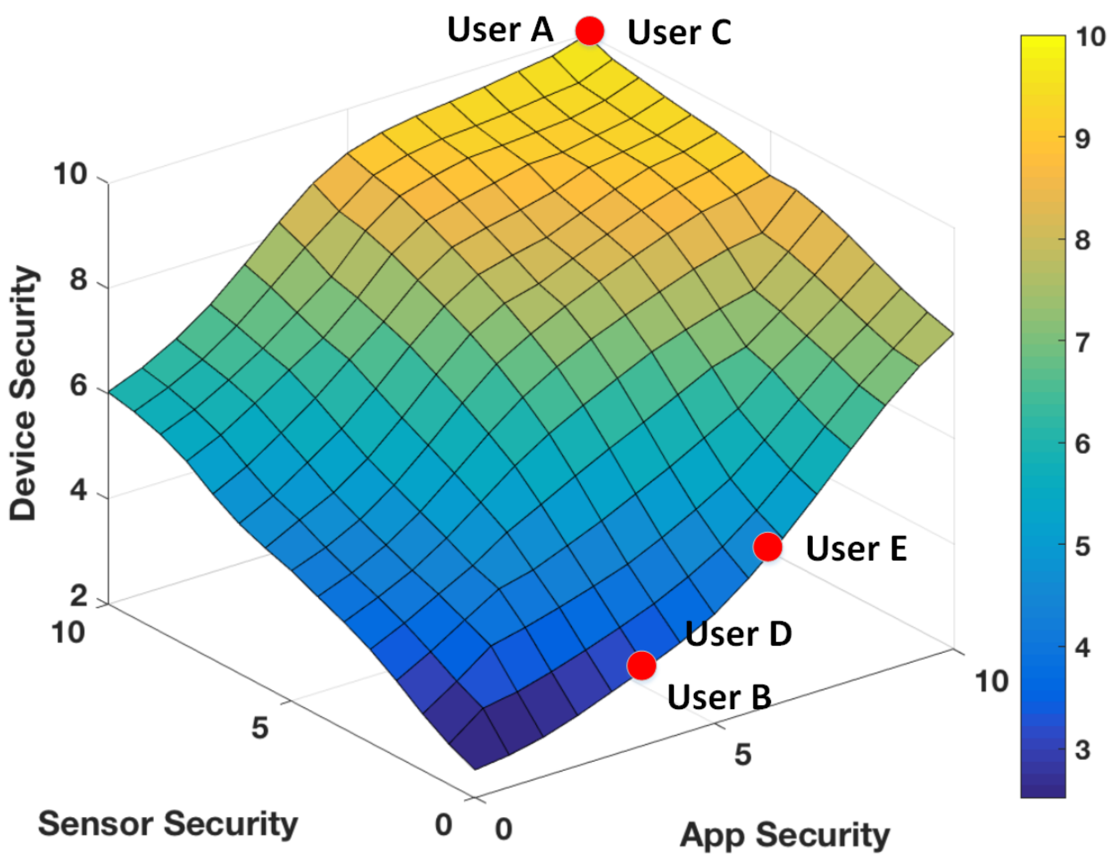


Figure 5.23: Device security dependency on sensor security and application security.

to prioritize necessary road works. Citizens may send reports about road potholes using their smartphones. We employ a ranking mechanism to evaluate a report's priority. Each road pothole report represents the data object (DO) and has DQ associated with it. DQ of the report includes such metrics as image quality, report completeness, and security characteristics.

5.8.1 Knowledge Graph Vocabulary for DQ Evaluation of Road Pothole Reporting System

In this example, we extend the initial vocabulary with new entities and their relations. The report is the DO that is augmented with its DQ value. Report is a logical entity that has such properties as “has”, “time”, and “belongs_to”. In addition, this entity has various data (values) that have to be present in the road-pothole report (see table 5.10). This report entity has “image quality”, “completness”, and “DQ”.

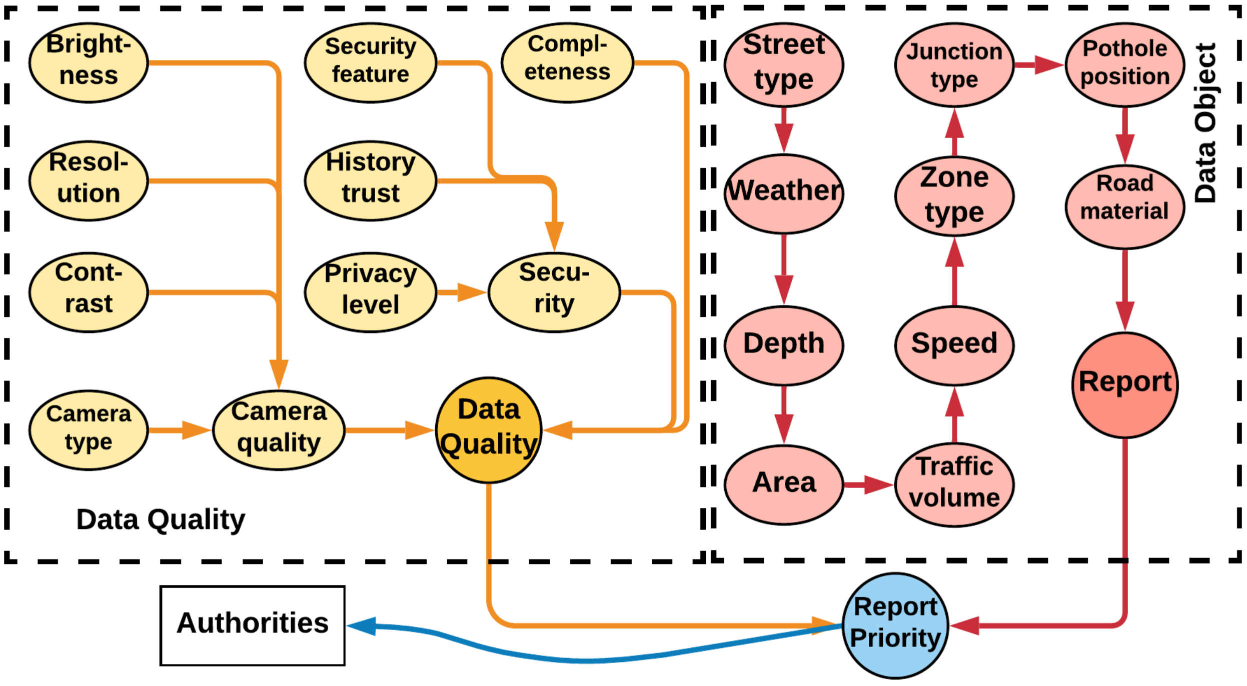


Figure 5.24: DQ indicator (report priority) calculation engine.

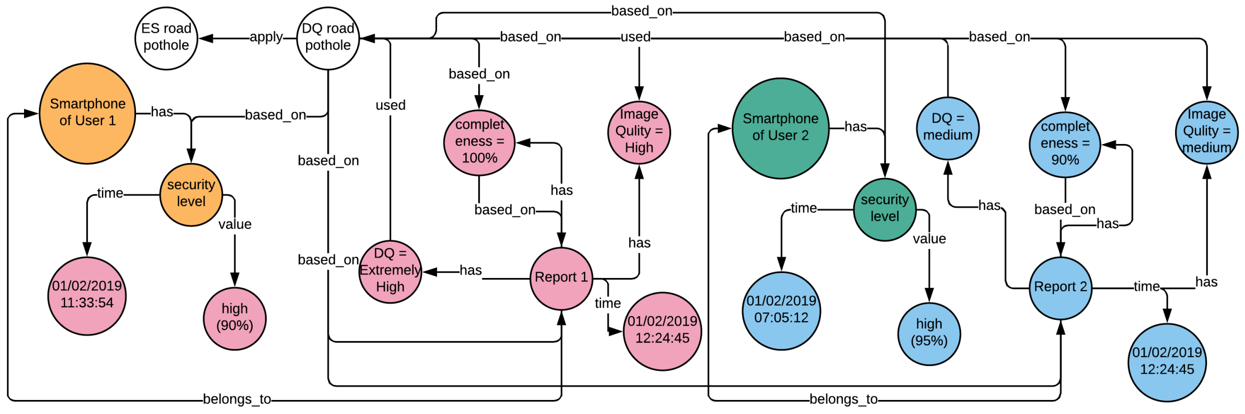


Figure 5.25: Knowledge graph and road pothole report DQ evaluation.

Image quality is calculated based on the quality of the camera that captured the pothole image. In this example, image quality is calculated using such image characteristics as camera type (smart-

Table 5.10: Pothole Cases Table

	Factors	Instance 1	Instance 2
Metrics	**Completeness**	100%	90%
	Image Quality	High	Medium
	Device Security	High	High
Data Object	Weather	Sunny	Sunny
	Street Type	Arterial	Arterial
	Depth	16 cm	***No Data***
	Area	20 sq. ft.	20 sq. ft.
	Traffic Vol.	Extremely High	Extremely High
	Avg. Speed	High	High
	Safety Zone	School Zone	School Zone
	Junction	Intersection	Intersection
	PPRRW	Center	Center
	Material	Asphalt	Asphalt
	Repair	Permanent	Permanent
DQ	*Priority*	Extremely High	Medium

phone type), brightness, contrast, resolution, noise, and exposition. The good quality of the image allows the system to estimate the required work. Data completeness specifies a user trust measure based on the ratio of the number of questions answered by the user to the number of answers requested by the reporting software. In the case of new sensors employment, the DQ metric set can be further extended to include such metrics as noise, consistency, time resolution, and data freshness. Security characteristic measure is based on techniques presented in the section 4.1 an in [69,70,78,138]. To determine the report's trustworthiness, a hierarchical framework that analyzes installed applications, embedded mobile security features, device privacy settings, and a historical device trust is employed.

The left part of figure 5.24 shows the data quality rule engine. Data quality is determined by taking into consideration the data completeness, camera quality, and security. The quality degree is scored as a percentage. Completeness score is calculated based on the ratio between the number of questions answered by the user to the number of answers requested by the reporting application. Camera quality score is determined from image quality parameters, and meta-data (EXIF) provided along with the photo image. The security score is determined by analyzing the smartphone security characteristics. After calculating the score for each factor, it is multiplied by each factor's weight. The final weighted sum gives the calculated data quality. The priority score is calculated through the forward-chained, rule-based intelligent system and represents the integrated DQ value. The priority (DQ) is represented in percents.

When all the data is gathered, it is incorporated into a database, which may be used to analyze and schedule potholes repair works. In addition, a new entity of the report and its DQ is created in the knowledge graph. The goal is to analyze the data and to detect patterns like regions, which are extremely prone to specific types of potholes, specifically sized potholes, how the weather conditions affect potholes in a particular area. After the sufficient amount of data accumulated in the database, it may increase the ability to find answers on many questions related to road maintenance and help in the effort of prognosis and actions on pothole-related maintenance.

