SECURE DATA PROVENANCE FOR INTERNET OF THINGS(IOT) DEVICES

EBELECHUKWU NWAFOR

Department of Computer Science

SECURE DATA PROVENANCE FOR INTERNET OF THINGS(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. The numerous heterogeneous connected devices that exist throughout the IoT system creates new security and privacy paradigm which encompases various security issues from different areas of research . .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 been explored in the areas of scientific computing and buisness to track the workflow processes,also in field of computer security for forensic analysis and intrusion detection.Data provenance has immense benefits that provenance offers in detecting and mitigating current and future malicious attacks.

In this dissertation, we explore provenance with a focus on IoT devices. We take an in depth look in the creation, security and applications of provenance data to mitigating malicious attacks. We define our model constructs for representing IoT provenance data. We address the issue of provenancecollecting by creating a secure provenance aware system that collects provenance data in 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 memory space fillage.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 data for security of malicious attacks. We focus on intrusion detection of malicious threats.We propose an intrusion detection system that detects malicous attaks based on provenance data 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. This dissertation evaluates the data pruning algorithms for efficient storage of provenance data and evaluates a proof-of-concept in by creating an intrusion detection system for IoT devices.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Over the years,the Internet of Things has garnered significant attention. According to a report released by Cisco, it is estimated that a total of 50 million devices will be connected to the internet by the year 2020. With the vast amounts of heterogeneous devices connected, security and privacy risks are inceased. Rapid7, an internet security and analytics organization, released a report. In their report, they outline that vulnerabilities exist in baby monitors which allowed intruders unauthorized remote access to these devices whereby a malicious intruder can remotely view live feeds from a remote location. Having a provenance aware device will be beneficial since we have a trail of all activities performed on the device, we can backtrack and analyze operations performed on devices to determine who, where, and how a malicious activity occurred.

In an IoT system, most of the devices (things) are embedded systems which require lightweight and efficient solutions compared to general purpose systems.This requirement is attributed to the constrained memory and computing power of such devices. A major issue arise in ensuring that data is properly secured and disseminated across an 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 IoT. The goal of data provenance is to determine causality and effect of actions or operations performed on a data object. Provenance ensures transparency between things connected in IoT systems. Data provenance ensures authenticity, integrity and transparency between information disseminated across an IoT network. By creating data transparency, we can track data objects to determine where, in an event of a malicious attack. 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 and also investigate its applications in providing intrusion detection to IoT devices.Security applications such as intrusion detection, digital forensics, and access control can be further enhanced by incorporating data provenance to IoT devices. Data is generated in at a fast rate in real-time by sensors and actuators. The provenance of this data tends to be larger than the data itself.

1.1.1 Data Provenance

The oxford English dictionary defines provenance 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 can be defined as the history of all transformations performed on a data object from the its creation to its current state. Data Provenance has been explored in the areas of scientific computing to track how an experiment is produced for reproducability, in buisness to determine the workflow of a process, in field of computer security for forensic analysis and intrusion detection. Provenance denotes the who, where and why of data.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 of all host systems that requests information from the server.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 systems to further enhance the security of computing devices. Most IoT devices are memory constrained devices TODO: Expand more on statement..

1.1.2 Internet of Things(IoT)

The internet of things(IoT) can be defined as a network of heterogeneous devices communicating together. With the recent data explosion,information is ubiquitous and disseminated at vast rate.

Creating a provenance aware system is beneficial to IoT device 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 which enables backtracking in an event of a malicious attack.

TODO: Create transition from Data provenance to IoT...

1.2 Provenance-Aware IoT Device Use Case

Consider a smart home which contains interconnected devices such as a thermostat connected a wireless network that automatically detects and regulates the temperature of a room based on prior information of a user's temperature settings, a smart lock system that can be controlled remotely and informs a user via short messaging when has been a breach, a home security camera motoring system, A smart fridge which sends a reminder when food produce are low. A malicous intruder successfully attempts to gain access to the smart lock system and security camea remotely. Provenance can be used to track the series of events to determine where and how a malicious attack originated.It can also be used as a safeguard to alert of a possible remote or insider compromise thereby protecting future or current malicious attacks.

