SECURE DATA PROVENANCE FOR IOT DEVICES

EBELECHUKWU NWAFOR

Department of Computer Science

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. 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.

Table of Contents

List of Figures

Chapter 1

Introduction

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 reproduced, in buisness to determine the workflow of processes, 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 tries to request 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..

TODO: Create transition from Data provenance to 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. 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 which enables backtracking in an event of a malicious attack.

1.1 Motivation

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 IT 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 an malicious intruder can remotely view live video feeds from the device. 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. Most of the 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 IoT. Data provenance ensures authenticity, integrity and transparency between information disseminated across an IoT network. Security applications such as intrusion detection, digital forensics, and access control can be further enhanced by incorporating data provenance 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. 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 and also investigate its applications in providing intrusion detection 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.

1.2 Provenance-Aware IoT Device Use Case

Consider a smart home which contains interconnected devices such as a thermostat connected to the internet. This device automatically detects and sets the temperature of a room based on previous information of your desired temperature settings, a smart lock system that can be controlled remotely, a home security camera motoring system, A smart fridge which sends a reminder when food produce are low. A malicous intruder tries 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 us from future or current malicious attacks.

1.3 Research Questions

collecting provenance data in IoT devices raises some key research issues. Some of these 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 data on memory contrained devices.
- How do we model provenance data for IoT devices? Do we use the OPM approach or create a specialized taxonomy for modeling provenance data.

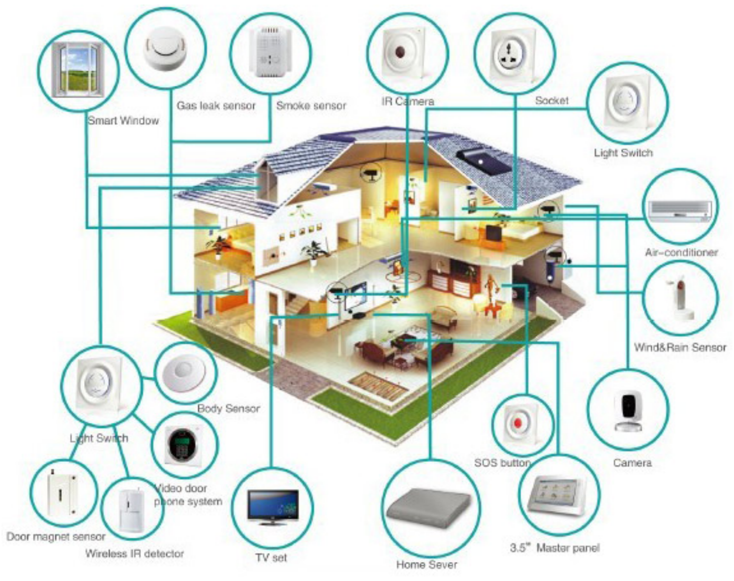

Figure 1.1: Place holder for use case Diagram

- 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 proposes 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.
- A novel framework for data storage on embedded systems.This addresses the storage issues of the memory constrained IoT devices.
- A framework for providing intrusion detection using provenance data in an IoT eco 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 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 for file systems, data compression techniques for data provenance, and provenance-aware intrusion detection systems.

2.1 Overview of Provenance Aware 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 and system level provenance but so far little attention has been given to provenance in IoT devices. Some the previous work done on data provenance collection systems are outlined below:

2.1.1 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.

2.1.2 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.

2.1.3 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.

2.1.4 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.

2.1.5 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.

2.1.6 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.

2.1.7 user space provenance with Sandboxing

TODO

2.1.8 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.

2.2 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.

2.2.1 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 enity requires the services of another enity 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.

