SECURE DATA PROVENANCE FOR IOT DEVICES

EBELECHUKWU NWAFOR

Department of Computer Science

APPROVED:

Gedare Bloom, Chair, Ph.D.

Wayne Patterson, Ph.D.

Gloria Washington, Ph.D.

Robert Rwebangira, Ph.D.

Messac Arenaz, Ph.D.
Dean of the Graduate School

SECURE DATA PROVENANCE FOR IOT DEVICES

by

EBELECHUKWU NWAFOR, M.S.

DISSERTATION PROPOSAL

Presented to the Faculty of the Graduate School of

Howard University

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

Department of Computer Science

HOWARD UNIVERSITY

FEB 2016

Acknowledgements

First and foremost, I would like to thank God for making this possible, without him i will not be where i am today. I would also like to thank my parents, Benneth and Chinwe for their constant encouragement, love and support.

I would like to thank my advisors Dr. Gedare Bloom and Dr. Legand Burge for their guidiance and encouragement.Also for believing in me and my research ideas. I would like to thank my lab partner Habbeeb Olufowobi and fellow graduate student Marlon Mejias for their constant constructive criticism and valuable input in various drafts revisions of my proposal.

Abstract

The concept of Internet of Things (IoT) offers immense benefits by enabling devices to leverage networked resources thereby making more intelligent decisions. The numerous heterogeneous connected devices that exist throughout the IoT system creates new security and privacy concerns. Some of these concerns can be overcome through data trust, transparency, and integrity, which can be achieved with data provenance. Data provenance also known as data lineage provides a history of transactions that occurs on a data object from the time it was created to its current state.Data provenance has immense benefits in detecting and mitigating current and future vulnerability attacks and has applications in intrusion detection, access control, and digital forensics.

This dissertation looks at provenance with a focus on IoT devices. It takes a holistic approach in the creation, security and application of provenance data to mitigating malicious attacks. We create a secure provenance aware system for IoT devices.This system ensures trust and helps establish causality for decisions and actions taken by an IoT connected system. As a result of the amount of data that is generated from our data provenance system, there arises an issue of running out of memory.To address this issue, we propose a novel data pruning technique that provides optimal storage of provenance data contained in the IoT device.We conclude by looking at the applications of provenance for security of malicious attacks by focusing on intrusion detection of malicious threats.We propose an intrusion detection system for IoT devices. The provenance generated from the IoT devices can provide valuable insights that could be used to detect and record abnormal patterns that might exist in a causal relationship between entities contained in the provenance chain.

Table of Contents

List of Figures

Chapter 1

Introduction

According to the oxford English dictionary, provenance can be defined as the place of origin or earliest known history of something. An example of provenance can be seen with a college transcript. A transcript can be defined as the provenance of a college degree because it outlines all of the courses satisfied in order to attain the degree.

In the field of computing, Data provenance also known as data lineage is the history of transformations performed on a data object from the beginning of time to its current state. An example of provenance for software systems is a web server's log file. This file contains metadata for various request and response time and ip addresses. Provenance denotes the who, where and why of data.Provenance data is represented as an acyclic graph which denotes casual relationship and dependencies between entities.Provenance ensures trust and integrity of data. The information in which provenance offers can be used in digital forensics to investigate the cause of a malicious attack and also in intrusion detection system to further enhance the security of computing devices.

TODO: Create transition...

The internet of things(IoT) can be defined as a network of heterogeneous devices communicating together. With the recent data explosion,information is disseminated at every communication level that exists. From mobile devices to desktop computers and servers.Making IoT systems provenance aware is of the essence because it ensures trust and integrity of systems. Enabling IoT device provenance aware allows devices to be able to capture information such as the who, where and how transformations occur on a data object enabling us to be able to trace back in the vent of a malicious attack.