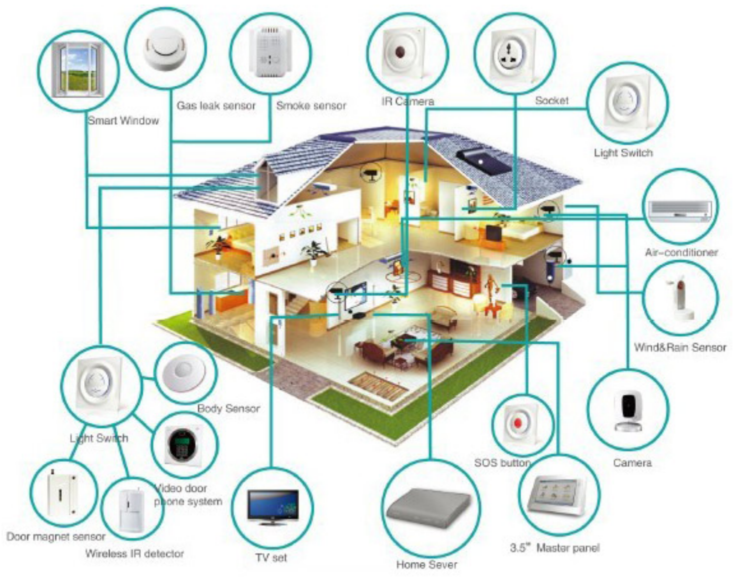

Figure 1.1: Place holder for use case Diagram

1.3 Research Questions

Data provenance collection in IoT devices raises some key research issues. Some of the issues raised are outlined below:

- Memory constraints on IoT Devices: The vast amounts of data generated leads high storage space utilization . Proper memory management/data pruning techniques should be employed for efficient storage of provenance data on memory contrained devices.

- How do we model provenance data collected for IoT devices? Do we use the OPM approach or create a specialized taxonomy for modeling provenance data.

- Provenance Query: How do we query provenance data in order to interprete provenance information and make anaylsis of data collected.

- Provenance Versioning: Provenance version creates cycles. When a file is read or edited do we create a new instance of the file? Tracking all transformations that occurs on a data object in memeory constrained devices might lead to running out of storage space. A new instance of the provenance of that file should be created and included in the provenance data.

- Securing Provenance: Due to the great level of sensitivity that provenance data entails, it is of utmost importance to ensure the confidentiality and integrity of provenance data stored on IoT devices while at rest or in transit. Proper encryption and authentication techniques need to be employed to ensure the confidentiality, integirty, and availability of provenance data.
- Provenance collection raises privacy issues.How do we ensure that the vast amount of data collected is not invasive to the privacy of the use and also is not used as a tool to perform malicious attacks.

1.4 Research Contribution

In this dissertation, we propose the following key contributions:

- A provenance collection framework which denotes causality and dependencies between entities contained in an IoT system.This system creates the groundwork for capturing and storing provenance data on IoT devices.
- An efficent algorithm for storing provenance data on IoT devices. We evaluate our contributions for an efficient provenance storage by using an intrusion detection system on IoT devices.

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 involved in collecting system level provenance, data pruning techniques for storage optimization and also applications of data provenance with provenance based intrusion detection system. chapter 3 discusses our proposed provenance collection system and focuses specifically on preliminary work done in creating a provenance aware system. Chapter 4 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 state of the art in the area of data provenance collection systems, data compression techniques for data provenance, and models for representing provenance.

2.1 IoT Architecture

IoT architecture consists of four distinct layers, sensor layer, device layer, gateway layer and cloud layer. The base of the architectural stack consist of sensors and actuators which gathers information from the IoT devices and interacts with the device layer. The device layer consists of devices such as (e.g mobile phones, laptops, smart devices) which are responsible for aggregating data from sensors and actuators. These device in turn forwards the aggregated data to the gateway layer.The gateway layer is concerned with the routing and forwarding of data collected. It could also serve as a medium of temporal storage. This layer serves as the intermediary between the device and the cloud layer. The cloud layer is involved with the storage and processing of data collected from the gateway layer. This information can be further processed with advanced analytics to derive insights from the data received. Figure 2 displays the IoT architectural stack and the interactions between the respective layers.

TODO: Discuss IoT architecture

2.2 Provenance and Metadata

Metadata and provenance are often considered related but yet subtle differences exist between them. Metadata contains descriptive information about data. Metadata can be considered as provenance when there exists relationship between objects and explains how objects are . For example a web server can have metadata information such as the host and destination IP addresses which might not be considered as provenance information for a specific application.It could also contain time stamps with location of IP addresses which are considered as provenance information. In summary metadata and provennace are not the same, metadata contains valuable provenance information but not all provenance information is metadata.

Metadata is used to represent properties of objects. Many of those properties have to do with provenance, so the two are often equated. How does metadata relate to provenance? Descriptive metadata only becomes part of provenance when one also specifies its relationship to deriving an object. For example, a file can have a metadata property that states its size, which is not considered provenance information since it does not relate to how it was created. The same file can have metadata regarding creation date, which would be considered provenance-relevant metadata. So even though a lot of metadata has to do with provenance, both terms are not equivalent. In summary, provenance is often represented as metadata, but not all metadata is necessarily provenance.