Table 5.10 presents two samples of road pothole reports. One can notice that *Instance 1* of the report and *Instance 2* of the report represent the data about the same road pothole. However, the DQ metrics of these instances are different. In addition, "Instance 2" lacks one of the fields, which impacts on DQ as well (because of the lower "completeness" score). As a result, the "Instance 1" of the report gets a higher priority than the "Instance 2". Authorities can have more trust in this report than in the "Instance 2". This application allows authorities to reduce data processing efforts and to focus on other tasks.

5.8.2 Knowledge Graph Employment in DQ Evaluation of Road Pothole Reporting System

Figure 5.25 presents the role of knowledge graph in road pothole report DQ evaluation. The report is represented by an entity with basic fields that are described in section 3.2.1 as well as with new properties. This example demonstrates how the knowledge graph facilitates the automation of DQ evaluation adoption by new IoT applications. Entities of pink and blue colors illustrate new knowledge generation after the report was sent and its DQ value evaluated. In the figure 5.25

two examples that are represented in table 5.10 are shown. Entity "ES road pothole" points to a metrics integration calculus. "Report 1" and "Report 2" entities correspond to "Instance 1" and "Instance 2" in the table 5.10.

5.9 Framework Validation Use Case Three: Data Quality Evaluation in the Crowd-sourcing Application

This use case demonstrates the framework application in the crowdsourcing project for non-sensor originated data. ICitizen is a crowdsourcing platform that bridges citizens and decision-makers. It is a nonpartisan civic engagement platform where people can influence action on the issues they care about, and decision-makers can inform policy with real-time feedback. People vote on and promote issues and policies that affect their lives. Decision-makers use that feedback to inform policy and make data-driven decisions. By working together online, people and their leaders build stronger, more connected communities in the real world. ICitizen users are general citizens, candidates, organizations, elected representatives, officials, and schools, which act as data consumers.

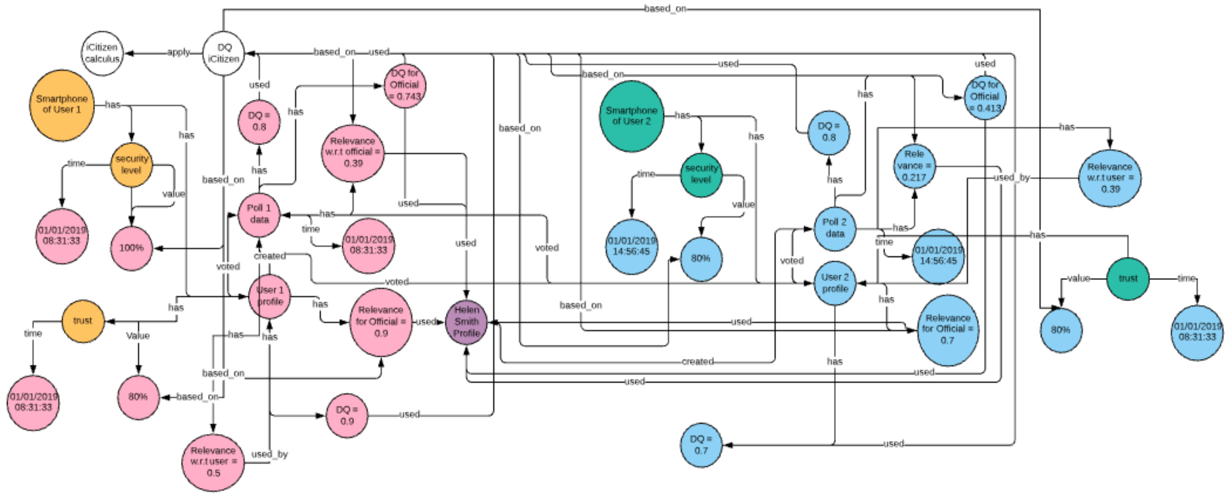


Figure 5.26: Knowledge graph employment in iCitizen DS calculation.

This platform uses an Android Application where a user can take or generate polls and ideas. Their responses are recorded and sent to the server where they are stored for further use. The responses, polls, and ideas are augmented with user profile information and other relevant meta-data.

5.9.1 Knowledge Graph Vocabulary for DQ Evaluation of Electronic Polling System

This example extends the initial vocabulary even further. We introduce new “poll”, “user”, and “decision-maker” logical entities. The “Decision-maker” entity represents an official who is interested in a poll results. “User” and “Poll” entities have “DQ” entity and “relevance” entities associated with them (more details in the next subsection). “Poll” entity may be created by a user or a decision-maker. We introduce “DQ for official” logical entity that represents the DQ value that is interesting for a decision-maker. In addition, we augment vocabulary with such relations as “created” and “voted”. “Created” relation (property) points to a “user” or “decision-maker” entities that created the “poll” entity. “Voted” relation connects “user” entities with “poll” entities and points to users that voted for the particular poll.

5.9.2 Calculus Implementation for DQ Evaluation of Electronic Polling System

To each poll or survey, relevant weights are assigned. As seen in the following example, these are w1, w2, etc. weights. The DQ evaluation is based on device security, data, and the relevance of the data to the user. In this example, linear equations are used to reduce the resources used for the computation of various metrics.

Let us consider a case of Helen Smith being a Democrat senator from Tennessee, TX. Her area of interest is in issues related to taxes, immigration policy, and Medicare. DQ is a function of a data object, DR, and device security. Relevance can be calculated using two aspects: the profile relevance and the poll (data object) relevance. These values depend on how relevant the profile and data are to the data consumer.

Table 5.11 presents profiles of five users with all their details. The profile credibility score in the table can be calculated using the following parameters:

- How frequently the application is used.
- How many users have used the same Device id.
- How many polls a user took.
- How many polls were posted.
- Posted polls credibility score = f(Total posted polls vote count, Presence of any anti-social agenda in the posted polls)
- If the user is blacklisted.

For this example, we consider five Polls.

The DQ in this example is calculated as follows:

$$DQ = \frac{\text{Rel(Profile data)} + \text{Rel(Polls data)} + (\text{Profile DQ} * \text{Poll DQ})/2}{3}$$

The DQ value is always between of 0 and 1

Further, we present the calculus of each equation component:

1. Rel(Profile data)

If Helen wants to target the issues faced by Democrats in her constituency, aged in between 30 - 45, who are white males Democrats, all the relevance value associated with the profile data will be derived using the following equation:

Relevance of Profile Data = w1*(Same region or Not) + w2*Gender matching + w3*Age matching + w4*Political affiliation matching + w5*(Matching interests points)

Table 5.11: Table for Profiles and there associated values

	User1	**User2**	**User3**	**User4**	**User5**
Name	Alex	Ram	Chan	Usain	Jane
Age	32	30	54	30	34
Gender	M	M	M	M	F
Race	White	Asian	Asian	Black	White
From	Tennessee, TX	Rochester, NY	San Francisco, CA	Washington DC	Tennessee, TX
Political	Democrat	Democrat	Republican	Republican	Democrat
Education Level	High School	Graduate	Under Graduate	No School	Graduate
Political Views	Moderate	Moderate	Conservative	Very Conservative	Liberal
Areas of Interests	1.City Planning 2.Education 3.Health-Care	1.Immigration 2.Jobs 3.Sports	1.City Planning 2.Education 3.Health Care	1.Sports 2.Wild Life 3.National Security	1.City Planning 2.Education 3.Sports Care
Security Level (Phone)	1	0.8	0.6	0.2	0.9
Polls Taken	121	82	43	17	110
Polls Created	24	7	9	2	12
Number of profiles with the associated Device Id	1	1	2	4	1
Black-listed	0	0	0	0	0
Profile Credibility Score	1	1	0.33	0.66	1
Profile Relatability Score	1	0.3	0.25	0.35	0.849
Completeness	1	1	1	1	1

Table 5.12: Table for profile relevance values

	Alex	**Ram**	**Chan**	**Usain**	**Jane**
Profile Relevance Score	1	0.3	0.25	0.35	0.849

Table 5.13: Table for polls relevance values

	Poll 1	**Poll 2**	**Poll 3**	**Poll 4**	**Poll 5**
Poll Relevance Score	0.39	0.217	0.1226	0.3817	0.6801

Weight for each of the data consumers interests = 1 / Interest count

So, in Helen's case, it is 0.33 for each interest.

We can apply this equation to all the user profiles to assign them a relative profile relevance score. The issues posted or answered by these user profiles, would have more value to the data consumer, which is Helen Smith in this scenario.

Table 5.12 presents profile relevance values for w1 = 0.5, w2 = 0.1, w3 = 0.2, w4 =-0.05 and w5 = 0.15.

2.Rel(Polls data)

Let us assume that a poll is completed, and Helen Smith needs the poll results, which are the most relevant to her. In this case, the poll data relevance is calculated using the same aforementioned relevance formula but with some different weights values. Relevance of Poll Data =(w1*(Number of people from same region) + w2*(Number of people from a particular gender) + w3* (Number of people in the required age range) + w4*(Number of people with matching political affiliation) + w5*(Matching Tag)) / Vote Count We apply this equation to all polls to assign them a relative poll relevance score. Let us suppose w1 = 0.5, w2 = 0.1, w3 = 0.2, w4 =-0.05, and w5 = 0.15 then the Table 5.13 for poll relevance values shows us all the relevance for all the polls.

3. Poll Data Quality

This value depends on the following constraints:

- Vote Count
  - total vote count ¿ 5000 = +1
  - total vote count between 2000 - 5000 = +0.5
  - total vote count between 0 - 2000 = +0.25
  - Otherwise = +0
- Tagged or Not
- Trending or not
- Freshness
- Question Count
- Not Controversial (Can be calculated using sentiment analysis)

By adding the points for all the above constraints, we obtain the normalized poll data quality by dividing it with the number of constraints. Table 5.12 presents the calculated values for profile relevance values.

4. Trust factor for the user profile.

The Trust level of a user profile will have a significant impact on the DQ. This trust for the user profile can be calculated using the following steps 5.27:

- Suppose the user information form has ten questions to be answered, then depending on the completeness of this form, the user profile completeness score can be calibrated as 0.1 for each question completed. For example, if the user has completed 7 out of 10 profile related questions, then this score component will be 0.7.
- The number of polls posted by the user and their quality will also have an impact on the user's trust score. The quality of the posted polls can be calculated by the aforementioned method, and the quality of this poll can be averaged to get a value in a range from 0 to 1.