1.1 Motivation

According to a report released by Cisco [4], it is estimated that a total of 50 million devices will be connected to the internet by 2020. With these heterogeneous devices connected, security and privacy risks increase. Rapid7 [5] discovered that vulnerabilities exist in baby monitors that allowed attackers unauthorized remote access to these devices whereby an attacker can remotely view live video feeds. Having a provenance aware system will be beneficial in this situation since we have a record of input and output operations performed on the device, we can be able to look back on operations performed on the device to determine who, where, and how a malicious activity occurred. Devices (things) connected in an IoT network are embedded systems, which require lightweight and efficient solutions compared with general purpose systems. This requirement is attributed to the constrained memory and computing of such devices. A major issue arises in ensuring that data is properly secured and disseminated across the IoT network. The vast amount of data generated from IoT devices requires stronger levels of trust which can be achieved through data provenance. Provenance has immense benefits in the IoT. Data provenance ensures authenticity, integrity and transparency between information disseminated across an IoT network. Security applications such as anomaly detection, digital forensics, and access control can be further enhanced by incorporating data provenance in IoT devices. The goal of data provenance is to determine causality and effect of actions or operations performed on data. Provenance ensures transparency between things connected in IoT systems. By creating data transparency, we can trace information to determine where, if and when a malicious attack occurs. To achieve transparency, we propose a secure provenance aware system that provides a detailed record of all data transactions performed on devices connected in an IoT network.

1.2 Provenance-Aware IoT Device Use Case

Consider a smart home which contains interconnected devices such as a thermostat connected to the internet that sets the temperature remotely and also learns the temperature based on previous temperature settings. A malicous intruder tries to gain access to the connected devices remotely. Provenance can be used to track the events that have happened to pin point where the malicious attack originates it can also be used as a safeguard to alert of a possible remote hijack thereby protecting us from future occurrence.

1.3 Research Questions

collecting provenance data in IoT devices raises some key research questions. Some of these questions are outlined below:

- Memory constraints,
- what approach to use. access control with IoT.
- Collecting system level provenance in embedded systems.
- Do we use the OPM approach or something else. Querying provenance data.

1.4 Research Contribution

This dissertation proposes the following contributions:

- A provenance collection framework which denotes causality and dependencies between entities in an IoT system.This system creates groundwork for capturing and storing provenance data.
- A novel framework for Data pruning.This addresses the memory overflow of the memory constrained IoT devices.

- A framework for providing anomaly detection using provenance data in an IoT system.

1.5 Organization of Dissertation proposal

The remaining portion of the dissertation proposal is organized as follows. Chapter 2 talks about background information on data provenance, some of the techniques of collecting system level provenance, Data pruning techniques and also provenance based access control, Provenance data model. chapter 3 discusses our proposed provenance collection system and focuses specifically on preliminary work done in creating a provenance aware system. Chapter 5 concludes the proposal and discusses the proposed framework and projected timeline for completion.

Chapter 2

Background and Related Work

This section outlines some of the work done in the area of data provenance for file systems and also data compression techniques for data provenance, and providing access control for using data provenance.

2.1 Overview of Provenance Aware systems

There have been a considerable amount of work done on collecting provenance data. Some of which has been focused on databases, sensore networks and system level provenance but so far little attention has been given to provenance in the IoT. Some the previous work done on data provenance collection are outlined below:

2.1.1 Provenance Aware Storage System(PASS)

This was developed at Harvard by MuniswamyReddy et al. There are two versions PASS v1 and PASS v2. v1 allows... while version 2...Provenance information is stored in the same location as the file system for easy accessibility, backup, restoration, and data management. Provenance information is collected and stored in the kernel space. The system also provides provenance query capabilities and cycle detection. PASS systems recognizes data pruning for efficient storage, which is in line with our research goal but data pruning was not fully explored in PASS.

The collector keeps track of system level provenance. It intercepts system calls which are translated into provenance data and is stored in an in kernel database. This database maps key value pairs for provenance data for fast index look up. It also provides functionalities

for querying provenance data in the database. PASS detects and eliminates cycles that might occur in provenance dependencies as a result of version avoidance.cycles violates the dependency relationships between entities. For example, a child node could depend on a parent node and also be an ancestor of a parent node. It achieves this by merging processes that might have led to the cycles.

2.1.2 HiFi

Bates et al. [2] developed system level provenance information for the Linux kernel using a Linux Provenance Modules, which tracks whole system provenance including interprocess communication, networking, and kernel activities. This is achieved by mediating access to kernel objects. Linuex Security Model is a framework that was designed for providing custom access control into the Linux kernel.It consists of a set of hooks which is executed before access decision is made.

HiFi contains three components, provenance collector, provenance log and provenance handler. collector and log are in the kernel space while the handler is in the user space.The log is a storage medium which transmits the provenance data to the user space.The collector contains the LSM which resides the kernel space. The collector records provenance data and writes it to the provenance log.The handler intercepts the provenance record fromt he log processes the data and stores it to the provenance record. LSM was designed to avoid problem created by direct system call interception. The provenance information from the kernel space is securely transmitted to the provenance recorder in the user space. This approach to collecting provenance data differs from our work since we focus on embedded systems and are concerned with input and output (I/O) data, which primarily involve sensor and actuator readings.