2.3 Provenance and Log data

Log data contains information about that activity of an object in a operating system or process.Provenance is a specialized form of Log data. It contains information specific to an application domain. Log files might contain unrealted information such as error messages, warnings which might not be considered as provenance data. Logging is limited, lacking information that shows the internal state of an object. Provenance allows for a holistic collection of information that relates to the change the internal state of an object.

2.4 Related Work on Provenance Collection Systems

There has been a considerable amount of work done on data provenance collection. Some of this work has been focused on databases, sensor networks, scientific workflow systems and file system provenance but so far little attention has been given to provenance in IoT devices. Some the prior work done on data provenance collection are outlined below:

2.4.1 Provenance Aware Storage System(PASS)

MuniswamyReddy et al developed a provenance aware system that tracks system level provenance of the linux file system.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. PASS is composed of 3 major components: provenance collector, and provenance storage, provenance query.The collector keeps track of system level provenance. It intercepts system calls which are translated into provenance data and initially stored in memory as inode catche. This provenance data is then transfered to a file system in an kernel database, BerkleyDB. 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. The query tool is built ontop of Berkley DB. For querying, users can process queries using the provenance explorer. Query is done using commands such as MAKEFILE GENERATION which creates the sequence of events that led to the final state of a file or process. DUMP ALL, gets the provenance of the requested file.PASS stores a refrence to the executable that created the provenance data, input files, A description of hardware in which provenance data is produced, OS information, process environment, process parameters, and other/data such as a random number generator seed.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. PASS eliminates cycles by merging processes that might have led to the cycles.

2.4.2 HiFi

Bates et al. [2] developed system level provenance information for the Linux kernel using Linux Provenance Modules(LPM), which tracks at whole system provenance including interprocess communication, networking, and kernel activities. This is achieved by mediating access to kernel objects. Linux 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.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.

HiFi contains three components, provenance collector, provenance log and provenance handler. The collector and log are contained in the kernel space while the handler is contained 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 from the log.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.

2.4.3 RecProv

RecProv is a provenance system which records user level provenance, avoiding the overhead incurred by kernel level provenance recording.It uses mozilla rr to perform deterministic record and replay by monitoring system calls and non deterministic input.Mozilla rr is a debugging tool for linux browser. It is developed for the deterministic recording and replaying of firefox browser in linux. It also ensure the integrity of provenance data up till the point at which a host is compromised by trace isolation. Mozilla rr relies on PTRACE which intercepts system calls during context switch.It uses PTRACE to monitor the CPU state ducring and after a system call. System calls such as execve, clone, fork, open, read, write, clode, dup, mmap, socket, connect, accept are monitored which is used for file versioning. 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 and storage.It does not require changes to the kernel like most provenance monitoring systems (include citations of other provenance monitoring systems that require kernel modification).

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

2.4.4 StoryBook

Spillance et al developed a user space provenance collection system, Storybook [**?**] 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. StoryBook allows developers to implemement provenance inspectors. these are custom provenance models which captures the provenance of specific 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.

2.4.5 Trustworthy whole system provenance for Linux kernel

Bates et al provide a security model for provenance collection in the linux kernel. This is achieved by creating a Linux Provenance Model(LPM). LPM serves as a security layer that provides a security abstraction for provenance data. It is involved in the attestation disclosure of the application layer, authentication channel for the network layer.The goal of LPM is to provide an end to end provenance collection system.

LPM ensures the following security guarantees: for LPM,the system must be able to detect and avoid malicous forgery of provenance data. The system must be trusted and easily verifiable.

When an application invokes a system call, the LPM tracks system level provenance which is transmitted to the Provenance module via the relay buffer . The provenance module registers contains hooks which records system call events. These events are sent to a provenance recorder in the user space. The provenance recorder ensures that the provenance data is sored in the appropriate backend of choice. Proveance recorders offer storage for Gzip, PostGreSQL, Neo4j and SNAP.

2.4.6 Towards Automated Collection of Application-Level Data Provenance

Tariq et all developed a provenance collection system which automatically collects the provenance of applications source code at run time. It takes a different approach from system level provenance capture. It achieves provenance capture by using LLVM compiler framework and SPADE provenance management system. LLVM is a framework that allows dynamic compilation techniques of applications written in C, C++, Objective-C, and Java.Provenance is inserted during compilation.LLVM contains an LLVM Reporter. This is a java class that parses the output file collected from the LLVM Tracer and forwards the provenance data collected to the SPADE Tracer. SPADE is a provenance management tool that allows for the tranformation of domain specific activity into provenance data. The LLVM Tracer module tracks provenance at the exit and entry of each function. The output is an Open Provenance Model in which functions are represented by an OPM process, arguments and return values are represented as an OPM artifact. To minimize provenance record, users are allowed to specify what functions they would like to collect provenance information at run time. The system discards provenance data that are not needed. This does not provide a whole system provenance of all activities that occurs on the system.The LLVM optimizer generates a graph of the target application. This graph is used to perform reverse reachability analysis from the functions contained in the graph.