Table 5.14: Table for DQ values of all the polls for Helen Smith

	Poll 1	**Poll 2**	**Poll 3**	**Poll 4**	**Poll 5**
Data Quality	0.7429	0.4128	0.3301	0.5578	0.6574

- The number of polls taken by the user will also be considered. The score would be:
  - 0.5 if the number of polls taken is between 1 to 25.
  - 0.75 if the number of polls taken is between 26 to 100.
  - 1.0 the number of polls taken is greater than 100.
- The duration for which the account is active will also have an impact on the trust level of the user. The user is termed as active if he/she is either posting or taking polls at regular intervals. The longer the user is active, the better. The dormant accounts will be loose points in this aspect.
- The number of accounts associated with the device being used by this user.
  - if only one account is associated with this users device then 1.0 point
  - if 2 - 4 accounts are associated with this users device then 0.5 point
  - otherwise 0
- The security of the device will also be used towards the trust factor of a profile.

The scores generated from the above-mentioned points are averaged out to get a profile trust score in the range from 0 to 1. All the scores above are nullified if the “Integro-System” identifies the account as ”Sybil”. To get the normalized profile data quality, we add all constraints and divide them by the number of them.

Now, let us reconsider the scenario where ”Helen Smith wants to target the issues faced by Democrats in her constituency, aged in between 30 - 45, who are white males Democrats” and calculate the data value for each issue:

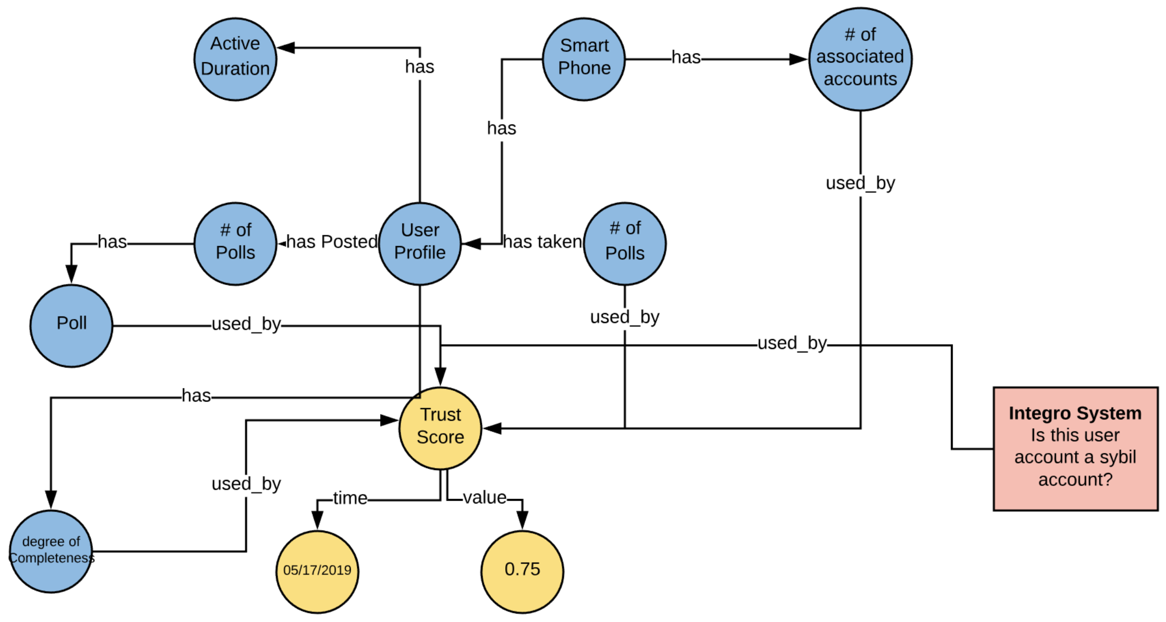


Figure 5.27: Knowledge graph employment in the calculation of the trust score in the polling example.

Relevance of Poll 5 = 0.6801

Relevance of Jane's Profile = 0.849

$$\begin{aligned} DQ &= \frac{\frac{\text{DQ of Poll 5}*\text{Trust score for Jane}}{2} + \text{Relevance of Poll 5} + \text{Relevance of Jane's Profile}}{3} = \\ &= \frac{\frac{0.9167*0.9667}{2} + 0.6801 + 0.849}{3} = 0.6574 \end{aligned} \tag{5.3}$$

Similarly, we calculate the DQ for all other polls. Values can be found in the Table 5.14.

The list of issues from top priority issues to the least priority for Helen Smith in this scenario would be Poll 1, Poll 2, Poll 3, Poll 4, Poll 5.

Security and trust play a significant role in any crowd-sourcing platform. Let us see how security and trust features can be built in to validate and improve the quality of data generated using this crowd-sourced application. The security and trust metrics in these kinds of applications can be divided into two parts: the security and trust level evaluation of the device, which is used to collect the data, and the evaluation of the profile that is generating this data. In this application, the security level is calculated based on the implementation presented in section 5.1. The trust level of the profile is determined as follows. We check how frequently and for how long the application is being used. Longer the time for which the user is active, the higher the score. The frequency of the usage also matters as the user using the application from the same device at a normal usage frequency is more trustworthy. We also check if the device id associated with the account is being used to create some other accounts as well and based on these number of devices associated with that account. In addition, we detect fake accounts. For blacklisting an account, the following criteria are used. The profile is blacklisted if it identified as a "sybil account" using the Integro-System [16]. Similarly, the content posted by the user is checked if it contains any malicious or harmful text; if the frequency at which such polls originate from a user profile is high, then the account is blacklisted.

5.9.3 Knowledge Graph Employment in DQ Evaluation of Electronic Polling System

Figure 5.26 presents a sub-graph of a knowledge graph that is used in this example. Each time a decision-maker requests some data and the DQ associated with it, a new entity in the knowledge graph is created (blue and pink color). Figure 5.26 presents two data polls and two users from the table 5.11. This sub-graph reflects DQ evaluation for two polls in relation to Helen Smith's 'decision-maker" entity. One poll was created by a "user 1", and Helen herself created the second poll. The graph shows that both users have voted for both polls, which is reflected with "voted" property in both users. "DQ iCitizen" entity indicates at what other entities and their relations a "DQ for official" is based on, and what calculus has to be used. One can see in the graph that after Helen requested information about two polls, two "DQ for official" entities were created. Both entities have a "used" relation that points on Helen's entity and "DQ iCitizen" entity.

5.10 Developed Tools for Data Quality and Security Evaluation

This section presents tools that have been developed in this project and are dedicated to the DQ and security evaluation as well as for the data collection.

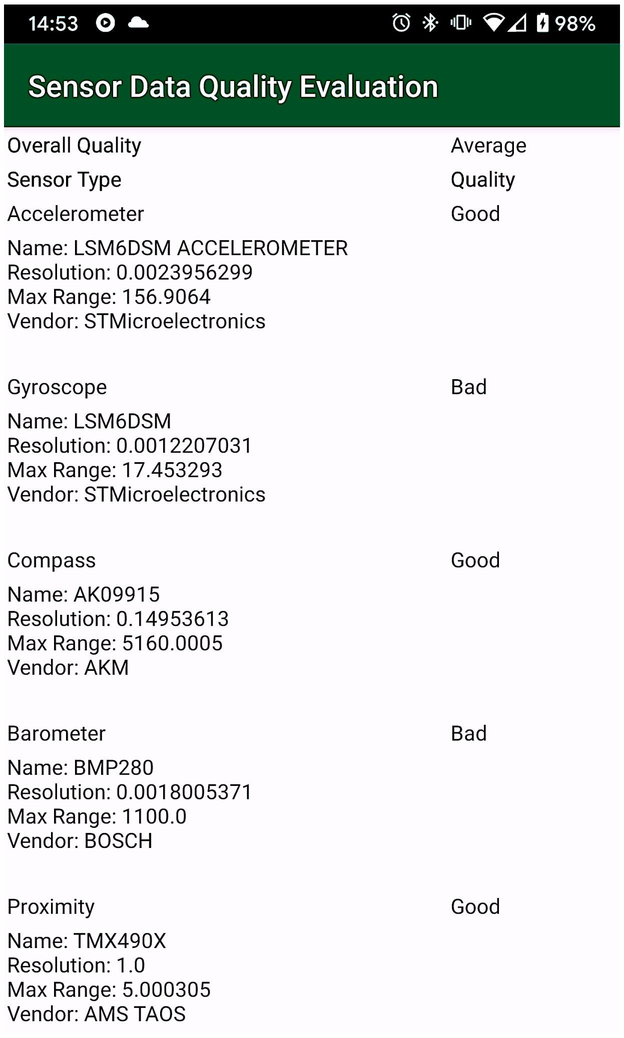
5.10.1 Sensor Quality Evaluation Tool

This tool implements the sensors' quality evaluation knowledge base, presented in section 3.3. To assist mobile device users in the quality analysis of sensors that are built into their devices and to provide further recommendations, we developed the specialized application [33]. It is based on the quality calculation procedures described in the previous section and can be executed on the Android OS devices. This application does not require any specific Android permissions and employs standard Android API to acquire and to present to the user sensor types that are built-in a smartphone as well as sensor characteristics. While basic sensor characteristics, such as the resolution or sensing range, may be directly accessed through the Android API, other specific information such as noise density or non-linearity is acquired from the developed database.

The screenshot of the application's user interface is presented in figure 5.28. The developed Android OS application reveals to the users those sensors that are built-in-to smartphones, their basic characteristics, and overall quality score.

