University of South Carolina
Scholar Commons

Theses and Dissertations

Spring 2023

Semantics-Based Data Security Models

Theppatorn Rhujittawiwat

Follow this and additional works at: https://scholarcommons.sc.edu/etd

 Part of the Computer Sciences Commons, and the Engineering Commons

Recommended Citation

Rhujittawiwat, T.(2023). *Semantics-Based Data Security Models.* (Doctoral dissertation). Retrieved from https://scholarcommons.sc.edu/etd/7349

This Open Access Dissertation is brought to you by Scholar Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact digres@mailbox.sc.edu.

Semantics-based Data Security Models

by

Theppatorn Rhujittawiwat

Bachelor of Engineering
Chulalongkorn University 2006

Master of Engineering
Chulalongkorn University 2010

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2023

Accepted by:

Csilla Farkas, Major Professor

Shankar Banik, Committee Member

Stephen Fenner, Committee Member

John Rose, Committee Member

Qiang Zeng, Committee Member

Cheryl L. Addy, Interim Vice Provost and Dean of the Graduate School

© Copyright by Theppatorn Rhujittawiwat, 2023
All Rights Reserved.

Abstract

In this dissertation, we studied how an adversary could attack databases and how the system could prevent or recover from such an attack. Our motivation to improve the current security capabilities of database management systems. We provided better recovery capabilities of database management systems by incorporating data provenance. We also expand our study to express security and privacy needs of data in the Internet of Things (IoT) environments such as a smart home environment. For this, we proposed a stream data security model to theoretically represent the data in the IoT network. We built a dynamic authorization model on our context-aware architecture and stream data model. We demonstrated the capabilities of our dynamic security policy to address security needs due to the changes in the context. Furthermore, we demonstrated the applicability of our approach by implementing our framework in a smart home IoT network. For our proof-of-concept implementation, we used a commercial and open-source home automaton software. Our approach to improve the system is expanding it by incorporating third party applications, such as a dynamic access control engine. We aim to incorporate a logic reasoner into smart home automaton to provide situation-aware capabilities to the system in this study.

The main research questions in this dissertation are as follows:

1. **How can we improve the efficiency of database recovery after a malicious transaction attack?**

   We proposed algorithms to reduce the downtime of the database during the recovery process. The traditional approach for recovery is to execute all non-malicious transactions from a consistent rollback point. However, this approach

is inefficient. First, the database will be unavailable until the restoration is finished. Second, all non-malicious transactions that are committed after the rollback state need to be re-executed. The intuition for our approach is to re-execute partial transactions, i.e., only the operations that were affected by the malicious transactions.

2. **How to support context-aware and dynamic security policy for stream data model?**

   We proposed a semantics-based authorization model for stream data. We demonstrate that current authorization models are insufficient to provide dynamic access control for emerging technologies, such as the Internet of Thing. We propose an authorization model using ontologies and rules to express security requirements for stream data. Our model supports secure interoperation and is independent from the data syntax. We propose security object patterns to express access control needs. These patterns are associated with the ontological concepts corresponding to the database schema. Data instances inherit the security protection needs from these ontological concepts. We aim to support dynamic security policies due to changes in the context. Such changes may result in new security assignments to the protected objects. We model contextual changes as finite state transition automata.

3. **How can we support situation-aware decision making for a smart home environment?**

   We proposde an architecture to enable third party applications within the popular home automation environment. Our claim is that the functionality of these popular smart platforms can be improved by enabling third party applications, such as situation-aware decision making. We show how third-party applications may support home automation systems to respond to complex environmental

changes. We implemented a proof-of-concept prototype system to demonstrate how home automation applications can interact with logic reasoners to support dynamic system policies. Our implementation was built using an open source Home Assistant platform and Protégé-OWL reasoner. We propose threat modeling, develop the use and misuse cases for a smart home environment, and support for data exchange between the home automation system and the reasoner.

Table of Contents

Chapter 1

Introduction

1.1 Research Motivation

Database has been a research topic for decades. In the past, the research were focused on performance which produced our current efficient database system which has been relied on by every organizations. Information is always a valuable resource for every organizations and database is where those valuable information is stored. Those information is valuable for both organization and attacker. Losing the control over database by an attack can create devastating loss in both direct and indirect costs. From IBM's annual report, the direct cost of a single data breach is $3.86 millions on average in 2020[1]. Moreover, the data breach can also damage the reputation of the organization which create an indirect cost. To protect the valuable resource and limit the cost from attacks, the security capabilities are needed. The core security objectives consist of confidentiality, integrity and availability (CIA). CIA principle has been used to address security requirements and design security policy for database. Confidentiality protects data in database from being disclosed to unauthorized users. Integrity ensures that data in database is accurate and attacker can not temper it. Availability guarantees that authorized users can access data when needed. To achieve CIA objectives, security mechanism must be developed. There are 3 security mechanism types: prevention, detection, and recovery. Access control is the most well-known prevention mechanisms to protect confidentiality of database. There are many access control models with different strengths and weaknesses such

as Discretionary access control (DAC), Mandatory access control (MAC), and Role-based access control (RBAC). Detection mechanism is relied on monitoring system to detect known attacks or anomalies. Also, in the case we fail to prevent the attack, recovery mechanism is required to return database to the consistent state such as rolling the data back to the state before being attacked.

In this dissertation, we focus on 3 research questions as follows. Does the current database recovery method sufficient? Does the stream data authorization model sufficient? How to provide IoT specific security requirement?.

1.2 Database Recovery

One of attacks specific to database is the malicious transaction. It is an attack which attacker commit fraud transactions to the database. Malicious transaction attacks are often detected at a later time, after malicious transaction as well as other transactions dependent of the malicious one are committed. The traditional recovery method is rollback the database to a consistent backup point then re-execute all non-malicious transactions again. However, this causes the database to be unavailable until the restoration is completed. Moreover, we have to re-execute transactions which were not affected by the malicious attack. To improve recovery speed, identify and re-execute only affected transactions is one of approaches. Many researchers proposed approaches to determine affected transactions. The approaches presented by Ammann et al. [2], Liu et al. [3], Panda et al. [4] have used transaction dependency to identify compromised transactions. The works of Kim et al. [5], and Chandra et al. [6] have used input comparison to determine unaffected transactions. Input comparison approaches provide simpler and faster solutions than the dependency approaches. However, transaction dependency approaches provide greater details about which transactions are affected and a higher possibility to salvage good transactions than the input comparison approaches. Therefore, there is a possibility that the dependency-

based approaches may provide better overall performance than input comparison-based approaches when it can salvage many good transactions. In this case, the reduced re-execution cost outweighs the slower determination process.

1.3 Stream Data Security

Stream data is a data model for continuous sequential data processing. Unlike conventional databases, the data will be processed once it arrives to the system with continuous queries. The current stream data security models are insufficient. They cannot provide flexible, fine granularity, and adaptive security policies. They provide security capabilities for stream data by using end-to-end encryption, in which all communications between data sources and the system are encrypted. This technique can provide CIA over the transmissions. Despite how efficient this approach is, it lacks granularity and a multi-level security capability. This approach cannot provide needed security capabilities to protect privacy or provide multi-level security policies because there are only two choices between encrypting the whole transmission or nothing at all. The limitation of all or nothing protection doesn't reflect the needed of multiple security levels in data stream contains critical personal information such as medical monitoring devices, social media, or other online activity monitors. Such limitations were addressed by researchers, so syntax-based security approaches have been introduced. Syntax-based approaches can provide multi-level security by specifying the security level of attributes in stream data. However, stream data tuple lacks metadata of its data. The attributes in stream data tuple are recognized by its position in tuple. This limitation introduces a security risk in the case that attribute positions are swapped.

1.4 Specific Need of Internet of Things

One of the area which heavily uses stream data is Internet of Things (IoT). IoT is the network of devices which connect and exchange data over the internet. Many devices, such as portable medical monitors, observe and send sensitive information of users. Privacy is the major concern of patients' medical information. Only patients and their authorized physicians should be able to access the information. However, that information is crucial during emergencies and first responders must be able to acquire it as fast as possible. For example, the GPS location of patients must be protected and no other people except the patient himself should be able to access it. However, his location must be provided to first responders in emergency situations so that he can be rescued in time. The first problem that has to be addressed is the definition of emergency situation. It cannot be defined by any specific value of attributes. Each patient is different; the same value of a heart rate can mean a normal situation for one patient but an emergency situation for another. The heart rate of the same person can also differ depending on his current activity. For example, if a patient's heart rate is slowly rising, it can be considered normal because he may currently work out. However, the same value of heart rate can be an emergency if it suddenly rises. Additionally, even if the heart rate is in the normal range, the patient can currently suffer from a heart attack if the rate is unstable. In this case, the value of any single tuple cannot present the situation at all. This problem shows that we have to understand the context of information corresponding to each individual to accurately classify the situation.

The semantic-based approaches can provide capabilities to express security needs in a context-aware manner. The semantic-based approaches allow to defines security policies that can adopt according to the context of the access. Moreover, such policies support interpolation and secured sharing of data.

1.5 Organization of the Dissertation

The organization of the proposal is as follows: Chapter 1 discusses the problems and motivating situations which drive the direction of research. Chapter 2 illustrates motivating examples driving this research. Chapter 3 describes the general knowledge and works which are related to this research. Chapter 4 discusses the problem, defines the formal definitions for database recovery, and presents a solution to recover database from a specific malicious attack. 5 discusses the home automaton environment and demonstrate a proof-of-concept improvement with logic reasoners. Chapter 6 defines the stream data model, security approach and presents an approach to solve a stream data security problem.

Chapter 2

Motivation

In this section we present motivating examples to demonstrate the shortcomings of current stream data authorization models.

In stream data, only a part of data can be processed at any given time via continuous queries. The data will be processed with the same queries over and over again. Different order of the stream tuples will result in different results. Similarly, missing tuples will impact the results of the continuous queries. Incorrect ordering can be detected in case the timestamps are correct. However, malicious users may modify the timestamp to achieve a desired evaluation schedule. Moreover, the completeness of the stream may be difficult to detect, if there is no predetermined schedule of the arrival of the stream tuples. Next, we demonstrate some of the scenarios that may occur in the presence of malicious users.

Health monitoring sensors broadcast vital signs, such as body temperature, pulse rate, respiration rate, blood pressure, and pulse oximetry. These vital signs are frequently observed by medical staffs during hospital admission. The frequency can vary depending on the condition of patients. For example, pulse oximetry measures the oxygen saturation in blood. The normal value is between 95-100%. The oxygen saturation value below 90% is referred as hypoxemia. Patients generally show signs of mental impairment at oxygen saturation below 85% and completely lose consciousness at below 75%[7]. The presence of hypoxemia is generally caused by underlying conditions of the heart or lungs, which may require emergency care. The lack of oxygen from hypoxemia can lead to a wide range of complications from minor headaches

to respiratory failure, which can lead to death or permanent brain damage in a matter of minutes without an immediate response. Oxygen saturation level is generally observed every 4 hours in admitted patient care but constantly observed in critically ill patient cases.