A workflow of how the framework works in collecting provenance is as follows:

- The application is compiled and converted into bitcode using the LLVM C compiler, clang.
- The LLVM Tracer module is compiled with the bitcode.
- Provenance instrumentation is added to the bitcode via the LLVM Trace module.
- instrumentation bitcode is converted into assembly code via the compiler llc
- LLVM Tacer and Reporter is compiled into assembly code.
- The instrumented application and assembly code is compiled into an executable binary and sent to the SPADE kernel.

One major limitation to this approach of collecting application level provenance is that a user is required to have the source code of the specific application in which provenance data is required.Also, provenance collection is limited to function's exit and entry points.

2.4.7 Provenance from Log Files: a BigData Problem

Goshal et al developed a framework for collecting provenance data from log files.They argue that log data contains vital information about a system and can be an important source of proveance information.Provenance is categorized based on two types: process provenance which involves provenance information of the process in which provenance is captured. Data provenance describes the history of data involved in the execution of a process.Provenance is collected by using a rule based approach which consists of a set of rules defined in XML.This framework consists of a Rule Engine. The rule engine processes raw log files to structured provence. There are three categories of rules as specified by the grammar. The match-set rules which selects provenance data based on a set of matching rules. Link Rules which specifies relationship between provenance events and Remap rules are used to specying an alias for a provenance object.

The framework consists of three major components. The Rule engine which contains XML specifications for selecting, linking and remaping provenance objects from log files. This component is integrated with the log processor. The Log processor component is involved in parsing log files with information contained in the rule engine. It converts every matching log information to a provenance graph representing entities(vertices) and their relationship(edges). The adaptor converts the structued provenance generated by the Log processor into serialized XML format which is compatible with karma a provenance service application that is emplyed for the storage and querying of the provenance data collected.

2.4.8 Towards a Universal Data Provenance Framework using Dynamic Instrumentnation

Gessiou et al propose a dyamic unstrumentation framework for data provenance collection that is universal and does not incure the overhead of kernel or source code modifications. This is achieved by using DTrace, a dynamic instumentation utility that allows for the instrumentation of user-level and kernel-level code and with no overhead cost when disabled. It allows to be included at every point in time in a rsystem runtime without overhead issues from diabled probes.

The goal is to provide an easy extension to add provenance collection to any application regardless of its size or complexity. They implement the provenance collection scheme on file system using PASS, on a database(SQLite) and a web browser(Safari). The logging component monitors all system calls for all processes involved. It contains information such as system-call arguments, return value, user id, process name, process if and timestamp. The logging components include functionalities to collect library and functional calls.They argue that applying data provenance to different layers of the software stack and not just the system level can provide varying levels of information. Provenance needs to be captured at the layer in which it is closest to the information that is needed to be captured. They collect provenance information that pertains to complex systems activities such as a web browser or a database. The information is collected from a process system-call and functional-call based on valid entry points contained in the process. It allows information from multiple entry points. DTrace dynamic instrumentation helps in discovering interesting paths in a system in which data propagates. This helps users use this information to collect a more accurate provenance data.

2.4.9 user space provenance with Sandboxing

TODO

2.4.10 Provenance for Sensors

Lim et al. 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.

2.4.11 Provenance Recording for Services

Grouth et al developed a provenance collection system, PReServ that allows software developers to integrate provenance recording into their applications mainly focused on scientific applications. They introduce P-assertion Recording protocol as a way of monitoring process documentation for Service Oriented Architecture(SOA) the standard for grid systems. PreServ is the implementation of PreP. PreP specifies how provenance can be recorded. PreServ contains a provenance store web service for recording and querying of p-assertions. P-assertions are provenance data generated by actors about the execution. It contains information about the messages sent and recieved as well as state of the message. PreServ is composed of three layers, the Message Translator which is involved with converting all messages receive via HTTP request to XML format for the provenance store layer. It also determine which plug-in handle to route incoming request, the Plug-Ins layer receives messages from the Message translator. It implements functionality that the Provenance Store component provides. This component allows third party developers to add new functionality to store provenance data without going through details of the code implementation. The backend component which stores the P-assertions. Various backend implementations could be attached to the system by the plug-in component

2.5 Related Works on Data Compression

2.5.1 Secure Data Provenance Compression Using Arithmetic Coding in Wireless Sensor Networks

The authors encode the provenance of the path in which a sensor provenance travels using arithmetic coding, a data compression algorithm. The WSN contains a base station and a newtork of nodes. The base station is incharge of verifying the integrity of the packets sent. It is also where the arithmetic coding is decoded.The provenance is encoded at the respective nodes and forwarded to the next node in the graph and it is decoded in the base station.