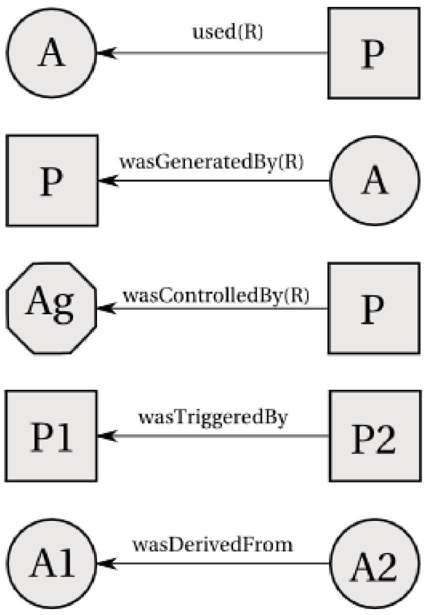

Figure 2.1: Edges and entities in OPM

2.2.2 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). 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 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 entities.

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

2.2.3 PROV-JSON

PROV-JSON is a W3C standard for representing PROV-DM in json 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 PROV-DM mappings.JSON format is lightweight, easy to parse, and allows an easy way of manipulating data programmatically. An example of PROV-JSON format is described below:

```
{
    "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

2.3 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

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 to the Intrusion Detection System which deals with detecting malicious attacks on IoT Devices.

3.1 ProvenanceCollection Model

TODO: Talks about how we use PROVDM and what kinds of provenance we are looking to store

3.2 Provenance-Collection System

In this section, we outline the components of our system and describe how provenance information is collected across the IoT ecosystem. 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.

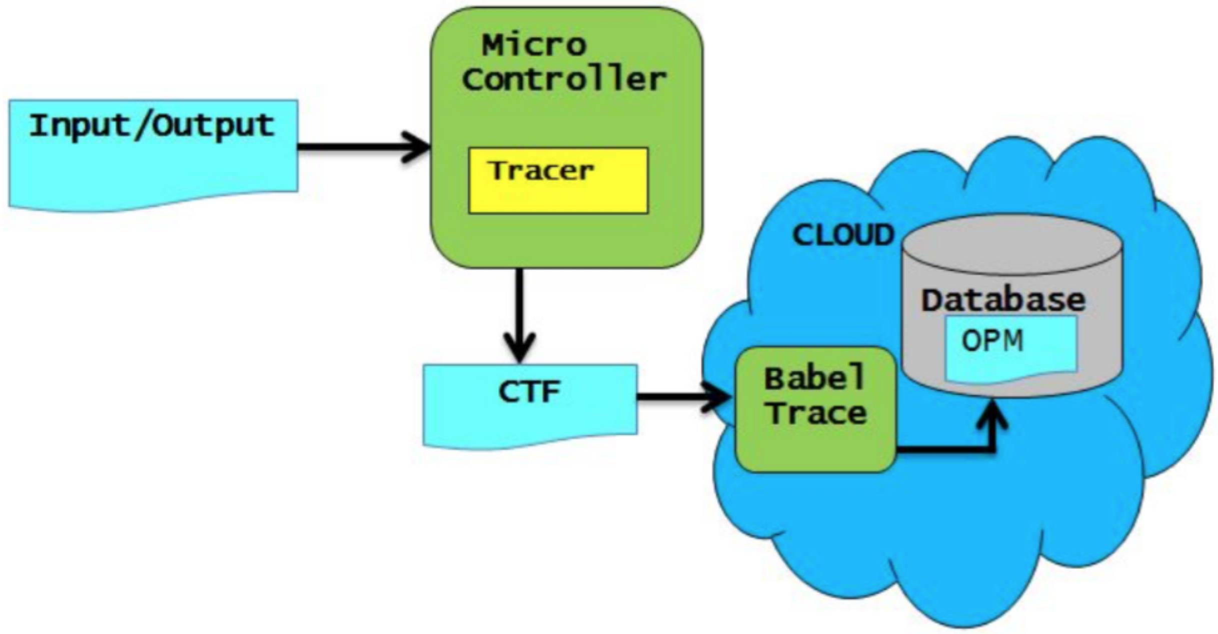

Figure 3.1: Proposed Model

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:

- 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.

3.3 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

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.

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.