5.10.2 Smartphone Initial Security Evaluation Tool

This application evaluates the overall security level of your smartphone based on the system's parameters. It can consider parameters separately or perform a complex evaluation where all the parameters are taken into account. As a result, you have a score that represents the overall security level of your smartphone. In addition, this application provides detailed recommendations on how to improve particular security aspects as well as general advice on increasing an overall


14:53 98%

Sensor Data Quality Evaluation

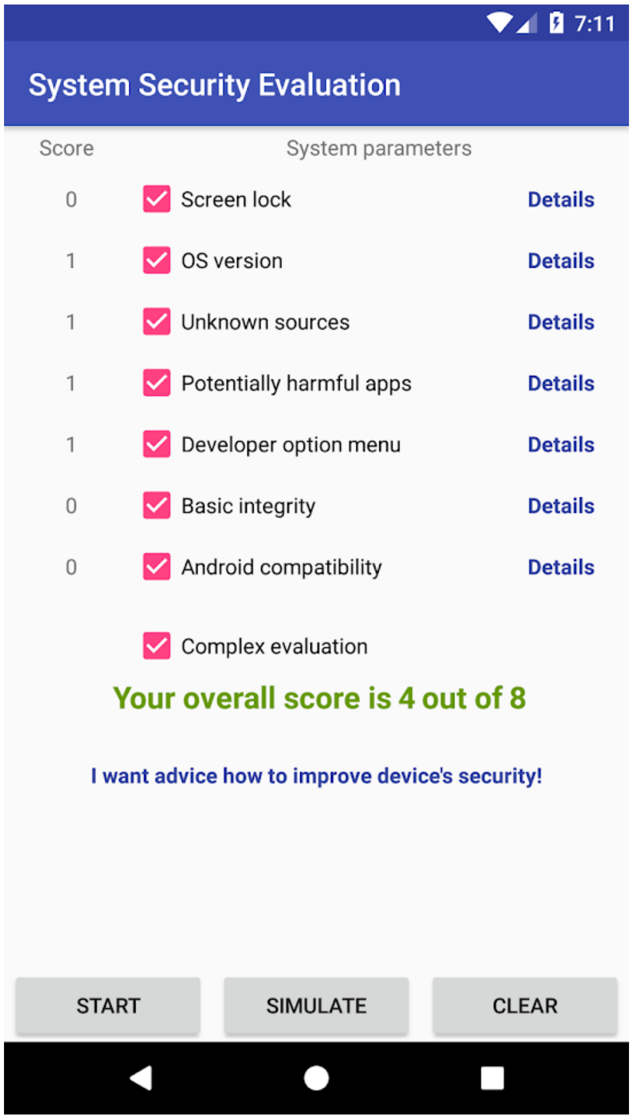
Overall Quality	Average
Sensor Type	Quality
Accelerometer	Good
Name: LSM6DSM ACCELEROMETER Resolution: 0.0023956299 Max Range: 156.9064 Vendor: STMicroelectronics	
Gyroscope	Bad
Name: LSM6DSM Resolution: 0.0012207031 Max Range: 17.453293 Vendor: STMicroelectronics	
Compass	Good
Name: AK09915 Resolution: 0.14953613 Max Range: 5160.0005 Vendor: AKM	
Barometer	Bad
Name: BMP280 Resolution: 0.0018005371 Max Range: 1100.0 Vendor: BOSCH	
Proximity	Good
Name: TMX490X Resolution: 1.0 Max Range: 5.000305 Vendor: AMS TAOS	

Figure 5.28: The screenshot of the Android OS application for sensor quality assessment

security level. The application also provides a possibility to simulate various operating conditions and system parameters values to demonstrate how they influence the overall security level. User interface of the application is presented in figure 5.29 and figure 5.30.

5.10.3 Data Collection Tool

This tool collects smartphone's data such as a device model, hardware description, onboard sensors, installed applications and their permissions, state of the developer options menu, device lock



Score	System parameters	
0	✓ Screen lock	Details
1	✓ OS version	Details
1	✓ Unknown sources	Details
1	✓ Potentially harmful apps	Details
1	✓ Developer option menu	Details
0	✓ Basic integrity	Details
0	✓ Android compatibility	Details

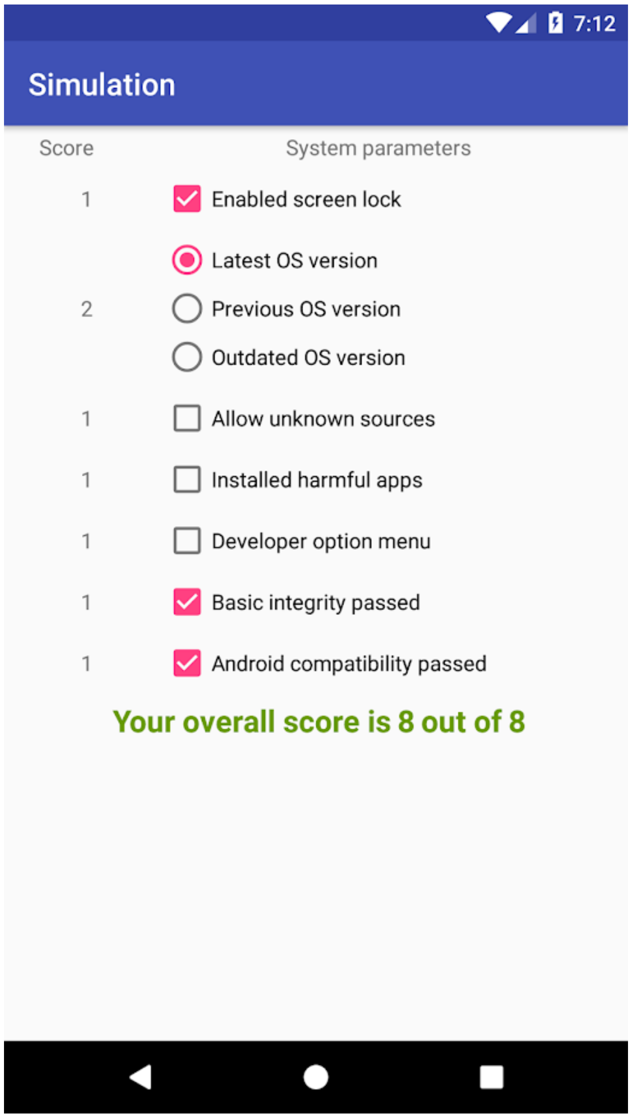
✓ Complex evaluation

Your overall score is 4 out of 8

I want advice how to improve device's security!

Figure 5.29: The screenshot of the Android OS application for the smartphone security evaluation. Main screen.

mechanism, basic integrity test, compatibility test, and presence of potentially harmful apps. This data is saved to a text file. No private information about the owner is recorded. It gives a user an option to prepare data for sending it to the specified email or remote folder. This tool may be used for developing rules for DQ and security evaluation as well for training various AI models. The interface of the tool is presented in figure 5.31.



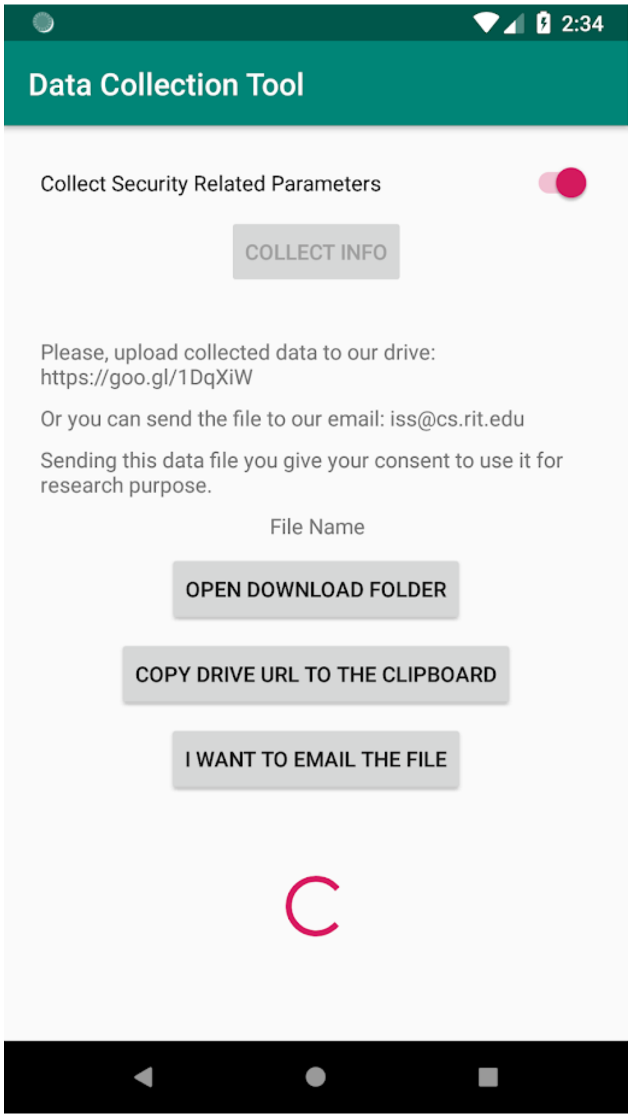
Score	System parameters
1	[✓] Enabled screen lock
2	(●) Latest OS version ( ) Previous OS version ( ) Outdated OS version
1	[ ] Allow unknown sources
1	[ ] Installed harmful apps
1	[ ] Developer option menu
1	[✓] Basic integrity passed
1	[✓] Android compatibility passed

Your overall score is 8 out of 8

Figure 5.30: The screenshot of the Android OS application for the smartphone security evaluation. Emulation screen.

5.10.4 Expert System Implementation on Android OS Smartphone with Neural Networks

This application implements fuzzy rules expert systems presented in section 5.5 through neural network models described in section 5.6. The application also utilizes the developed library presented in section 5.1, which is used for acquiring the security metrics of the smartphone. The interface of


2:34

Data Collection Tool

Collect Security Related Parameters

Please, upload collected data to our drive: https://goo.gl/1DqXiW

Or you can send the file to our email: iss@cs.rit.edu

Sending this data file you give your consent to use it for research purpose.

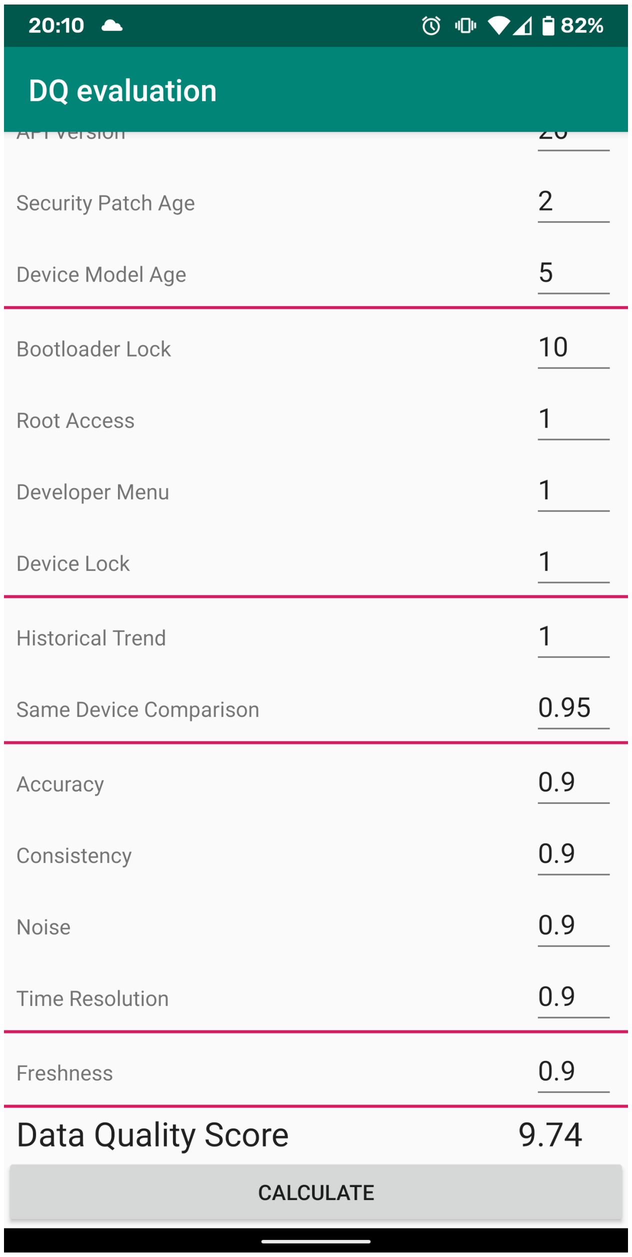
File Name

Figure 5.31: The screenshot of the Android OS application for the collecting data about security and sensor quality metrics of the Android OS smartphone.

the application is presented in figure 5.32.