The BS recovers the characters the same way they were encoded. It also receives the encoded interval which it uses to decode by using the probability values saved in its storage.

For their application,provenance are refered to as the path in which data travels. Provenance refers to where a packet is originated and how it is delivered to the base station. Each node in the WSN is represented as a string of characters and are encoded using arithmetic coding. The base station receives the code and decodes.

Contains two kinds of provenance-Simple provenance in which data are generated from the leaf nodes and are forwarded to surrounding nodes towards the Base Station. In the other form of provenance, data is aggregated at an aggregation node and forwarded to a BS.

Arithmetic coding assigns intervals to a string based probabilities for the cumulative occurrence of characters contained in the string. It assign probabilities by monitoring the activities of the WSN, this is known as the training phase. In this phase the number of times a node appears in a packet is counted. Each node in the WSN is regarded as a symbol. This is used as input to the arithmetic coding algorithm which generates a variable interval for the combination of symbols.

2.6 Model for representing provenance for IoT

In order to generate provenance, we need to satisfy the who, where, how, and what of data transformations.Provenance data is represented using a provenance model which is serialized as a JSON output. This model contains information such as sensor readings, device name, and device information. This information is mapped into an appropriate provenance data model to allow for interoperability and visualization. Provenance is represents causal dependencies between data objects. There are two models for representing provenance. These models have been applied in various literature and is considered a standard for representing provenance. Some of the models for representing provenance data are outlined below:

2.6.1 Open Provenance Model(OPM)

Open provenance model is a specification that was 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 amongst 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 if it is produced by a computer system or not.

An example of such is depicted in Figure 7. This OPM graphs represents a process of driving a car. TODO: work on OPM Example

OPM is represented as a directed acyclic graph which denotes causal dependency between entities. The edges in the graph denotes dependencies with its source denoting effect and its destination denoting cause.The definitions of the edges and their relationships are denoted below:

- 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 denotes a relationship in which an entity caused the creation of another entity.
- used(Role): Denotes an entity requires the services of another entity in order to execute.
- wasTriggeredBy: This relationship represents a process that was triggered by another process
- wasDerrivedFrom: This relationship indicates that the source needs to have been generated before the destination is generated.

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: A process is an event which is taking place.A process is represented by a square object.
- Agent: Agents are actors that facilitate the execution of a process.An agent is represented by a hexagon in an OPM graph.

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.

2.6.2 Provenance Data Model(Prov-DM)

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).

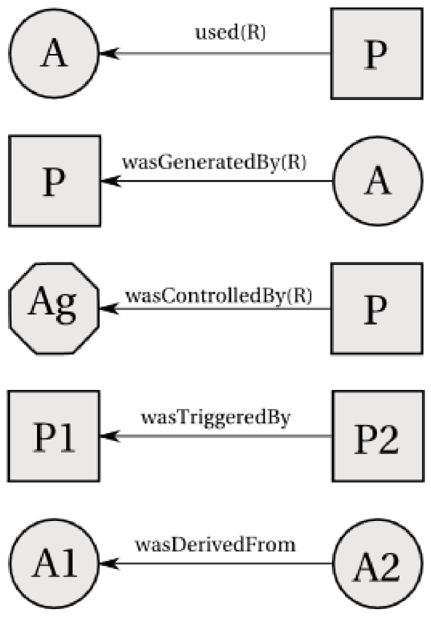

Figure 2.1: Edges and entities in OPM

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.
- 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).Entities, activities and agent are represented by oval, rectangle and hexagonal shape respectively.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 functionality 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 described 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 entities.

Since PROV-DM is built on OPM and contains easy to understand constructs of entities, we choose to use this instead of OPM.