2.1.3 RecProv

creates a provenance system which records user level provenance thereby avoiding the overhead incurred by kernel level provenance recording.It uses mozilla rr to perform deterministic record and replay by capturing system calls and non deterministic input. It also ensure the integrity of provenance data up till the point that the host is compromised by trace isolation. Mozilla rr relies on ptrace which intercepts system call during context switch.Mozilla rr is a debugging tool for linux browser. It is developed for the deterministic recording and replaying of firefox browser in linux. System calls such as execve, clone, fork, open, read, write, clode, dup, mmap, socket, connect, accept are recorded. the provenance information generated in converted into PROV-JSON a W3C standard for relaying provenacne information and also stores provenance data in Neo4j a graph database for visualization of provenance graphs. It does not require changes to the kernel like most provenance monitoring systems (include citations of other provenance monitoring systems that require kernel modification). it generates 20 percent overhead which is acceptable according to the authors.

Recprov uses PTRACE_PEEKDATA from PTRACE to access the derefrenced address of the traced process from the registers.

2.1.4 StoryBook

Spillance et al developed a user space provenance collection system, Storybook [14] that allows the collection of provenance data from the user space thereby reducing performance oververhead from kernel space provenance collection. This system is modular.It allows the use of application specific extensions allowing additions such as database provenance, system provenance, and web and email servers. It achieves provenance capture by using FUSE for system level provenance and MYSQL for database level provenance capture. Story Book allows developers to implememtn provenance inspectors. these are custom provenance models which captures the provenance of applications which are often modified

by different application(e.g web servers, databases). When an operation is performed on a data object, the appropriate provenance model is triggered and provenance data for that data object is captured. Sotrybook stores provenance information such as open, close, read or write, application specific provenance, causality relationship between entities contained in the provenance system(?). Provrenance is stored in key value pairs and It uses Fable as the storage backend. Storybook allows for provenance query.It achieves this by looking up inode in the file, ino hashtable.

2.1.5 Trustworthy whole system provenance for Linux kernel

2.1.6 Towards Automated Collection of Application-Level Data Provenance

2.1.7 user space provenance with Sandboxing

2.1.8 Provenance for Sensors

Lim et al. [3] developed a model for calculating the trust of nodes contained in a sensor network by using data provenance and data similarity as deciding factors to calculate trust. The value of provenance signifies that the more similar a data value is, the higher the trust score. Also, the more the provenance of similar values differ, the higher their trust score. This work differs from our approach since the authors focus on creating a trust score of nodes connected in a sensor network using data provenance and do not emphasize how the provenance data is collected. We are focused on creating a secure provenance aware system for I/O operations which is used to ensure trust of connected devices.

2.2 Model for representing provenance for IoT

In order to generate provenance, we have to satisfy the who, where, how, and what of data transformations.provenance data is represented using a provenance model which is serialized as JSON output. This model contains information such as sensor readings, device name,, device information, provenance information. This information will be converted into provenance data model and in order to allow for interoperability and visualization.

2.2.1 Open Provenance Model(OPM)

Open provenance model was a specification derived as a result of a meeting at the International Provenance and Annotation Workshop (IPAW) workshop in may 2006. OPM was created to address the need of allowing a unified way of representing provenance data among various applications. It allows for interchangeability between various provenance models that might exist. The goal of OPM is to develop a digital representation of provenance for entities regardless of whether it is produced by a computer system. An example of such is depicted in Figure 7. This OPM graphs represents a process of driving a car. It is represented as a directed acyclic graph which denotes causal dependencies between entities. The edges in the graph denotes dependencies with its source denoting effect and its destination denoting cause.The edges and their relationships are denoted below:

When muliple process has been used by an artifact, roles(denopted by R) should be defined.

- wasGeneratedBy: Shows relationship in which an entity(e,g artifact) is utilized by one or more entities(e.g process). An entity can use multiple enities so it is important to define the role.
- wasControlledBy: This showsa the relationship in which an entity caused the creation of another entity.
- used(Role): denotes an enity requires the services of another enity in order to execute.

- wasTriggeredBy: This represents a process that was triggered by another process
- wasDerrivedFrom:

There are three entities contained in the OPM model: artifact, process, agent.