5.11 Conclusions

In response to the research question posed in this chapter: ***"How to develop effective and efficient framework implementation on a mobile platform, such as Android OS-based***


20:10 82%

DQ evaluation

Security Patch Age	2
Device Model Age	5
Bootloader Lock	10
Root Access	1
Developer Menu	1
Device Lock	1
Historical Trend	1
Same Device Comparison	0.95
Accuracy	0.9
Consistency	0.9
Noise	0.9
Time Resolution	0.9
Freshness	0.9

Data Quality Score 9.74

Figure 5.32: The screenshot of the Android OS application that implements ES modules through NN models for the DQ and security evaluation.

smartphones?", *we achieved the following* ***major*** *research results, which are further described in details:*

- *We developed methods and tools for* ***security metrics acquisition*** *in the form of Android*

OS library;

- *We developed methods and tools for* ***data quality and security metrics integration*** *in the form of fuzzy rules expert system;*
- *We* ***developed a fuzzy rules expert systems implementation for resource-constraint mobile devices*** *through neural networks ;*
- *We* ***developed detectors of novel and sophisticated attacks on users' privacy*** *that can be used by untrained smartphone users on a modern Android smartphone without firmware modification.*
- *We* ***developed a set of various tools for data quality and security evaluation****.*

To validate the performance of the developed framework, we implemented and tested it on important real-life use cases. To integrate all metrics into the overall DQ value, we implemented developed calculi through expert systems based on fuzzy logic, neural networks, and algebraic functions. To incorporate security and privacy aspects into the DQ evaluation process, we employed the knowledge graph as a major development methodology. In addition, the knowledge graph allows an easy framework extension and adjustment by extending vocabulary with new entities and adding new calculus.

For the data privacy evaluation, we developed novel intelligent classifiers for real-time anomaly detection on Android OS-based smartphones. We tested the developed attack detectors on multiple diverse "colluded applications" attack cases. The novelty of the developed attack detectors is that they do not require a sensor platform firmware modification and employ system resources monitoring data collected and analyzed in real-time. The developed attack detectors employ various machine learning techniques and models such as feed-forward neural networks, simple recurrent neural networks, long short-term memory, and bidirectional long short-term memory.

In order to achieve better results in attack detection, models that utilize memory consumption and CPU frequency were developed. The developed attack detectors employ feed-forward neural networks and long short-term memory ML techniques. Each detector was tested on the data from selected four attack scenarios, and results were analyzed. The best results were achieved with LSTM based attack detectors.

Our analysis shows that reliable attack detection accuracy can be achieved using system data collected over one second with even higher accuracy with data collected over two seconds. While the LSTM model has a higher performance than the FFNN model, it also has much higher complexity. On the other hand, the LSTM model achieves this high accuracy using only CPU cores clock speed, while the FFNN model needs memory usage as well. The developed attack detectors play an important role in data privacy evaluation.

To reduce a false positive rate of the developed attack detectors, we developed and implemented the permission analysis of the installed application. This additional step allows excluding the applications that do not benefit from the attack execution.

We validated the developed framework on the set of important real-life tasks, such as framework employment in the crowd-sensing application, road maintenance based on the user reports, and autonomous self-driving vehicles. We demonstrated the ways of data security and privacy aspects incorporation into the overall DQ evaluation processes.

We implemented the DQ evaluation calculus through the expert systems with fuzzy rules. We developed a novel and effective implementation of the developed expert system on the resource constraint sensor-platforms that are based on the Android OS smartphones. The presented solution employs neural network models that approximate the input/output hypersurfaces of the developed ES modules. The developed NN models may take advantage of the modern NN dedicated chips.

We rendered how to bind together the metrics integration calculi and the application context by employing the knowledge graph. We demonstrated how to extend the framework application area by augmenting the knowledge graph vocabulary.

To acquire initial data about a device for the security evaluation component, we produced a dedicated Android OS library that can be installed on any modern Android OS-based device. We also developed the Android OS applications that implement concepts and methods developed in this research:

- An Android application that analyzes the quality of the sensors that are available on the device and provides this information to the device's users. The developed tool enables the education of sensor users and facilitates the sensor's secure and safe applications.
- System Security Evaluation app based on a security evaluation library - https:// play.google.com /store /apps/details?id= com.igorkh.trustcheck. securitycheck
- System Security Evaluation app based on a security evaluation library with the cloud support - https://play.google.com /store/apps/ details?id= com.igorkh.trustcheck. securitycheckcloud
- Detector of Unverified Apps - https://play.google.com/ store/apps/details?id= dataquality-lab.rit. ver_app_finder
- Implementation of the expert system on resource-constraint devices with neural network models. Source code of the NN calculus for security evaluation component - https://drive.google.com/ file/d/ 1jwii74qSpOPMHb4EuEwTtydSiF8Bf6mW/ view? usp=sharing
- Data collection tool for acquisition data quality and security metrics of Android smartphone for further analysis - https:// play.google.com/store/apps /details?id= com.dataqualitylab. collectinfo.collectinfo

Chapter 6

Conclusions

In the modern world of the constantly growing number of applications and systems that are fueled by data, data quality and security are essential concepts in data management and decision-making that are often separated in practice. Spreading of the citizens-science and crowdsensing elevated the importance of DQ evaluation even higher. Bad quality of the data leads to wrong conclusions and poor decisions that might produce bad results ranging from resource waste the loss of lives.

Data scientists and system developers have already produced a wide variety of DQ related applications, with most of them aiming at improving one or a few specific DQ aspects with no consideration of many others. In addition, the security components of data quality are often underestimated or even ignored. Nowadays, data security and privacy are extraordinarily important, especially considering the amount of data collected through mobile phones. We believe that data security as one of the significant and essential components of the DQ evaluation process.

We developed a systematic approach to DQ evaluation system design and implementation that takes into account multiple diverse metrics and enables an effortless adjustment to various areas. We implemented this approach in a framework designed for DQ and security evaluation in resource constraint sensor platforms such as mobile devices. The framework includes models and methods for DQ and security metrics assignment, selection, composition, their integration calculus, and the knowledge graph that bonds them together and allows the further development through knowledge accumulation. As an implementation platform, we chose an Android OS smartphone. However, our framework design is not limited to mobile phones only and can be further adjusted to various data sources and sensor platforms' operations.

We analyzed the existing methods of DQ evaluation and revealed that current approaches to DQ estimation are often limited to a very narrow application field. Consequently, those DQ evaluation methods use a few application-specific metrics. Therefore, it is very hard to modify existing DQ evaluation methods for employment in new application fields.

We **classified existing DQ metrics design approaches** into three categories: empirical, theoretical, and intuitive. In our DQ and security evaluation framework, we employ the theoretical approach to metrics design and selection due to it considers ways on how data may become imperfect during the data production or acquisition processes and results in a broad spectrum of various DQ metrics.

We studied the variety of data quality and security metrics and developed a **novel multilevel hierarchical knowledge** structure that facilitates relatively effortless metrics adoption from new application domains and extends framework employment in solving broad task spectrum.

We also **reviewed and classified existing DQ concept types** into two major categories: intrinsic and contextual, where the intrinsic approach to DQ definition isolates DQ from the application context, and the contextual DQ considers users' needs. Our analysis of the existing approaches to DQ definition directed us to use intrinsic first-level DQ metrics and consider application context on a stage of metrics integration.

Our analysis of existing approaches to DQ evaluation **revealed insufficient attention to data security and privacy**. This lack of security and privacy consideration in the overall DQ evaluation pipeline led us to investigate existing methods of security and privacy aspects evaluation and ways of their inclusion to the overall DQ evaluation processes. We chose to use diverse security and privacy-related metrics that are accessible without sensor-platform modification. We **investigated privacy threats** and **classified existing approaches to the privacy protection**. We also considered novel and sophisticated threats to data privacy, such as the "colluded applications" attack. We **analyzed existing methods of this attack detection** and **revealed the absence of the attack detection methods that can be used by untrained users**.

We **developed methods to integrate the security and privacy aspects into the overall DQ evaluation processes**. To integrate all security and DQ metrics into the overall DQ value, we developed and implemented various calculi such as an expert system based on fuzzy logic, neural networks, and weighted sum. The metrics integration hierarchy follows up the metrics hierarchy and allows us to adjust DQ evaluation with a little framework modification. We investigated **the knowledge graph concept** and developed the methodology of its employment as the novel implementation platform of our DQ evaluation framework. The knowledge graph allowed us to integrate security and privacy aspects into the conventional DQ evaluation techniques and to consider the application context. The knowledge graph component of the framework facilitates an easy framework extension and adjustment by extending vocabulary with new entities and adding new calculus. The knowledge graph also allows considering the application context on a stage of metrics integration, which aids in adopting of more application domains. We formulated **an initial knowledge graph vocabulary**, which facilitates the framework usage in crowd-sensing tasks for evaluation sensor originated data. We investigated and analyzed sensors incorporated in the modern mobile devices, produced the comprehensive knowledge base and **made it available for community use**. The presented framework is a "skeleton", which later can be easily expanded and modified for various applications.

We investigated modern Android OS based smartphones and developed **novel multilayered hierarchical security-related metrics structure** that includes three major branches. We **developed the initial calculus for metrics integration** from each of the security branches. On our way to integrating security and privacy aspects into conventional DQ evaluation, we **developed**

novel security and privacy evaluation methods. We considered security and privacy-related characteristics of the Android OS-based smartphone and analyzed ways of violating security and privacy data aspects. Our analysis resulted in the initial set of metrics that are used for security evaluation. We **investigated current threats** to user privacy and **revealed the lack of tools for detection novel and sophisticated attacks**. On the way of data privacy evaluation, we conducted an empirical study of the novel and sophisticated attacks on data privacy on the example of "colluded application" attacks. We **formalized "colluded applications" attack and developed its model**. We **conducted an empirical study of this attack**, and **created a dataset** that can be used for developing machine learning-based attack detectors. We **investigated and analyzed various artificial intelligence techniques for the detection attacks** on users' privacy. We **developed detectors of novel and sophisticated attacks on users' privacy** that can be used by untrained smartphone users on a modern Android smartphone without firmware modification.