2.6.3 PROV-JSON

PROV-JSON is a W3C specification for relaying PROV-DM specification in JSON(Javascript Object Notation) format. It contains all of the components and relationships contained in PROV-DM. It ensures interoperability between applications using PROV-DM. It also allows for easy serialization and deserialization of PROVDM mappings.JSON is lightweight, easy to parse, and allows an easy way of manipulating data programmatically. An example of PROV-JSON format is described below:

2.7 Overview of Relevant Compression techniques

2.7.1 Graph Compression

2.7.2 Lossy Compression

Lossy compression is a form of data compression in which the original data can be reconstructed with some tolerance of loosing some information contained in the original data. This enables higher compression ration than lossless compression techniques.Lossy Compression is mostly used in applications that does not have strict requirements as to loosing some information. An example of a lossless compression application can be seen in image compression software where the quality of the image is not seen by the naked eye. It is also used in speech transmission.

```
{
    "entity": {
        "e1": {
            "ex:cityName": {
                "$": "Londres",
                "lang": "fr"
            }
            ...
        }
    },
    "agent": {
        "e1": {
            "ex:employee": "1234",
            "ex:name": "Alice",
            "prov:type": {
                "$": "prov:Person",
                "type": "xsd:QName"
            }
        }
        ...
    },

    "activity": {
        "a1": {
            "prov:startTime": "2011-11-16T16:05:00",
            "prov:endTime": "2011-11-16T16:06:00",
            "ex:host": "server.example.org",
            "prov:type": {
                "$": "ex:edit",
                "type": "xsd:QName"
            }
        }
    }

}
```

Figure 2.2: PROV-JSON MODEL

2.7.3 Lossless Compression

Lossless compression is a data compression technique in which the original data is reconstructed without loss of any information from. It can be used by applications which requires that the data compressed and the original data be identical. An example of such an application that requires lossless compression is text compression. Any compression that alters the structure of the original text results to a different meaning of the text. For instance, the sentence ”I have a black cat” would have a different meaning if any information is lost from it.

Arithmetic Coding

Arithmetic coding is a form of lossless compression which encodes a stream of characters into a variable size interval between [0,1). A probability is assigned to each characters contained in the string. A cumulative probability is used to calculate the interval for the respective character.The more frequent characters are encoded with shorter codes than the less frequent characters.

Dictionary Coding

Pruning Technique

Chapter 3

Sensor Data Provenance Collection Framework for the Internet of Things

In this chapter, we define how provenance is collected and modeled along the IoT architectural stack. We also define implementation specifics of a provenance collection framework for IoT device.

3.1 IoT ProvenanceCollection Data Model

A provenance model is used to represents causal relationship between objects and it is usually modeled as a Directed Acyclic Graph(DAG).This enables for better visualization of provenance relationships and also a unified format for representing provenance data. There are two major models for modelling provenance, OPM and ProvDM. PROVDM is a predecessor and standardized version of OPM. It allows the modeling of object either physical or digital. PROVDM is chosen as the model to represent provenance for our implementation because it allows for proper representation of all of the relationships in which we envision for IoT devices. This section defines a model for relaying provenance in IoT systems which is built on top of PROVDM.

From the IoT architecture described in Section 2, we can see that data is disseminated from sensors and actuators across various layers contained in the IoT architectural stack. The provenance data produced from various sensors is collectively aggregated at the gateway layer and/or the cloud layer. We allow for provenance data to be translated to the appropriate PROVDM format at the various levels of the IoT architecture. This allows for fault tolerant processing of information even in the case of a network failure at all layers. Sensor readings are collected from devices as specified by a policy. This helps address the memory constraint problems of collecting provenance in IoT devices.Policy specification and implementation are discussed in detail in chapter 4.

Using the example of a smart home use case as described in chapter 2 of this proposal, an example of how our provenance collection framework can be applied is described below

- provenance trace data is collected from sensor from every device contained in the home(e.g thermostat, refrigerator, and smart doors).The provenance data collected is specified in the policy document which can be only be defined by the house owner.
- Each device in the smart home aggregates sensor readings and this information is passed to the gateways. The information is further analyzed and transmitted to the cloud for storage and further analysis could be conducted to derive more insights from the provenance data collected. This information is then transferred to the cloud where it is mapped to a provenance model and visualized. The cloud contains all of the aggregated provenance from the respective devices contained in the smart home. The respective devices could also map the sensor readings into provenance model and perform computations on the data.

3.1.1 Provenance-Collection Model Definition

It is assumed that provenance data is collected from the underlying IoT device using the framework outline in section 3 of this chapter below. Data is collected from sensors and actuators attached to an IoT device.The underlying construct is represented as a graph which denotes the relationships between multiple sensors and actuator readings. Since PROVDM provides a generic model for relaying provenance information, we define a more specific construct for representing provenance data in IoT devices built on top of PROV-DM. We define provenance data point(PDP) as a user transaction for storing provenance data. We use the same instance of Entity, activity and actors from PROVDM to represent data object. In the context of IoT provenance the entity, process and agent are redefined below:

- Entity:

3.1.2 Provenance Characteristics

provenance denotes the who, where and why of data transformation. These characteristics outlines essential provenance data that is collected in any system.In an IoT system, It is imperative that we collect sensor and actuator reading that satisfies the required conditions. The characteristics of data provenance are outlined in details below.