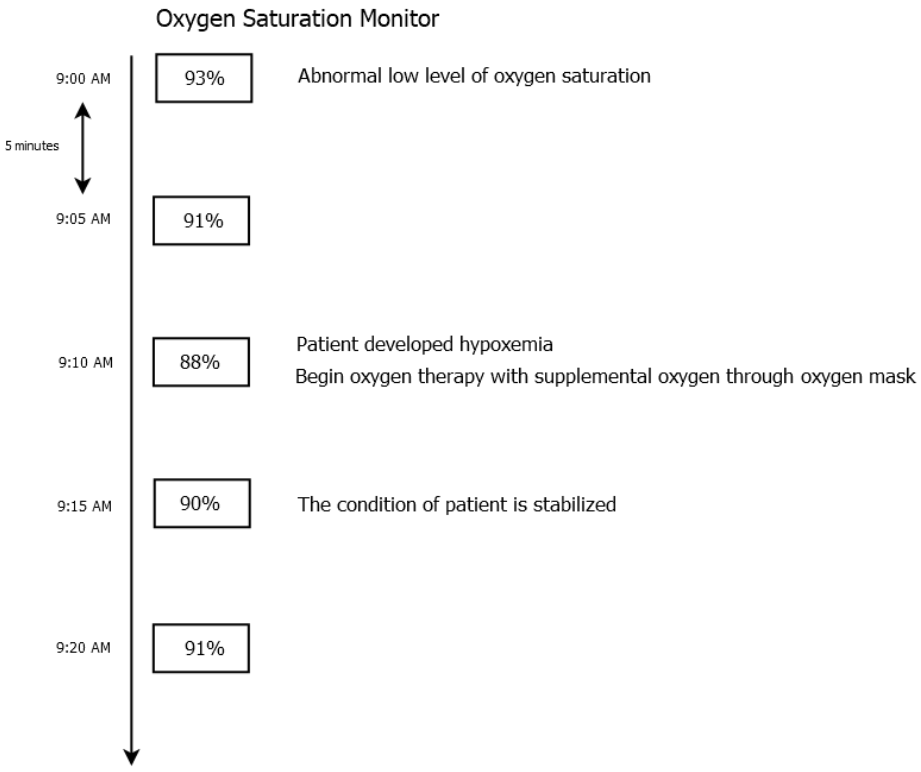

Figure 2.1 Oxygen saturation monitoring results in an emergency patient

Figure 2.1 shows an example of oxygen saturation monitoring results. In this example, as soon as hypoxemia was detected treatment was initiated. The timely treatment prevented long-term complications for the patient.

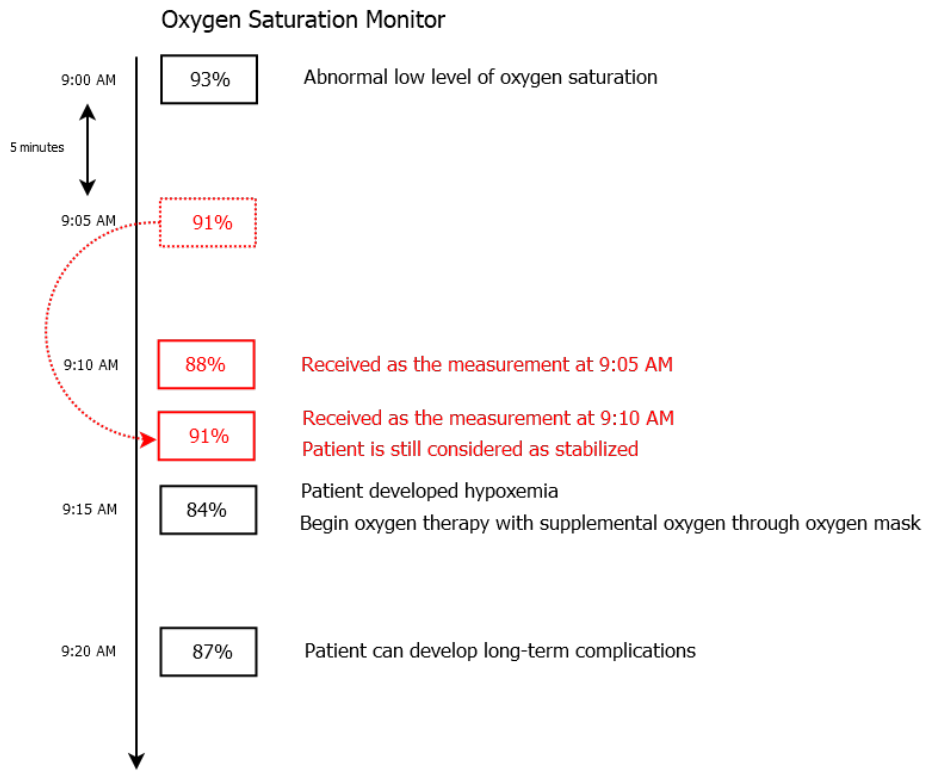

Figure 2.2 Incorrect order of monitoring results

Figure 2.2 shows an example of the same situation as Figure 2.1 with an incorrect ordering. The medical staff missed the early low oxygen saturation sign at 9:10 AM and didn't initiate the emergency procedure. To avoid penalty, they switched the timestamp of the measurements, claiming that the readings were inconclusive. The patient did not receive appropriate treatment in time which lead to long-term complications.

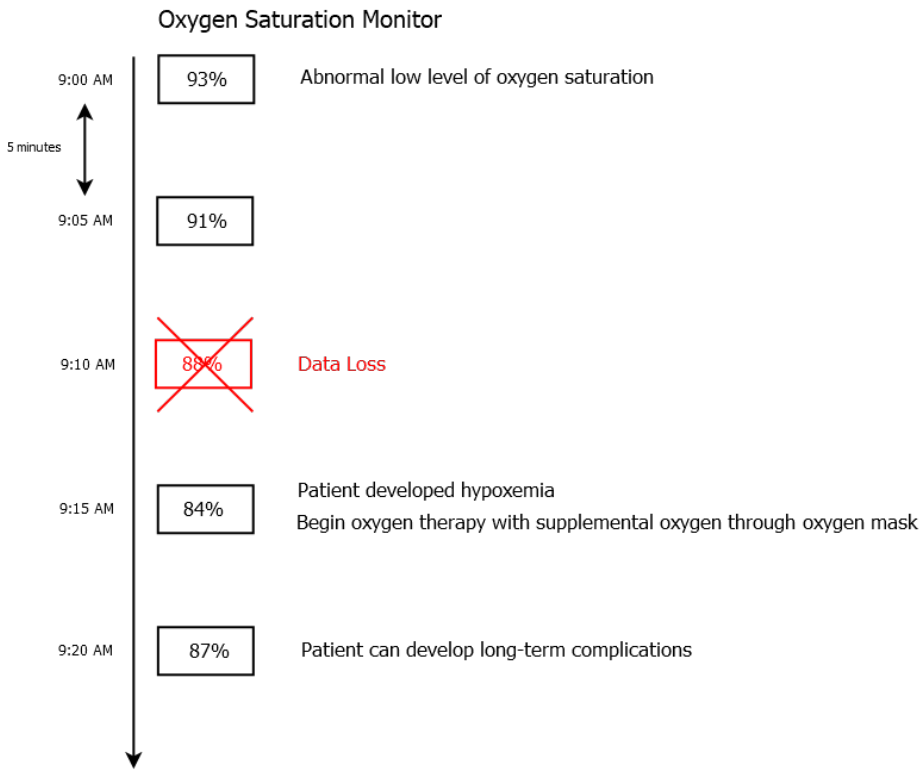

Figure 2.3 Incomplete monitoring results

Figure 2.3 shows an example of the same situation as Figure 2.1 but demonstrating the problem of incomplete data. In this case, the reading at 9:10 AM is missing or intentionally removed. The missing data resulted in late diagnosis, thus the patient did not receive appropriate treatment in time. This lead to long-term complications.

Chapter 3

Related Work

3.1 Database Recovery

Data provenance is the historic records of how the data was created and processed over time, it can be used to support many analysis and forensic activities. For example, analyzing how reliable of the new data by comparing it to the past records as in the work of He et al.[8], detecting the source of data leakages by tracking the accesses of data as in the work of Zhang et al.[9], detecting the malicious activities by analyzing the signatures of entities as in the work of Backes et al.[10], or providing information to recreate the data as in the work of Foster et al. [11]. As illustrated by earlier examples, data provenance can be used to support many activities such as detecting unreliable data, system weaknesses, malicious activities, and data recreation. It can be used to provide data security by detecting those malicious attacks before any damage has been done to the data and also recover data from any successful attacks. An effective data recovery can reduce the cost to return a system to a reliable state after malicious attacks. We aim to incorporate data provenance to improve data recovery by minimize the data recreation process after malicious attacks.

3.2 Access Control Model

To provide security capabilities, access control models were introduced. Each type of access control have pro and con as follows.

3.2.1 Mandatory Access Control (MAC)

It is an access control model which the system assigns security levels of subjects and objects[12][13]. The user(subject) can access a file(object) only when the user has at least the same security level as that file. For example, giving security levels; "top secret" > "secret" > "public". An user that have "secret" clearance can access files with "public" or " secret" security label but not a file with "top secret" security label.

MAC provides intuitive and centralize security policies. Only the system administrator may access or alter security control. This reduces the chance having vulnerabilities created by human errors from untrained users. However, the system also requires constant attention from administrators to assign or alter security levels for each subjects and objects. It also doesn't scale well with a huge organization because security levels must be assigned every time a file is created or an user's position is changed.

3.2.2 Discretionary Access Control (DAC)

It is an access control model which the owners of each files decide who can access their files[13]. Each objects in DAC are assumed to have owner who control the permission to access those objects. The owner has full control over his files and can give other users the permission to access his files. For example, Alice creates file A so she is the owner of it. She can give the permission to access file A to Bob. Bob will be able to access file A but Claire cannot access it.

DAC provides simple and flexible security policies. Each users can decide the permission of their own files as soon as the files are created. This requires less attention from the system administrator to manage the system. It's also user-friendly since users can manage their own files without having to contact the system administrator. However, the system will be more likely to have vulnerabilities because of human errors from users who lack security awareness.