- artifact: This represents the state of an entity.An artifact is graphically represented by a circle.
- Process: This denotes an event which is taking place.A process is represented by a square object.
- Agent: These are actors that facilitate the execution of a process.An agent is represented by a hexagon.

OPM denotes all previous and current actions that have been performed on an entity and the relationship between each entities contained in the graph. Figure 2 represents an example of an OPM acyclic graph with all of its causal dependencies. The goal of OPM is to be able to model the state of how things both digital or physical are at a given state.

2.2.2 Provenance Data Model(ProvDM)

PROV-DM is a W3C standardized extension of OPM. Prov-DM is a model that is used for depict causal relationship between entities, activities and , and agents(digital or physical). It creates a common model that allows for interchange of provenance information between heterogeneous devices. It contains two major components: types and relations. Figure below shows an example of a causal relationship between an entity, agent, and activity in a PROV-DM

- entity: An entity is a physical or digital object. An example of an entity is a file system, a process, or an motor vehicle.

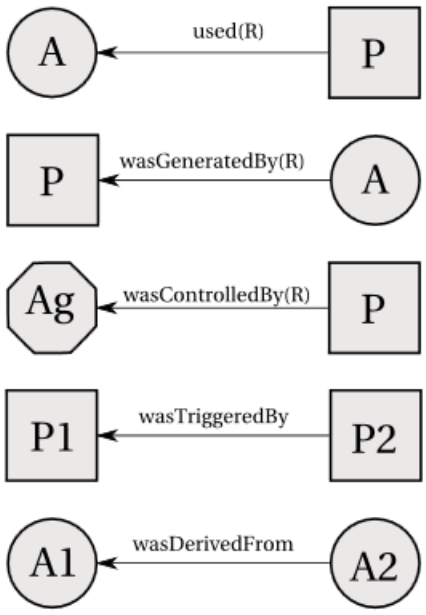


Figure 2.1: Edges and entities in OPM

- Activity: An activity represents some form of action that occurs over a time frame.Actions are acted upon by an entity. an example of an activity is a process opening a file directory, Accessing a remote server.

- Agent: An agent is a thing that takes ownership of an entity, or performs an activity. An example of an agent is a person, a software product, a process.

PROV-DM relations represents causal dependencies which denotes relationship between the core types(entity, activity, agent).The names of relations make use of past tense to denote past occurrence of provenance information. provenance does not keep track/estimate of future events. PROV relations are outlined below:

- wasGeneratedBy: This relation signifies the creation of an entity by an activity.

- used: This relation denotes that the functionalities of an entity has been adopted by an activity.

- wasInformedBy: This relation denotes an causality that follows the exchange of two activities.
- wasDerievedFrom This relation represents a copy of information from an entity.
- wasAttributedTo: This denotes relational dependency to an agent. It is used to denote relationship between entity and agent when the activity that created the agent is unknown.
- wasAssociatedWith:This relation denotes a direct association to an agent for an activity that occurs.This indicates that an agent plays a role in the creation or modification of the activity.
- ActedOnBehalfOf: This denotes assigning authority to perform a particular responsibility to an agent. This could be by itself or to another agent.

Prov-DM contains similar yet subtle differences between OPM.Some of the difference between OPM and PROV-DM are outlined below:

- the main components Artifact, process and agent in the OPM model are changed to Entity, Action, and agent.
- additional causal dependencies such as wasAttributedTo and actedOnBelafOf are included to represent direct and indirect causal dependencies respectively between agents and entity.

Entities, activities and agent are represented by oval, rectangle and hexagonal shape respectively. Since PROV-DM is built on OPM and contains easy to understand constructs of enities, we choose to use this instead of OPM.