- Who: This characteristics provides information on who made modifications to a data object. Knowing the "who" characteristics made is essential because it maps identity of modification to a particular data object. An example of who in an IoT use case is a sensor device ID.

- Where: This characteristic denotes location information in which data transformation was made. This characteristic is optional since not every data modification contains location details.

- When: This characteristic denotes the time information in which data transformation occurred. This is an essential provenance characteristic. Being able to tell the time of a data transformation allows for varied use scenarios of tracing data anomalies.

- How: This characteristic denotes the process in which transformation is applied on a data object. A use case for IoT can be seen in the various operations(delete, create, open) that could be performed on a file.

3.2 Provenance-Collection System

In this section, we outline the components of our system and describe how provenance system trace is collected across the IoT device. 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 events 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. This is achieved by using a policy based approach where a user with administrative privileged is given access to create a policy which dictates what provenance data to collect. The remaining information is discarded. 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 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:

- BeagleBone Black Board. This device is a microcontrollers used to evaluate our approach. We choose the BeagleBone Board 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, BeagleBone 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

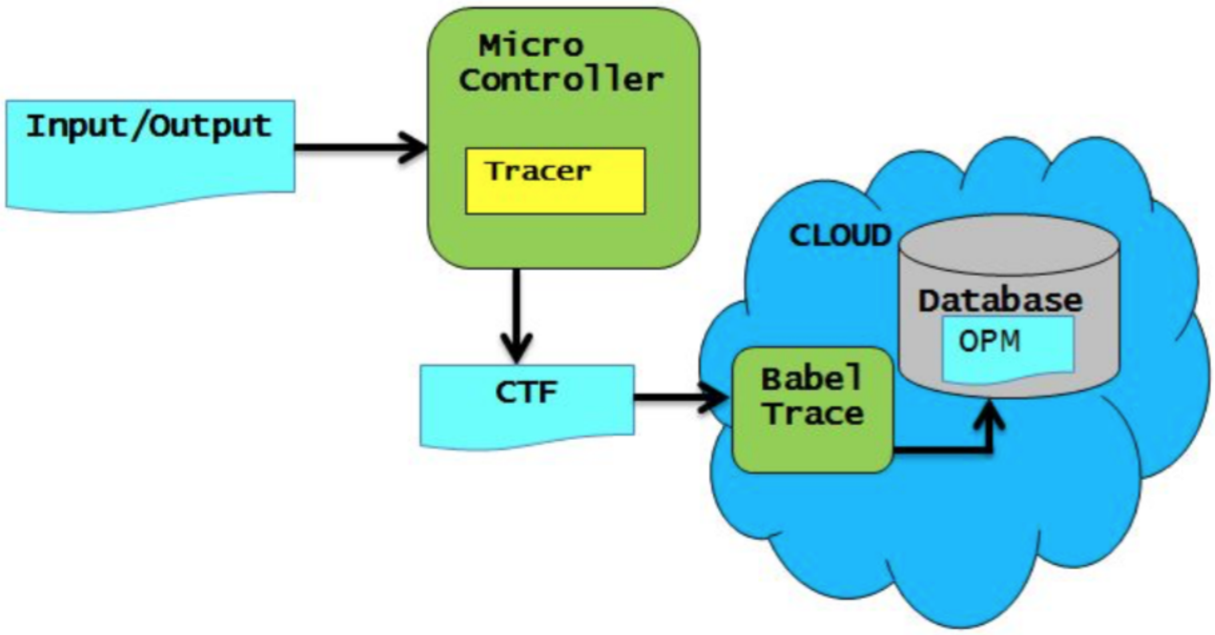

Figure 3.1: Proposed Model

operating system (RTOS) for embedded systems. This operating system is a typical RTOS that may be deployed in IoT devices.

Chapter 4

Storage Optimization Framework for Provenance data storage in IoT devices

In this chapter we define our model and algorithms for storage optimization of provenance data in IoT devices. We focus on all of the proposed techniqes such as data pruning and data compresion of graphs. We evaluate our hypothesis the proof of concept ideology.

4.1 Definitions

Definition 1 We say that the number $\tilde{x}$ represents a number x with *absolute accuracy* $\Delta > 0$ if $|x - \tilde{x}| \leq \Delta$.