3.2.3 Role-based Access Control (RBAC)

It is an access control model which the system assigns security levels of roles and objects[12][13]. This access control model is closely related to MAC. Instead of assigning security levels to each users, users will be assigned roles which contain permissions. For example, a role A has permissions to access file a. If role A is assigned to Ann and Ben, they will be able to access file a.

RBAC provides easy-to-maintain security policies for huge organizations. The huge organizations usually contain many users with the same job positions. Their privileges should be the same so grouping them together will greatly reduce the size of policy compared to MAC. However, if each users have unique permissions or cannot be grouped together into roles, RBAC will have more cost than MAC because an extra step from assigning permissions to roles instead of assigning permissions directly to users.

3.2.4 Context-Aware Access Control

It is an access control which the permission is decided based on the context[14][15]. It usually be used as another layer on the top of other access control models. The context can be many kinds of contextual knowledge such as user identity, location, time, or situation. For example, the medical records of patients are confidential in a normal situation and should be disclosed to those patients' physician only. But if a patient is in an emergency situation, the first responder should be allowed to access that patient's medical record.

Context-Aware Access Control proposes a flexible and dynamic access control which can adapt to situations. It can provide a responsive access control which decide the appropriate permission to users. However, it requires much higher cost and knowledge to design the dynamic policy more than access controls with static policies.

3.3 Stream Data Security

Current work on stream data security focuses on confidentiality[16][17][18]. The first priority of stream data design is performance. To provide security capabilities with the least effect on performance, the major methods providing confidentiality are based on end-to-end cryptography[19][20][21][22][23]. However, end-to-end encryption techniques provide confidentiality over only the whole package. They are unable to provide fine granularity authorization, and cannot enforce advanced security requirements, such as role-based, lattice-based access control, or dynamic security policies. Therefore, such requirements must be enforced by the receiving applications. Many researchers introduce syntax-based security techniques for streaming data. The security requirements are assigned to each attribute by their positions in the stream tuple. Nehme et al.[24] proposed a technique called security punctuation to provide efficient access control for stream data. A security punctuation technique focuses on providing privacy control for the data source side. The current security policy is sent from the data source and specifies the security labels of attributes in the data tuple. This approach allows the data source side to decide security needs which are suitable for privacy sensitive data domains such as medical information.

Semantic approaches have been used in stream data since the field was established. However, their main focus is how to interpret data and use it for data analytics[25][26]. Sabelfeld et al.[27] demonstrated that semantic security policies can provide precise policies. They can be used to express and counter different kinds of attacks including attacks based on timing and probability. Many group of researchers have introduced semantic-based access controls. There are two major approaches to incorporate semantics to security policy. The first one is the subject-based approach[28]. It uses semantics to decide if the user is allowed to access the data or not. Even if the user has legitimate credentials, the system may disallow him to access if he accesses it from a suspicious place or a suspicious time. The second one is the context-based

approach[29][30][31]. In this approach, whether the access is granted or not is based on action. The semantic is built on the relation between user, data and the action that will be performed. SeCoMan is one of the context-based approaches which provides privacy control over the IoT network[29]. It has been used over stream data but the technique is not defined on a stream data model. This approach is built on a relational model.

Stream data requires dynamic security policies because it is time sensitive. The same attribute in stream data may require different security levels in different times such as healthcare information in emergency. Lu et al.[32] proposed a framework to provide privacy on mobile health monitoring devices. This approach uses mathematical models on monitored information to detect emergency situation and release information to qualified helpers. Other mathematical models to express emergency situation were also proposed by many researchers[33][34][35]. Also, the network is a crucial part of stream data. This make stream data inherits some problems from the network such as out-of-order message arrival. The delay from the network can affect the timing of incoming messages. Attackers can masquerade attacks as regular network delays. The hypothetical reordering attack can break security on stream data as follows. Security level of each attribute may be differ depending on time. It is possible for attackers to change the security level by delaying messages or swapping the message order. The integrity of message order is required to counter such attack.

3.4 IoT Data Security

Privacy is the major security concern related to commercial IoT devices. For example, police officers can observe any camera footage from Ring™Doorbell without permission from users[36]. This practice improves public security while completely sacrifices each individual's privacy. Not all footage will contribute to public security, sacrifice all privacy is not a suitable cost. Based on many studies[37][38][39], most

IoT devices in the market only focus on having strong secure connection between devices and the server. They lack a good access control policy to protect sensitive information. The information can be exposed to the third-party that operates the service such as Amazon Web Services (AWS)[39]. Many devices even have outright flaws which allow attackers to directly access the information from the server through its API[38]. To improve this practice, giving away the appropriate footage is the key. The home owners may feel comfortable to give away their door footage when they are away from home but not when their kids playing in front of it. The suitable security policy is depending on situations. Due to the fact that every homes are different, it's impossible to come up with the single policy to handle all footage. We propose our dynamic security policies as a solution for this problem. Since the suitable policy is depending on the situations, the way to express situation is crucial. Semantic is required to express the situations.

Many researchers proposed some forms of dynamic security policies as the solution to protect privacy in normal situations while does not put users at risk in emergency situations. For example, consider the case of medical monitoring devices in emergency situations, current approaches still use static security policies which patients have to give up their privacy so experts can observe those information[40] or use dynamic security policies based on mathematical models which is complicate in practical. There are approaches to provide privacy while working around current static model such as using statistic model to detect anomaly in data and grant access to all data if they are requested during the present of anomaly[16]. We know that semantic-based security policies can provide needed dynamic capabilities to protect privacy without hinder emergency management. An architecture which incorporates semantic-based security policy into a stream data model has the potential to provide dynamic security policy which provide functionalities in emergency without sacrifice privacy.

Chapter 4

Database Recovery

Data provenance is a term referred to a record trail including the origin of a piece of data and a track explaining how and why the data got to the current state. For example, in Wikipedia, the view history provides a track record of who created the page, who edited the page, when it was edited, and why it was edited. Data provenance can provide information indicating the reliability of the data. Data reliability is crucial for data security. Processing unreliable data can make a significant impact. For example, assume that a bank processes a fake money transfer transaction. This can create a loss for both customers and the bank in terms of capital loss and reputation damage.

Additionally, in a database, The damage from a malicious transaction committing unreliable data can also spread to other sections of the database which used corrupted data to commit new data. Traditional data recovery method is rolling back the database to a consistent backup point then re-execute all non-malicious transactions which were committed after the backup point. However, this causes the database to be unavailable until the restoration is completed. We propose a provenance-based approach to efficiently recover database from malicious transactions.

4.1 Problem Specification

Malicious transactions can potentially be committed before detected thus propagating their damage across the database. The problem of malicious transaction recovery has 2 major parts: identifying the scope of damage and repairing the damage. The

damage from a malicious transaction can spread to other transactions. For example, if a malicious transaction T_i updates the value of attribute A_1, every transaction that uses this A_1 will be affected by T_i and will need to be corrected. Furthermore, if an affected non-malicious transaction T_j uses A_1 to update A_2, then A_2 will need to be corrected therefore spreading the damage and the scope of the correction. On the other hand, if a transaction T_k which does not use A_1 can be indirectly affected if it uses A_2 from T_j.

Our goal is to reduce the cost of locating every affected transaction, limit the time for spreading of infections, and reduce the recovery work.

4.2 Preliminaries

In this section, we present our data model and corresponding definitions.

Definition 1. A relation schema R is denoted by $R(A_1, ..., A_n)$, where R is the name of the relation and $A_i | i \in \{1, ..., n\}$ is an attribute name.

Definition 2. An attribute value pair is denoted by (A, v), where v is the value of an attribute A such that v is in the domain of attribute A, i.e., $v \in Dom(A)$.

Definition 3. A relation instance r is denoted by r(R), where $R(A_1, ..., A_n)$ is the relation schema of r. r is a set of tuples $r = \{t_1, ..., t_m\}$, where each tuple is a set of n attribute value pair $t = \{(A_1, v_1), ..., (A_n, v_n)\}$ such that each value $v_i \in Dom(A_i)$. We also represent a tuple as an order list of n values $t =< v_1, ..., v_n >$ such that each value $v_i \in Dom(A_i)$.

Definition 4. An attribute-value assignment is denoted by

$$t = \{(A_1, v_1), .., (A_j, v_j), .., (A_n, v_n)\}$$
$$\longrightarrow t' = \{(A_1, v_1), .., (A_j \rightarrow exp), .., (A_n, v_n)\} \quad (4.1)$$

for some attribute value pair of $A_j \in R, (A_j, v_j)$ will be set to (A_j, exp) such that exp is either;

1. A constant value $v'_j \in Dom(A_j)$.
2. An attribute $A_k \in R$ with value v_k, such that $v_k \in Dom(A_k)$.
3. An arithmetic expression which includes constant values, attribute names in R, and/or arithmetic operators.

The representations of value assignment mentioned above are explained using the following transaction on a relation r with schema $R(A_1, A_2, A_3)$.

Example 4.1 Transaction T_1

UPDATE r

SET

$A_1 = 10$,

$A_2 = A_1$,

$A_3 = 2A_1 + 20$;

In the above transaction, the attribute A_1 is assigned by a constant value 10. The attribute A_2 is assigned the value of A_1. The attribute A_3 is assigned by arithmetic expression containing attribute A_1, constant values and arithmetic operator, the value of A_3 will be the result derived from the expression.

Definition 5. An attribute-value conditional assignment is denoted by

$$t\{(A_1, v_1), .., (A_j, v_j), .., (A_n, v_n)\} \longrightarrow t'\{(A_1, v_1), .., (A_j \rightarrow exp, c), .., (A_n, v_n)\} \quad (4.2)$$

for some attribute value pair of $A_j \in R, (A_j, v_j)$ will be set to $(A_j, \{exp, c\})$ when condition c is satisfied, such that c is a boolean expression which may include truth values, boolean variables, arithmetic expressions, relational operators, and/or boolean operators.

The multiple attributes assignment $t\{(A_1, v_1), .., (A_n, v_n)\}$
$\longrightarrow t'\{(A_1, v_1), .., (A_i \rightarrow exp_i, c_i), ..., (A_j \rightarrow exp_j, c_j), .., (A_n, v_n)\}$ is interpreted as a sequence of update to attributes $A_i, ..., A_j$

The representations of value conditional assignment mentioned above are explained using the following transaction on a relation r with schema $R(A_1, A_2, A_3)$.

Example 4.2 Conditional Update Transaction T_2

UPDATE r
SET A_1 = **CASE**
WHEN $A_1 > 10$ **THEN** $A_1 = 10$
ELSE NULL
END,
SET A_2 = **CASE**
WHEN $A_1 < 10$ **AND** $A_2 > A_1$ **THEN** $A_2 = A_1$
ELSE NULL
END

In the above transaction, the attribute A_1 is assigned by a constant value 10 only if the boolean expression $A_1 > 10$ is satisfied. The attribute A_2 is assigned by attribute A_1 and A_2 will end up be assigned by the value of A_1 only if the boolean expression $A_1 < 10 \wedge A_2 > A_1$ is satisfied.