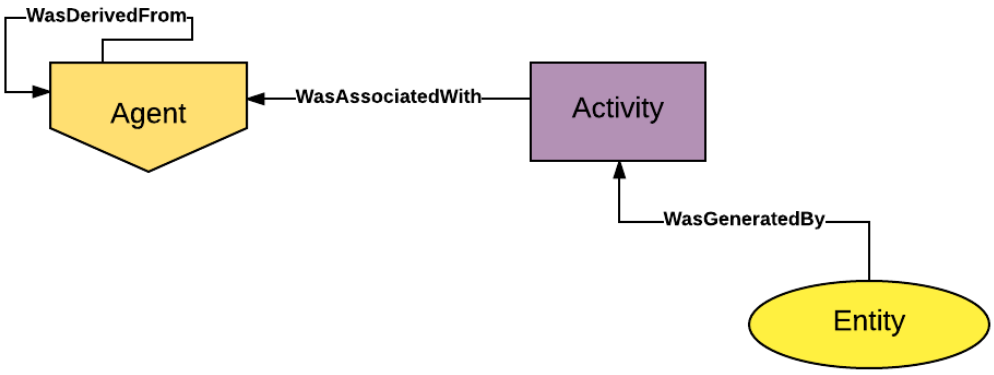
2.3 Overview of Data pruning techniques

2.3.1 Graph Compression

2.3.2 Dictionary Encoding

2.3.3 Arithmetic encoding

2.4 Intrusion Detection for IoT


WasDerivedFrom
Agent
WasAssociatedWith
Activity
WasGeneratedBy
Entity

Chapter 3

Proposed Model

In this chapter we define all of the components of our proposed model. From the provenance-aware system which deals with the collection, storage and querying of provenance data from IoT devices to the IDS system which deals with detecting malicious attacks on IoT Devices.

3.1 Provenance-Collection System

In this section, we outline the components of our system and describe how provenance information is collected across the IoT system. Figure 1 displays the system architecture of our approach. Sensor and actuator readings in the form of I/O are recorded by the tracer component. This component intercepts system level I/O and produces trace information in the Common Trace Format (CTF). CTF represents binary trace information containing multiple streams of binary events such as I/O activity. Trace information is converted into the Open Provenance Model (OPM), which represents the relationship between provenance entities contained in the system. Our system relies heavily on data pruning to reduce and remove unimportant provenance in order to conserve memory. The key research challenge is in creating an efficient data pruning technique that ranks I/O operations based on importance. Pruned provenance information is later securely transmitted and stored in a private cloud backend.

The goal of our research is to create a provenance aware system that records I/O operations on data for devices connected in an IoT system. For our implementation, several tools and hardware components are utilized in the development of our prototype, some of the tools utilized are outlined below:

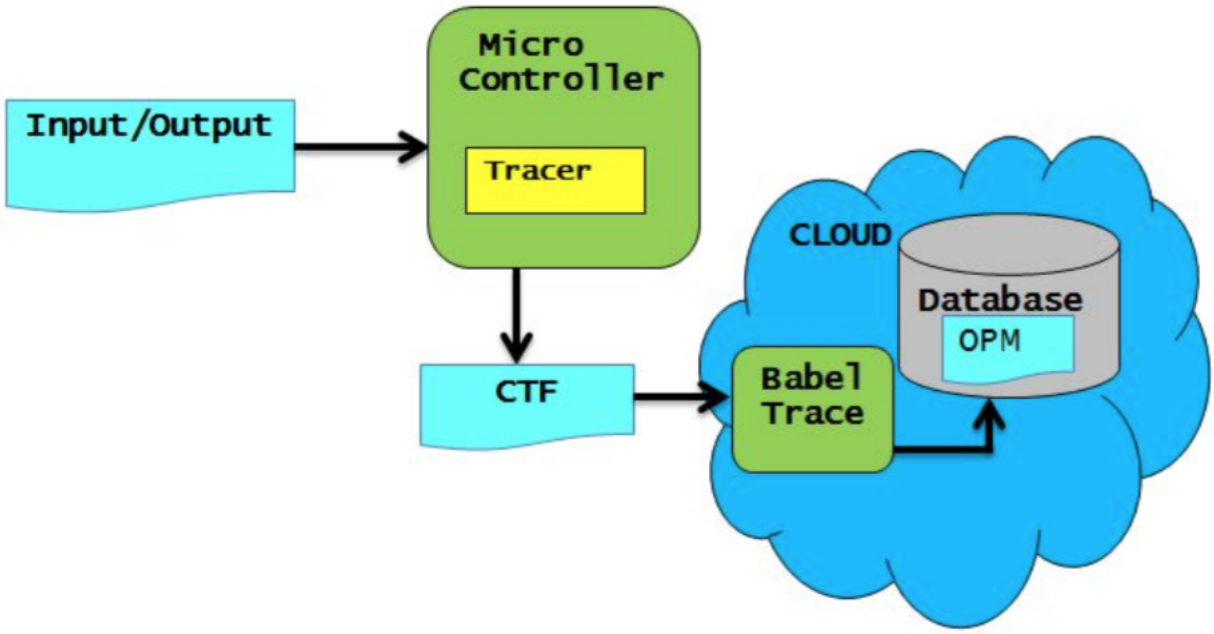


Figure 3.1: Edges and entities in OPM

- Xilinx Zynq ZedBoard and TI TM4C129E. These devices are microcontrollers used to evaluate our approach. We choose the Xilinx Zynq ZedBoard because it is a representation of what can be found on an IoT gateway device and it has the capability to include custom hardware in programmable logic. Also, TM4C is a lowcost, simple IoT demonstrator that was chosen for its highperformance, onboard emulation, and IoT gateway projects can be programmed without additional need for hardware tools.

- Real Time Executive for Multiprocessor Systems (RTEMS) is an open source realtime operating system (RTOS) for embedded systems. This operating system is a typical RTOS that may be deployed in IoT devices.

3.2 PAIST: Provenance Aware IDS System for IoT

This section outlines the core functionalities of our model PAIST.

Chapter 4

Concluding Remarks

4.1 Significance of the Result

Now that we have shown that we cannot solve narrow-interval linear equation systems in general, what does this mean to the computing and mathematical communities? Unless P=NP, attempts at developing a feasible algorithm to solve this problem in general will assuredly fail; however, the very fact that there exists an algorithm for solving a large number of these systems (see [9]) shows that work on solving a subclass of the class of all narrow-interval linear equation systems is certainly a worthwhile endeavor.

4.2 Future Work

The problem now shifts to identifying new subclasses of the class of all narrow-interval linear equation systems for which the problem of solving them is possible with the development of new algorithms. Also, if the general problem (or any problem in the class NP) shows up often enough in industry, science, research, etc., work on improving existing and/or creating new approximation methods (including heuristic and/or statistical methods, where applicable) is certainly warranted. Since we cannot compute the exact bounds for the general case, good approximation methodologies are the most we can hope for or expect.

References

[1] S. Cook, "The complexity of theorem-proving procedures," *Proceedings of the 3rd ACM Symposium on Theory of Computing*, Shaker Heights, Ohio, 1971, pp. 151–158.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of* NP-*Completeness*, W. H. Freeman, San Francisco, 1979.

[3] R. Karp, "Reducibility among combinatorial problems," in: R. Miller and J. Thatcher (eds.), *Complexity of Computer Computations*, Plenum Press, New York, 1972, pp. 85–103.

[4] R. B. Kearfott and V. Kreinovich (eds.), *Applications of Interval Computations*, Kluwer Academic Publishers, Norwell, MA, 1996.

[5] V. Kreinovich, A. V. Lakeyev and S. I. Noskov, "Optimal solution of interval linear systems is intractable (NP-hard)," *Interval Computations*, 1993, No. 1, pp. 6–14.

[6] V. Kreinovich, A. V. Lakeyev and J. Rohn , "Computational complexity of interval algebraic problems: some are feasible and some are computationally intractable: a survey," in: G. Alefeld and A. Frommer (eds.), *Scientific Computing and Validated Numerics*, Akademie-Verlag, Berlin, 1996, pp. 293–306.

[7] V. Kreinovich, A. V. Lakeyev, J. Rohn and P. Kahl, *Feasible? Intractable? On Computational Complexity of Data Processing and Interval Computations*, Kluwer Academic Publishers, Norwell, MA, 1996 (to appear).

[8] U. Kulisch and W. L. Miranker, *Computer Arithmetic in Theory and Practice*, Academic Press, NY, 1981.

[9] A. V. Lakeyev and V. Kreinovich, "If input intervals are small enough, then interval computations are almost always easy," *Reliable Computing*, Supplement (Extended

Abstracts of APIC'95: International Workshop on Applications of Interval Computations), 1995, pp. 134–139.

[10] L. Levin, "Universal sequential search problems," *Problems of Information Transmission*, 1973, Vol. 9, No. 3, pp. 265–266.

[11] R. E. Moore, "Automatic error analysis in digital computation," *Technical Report LMSD-48421*, Lockheed Missiles and Space Co., Palo Alto, CA, January 1959.

[12] R. E. Moore, *Interval Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1966.

[13] A. Neumaier, *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, 1990.

[14] S. G. Rabinovich, *Measurement Errors: Theory and Practice*, American Institute of Physics, NY, 1995.