Definition 2 By *absolute half-width* of the interval $\mathbf{x} = [x^-, x^+]$, we mean the smallest number $\Delta > 0$ for which every number in $\mathbf{x}$ is represented with an absolute accuracy Δ by $\tilde{x} = \dfrac{x^- + x^+}{2}$.

Proposition It can be seen that the absolute half-width of $\mathbf{x}$ is

$$\max_{x \in [x^-, x^+]} |x - \tilde{x}| = \frac{x^+ - x^-}{2}.$$

Definition 3 By *absolute width* W of an interval $\mathbf{x} = [x^-, x^+]$, we mean twice the absolute half-width of $\mathbf{x}$, i.e.,

$$W([x^-, x^+]) = x^+ - x^-.$$

Definition 4 We say that an interval $\mathbf{x}$ is Δ-*narrow in the sense of absolute accuracy* if $W(\mathbf{x}) \leq \Delta$.

Definition 5 Let $\varepsilon > 0$ be a real number, let $D \subseteq R^N$ be a closed and bounded set of positive N-dimension volume $V(D) > 0$, and let $P(x)$ be a property that is true for some points $x \in D$. We say that $P(x)$ is true for (D, ε)-*almost all* x if

$$\frac{V(\{x \in D | \neg P(x)\})}{V(D)} \leq \varepsilon.$$

Definition 6 Let $\eta > 0$ be a real number. We say that intervals

$$[r_1 - d_1, r_1 + d_1], \ldots, [r_n - d_n, r_n + d_n]$$

are η-close to intervals

$$[\tilde{x}_1 - \Delta_1, \tilde{x}_1 + \Delta_1], \ldots, [\tilde{x}_n - \Delta_n, \tilde{x}_n + \Delta_n]$$

if $|r_i - \tilde{x}_i| \leq \eta$ and $|d_i - \Delta_i| \leq \eta$ for all i.

Definition 7 Let $\mathcal{U}$ be an algorithm that solves some systems of interval linear equations, and let $\eta > 0$ be a real number. We say that an algorithm $\mathcal{U}$ is η-*exact* for the interval matrices $\mathbf{a}_{ij}$ and $\mathbf{b}_i$ if for every interval matrix $\mathbf{a}'_{ij}$ and $\mathbf{b}'_i$ that are η-close to $\mathbf{a}_{ij}$ and $\mathbf{b}_i$, the algorithm $\mathcal{U}$ returns the exact solution to the system

$$\sum_{j=1}^{n} \mathbf{a}'_{ij} x_j = \mathbf{b}'_i.$$

Definition 8 We say that an algorithm $\mathcal{U}$ is *almost always exact for narrow input intervals* if for every closed and bounded set $D \subseteq R^N (N = n \cdot m + n)$ there exist $\varepsilon > 0$, $\Delta > 0$ and $\eta > 0$ such that, for (D, ε)-almost all $\tilde{a}_{ij}$ and $\tilde{b}_i$, if all input intervals $\mathbf{a}_{ij}$ and $\mathbf{b}_i$ (containing $\tilde{a}_{ij}$ and $\tilde{b}_i$ respectively) are Δ-narrow (in the sense of absolute accuracy), then the algorithm $\mathcal{U}$ is η-exact for $\mathbf{a}_{ij}$ and $\mathbf{b}_i$.

4.2 Theorem

Theorem 1 *There exists a feasible (polynomial-time) algorithm $\mathcal{U}$ that is almost always exact for narrow input intervals.*

This theorem was proven in 1995 by Lakeyev and Kreinovich (see [**?**]).

4.3 Open Problem

Theorem 1 says that we can have a feasible algorithm that solves *almost all* narrow-interval linear equation systems, but it does not say whether we can solve *all* of them in reasonable time. Thus, there still remains an open question:

> *Can a feasible algorithm be developed for the general problem of solving systems of linear equations with narrow-interval coefficients?*

The answer to this open question is the main concern of this thesis.

We will show that the problem of solving all narrow-interval linear equation systems is NP-hard; moreover, we will show that the problem is NP-hard not only for intervals that are narrow in the sense of absolute accuracy, but also in the sense of relative accuracy.

Chapter 5

Concluding Remarks

5.1 Significance of the Result

In this research, we present a provenance aware system that collects provenance information for IoT devices. This information can be beneficial in correcting and mitigating current and future attacks. We plan to continue refining our prototype, construct a model to guide data pruning, and create a taxonomy for secure data provenance in IoT systems that can inform our model.

endeavor.

5.2 Future Work

- Provenance collection raises privacy issues.How do we ensure that the vast amount of data collected is not invasive to the privacy of the use and also is not used as a tool to perform malicious attacks.