Definition 6 (Data Dependency Pair)**.** A data dependency pair is denoted by

$$Dp_{A_i} = (A_i, \{A_1, ..., A_n\}) \tag{4.3}$$

We also represent it as

$$Dp_{A_i} = (A_i, Set_{A_i}) \tag{4.4}$$

Where an attribute A_i depends on a set of attributes $Set_{A_i} = \{A_1, ..., A_n\}$ such that $A_i \in R, i \in \{1, ..., n\}$. The data dependency is transitive, i.e., if (A_i, A_j) and (A_j, A_k) then (A_i, A_k). However, it is not reflexive, not symmetric.

For a transaction T, $Dp_T = \{Dp_{A_1}, ..., Dp_{A_n}\}$ denotes all the dependency pairs $Dp_{A_1}, ..., Dp_{A_n}$ such that $A_i | i \in \{1, ..., n\}$ were modified by T.

Definition 7 (Last Assignment Table)**.** A last assignment table L has the schema

$$L(A, T) \tag{4.5}$$

Where A is the column with domain $\{A_1, ..., A_n\} \in R$ and T is the column for representing the last transaction that updated attribute A_i, the domain of T is the identities of the transaction.

Definition 8 (Transaction Dependency)**.** Transaction T_j depends on transaction T_i iff there is an attribute A_j that is modified by T_j and A_j depends on an attribute A_i that was updated by T_i. The transaction dependency is transitive, i.e., transaction T_k also depends on transaction T_i if there is an attribute A_k that is modified by T_k and A_k depends on an attribute A_j that was updated by T_j. Where A_j depends on an attribute A_i that was updated by T_i.

We also say that transaction T_i affects transaction T_j if transaction T_j depends on transaction T_i

Definition 9. Data dependency record of provenance for transaction T is denoted by

$$P_T =< ID_T, ts, R\_s, W\_s, Dp > \tag{4.6}$$

Where ts is a timestamp when transaction T committed, ID_T is the id of transaction T_i, R_s is a set of read attributes, W_s is a set of modified attributes, Dp is the data dependency set of T

Assuming a transaction can read and write each attribute only once, we can recreate the partial order of T as a directed acyclic graph (DAG) G_T from a record of T in P_T by doing the following steps:

1. For each attribute A_i in R_s, create a node $r[A_i]$.
2. For each attribute A_j in W_s, create a node $w[A_j]$ and read Dp_{A_j} in Dp.
3. Create a directed edge from each $r[A_i]$ where A_i is in Set_{A_j} to $w[A_j]$.
4. If any subgraph G'_T has only one node with no edge, create a directed edge from the node in G'_T to another node outside of G'_T that has no directed edge pointing to it.
5. If any subgraph G'_T contains an edge but does not contain any edge connect to any node outside of G'_T, create a directed edge from any node that has no directed edge pointing from it in G'_T to another node outside of G'_T that has no directed edge pointing to it.

Definition 10 (Conflict transactions)**.** Transaction T_1 and transaction T_2 are in conflict if at least one of the following conditions is met:

1. $R\_s_{T_1} \cap W\_s_{T_2}$ is not $\varnothing$.
2. $R\_s_{T_2} \cap W\_s_{T_1}$ is not $\varnothing$.
3. $W\_s_{T_1} \cap W\_s_{T_2}$ is not $\varnothing$.

Notation 1. Given relation R with a single column. $S[R]$ denotes the set containing the attribute values of relation R

4.3 Proposed Solution

In this section, we will present the algorithms needed to properly recover from malicious transactions. Algorithm 1 creates a data dependency record of provenance for each transaction. Each record contains transaction ID, timestamp, attributes which are read or wrote by this transaction, and a set of data dependency pairs. These records will be used to find the minimal number of transactions and attributes that have to be recovered.

Algorithm 2 finds the affected transactions and affected attributes. The records generated by Algorithm 1 provide information on how each attribute are updated. By scanning through those records, all affected transactions and affected attributes can be found. All affected transactions will be added into a table containing their transaction ID and ID of transactions which they depend on. All affected attributes will be locked until they are repaired. The last assignment table L is generated so the system knows when affected attributes can be unlocked. Once affected attributes are updated with values from re-execution of their last assignment transactions, they will be unlocked.

Algorithm 3 repairs the damage. The affected transaction table AT generated by Algorithm 2 provides information on which transactions have to be re-executed. The re-execution will be processed on a snapshot of database. The unaffected part of database can be operated during this repair process. The affected transactions will be re-executed in a topological order based on transaction dependency. The affected attributes will be unlocked once the transactions which last assigned them are re-executed and committed. This provides availability of those attributes as soon as possible.

Next, we analyze and formally prove the properties of our solution.

Theorem 1. Given a malicious transaction T_k and a data provenance table P_T. The

Algorithm 1 Create Data Dependency Provenance Record

Input: Transaction, T_i and data provenance table P_T

Output: Data dependency provenance record of T_i as $P_{T_i} =< ID_{T_i}, ts, R\_s, W\_s, Dp >$

1: Initialization
2: If data provenance table is not available, create an empty table $P_T(ID, ts, R\_s, W\_s, Dp)$
3: Let ts := "", R_s := {}, W_s := {}, Dp := {}, ts = timestamp when T_i is committed, ID_{T_i} = generated id based on transaction committed order
4:
5: **for all** *attributes* A_i *in* T_i **do**
6: **if** *attribute* A_i *is assigned by* $\{exp, c\}$ **then**
7: $Set_{A_i} = \{\}$
8: $Dp_{A_i} = (A_i, Set_{A_i})$
9: $W\_s = W\_s \cup A_i$
10: **for all** *attributes* A_j *in* $\{exp, c\}$ **do**
11: $Dp_{A_i} = (A_i, \{Set_{A_i} \cup A_j\})$
12: **end for**
13: *Add* Dp_{A_i} *to* Dp
14: **else**
15: $R\_s = R\_s \cup A_i$
16: **end if**
17: **end for**
18: Insert record $(ID_{T_i}, ts, R\_s, W\_s, Dp)$ into P_T

tables AT and L, generated by Algorithm 2 are complete. That is, table AT contains all transactions that are affected by T_k and a table L contains all attributes that are affected by T_k.

This theorem shows that Algorithm 2 found all affected transactions and affected attributes.

Proof. By induction;

Let $T_1, T_2, ..., T_n$ be n transactions following T_k in transaction log, AT_n contains all transactions affected by T_k from T_1 to T_n, and L_n contains all attributes affected by T_k from T_k to T_n.

The base case: when n = 1.

Algorithm 2 Finding affected transaction

Input: Data dependency provenance table P_T, Transaction ID T_k of a malicious transaction, and a transaction log

Output: The affected transaction table $AT(T, Set_T)$ and the last assignment table $L(A, T)$

Initialization
Create empty tables $AT(T, Set_T)$, and $L(A, T)$

Abort all incomplete transactions
We are locking the database
Read the data dependency provenance record P_T of the malicious transaction T_k in the table
for all *dependency pair* (A_i, Set_{A_i}) *in* Dp **do**
    Insert record (A_i, T_k) into L
end for
for all T_i following T_k in transaction log ordered by committed time **do**
    Read record of T_i in P_T
    for all dependency pair (A_i, Set_{A_i}) in Dp **do**
        if $Set_{A_i} \cap S[\pi_A(L)]$ is not $\varnothing$ **then**
            if $T_i \notin S[\pi_T(AT)]$ **then**
                $Set_{T_i} = \{\}$
                for all $A_j \in Set_{A_i} \cap S[\pi_A(L)]$ **do**
                    $Set_{T_i} = Set_{T_i} \cup \pi_T(\sigma_{A=A_j}(L))$
                end for
                Insert record (T_i, Set_{T_i}) into AT
            else
                $Set_{T_i} = \pi_{Set_T}(\sigma_{T=T_i}(AT))$
                for all $A_j \in Set_{A_i} \cap S[\pi_A(L)]$ **do**
                    $Set_{T_i} = Set_{T_i} \cup \pi_T(\sigma_{A=A_j}(L))$
                end for
                Update record $\sigma_{T=T_i}(AT)$ to (T_i, Set_{T_i})
            end if
            if $A_i \notin S[\pi_A(L)]$ **then**
                Insert record (A_i, T_i) into L
            else
                Update record $\sigma_{A=A_i}(L)$ to (A_i, T_i)
            end if
        else if $A_i \in S[\pi_A(L)]$ and $Set_{A_i} \cap S[\pi_A(L)]$ is $\varnothing$ **then**
            Delete record $\sigma_{A=A_i}(L)$
        end if
    end for
end for
We are unlocking only the attributes in database $\notin S[\pi_A(L)]$

Algorithm 3 Repairing

Input: The affected transaction table AT, the last assignment table L, and the snapshot of database at the time before the malicious transaction committed

Output: A consistent database state which all malicious and affected transactions are undone

Initialization
Create empty ordered list $FixQueue =< T_1, T_2, ... >$ with the size equal to the number of records in AT

for all $\pi_{Set_T}(AT)$ **do**
    $Set_T = Set_T - T_k$
end for
while AT is not $\varnothing$ **do**
    if $\pi_{Set_{T_i}}(AT)$ is empty **then**
        Add T_i to FixQueue
        Delete record $\sigma_{T=T_i}(AT)$
        for all $\pi_{Set_T}(AT)$ **do**
            $Set_T = Set_T - T_i$
        end for
    end if
end while
if $T_k \in S[\pi_T(L)]$ **then**
    Fixed attributes F := $\pi_A(\sigma_{T=T_k}(L))$
    for all $A_i \in S[F]$ **do**
        Unlock A_i
    end for
end if
for all $FixQueue[i]$ where i = 0, i++ **do**
    execute FixQueue[i]
    if $FixQueue[i] \in S[\pi_T(L)]$ **then**
        Fixed attributes F := $\pi_A(\sigma_{T=FixQueue[i]}(L))$
        for all $A_i \in S[F]$ **do**
            Update database with A_i
            Unlock A_i
        end for
    end if
end for

1. Let L_1 initially contains all attributes updated by T_k.

2. AT_1 contains T_1 if T_1 update an attribute A_1 depending on an attribute A_k in L_1.

3. Records of T_1 in data provenance table P_T shows whether A_1 depends on A_k.

4. If A_1 depends on A_k, A_1 is also affected by T_k so T_1 is added to AT_1 and A_1 is added to L_1.

The induction case: assume AT_{n-1} contains all transactions that are affected by T_k from T_1 to T_{n-1}, and L_{n-1} contains all attributes affected by T_k from T_k to T_{n-1}.