We implemented the proposed methods and techniques on the resource-constraint sensor platforms, such as Android OS-based smartphones. We **developed a systematic security evaluation pipeline** that considers smartphones' resource-constraint nature and **includes a comprehensive set of security and privacy evaluation approaches**. To acquire initial intrinsic DQ metrics, we **developed an application that collects DQ and security metrics** from most modern Android OS-based smartphones. We **developed a dedicated Android OS library** that can be installed on any modern Android OS-based device, collect security-related metrics, ad integrate them into the security score. We **developed a fuzzy rules expert systems implementation for resource-constraint mobile devices** with an application of neural network models. Developed NN models approximate input/output hypersurfaces of the developed ES modules. NN models are an effective and efficient approach to ES implementation on modern mobile phones that takes advantage of the modern NN chips that are built into modern smartphones.

We **validated the developed framework on a set of diverse important real-life examples**. We demonstrated the knowledge graph role in DQ and security evaluation and how it allows a framework augmentation with new applications on several use cases. These use cases present DQ evaluation for sensor originated data in the various application domain. We demonstrate the comparison of the DQ evaluation with and without the security component and render the importance of the security component in overall DQ evaluation. We presented how the DQ evaluation with a specific calculus is implemented through the knowledge graph. The presented use cases also render how the knowledge graph vocabulary can be extended and how to include a new calculus implementation.

The **developed Android OS applications** that implement concepts and methods presented in this research are made available for public use:

- An Android application that analyzes the quality of the sensors that are available on the device and provides this information to the device's users. The developed tool enables the education of sensor users and facilitates the sensor's secure and safe applications.
- System Security Evaluation app based on a security evaluation library - https:// play.google.com /store /apps/details?id= com.igorkh.trustcheck. securitycheck
- System Security Evaluation app based on a security evaluation library with the cloud support - https://play.google.com /store/apps/ details?id= com.igorkh.trustcheck. securitycheckcloud
- Detector of Unverified Apps - https://play.google.com/ store/apps/details?id= dataquality-lab.rit. ver_app_finder
- Implmentation of the expert system on resource-constraint devices throu neural network models. Source code of the NN calculus for security evaluation component - https://drive.google.com/ file/d/ 1jwii74qSpOPMHb4EuEwTtydSiF8Bf6mW/ view? usp=sharing
- Data collection tool for acquisition data quality and security metrics of Android smartphone for further analysis - https:// play.google.com/store/apps /details?id= com.dataqualitylab. collectinfo.collectinfo

The integration of conventional DQ metrics and security properties into the single framework allowed us to produce the overall DQ evaluation pipeline that covers diverse data aspects. The employment of the knowledge graph facilitates the framework adaptation to a broad spectrum of applications. The knowledge graph allowed us to integrate data collection and DQ evaluation into the complete data management pipeline. It enables knowledge accumulation and, at the same time, acts as data storage. In addition, the knowledge graph application allows the employment of versatile DQ calculi. The framework could be employed in combination with other products in various applications, for example, for data classification in data access control or security enhancement with data object tracing.

Bibliography

[1] Davis Aaron, Lazo Luz, and Schemm Paul. Additional software problem detected in boeing 737 max flight control system, officials say, 2019. www.washingtonpost.com/world/africa/ethiopia-says-pilots-performed-boeings-recommendations-to-stop-doomed-aircraft-from-diving-urges-review-of-737-max-flight-control-system/2019/04/04/3a125942-4fec-11e9-bdb7-44f948cc0605_story.html. Accessed: April 5, 2019.

[2] Acid project, 2019. http://acidproject.org.uk/. Accessed: January 17, 2019.

[3] Hussein Alrubaye, Mohamed Wiem Mkaouer, Igor Khokhlov, Leon Reznik, Ali Ouni, and Jason Mcgoff. Learning to recommend third-party library migration opportunities at the api level. *Applied Soft Computing*, 90:106140, 2020.

[4] Amazon.com, 2019. https://www.amazon.com/. Accessed: January 19, 2019.

[5] andr7e. Device info hw database, 2019. http://www.deviceinfohw.ru/devices/index.php. Accessed: June 15, 2019.

[6] Safetynet, 6 2018. https://developer.android.com/training/safetynet/index.html. Accessed: March 25, 2020.

[7] Inc. Apogee Instruments. Applications and uses of uv sensors, 2019. https://www.apogeeinstruments.com/applications-and-uses-of-uv-sensors/. Accessed: June 15, 2019.

[8] App store - apple, 2019. https://www.apple.com/ios/app-store/. Accessed: January 19, 2019.

[9] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[10] Irina Mariuca Asavoae, Jorge Blasco, Thomas M Chen, Harsha Kumara Kalutarage, Igor Muttik, Hoang Nga Nguyen, Markus Roggenbach, and Siraj Ahmed Shaikh. Towards automated android app collusion detection. *arXiv preprint arXiv:1603.02308*, 2016.

[11] Irina Măriuca Asăvoae, Hoang Nga Nguyen, and Markus Roggenbach. Software model checking for mobile security–collusion detection in K. In *International Symposium on Model Checking Software*, pages 3–25. Springer, 2018.

[12] Amotz Bar-Noy, Greg Cirincione, Ramesh Govindan, S Krishnamurthy, TF LaPorta, Prasant Mohapatra, M Neely, and Aylin Yener. Quality-of-information aware networking for tactical military networks. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 2–7. IEEE, 2011.

[13] Ataul Bari, Shamsul Wazed, Arunita Jaekel, and Subir Bandyopadhyay. A genetic algorithm based approach for energy efficient routing in two-tiered sensor networks. *Ad Hoc Networks*, 7(4):665–676, 2009.

[14] Chatschik Bisdikian, Lance M Kaplan, and Mani B Srivastava. On the quality and value of information in sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 9(4):48, 2013.

[15] Jorge Blasco, Thomas M Chen, Igor Muttik, and Markus Roggenbach. Detection of app collusion potential using logic programming. *Journal of Network and Computer Applications*, 105:88–104, 2018.

[16] Yazan Boshmaf, Dionysios Logothetis, Georgos Siganos, Jorge Leria, Jose Lorenzo, Matei Ripeanu, Konstantin Beznosov, and Hassan Halawa. Integro: Leveraging victim prediction for robust fake account detection in large scale osns. *Computers & Security*, 61:142–168, 2016.

[17] Amiangshu Bosu, Fang Liu, Danfeng Daphne Yao, and Gang Wang. Collusive data leak and more: Large-scale threat analysis of inter-app communications. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 71–85. ACM, 2017.

[18] Amiangshu Bosu, Fang Liu, Danfeng Daphne Yao, and Gang Wang. Android collusive data leaks with flow-sensitive dialdroid dataset, 2018. http://people.cs.vt.edu/gangwang/DIALDroid-poster.pdf. Accessed: January 17, 2019.

[19] Sven Bugiel, Lucas Davi, Alexandra Dmitrienko, Thomas Fischer, and Ahmad-Reza Sadeghi. Xmandroid: A new android evolution to mitigate privilege escalation attacks. *Technische Universität Darmstadt, Technical Report TR-2011-04*, 2011.

[20] Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani. Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 15–26. ACM, 2011.

[21] Li Cai and Yangyong Zhu. The challenges of data quality and data quality assessment in the big data era. *Data Science Journal*, 14, 2015.

[22] Derya Cansever. Value of information. In *Military Communications Conference, MILCOM 2013-2013 IEEE*, pages 1105–1108. IEEE, 2013.

[23] Deyan Chen and Hong Zhao. Data security and privacy protection issues in cloud computing. In *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*, volume 1, pages 647–651. IEEE, 2012.

[24] Zhenguo Chen, Liqin Tian, and Chuang Lin. Trust model of wireless sensor networks and its application in data fusion. *Sensors*, 17(4):703, 2017.

[25] Sung-Bae Cho and Sang-Jun Han. Two sophisticated techniques to improve hmm-based intrusion detection systems. In *International Workshop on Recent Advances in Intrusion Detection*, pages 207–219. Springer, 2003.

[26] Sergey Chuprov, Egor Marinenkov, Ilya Viksnin, Leon Reznik, and Igor Khokhlov. Image processing in autonomous vehicle model positioning and movement control. In *2020 IEEE World Forum on Internet of Things*. IEEE, 2020. Accepted for publication.

[27] Sergey Chuprov, Ilya Viksnin, Iuliia Kim, Leon Reznik, and Igor Khokhlov. Reputation and trust models with data quality metrics for improving autonomous vehicles traffic security and safety. In *IEEE/NDIA/INCOSE Systems Security Symposium 2020, Crystal City, Virginia, USA*. IEEE, 2020. Accepted for publication.

[28] Catalin Cimpanu. 100 million android users may have a backdoor on their device thanks to the baidu sdk, 2015. http://news.softpedia.com/news/100-million-android-users-may-have-a-backdoor-on-their-device-thanks-to-the-baidu-sdk-495673.shtml. Accessed: January 17, 2019.

[29] Catalin Cimpanu. 21 android apps spotted using app collusion attacks, 2016. http://news.softpedia.com/news/21-android-apps-spotted-using-app-collusion-attacks-505252.shtml. Accessed: January 17, 2019.

[30] Igino Corona, Davide Ariu, and Giorgio Giacinto. Hmm-web: A framework for the detection of attacks against web applications. In *2009 IEEE International Conference on Communications*, pages 1–6. IEEE, 2009.

[31] Anupam Das. Sensor data collection, 2019. https://www.cs.cmu.edu/ anupamd/SensorDataCollection.html. Accessed: November 9, 2019.

[32] Data quality lab - sensor dataset, 2019. https://drive.google.com/drive/folders/ 1vnwpp3hzmbOVf9exXVYLtN38UsT7KEIU?usp=sharing. Accessed: June 15, 2019.

[33] DataQualityLab. Sensor quality assessment - apps on google play, 2019. https://play.google.com/store/apps/details?id=com.dataqualitylab.sensorquality. Accessed: December 9, 2019.

[34] Lucas Davi, Alexandra Dmitrienko, Ahmad-Reza Sadeghi, and Marcel Winandy. Privilege escalation attacks on android. In *international conference on Information security*, pages 346–360. Springer, 2010.

[35] Seta Davidian. How to select a humidity sensor, 2019. http://blog.servoflo.com/how-to-select-a-humidity-sensor. Accessed: June 15, 2019.

[36] Elaine Davies and James Rodger. These 145 downloads from google play store have hidden malware, 2018. https://www.coventrytelegraph.net/news/uk-world-news/google-playstore-apps-hidden-malware-15012679. Accessed: October 18, 2018.

[37] Dbpedia, 2019. https://wiki.dbpedia.org. Accessed: March 13, 2019.

[38] Devicespecifications - mobile device specifications, comparisons, news, user reviews and ratings, 2019. https://www.devicespecifications.com/en. Accessed: June 15, 2019.

[39] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):5, 2014.