1. Consider transaction T_n update an attribute A_n.

2. Records of T_n in P_T shows whether A_n depends on any attribute A_l in L_{n-1}.

3. If A_n depends on A_l, A_n is also affected by T_k so T_n is added to AT_n and A_n is added to L_n.

Conclusion: By the principle of induction, AT_n contains all transactions affected by T_k and L_n contains all attributes affected by T_k. □

Theorem 2. Given an affected transaction list AT, the last assignment table L, and the snapshot of database at the time before the malicious transaction T_k committed. Then Algorithm 3 recovers the database to consistent state without any affect from T_k and preserves all other transaction results.

This theorem shows that Algorithm 3 removed all damage from the malicious transaction and preserved all non-malicious transaction results.

Proof. By contradiction;

1. Assume that some inconsistent attribute A_i that is affected by malicious transaction T_k and A_i is not fixed by algorithm 3 .

2. But attribute A_i must be assigned by a transaction T_i that is affected by T_k.

3. Algorithm 3 executes all transactions in AT by their committed order on the given snapshot and AT contains all transactions that are affected by T_k.

4. AT contains all transaction id which are affected by T_k. So T_i is in AT.

5. Since T_i is re-executed from a consistent state without affect from T_k by algorithm 3, all attributes assigned by T_i are consistent including A_i.

6. So A_i must be consistent. This contradict the supposition that A_i is inconsistent.

□

Theorem 3. Let transaction T_i be any transaction in AT and transaction T_j is any incoming transaction. Assume that all transactions obey the Two-phase locking rule. Then the history containing all operations of T_i and T_j is conflict-serializable.

This theorem shows that executing incoming transactions and repairing can be run concurrently. The result of database state will be correct and consistent with respect to conflict serializability.

Proof.

Case 1: There is no conflict operation between transaction T_i and T_j.

1. In this case, the following conditions must be true by definition of conflict transactions

   a) $R\_s_{T_i} \cap W\_s_{T_j}$ is $\varnothing$.

   b) $R\_s_{T_j} \cap W\_s_{T_i}$ is $\varnothing$.

   c) $W\_s_{T_i} \cap W\_s_{T_j}$ is $\varnothing$.

2. The partial order of transaction T_i and T_j can be represent using directed graph.

3. There is no edge between node of T_i and T_j because of the conditions above.

4. So the graph is a directed acyclic graph. This history is conflict-serializable.

Case 2: There is a conflict operation between transaction T_i and T_j.

1. In this case, at least one of the following conditions must be true by definition of conflict transactions

   a) $R\_s_{T_i} \cap W\_s_{T_j}$ is not $\varnothing$.

   b) $R\_s_{T_j} \cap W\_s_{T_i}$ is not $\varnothing$.

   c) $W\_s_{T_i} \cap W\_s_{T_j}$ is not $\varnothing$.

2. Case (a): $R\_s_{T_i} \cap W\_s_{T_j}$ is not $\varnothing$.

   1 T_i always read attribute value from the snapshot which is a database state preceding a state that T_j writes. So the read operation of T_i always precedes the write operation of T_j.

   2 The partial order of transaction T_i and T_j can be represented using a directed graph.

   3 The edge between node of T_i and T_j is in a direction from T_i to T_j because T_i always precedes T_j.

   4 So the graph is a directed acyclic graph. This history is conflict-serializable.

3. Case (b): $R\_s_{T_j} \cap W\_s_{T_i}$ is not $\varnothing$.

   1 Algorithm 2 discovers all attributes that are affected by a malicious transaction T_k as claimed in Theorem 1 and it lock all those attributes.

   2 There are 2 possible cases as follows:

      i) The attribute A_i in $R\_s_{T_j} \cap W\_s_{T_i}$ is not an affected attribute.

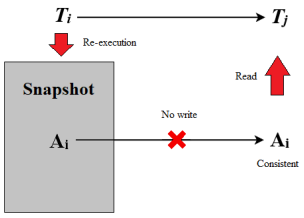

Figure 4.1 Transaction Dependency in Case 2.b.i

ii) The attribute A_i in $R\_s_{T_j} \cap W\_s_{T_i}$ is an affected attribute.

2.1 Case (i): the attribute A_i in $R\_s_{T_j} \cap W\_s_{T_i}$ is not an affected attribute.

2.1.1 The attribute A_i which was written by affected transaction T_i is consistent and will not be overwritten after T_i is re-executed in snapshot. So the history is preserved and the write operation of T_i over A_i precedes the read operation of T_j as in the original history.

2.1.2 The partial order of transaction T_i and T_j can be represent using a directed graph.

2.1.3 The edge between node of T_i and T_j is in a direction from T_i to T_j because T_i always precedes T_j.

2.1.4 So the graph is a directed acyclic graph. This history is conflict-serializable.

2.2 Case (ii): The attribute A_i in $R\_s_{T_j} \cap W\_s_{T_i}$ is an affected attribute.

2.2.1 The attribute A_i is locked by Algorithm 2 until the re-execution of affected transactions which modify A_i is finished and the consistent value of A_i is updated. So the write operation of T_i always precedes the read operation of T_j.

2.2.2 The partial order of transaction T_i and T_j can be represented using a directed graph.

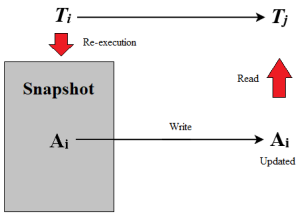

Figure 4.2 Transaction Dependency in Case 2.b.ii

2.2.3 The edge between node of T_i and T_j is in a direction from T_i to T_j because T_i always precedes T_j.

2.2.4 So the graph is a directed acyclic graph. This history is conflict-serializable.

4. Case (c): $W\_s_{T_i} \cap W\_s_{T_j}$ is not $\varnothing$.

1 Algorithm 2 discovers all attributes that are affected by a malicious transaction T_k as claimed in Theorem 1 and it locks all those attributes.

2 There are 2 possible cases as follows:

i) The attribute A_i in $W\_s_{T_j} \cap W\_s_{T_i}$ is not an affected attribute.

ii) The attribute A_i in $W\_s_{T_j} \cap W\_s_{T_i}$ is an affected attribute.

2.1 Case (i): the attribute A_i in $W\_s_{T_j} \cap W\_s_{T_i}$ is not an affected attribute.

2.1.1 The attribute A_i which was written by affected transaction T_i is consistent and will not be overwritten after T_i is re-executed in snapshot. So the history is preserved and the write operation of T_i over A_i precedes the write operation of T_j as in the original history.

2.1.2 The partial order of transaction T_i and T_j can be represented using a directed graph.

2.1.3 The edge between node of T_i and T_j is in a direction from T_i to T_j because T_i always precede T_j.

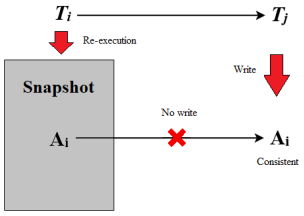

Figure 4.3 Transaction Dependency in Case 2.c.i

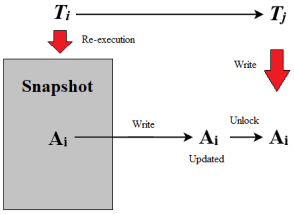

Figure 4.4 Transaction Dependency in Case 2.c.ii

2.1.4 So the graph is a directed acyclic graph. This history is conflict-serializable.

2.2 Case (ii): the attribute A_i in $W\_s_{T_j} \cap W\_s_{T_i}$ is an affected attribute.

2.2.1 The attribute A_i is locked by Algorithm 2 until the re-execution of affected transactions which modify A_i is finished and the consistent value of A_i is updated. So the write operation of T_i always precedes the write operation of T_j.

2.2.2 The partial order of transaction T_i and T_j can be represented using a directed graph.

2.2.3 The edge between node of T_i and T_j is in a direction from T_i to T_j because T_i always precedes T_j.

2.2.4 So the graph is a directed acyclic graph. This history is conflict-serializable.

5. Consider all the possible cases, the history containing all operations of T_i and T_j is conflict-serializable.

□

4.4 Provenance Protection

We have to protect our provenance records. In addition to malicious transaction attacks, attackers can also target our provenance records. This can hinder the recovery process by rendering the provenance records unusable. To protect our provenance records, we can implement protection mechanism. One of the possible approaches is blockchain technology. Many researchers proposed approaches using blockchain to protect provenance data [41][42][43]. The blockchain can guarantee the integrity of its records as long as the honest nodes are the majority of the blockchain network. To apply blockchain to our approach, we can store our data dependency provenance records in form of blocks in the blockchain. After the record blocks are validated and stored on the blockchain network, attackers cannot temper them without having computational power more than the rest of the network. However, blockchain approach also have disadvantages. The records on blockchain are public, anyone can read the records. It's also computationally expensive and slow.

Chapter 5

Smart Home Improvement

5.1 System Architecture

Smart home automation systems aim to allow home owners to seamlessly integrate multiple devices. Typical functionalities provided by home automation systems include fire and carbon monoxide monitoring, remote control of lights, thermostat, and smart appliances; security systems and cameras; real-time alerts; and integration with smart platforms (e.g., Alexa, Siri, Google Home). Some of the home automation systems also support data analysis to help data-driven decision making. However, current home automation systems fall short in incorporating the achievements of the situation-aware decision making research or to allow easy plug-in third party applications.

5.1.1 System Overview

We present a local architecture for extending the popular home automation systems with third party applications. Figure 5.1 shows the system overview for our application. The current implementation focuses on connecting Home Assistant, an open source home automation system, with a Protégé-OWL reasoner. The current proof-of-concept implementation is based on using local communication between Home Assistant and the reasoner. Device data from Home Assistant is converted to XML format and pushed to the reasoner. The result from the reasoner is sent back in XML format to be inserted to the Home Assistant database. Based on the received feedback from the reasoner, Home Assistant carries out the appropriate actions.

The advantage of decoupling the reasoner from the Home Assistant application is that our approach can also be used with other home automation software. The data exchange schema is based on data semantics and independent from local data models. The same schema can support other home automation platforms with mapping their data models to the data exchange format.

5.1.2 Home Assistant Overview

Home Assistant is a free and open source software platform that can be used for integrating IoT devices in a home network. It has a web user interface that can be used as a central system for controlling the devices in the home IoT network. The Home Assistant Operating System can be installed on various platforms. After the installation, the user creates an owner account with administrator privileges. Next, the Home Assistant shows the discovered devices in the network and provides the interface for integrating these devices. The user then can write automation rules based on triggering of an event (e.g., "Turn the porch light off at sunrise"), or based on user detection (e.g., "Turn the front door light on when user Alex enters the house").