[40] International Organization for Standardization. ISO/IEC 25012:2008 - Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Data quality model, 2008. https://www.iso.org/standard/35736.html. Accessed on January 08, 2019.

[41] International Organization for Standardization. ISO/TS 8000-1:2011 - Data quality – Part 1: Overview, 2011. https://www.iso.org/standard/50798.html. Accessed on January 08, 2019.

[42] Tech news, latest technology, mobiles, laptops – ndtv gadgets 360, 2019. https://gadgets.ndtv.com/. Accessed: June 15, 2019.

[43] Technology news, latest & popular gadgets reviews, specifications, prices, mobile comparison, technology videos & photos — gadgets now, 2019. https://www.gadgetsnow.com/. Accessed: June 15, 2019.

[44] Raghu K Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, 2011.

[45] Antonio Ghezzi, Donata Gabelloni, Antonella Martini, and Angelo Natalicchio. Crowdsourcing: a review and suggestions for future research. *International Journal of Management Reviews*, 20(2):343–363, 2018.

[46] Dan Goodin. Google play malware used phones' motion sensors to conceal itself, 2019. https://arstechnica.com/information-technology/2019/01/google-play-malware-used-phones-motion-sensors-to-conceal-itself/. Accessed: January 18, 2019.

[47] Introducing the knowledge graph: things, not strings, May 2012. https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html. Accessed: March 13, 2019.

[48] Pixel visual core: image processing and machine learning on pixel 2, 2018. https://www.blog.google/products/pixel/pixel-visual-core-image-processing-and-machine-learning-pixel-2/. Accessed: December 11, 2018.

[49] Google play, 2019. https://play.google.com. Accessed: January 19, 2019.

[50] Shielding you from potentially harmful applications, 2019. https://www.blog.google/technology/safety-security/shielding-you-potentially-harmful-applications/. Accessed: March 26, 2019.

[51] Nicolas Granger and Mounîm A el Yacoubi. Comparing hybrid nn-hmm and rnn for temporal modeling in gesture recognition. In *International Conference on Neural Information Processing*, pages 147–156. Springer, 2017.

[52] Gsmarena.com - mobile phone reviews, news, specifications and more..., 2019. https://www.gsmarena.com/. Accessed: June 15, 2019.

[53] Roee Hay, Omer Tripp, and Marco Pistoia. Dynamic detection of inter-application communication vulnerabilities in android. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis*, pages 118–128. ACM, 2015.

[54] Frederik Hermans, Norman Dziengel, and Jochen Schiller. Quality estimation based data fusion in wireless sensor networks. In *Mobile Adhoc and Sensor Systems, 2009. MASS'09. IEEE 6th International Conference on*, pages 1068–1070. IEEE, 2009.

[55] Katy Hilgenberg. Sensor data collection framework, 2019. https://sourceforge.net/projects/sdcf/files/ProjectReport.pdf/download. Accessed: September 9, 2019.

[56] Xue Li Hu, Lian Cheng Zhang, and Zhen Xing Wang. An adaptive smartphone anomaly detection model based on data mining. *EURASIP Journal on Wireless Communications and Networking*, 2018(1):148, 2018.

[57] Anne Immonen, Pekka Pääkkönen, and Eila Ovaska. Evaluating the quality of social media data in big data architecture. *IEEE Access*, 3:2028–2043, 2015.

[58] Audun Jøsang and Svein J Knapskog. A metric for trusted systems. In *In Proceedings of the 21st National Security Conference. NSA*. Citeseer, 1998.

[59] Stephen H Kan. *Metrics and models in software quality engineering.* Addison-Wesley Longman Publishing Co., Inc., 2002.

[60] Paul A Karger and Roger R Schell. Multics security evaluation: Vulnerability analysis. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, pages 127–146. IEEE, 2002.

[61] Igor Khokhlov, , Leon Reznik, and Sergey Lyshevski. Adaptive data fusion in inertial sensors and data quality of sensor networks. In *2020 IEEE 40th International Conference on Electronics and Nanotechnology (ELNANO)*. IEEE, 2020. DOI: 10.1109/ELNANO50318.2020.

[62] Igor Khokhlov, Chinmay Jain, Ben Miller-Jacobson, Andrew Heyman, Leonid Reznik, and Robert St Jacques. Meetci: A computational intelligence software design automation framework. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2018.

[63] Igor Khokhlov, Qiaoran Li, and Leon Reznik. D.I.F.E.N.S.E.: Distributed intelligent framework for expendable android security evaluation. In *The 14th Annual Symposium on Information Assurance (ASIA '19)*, page 18–27, 2019.

[64] Igor Khokhlov, Ligade Ninad, and Leon Reznik. Recurrent neural networks for colluded applications attack detection in android os devices. In *IEEE World Congress on Computational Intelligence (WCCI) 2020*. IEEE, 2020. Accepted for publication.

[65] Igor Khokhlov, Michael Perez, and Leon Reznik. Machine learning in anomaly detection: Example of colluded applications attack in android devices. In *18th IEEE International Conference on Machine Learning and Applications - ICMLA 2019*. IEEE, 2019.

[66] Igor Khokhlov, Michael Perez, and Leon Reznik. System signals monitoring and processing for colluded application attacks detection in android os. In *2019 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)*, pages 1–5. IEEE, 2019.

[67] Igor Khokhlov, Akshay Pudage, and Leon Reznik. Sensor selection optimization with genetic algorithms. In *IEEE Sensors 2019, Montreal, Canada*, 2019.

[68] Igor Khokhlov and Leon Reznik. Colluded applications vulnerabilities in android devices. In *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 462–469. IEEE, 2017.

[69] Igor Khokhlov and Leon Reznik. Data security evaluation for mobile android devices. In *Open Innovations Association (FRUCT), 2017 20th Conference of*, pages 154–160. IEEE, 2017.

[70] Igor Khokhlov and Leon Reznik. Android system security evaluation. In *Consumer Communications & Networking Conference (CCNC), 2018 15th IEEE Annual*, pages 1–2. IEEE, 2018.

[71] Igor Khokhlov and Leon Reznik. Knowledge graph in data quality evaluation for iot applications. In *2020 IEEE World Forum on Internet of Things*. IEEE, 2020. Accepted for publication.

[72] Igor Khokhlov and Leon Reznik. What is the value of data value in practical security applications. In *IEEE/NDIA/INCOSE Systems Security Symposium 2020*. IEEE, 2020.

[73] Igor Khokhlov, Leon Reznik, and Sahil Ajmera. Sensors in mobile devices knowledge base. *IEEE Sensors Letters*, 4(3):1–4, 2020.

[74] Igor Khokhlov, Leon Reznik, and Rohit Bhaskar. The machine learning models for activity recognition applications with wearable sensors. In *18th IEEE International Conference on Machine Learning and Applications - ICMLA 2019*. IEEE, 2019.

[75] Igor Khokhlov, Leon Reznik, Justin Cappos, and Rohit Bhaskar. Design of activity recognition systems with wearable sensors. In *2018 IEEE Sensors Applications Symposium (SAS)*, pages 1–6. IEEE, 2018.

[76] Igor Khokhlov, Leon Reznik, and Sergey Chuprov. Framework for integral data quality and security evaluation in smartphones. *IEEE Systems Journal*, 2020. DOI: 10.1109/JSYST.2020.2985343.

[77] Igor Khokhlov, Leon Reznik, Suresh Babu Jothilingam, and Rohit Bhaskar. What can data analysis recommend on design of wearable sensors? In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–2. IEEE, 2018.

[78] Igor Khokhlov, Leon Reznik, Ashish Kumar, Ankan Mookherjee, and Rohan Dalvi. Data security and quality evaluation framework: Implementation empirical study on android devices. In *Open Innovations Association (FRUCT), 2017 20th Conference of*, pages 161–168. IEEE, 2017.

[79] Akhil Killawala, Igor Khokhlov, and Leon Reznik. Computational intelligence framework for automatic quiz question generation. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2018.

[80] Michael Kläs, Wolfgang Putz, and Tobias Lutz. Quality evaluation for big data: A scalable assessment approach and first evaluation results. In *Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2016 Joint Conference of the International Workshop on*, pages 115–124. IEEE, 2016.

[81] Anja Klein. Incorporating quality aspects in sensor data streams. In *Proceedings of the ACM first Ph. D. workshop in CIKM*, pages 77–84. ACM, 2007.

[82] Raghavendra Kumar, Prasant Kumar Pattnaik, and Priyanka Pandey. *Detecting and Mitigating Robotic Cyber Security Risks.* IGI Global, 2017.

[83] Smita Kumari and Santanu Kumar Rath. Performance comparison of soap and rest based web services for enterprise application integration. In *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*, pages 1656–1660. IEEE, 2015.

[84] Donald C Latham. Department of defense trusted computer system evaluation criteria. *Department of Defense*, 1986.

[85] Grégoire Lefebvre, Samuel Berlemont, Franck Mamalet, and Christophe Garcia. Blstm-rnn based 3d gesture classification. In *International conference on artificial neural networks*, pages 381–388. Springer, 2013.

[86] Li Li and Wu Chou. Design and describe rest api without violating rest: A petri net based approach. In *Web Services (ICWS), 2011 IEEE International Conference on*, pages 508–515. IEEE, 2011.

[87] Genghong Lu, Dongqin Feng, and Biao Huang. Hmm-based attack detection for networked control systems subject to random packet dropouts. *IEEE Transactions on Industrial Electronics*, 2020.

[88] Mcafee labs threats report, 2016. https://www.mcafee.com/us/resources/reports/rp-quarterly-threats-may-2016.pdf. Accessed: January 17, 2019.

[89] Fred McConnel. Youtube is 10 years old: the evolution of online video, 2015. https://www.theguardian.com/technology/2015/feb/13/youtube-10-years-old-evolution-of-online-video?CMP=fb_gu. Accessed: May 31, 2019.

[90] Yajie Miao, Mohammad Gowayyed, and Florian Metze. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 167–174. IEEE, 2015.

[91] James R Michaelis. Value of information driven content management in mixed reality infrastructures. In *Next-Generation Analyst VI*, volume 10653, page 106530O. International Society for Optics and Photonics, 2018.

[92] Manar Mohamed, Babins Shrestha, and Nitesh Saxena. Smashed: Sniffing and manipulating android sensor data for offensive purposes. *IEEE Transactions on Information Forensics and Security*, 12(4):901–913, 2016.

[93] Quentin Mourcou, Anthony Fleury, Céline Franco, Frédéric Klopcic, and Nicolas Vuillerme. Performance evaluation of smartphone inertial sensors measurement for range of motion. *Sensors*, 15:23168–23187, 09 2015.

[94] Igor Muttik, Jorge Blasco, Tom Chen, Harsha K Kalutarage, and Siraj A Shaikh. Android–collusion conspiracy, 2015. https://www.researchgate.net/profile/Jorge_Blasco/publication/308294888_Android_-_Collusion_Conspiracy/links/57dfcafb08ae0c5b65647eb3/Android-Collusion-Conspiracy.pdf. Accessed: January 17, 2019.