5.1.3 Context and Situation-Aware Reasoning

Context-aware decision support has been studied extensively in the context of access control policies [14][15][44]. Weather, location, fire, computer network, and user security clearances are examples of contextual information. Context-Aware Access Control proposes a flexible and dynamic access control which can adapt to specific context characteristics. During the last decade, situation-aware security approaches were developed to address the dynamic and ad-hoc nature of IoT networks [45][46][47], [48]. These efforts aim to extend the security capabilities to identify situations and adopt the appropriate policy accordingly. Semantics-based data integration and on-

tological reasoning are the backbone of these approaches. Despite the achievements of the situational-aware security researchers, these technologies are not sufficiently incorporated in smart home environment. The closest to our work is by Chen et al. [46]. The authors address the specific needs in smart home environment, focusing of assisted living. Our primary focus on the security, safety, and privacy of smart homes, in particular, in the context of home automation systems.

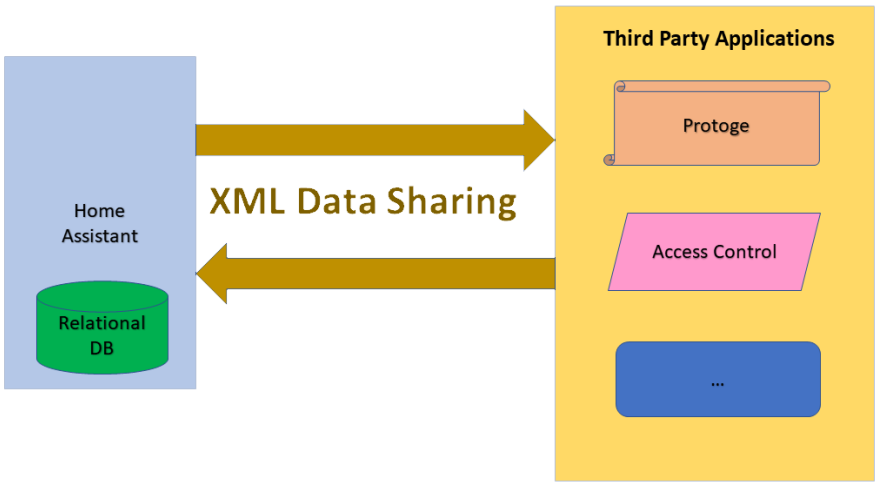

Figure 5.1 Smart Home Automation System Architecture

5.2 Use Cases and Misuse Cases

Use cases are widely used to model functional requirements is software and system engineering. Uses cases allow the developers to analyze how the system is planned to be used and who are the intended actors. Actors may be human users of external systems or events. During the last couple of decades, the concept of misuse cases (or abuse cases) have been developed. Misuse cases allow the developers to to analyze negative scenarios and define how the system should behave under such circumstances (see Figure 5.2 for a representative example on misuse cases). Misuse cases impact the use cases and often lead to new functionality requirements. Modelling potential threats against the system has been shown to increase the security, safety, and

reliability of such systems.

Understanding threats against the smart home environment is crucial. Most of the research and development efforts focused on the security and privacy threats against the underlying technologies and infrastructure (see [49] for an overview). More recently, researchers addressed the threat against smart home environment from the behavioral aspect of the malicious users, e.g., insiders' misuse of the technology [50].

Our aim is to discovers new use cases based on understanding both technical and behavioral threats. For example, consider the use case of providing fire safety for a smart home. The interleaved use case and misuse case diagrams are presented in Figure 5.2.

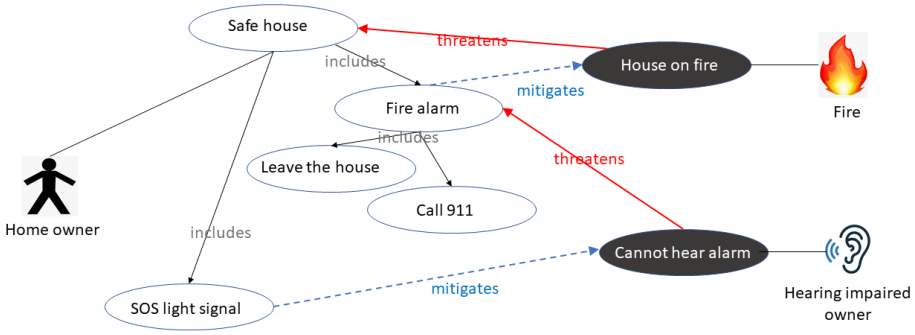

Figure 5.2 Home Assistant's response to fire in case of hearing impaired home owner

The main actors are the home owner (intended user); the house fire, and the hearing impairment of the home owner (malicious actors). The house fire threatens the safety of the house and the owners. The sound of the fire alarm mitigates the threat by alerting the home owner to the threat; so the owner could leave the house and call 911. However, if the home owner is hearing impaired, the alarm may go unnoticed, thus threatening the effect of the fire alarm. To mitigate this threat, we extended the home automation system to turn on the light and signal the Morse code for S.O.S. signal repeatedly. The three short, three long, and three short signal sequence is universally known to indicate a crisis situation. By projecting S.O.S., the

home automation system would alert the home owner, as well as external observers for the crisis.

5.3 Proposed Solution

In this section, we proposed a local system architecture to support the extension of home automation systems with third party applications. We argued that such extensions would enhance the functionality of the home automation environment. In particular, we focused on supporting semantics-based data sharing and dynamic policy support. We proposed the use of XML data exchange format to allow interoperation and make our proposed approach applicable to a variety of home automation systems. We also demonstrated that our reasoner was able to derive situation-aware information from the device data. This information, when received by Home Assistant, lead to the desired activity, i.e., the signalling of the S.O.S. Morse code via smart light bulbs. The uniqueness of this approach is that we were able to control the behavior of the home automation system based on the situation decision derived by the Protégé reasoner. Such enhancement of smart platforms are critical in supporting complex requirements for owners with disabilities.

5.3.1 Protégé Reasoner

We implemented our dynamic policy reasoner using Protégé[51] and Semantic Web Rule Language (SWRL). Protégé is an open-source ontology tool for Web Ontology Language (OWL) development. It provide various functionalities such as creating, modifying, reasoning, querying, and visualizing an ontology. SWRL can enhance reasoning capabilities of OWL by providing a rule language.

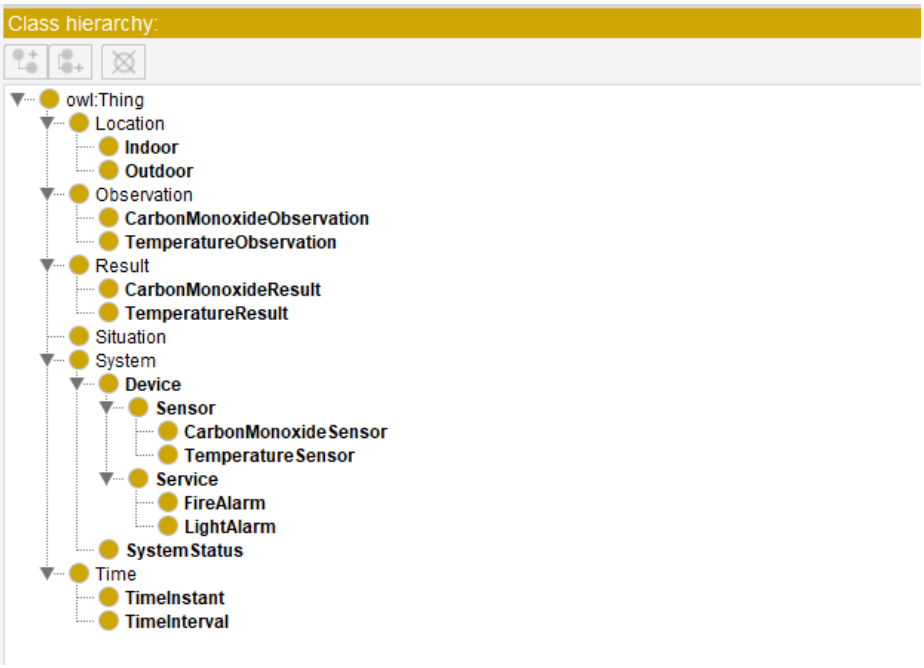

Figure 5.3 The Simplified Temperature Reasoner Ontology in the Protégé

Figure 5.3 shows a simplified temperature reasoner ontology. We used this simplified ontology to derive the current situation based on the observed temperature. Consider the following examples where the room temperature indicates normal, caution, or emergency situations.

SWRL rule 1. If the observed temperature at any given locations and time is between 70 to 80 degrees Fahrenheit, the temperature status is Normal.

$$Observation(?\,x) \wedge hasResult(?\,x, ?\,y) \wedge hasTemperature(?\,y, ?\,r) \wedge hasUnit(?\,y, "Fahrenheit")$$
$$\wedge\, swrlb : lessThan(?\,r, 80) \wedge swrlb : greaterThanOrEqual(?\,r, 70) \rightarrow$$
$$SystemStatus(CurrentTemperatureStatus) \wedge hasStatus(CurrentTemperatureStatus, Normal)$$

SWRL rule 2. If the observed temperature at any given locations and time is between 80 to 120 degrees Fahrenheit, the temperature status is Caution.

$$Observation(?\,x) \wedge hasResult(?\,x, ?\,y) \wedge hasTemperature(?\,y, ?\,r) \wedge hasUnit(?\,y, "Fahrenheit")$$
$$\wedge\, swrlb : lessThan(?\,r, 120) \wedge swrlb : greaterThanOrEqual(?\,r, 80) \rightarrow$$
$$SystemStatus(CurrentTemperatureStatus) \wedge hasStatus(CurrentTemperatureStatus, Caution)$$

SWRL rule 3. If the observed temperature at any given locations and time is more than 120 degrees Fahrenheit, the temperature status is Emergency.

$$Observation(?\,x) \wedge hasResult(?\,x, ?\,y) \wedge hasTemperature(?\,y, ?\,r) \wedge hasUnit(?\,y, "Fahrenheit")$$
$$\wedge\, swrlb : greaterThanOrEqual(?\,r, 120) \rightarrow SystemStatus(CurrentTemperatureStatus)$$
$$\wedge\, hasStatus(CurrentTemperatureStatus, Emergency)$$