[95] Peter G Neumann. Computer system security evaluation. In *National Computer Conference*, 1978.

[96] Brendan Nicholson. Wrong computer numbers caused emirates jet to almost crash at melbourne airport. *The Age*, 5 2009.

[97] David Nield. All the sensors in your smartphone, and how they work, 2019. https://gizmodo.com/all-the-sensors-in-your-smartphone-and-how-they-work-1797121002. Accessed: June 15, 2019.

[98] Steve Olenski. The state of crowdsourcing, 2015. https://www.forbes.com/sites/steveolenski/2015/12/04/the-state-of-crowdsourcing/. Accessed: April 17, 2019.

[99] Yantao Pan and Xicheng Lu. Energy-efficient lifetime maximization and sleeping scheduling supporting data fusion and qos in multi-sensornet. *Signal Processing*, 87(12):2949–2964, 2007.

[100] Maximilian Panzner and Philipp Cimiano. Comparing hidden markov models and long short term memory neural networks for learning action representations. In *International Workshop on Machine Learning, Optimization, and Big Data*, pages 94–105. Springer, 2016.

[101] Catalogue for smartphones - specifications and reviews - phonesdata, 2019. http://phonesdata.com. Accessed: June 15, 2019.

[102] Leo L Pipino, Yang W Lee, and Richard Y Wang. Data quality assessment. *Communications of the ACM*, 45(4):211–218, 2002.

[103] R Prabha, M Krishnaveni, SH Manjula, KR Venugopal, and LM Patnaik. Qos aware trust metric based framework for wireless sensor networks. *Procedia Computer Science*, 48:373–380, 2015.

[104] Withings Pulse. Withings pulse - what does spo2 mean? what is a normal spo2 level?, 2019. https://support.withings.com/hc/en-us/articles/201494667-Withings-Pulse-What-does-SpO2-mean-What-is-a-normal-SpO2-level-. Accessed: June 15, 2019.

[105] John Quiggin. The value of information and the value of awareness. *Theory and Decision*, 80(2):167–185, 2016.

[106] How many new apps are added to google play everyday?, May 2017. https://www.quora.com/How-many-new-apps-are-added-to-Google-Play-everyday. Accessed: January 17, 2019.

[107] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[108] Irum Rauf and Ivan Porres. Designing level 3 behavioral restful web service interfaces. *ACM SIGAPP Applied Computing Review*, 11(3):19–31, 2011.

[109] Xueqi Ren, Xinming Wang, Hua Tang, Zhaohui Ma, Jiechao Wu, and Gansen Zhao. A unified model for detecting privacy leakage on android. In *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*, pages 474–486. Springer, 2017.

[110] Francesco Restuccia, Nirnay Ghosh, Shameek Bhattacharjee, Sajal K Das, and Tommaso Melodia. Quality of information in mobile crowdsensing: Survey and research challenges. *ACM Transactions on Sensor Networks (TOSN)*, 13(4):34, 2017.

[111] Shawn Rogers. Big data is scaling bi and analytics-data growth is about to accelerate exponentially—get ready. *Information Management-Brookfield*, 21(5):14, 2011.

[112] Pampa Sadhukhan. Performance analysis of clustering-based fingerprinting localization systems. *Wireless Networks*, 25(5):2497–2510, 2019.

[113] Laurel Sadler. A research and experimentation framework for exploiting voi-based methods within analyst workflows in tactical operation centers. In *Next-Generation Analyst V*, volume 10207, page 102070O. International Society for Optics and Photonics, 2017.

[114] Navrati Saxena, Abhishek Roy, and Jitae Shin. Quest: a qos-based energy efficient sensor routing protocol. *Wireless Communications and Mobile Computing*, 9(3):417–426, 2009.

[115] Iot and schema.org: Getting started, 2019. https://iot.schema.org/docs/iot-gettingstarted.html. Accessed: December 11, 2018.

[116] Aubrey-Derrick Schmidt, Frank Peters, Florian Lamour, Christian Scheel, Seyit Ahmet Çamtepe, and Sahin Albayrak. Monitoring smartphones for anomaly detection. *Mobile Networks and Applications*, 14(1):92–106, 2009.

[117] International Technology Roadmap for Semiconductors, 2011 Edition, Micro-Electromechanical Systems (MEMS)., 2014. 03.

[118] Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. "andromaly": a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, 38(1):161–190, 2012.

[119] Fadilah Ezlina Shahbudin and Fang-Fang Chua. Design patterns for developing high efficiency mobile application. *Journal of Information Technology & Software Engineering*, 3(3):1, 2013.

[120] Ubejd Shala and Angel Rodriguez. Indoor positioning using sensor-fusion in android devices, 01 2011.

[121] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1928–1943, 2018.

[122] Richard Sizer. Information technology security evaluation criteria. *Computer Bulletin(London, 1986)*, 5, 1993.

[123] Software quality matters blog. http://blog.smartbear.com/apis/tools-to-help-you-write-apps-that-use-sensors/. Accessed: May 31, 2019.

[124] Sandeep K Sood. A combined approach to ensure data security in cloud computing. *Journal of Network and Computer Applications*, 35(6):1831–1838, 2012.

[125] Number of available applications in the google play store from december 2009 to september 2018, 2018. https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/. Accessed: October 18, 2018.

[126] Diane M Strong, Yang W Lee, and Richard Y Wang. Data quality in context. *Communications of the ACM*, 40(5):103–110, 1997.

[127] Niranjan Suri, Giacomo Benincasa, Rita Lenzi, Mauro Tortonesi, Cesare Stefanelli, and Laurel Sadler. Exploring value-of-information-based approaches to support effective communications in tactical networks. *IEEE Communications Magazine*, 53(10):39–45, 2015.

[128] Chung Huat J Tan and Duncan F Gillies. Generating reliable quality of information (qoi) metrics for target tracking. In *Signal Processing, Sensor Fusion, and Target Recognition XVIII*, volume 7336, page 733607. International Society for Optics and Photonics, 2009.

[129] Vincent F Taylor, Alastair R Beresford, and Ivan Martinovic. Intra-library collusion: A potential privacy nightmare on smartphones. *arXiv preprint arXiv:1708.03520*, 2017.

[130] I Todoran, Laurent Lecornu, Ali Khenchaf, and Jean-Marc Le Caillec. Assessing information quality in fusion systems. In *NATO SAS-106 Symposium on Analysis Support to Decision Making in Cyber Defence & Security*, pages 09–10, 2014.

[131] Android now has 2 billion monthly active users, May 2017. http://www.ubergizmo.com/2017/05/android-2-billion-monthly-users/. Accessed: January 17, 2019.

[132] Kdd cup 1999 data, 1999. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed: May 8, 2019.

[133] Rossouw von Solms, H Van Der Haar, Sebastiaan H von Solms, and William J Caelli. A framework for information security evaluation. *Information & Management*, 26(3):143–153, 1994.

[134] Ayush Vora, Leon Reznik, and Igor Khokhlov. Mobile road pothole classification and reporting with data quality estimates. In *2018 Fourth International Conference on Mobile and Secure Services (MobiSecServ)*, pages 1–6. IEEE, 2018.

[135] Kaveh Waddell. When apps secretly team up to steal your data, 2017. https://www.theatlantic.com/technology/archive/2017/04/when-apps-collude-to-steal-your-data/522177/. Accessed: January 17, 2019.

[136] Richard Y Wang, Henry B Kon, and Stuart E Madnick. Data quality requirements analysis and modeling. In *Data Engineering, 1993. Proceedings. Ninth International Conference on*, pages 670–677. IEEE, 1993.

[137] Richard Y Wang and Diane M Strong. Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4):5–33, 1996.

[138] R. Weiss, L. Reznik, Y. Zhuang, A. Hoffman, D. Pollard, A. Rafetseder, T. Li, and J. Cappos. Trust evaluation in mobile devices: An empirical study. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 25–32, 2015.

[139] Davey Winder. Google confirms android camera security threat: ”hundreds of millions” of users affected, November 2019. https://www.forbes.com/sites/daveywinder/2019/11/19/google-confirms-android-camera-security-threat-hundreds-of-millions-of-users-affected/. Accessed: January 06, 2020.

[140] Michelle Y Wong and David Lie. Intellidroid: A targeted input generator for the dynamic analysis of android malware. In *NDSS*, volume 16, pages 21–24, 2016.

[141] Daoyuan Wu, Yao Cheng, Debin Gao, Yingjiu Li, and Robert H Deng. Sclib: A practical and lightweight defense against component hijacking in android applications. *arXiv preprint arXiv:1801.04372*, 2018.

[142] Erez Yalon. How attackers could hijack your android camera to spy on you, November 2019. https://www.checkmarx.com/blog/how-attackers-could-hijack-your-android-camera. Accessed: February 03, 2020.

[143] Chunyong Yin, Sun Zhang, and Kwang-jun Kim. Mobile anomaly detection based on improved self-organizing maps. *Mobile Information Systems*, 2017, 2017.

[144] Xiaoyong Yuan, Chuanhuang Li, and Xiaolin Li. Deepdefense: identifying ddos attack via deep learning. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8. IEEE, 2017.

[145] Heiga Zen, Yannis Agiomyrgiannakis, Niels Egberts, Fergus Henderson, and Przemysław Szczepaniak. Fast, compact, and high quality lstm-rnn based statistical parametric speech synthesizers for mobile devices. *arXiv preprint arXiv:1606.06061*, 2016.

[146] Xinglin Zhang, Zheng Yang, Wei Sun, Yunhao Liu, Shaohua Tang, Kai Xing, and Xufei Mao. Incentives for mobile crowd sensing: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):54–67, 2016.

[147] Di Zhao, Jiqiang Liu, Jialin Wang, Wenjia Niu, Endong Tong, Tong Chen, and Gang Li. Bidirectional rnn-based few-shot training for detecting multi-stage attack. *arXiv preprint arXiv:1905.03454*, 2019.

[148] Zhibo Zhao and Fernando C Colon Osono. “trustdroid”: Preventing the use of smartphones for information leaking in corporate networks through the used of static analysis taint tracking. In *Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on*, pages 135–143. IEEE, 2012.

[149] Zhizhong Ma, Yuansong Qiao, B. Lee, and E. Fallon. Experimental evaluation of mobile phone sensors. In *24th IET Irish Signals and Systems Conference (ISSC 2013)*, pages 1–8, 2013.

[150] Chris Ziegler. Google unveils 'bouncer' service to automatically detect android market malware, 2012. https://www.theverge.com/2012/2/2/2766674/google-unveils-bouncer-service-to-automatically-detect-android-market . Accessed: January 17, 2019.