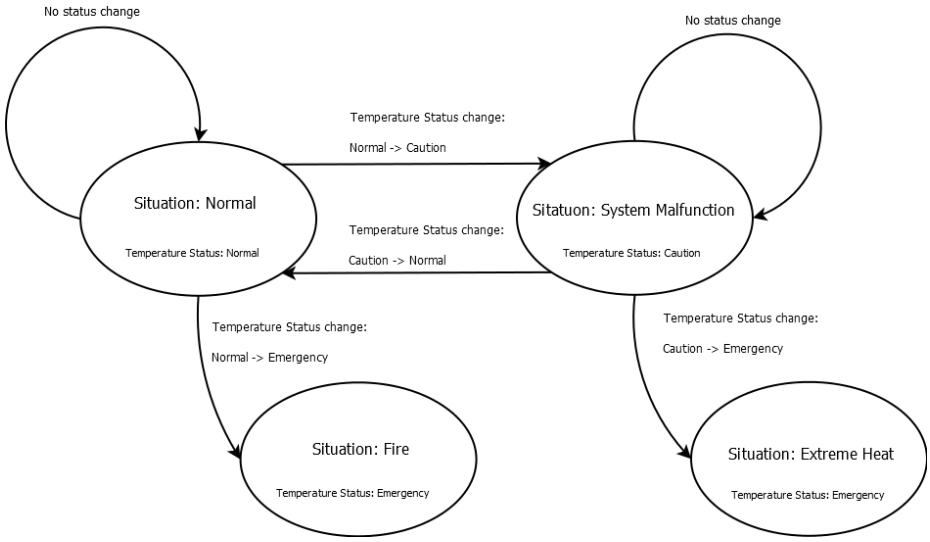

Figure 5.4 The Finite-State Machine for Situation Derivation

We use the temperature status (conclusions of the SWRL rules) to derive situational information. These conclusions trigger state changes in our finite-state machine in Figure 5.4. We derive four situations; normal, system malfunction, extreme heat, and fire.

Situation: Normal. The temperature status is normal. The system resumes normal operation and waits for the next observation.

Situation: Fire. If the temperature status changes rapidly from normal to emergency temperature. The system can derive that the fire occurs. The emergency procedures will be followed such as raising the alarm, calling the fire service, and activating water sprinklers. Note, that after reaching this situation state, the state will have to be changed manually by the system administrator to ensure that the emergency situation is resolved.

Situation: System Malfunction. If the temperature status changes slowly from normal to caution temperature. The system can derive that there is a potential malfunction in the temperature control system. The maintenance should be requested to fix the issue. The system submits the alert and waits for the next observation to update the situation. The situation can return to normal after the malfunction is fixed and temperature status returns to normal.

Situation: Extreme Heat. If the temperature status changes slowly from caution to emergency temperature. The system can derive that there is a potential health hazard from built-up heat. To avoid heat stress, all people in the area should be evacuated. Note, that after reaching this situation state, the state will have to be changed manually by the system administrator to ensure that the emergency situation is resolved.

5.3.2 Home Assistant Data Model

Figure 5.5 shows the high-level overview of the database that stores the device data and control information of Home Assistant. Detailed information about the data structure and user support to query the database is available at the Home Assistant's website [52]. Our research team used the data model to answer the following two specific questions: 1) How to retrieve data that is needed for the Protégé decision making? 2) How to store the result from the reasoner such that the appropriate automation is carried out?

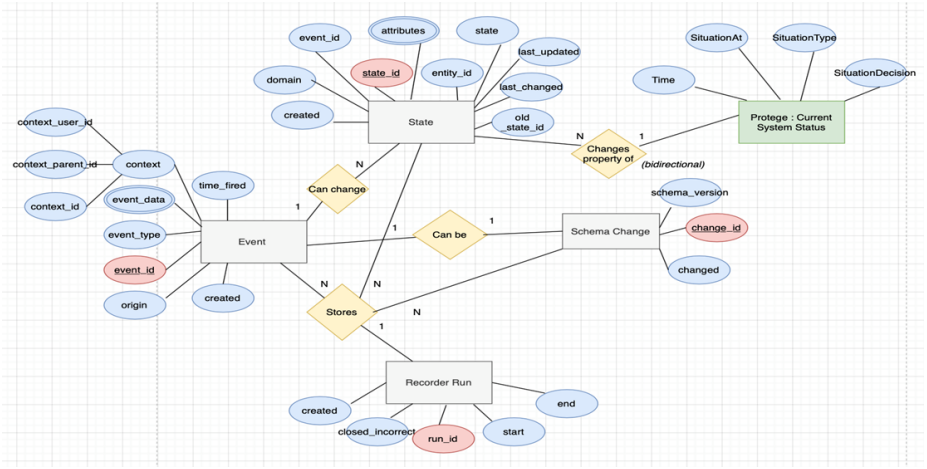

Figure 5.5 Home Assistant database Entity-Relational Data Model

The Home Assistant's database contains four tables: Events, States, Recorder Runs, and Schema Changes. The most important one is the events table. Everything that "happens" in Home Assistant is represented as an event. Each event has an unique ID, type, data, and other attributes such as context. The context is used to tie events and states together. Whenever an automation or user interaction triggers a new change, a new context is created. This new context is attached to all events and states that occur as the result of the change. States are representations of Entities in Home Assistant. Each entity is a certain function of a device. For example, a motion sensor is one device with three entities: motion, temperature, and light. Each entity has a state such as on/off associated with it. States have an unique state id, the event id that changed the state, and certain attributes that reveal extra state data about the entity. The value last changed is the last time the state changed, whereas last updated is the last time the state was checked, but not necessarily changed. A schema change is an event that changes the database schema. This can happen when an integration for a sensor platform is added. This allows certain data to be added and available. The recorder is responsible for storing all the data. Each time Home Assistant starts, the recorder starts a new recorder run. A run is finished when Home Assistant gracefully shuts down or is restarted.

5.3.3 Home Assistant and Protégé Data Sharing

Home Assistant supports built in functionalities as well as provides the capability to code additional functionalities. To facilitate data exchange between the Home Assistant platform and the reasoner, we developed an XML schema for data sharing. This approach allows to extend the data sharing for future applications as well as supported by most computing platforms. XML has been used to support data sharing among heterogeneous data sources. Figure 5.6 outlines how the data is extracted from the Home Assistant database, converted to the XML schema, and shared with the reasoner. Similarly, the decision reached by the reasoner is converted to the XML format, and incorporated in the Home Assistant database.

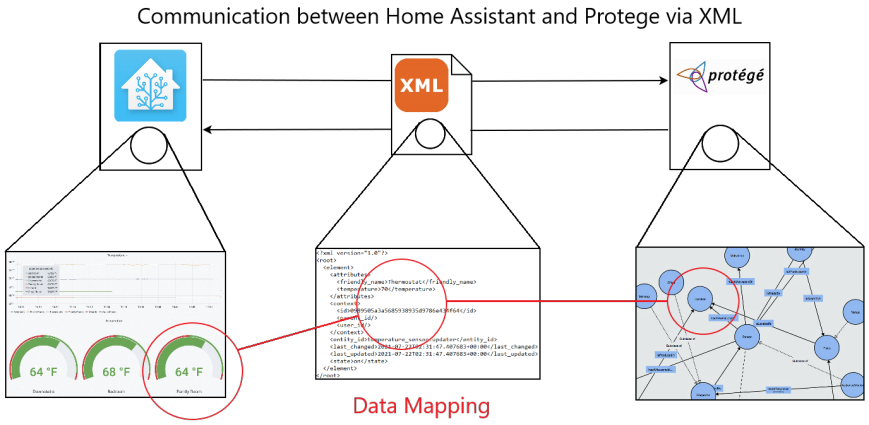

Figure 5.6 Communication between Home Assistant and Protégé via XML

To get data from Protégé to Home Assistant, we converted the CSV file that was exported by Protégé. We used basic bash commands to create a script that got the emergency status and the location of the emergency. To represent the decision data, a state was created within the Home Assistant system that contained the hazard status (normal/emergency), hazard location (bedroom, kitchen, etc.), and the hazard type (fire, burglar, etc.). To push the data, simple REST API calls were made. To monitor for emergency scenario, we implemented a Home Assistant script to monitor the state

containing the Protégé decision. In case the hazard status changed to emergency, Home Assistant sat the emergency flag and ran the corresponding automation. In our fire emergency scenario, this automation is the repeated flashing of the S.O.S. light signals.

Chapter 6

Stream Data Security

6.1 Stream Data Model

In addition to traditional data security requirements. stream data requires security capabilities to provide completeness of data, correct ordering, and semantic-based dynamic security policy. Incomplete data and incorrect order affect integrity of stream data. They also can not be addressed with relational database security approaches. Semantic-based dynamic security policy is needed to provide fast and situation-responsive availability. It can also counter attribute shuffle attacks which are unique to stream data model due to the lack of metadata in tuples.

In this section we present our framework to address the above requirements.

Definition 11. Data Schema

A data schema D is a set of attribute names denoted by $D(A_1, ..., A_n)$, where D is the name of the data schema and $A_i|i \in \{1, ..., n\}$ is an attribute name.

Definition 12. Data Instance on Data Schema D

A data instance d on data schema $D(A_1, ..., A_n)$ is denoted as $d =< a_1, ..., a_n >$, where $a_i|i \in \{1, ..., n\}$ is the value of attribute A_i such that a_i is in the domain of attribute A_i, i.e., $a_i \in Dom(A_i)$.

We also denote data instance d as $d =< A_1 = a_1, ..., A_n = a_n >$ to explicitly identify the attribute values.

Definition 13. Stream Tuple

A stream tuple t from source l is denoted as $t = (l, d, ts)$, where ts is a timestamp

representing the time when t was generated.

Example 6.1 A stream tuple t_1 from source $sensor_1$ at 1:00AM

$t_1 = (sensor_1, < A_1 = 10, A_2 = 20, A_1 = 30 >, 1{:}00{:}00\text{AM})$

The above stream tuple t_1 was generated by source $sensor_1$ at 1:00:00AM. It contains a data instance $d =< A_1 = 10, A_2 = 20, A_1 = 30 >$ which specify the value of attribute A_1 is 10, the value of attribute A_2 is 20, and the value of attribute A_3 is 30.

Definition 14. Data Stream

A data stream S from source l during a time period from ts_j to ts_k is denoted as $S_l =< t_j, ..., t_k >$, where $t_i = (l_i, d_i, ts_i)$ such that $l_i = l$ and $< ts_j, ..., ts_k >$ is a sequence of timestamps during a time period from ts_j to ts_k.

We also denote a data stream S from source l during time interval from ts_j to ts_k as $S_l[ts_j, ts_k] =< t_j, ..., t_k >$ or $S_l[ts_j, ts_k] =< (l, d_j, ts_j), ..., (l, d_k, ts_k) >$.

Example 6.2 A data stream S from source $sensor_1$ from 1:00AM to 1:05AM

$S_{sensor_1}[1:00:00AM, 1:05:00AM] = < t_2, t_2, t_3, t_4, t_5 > =$

$<$

$t_1 = (sensor_1, < A_1 = 10, A_2 = 20, A_1 = 30 >, 1{:}00{:}00\text{AM}),$

$t_2 = (sensor_1, < A_1 = 10, A_2 = 20, A_1 = 27 >, 1{:}01{:}00\text{AM}),$

$t_3 = (sensor_1, < A_1 = 11, A_2 = 20, A_1 = 27 >, 1{:}02{:}00\text{AM}),$

$t_4 = (sensor_1, < A_1 = 11, A_2 = 20, A_1 = 30 >, 1{:}03{:}00\text{AM}),$

$t_5 = (sensor_1, < A_1 = 10, A_2 = 22, A_1 = 33 >, 1{:}05{:}00\text{AM})$

$>$

The above data stream S_{sensor_1} was generated by source $sensor_1$ during 1:00:00AM to 1:05:00AM. It contains stream tuples t_1, t_2, t_3, t_4, and t_5, where the tuples were generated in 1 minute intervals.

Definition 15. Stream Bundle

Let ts denotes a timestamp and $l_1, ..., l_n$ denote stream data sources. A stream bundle at the time ts, denoted as

$B_{ts} = \{(l_1, d_1, ts), ..., (l_n, d_n, ts)\}$, is the set of stream tuples generated by the sources $l_1, ..., l_n$ at time ts.

Note, at this point we do not address the issue of incompatible time stamps due to different levels of precision of the sources or due to the lack of synchronization.

Definition 16. Atomic Protected Object

An atomic protected object is a triple consisting of 3 elements; a source description, a data description, and a timestamp description. A protected object o is denoted as $o = ((source = l, exp_l), (d, exp_d), (timestamp = ts, exp_{ts}))$ where each elements are as following;

1. A source description $(source = l, exp_l)$ where l is either;

   a) A constant value c such that c is in the domain of authorized sources.

   b) A variable v.

   And exp_l is a Boolean expression of the form $component_1\ AND\ ...\ AND\ Component_n$, where $Component_i$ is;

   a) A truth value TRUE.

b) v *op* c where v is the variable in $source = v$, c is a constant, and *op* is either $=, <$.

We also denote a source description (l, exp_l) representing any source as $*$ where $*$ represents $(v_1, TRUE)$.

2. A data description (d, exp_d) where d is a data instance on data schema D, denoted as $d =< A_1 = k_1, ..., A_n = k_n >$ such that k_i is either;

   a) A constant value c such that $c \in Dom(A_i)$.

   b) A variable v.

   And exp_d is a Boolean expression of the form $component_1$ AND ... AND $Component_n$, where $Component_i$ is;

   a) A truth value TRUE.

   b) v_i *op* c where v_i is the variable in d, c is a constant, and *op* is either $=, <$.

   c) v_i *op* v_j where v_i, v_j are the variables in d, and *op* is either $=, <$.

   d) A_i *op* c where A_i is the attribute name in d, c is a constant, and *op* is either $=, <$.

   e) A_i *op* v_i where A_i is the attribute name in d, v_i is the variable in d, and *op* is either $=, <$.

   f) A_i *op* A_j where A_i, A_j are the attribute names in d, and *op* is either $=, <$.

   We also denote a data description (d, exp_d) representing any data instance on data schema D as $*$ where $*$ represents $(< A_1 = v_1, ..., A_n = v_n >, TRUE)$ such that none of variable v_i are the same.

3. A timestamp description $(timestamp = ts, exp_{ts})$ where ts is a timestamp such that ts is either;

a) A constant value c such that c is in the domain of legitimated timestamps.

b) A variable v.

And exp_{ts} is a Boolean expression of the form $component_1$ AND ... AND $Component_n$, where $Component_i$ is;

a) A truth value TRUE.

b) v op c where v is the variable in $timestamp = v$, c is a constant, and op is either $=, <$.

We also denote a timestamp description (ts, exp_{ts}) representing any timestamp as $*$ where $*$ represents $(v_1, TRUE)$.

We also denote $O = \{o_1, ..., o_n\}$ as a set of protect objects.

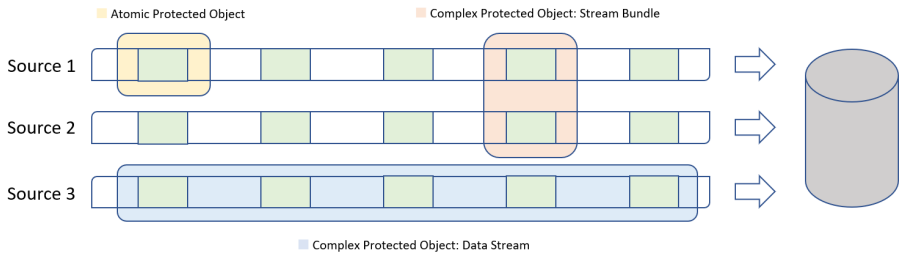

Figure 6.1 Examples of different kinds of protected objects; 1. Atomic Protected Object:A tuple, 2. Stream Bundle: Tuples from multiple sources which arrive at the same time period, 3. Data Stream: Multiple tuples which continuous arrive from a single source

Definition 17. Simple Object Valuation

Let ν be a symbol mapping, such that;

1. ν maps a constant c to itself, i.e.,

$$\nu : c \longrightarrow c.$$

2. ν preserves equality of variables, i.e.,

$$\nu : v \longrightarrow c_1.$$

$$\nu : v \longrightarrow c_2.$$

$$\Longrightarrow \ c_1 = c_2.$$

3. ν maps special symbols to themselves, i.e.,

$$\nu :=\longrightarrow= .$$

$$\nu :<\longrightarrow< .$$

$$\nu : AND \longrightarrow AND.$$

$$\nu : TRUE \longrightarrow TRUE.$$

$$\nu : FALSE \longrightarrow FALSE.$$

$$\nu : A_i \longrightarrow A_i.$$

Definition 18. Stream Pattern Mapping

Let $o = ((source = l, exp_l), (d =< A_1 = k_1, ..., A_n = k_n >, exp_d),$
$(timestamp = ts, exp_{ts}))$ be an atomic protection object, and ν a symbol mapping. A stream pattern mapping N is defined as;

1. $N(source = l, exp_l)$ is defined as;

   a) $N(source = l) = N(l) = \nu(l)$.

   b) $N(exp_l)$ is defined by replacing each symbol s in exp_l with $\nu(s)$, i.e., if exp_l is of the form;

      i. $(l \ op \ c)$ will become $\nu(l) \ \nu(op) \ \nu(c)$.

2. $N(< A_1 = k_1, ..., A_n = k_n >, exp_d)$ is defined as;

a) $N(d) =< \nu(A_1) = \nu(k_1), ..., \nu(A_n) = \nu(k_n) >$.

b) $N(exp_d)$ is defined by replacing each symbol s in exp_d with $\nu(s)$, i.e., if exp_d is of the form:

i. $(k_i\ op\ c)$ will become $\nu(k_i)\ \nu(op)\ \nu(c)$.

ii. $(k_i\ op\ k_j)$ will become $\nu(k_i)\ \nu(op)\ \nu(k_j)$.

iii. $(A_i\ op\ c)$ will become $\nu(A_i)\ \nu(op)\ \nu(c)$.

iv. $(A_i\ op\ k_j)$ will become $\nu(A_i)\ \nu(op)\ \nu(k_j)$.

v. $(A_i\ op\ A_j)$ will become $\nu(A_i)\ \nu(op)\ \nu(A_j)$.

c) $N(A_i) = N(k_i) = \nu(k_i)$.

3. $N(timestamp = ts, exp_{ts})$ is defined as;

a) $N(timestamp = ts) = N(ts) = \nu(ts)$

b) $N(exp_{ts})$ is defined by replacing each symbol s in exp_{ts} with $\nu(s)$, i.e., if exp_{ts} is of the form;

i. $ts\ op\ c$ will become $\nu(ts)\ \nu(op)\ \nu(c)$.

Definition 19. Stream Pattern Mapping Satisfiability

Let o be an atomic protected object. We say that the stream tuple t satisfies o iff there is a pattern mapping N from o to t denoted as $N(o) = t$, such that $N(exp_l) = TRUE$, $N(exp_d) = TRUE$, and $N(exp_{ts}) = TRUE$.

Note, from now on, we only consider satisfied mapping when we use the phrase "pattern mapping".

Definition 20. Complete Data Stream

Let $S =< t_1, ..., t_n >$ be a data stream generated by a source. We say that $S' =< t'_1, ..., t'_n >$ is a strong complete and ordered data stream iff all tuples t_i in S are in S' and any two tuples t_i and t_j, $t_i < t_j$ (i.e., t_i precedes i_j in S) then $t'_i < t'_j$, where $t'_i \equiv t_i$ and $t'_j \equiv t_j$

For data stream, where data is not transmitted at regular interval, we propose the concept of semantic completeness.

Definition 21. Insignificant Tuple

Let $S =< t_1, ..., t_n >$ be a data stream generated by a source and $S' =< t'_1, ..., t'_m >$ is an incomplete data stream of S missing a tuples $t_i == (l_i, d_i, ts_i)$. We say that t_i is an insignificant tuple iff $|d_{i-1} - d_{i+1}| < \Delta$ where Δ is a domain-specific threshold.

Definition 22. Semantic Completeness

Let $S =< t_1, ..., t_n >$ be a strong complete data stream and S' is an incomplete data stream of S where $\{t_i, ..., t_j\} | i, j \in \{1, ..., n\}$ are lost, S' is semantic complete iff all $\{t_i, ..., t_j\}$ are insignificant tuples.

Definition 23. Well-formed Data Stream

A data stream, $S =< t_1, ..., t_i, ..., t_j, ..., t_k >$ where $t_n = (l_n, d_n, ts_n)$, is well-formed iff it satisfies the following conditions;

1. The tuples are ordered, i.e., for any two tuples with timestamps $ts_j > ts_i$, the tuple with timestamp ts_j was generated after the tuple with the ts_i, ts_1 is the earliest timestamp, i.e., $ts_1 < ts_i$ for all $1 < i \leq k$, and ts_k is the latest timestamp, i.e., $ts_k > ts_i$ for all $1 \leq i < k$.

2. The data stream is complete.

Example 6.3 Protected Object Mapping

Given protected object o_1 and tuples t_1, t_2

o_1 = ((source=$sensor_1$,TRUE),($< A_1 = v_1, A_2 = 20 >$,$v_1 < 20$ AND $10 < v_1$),($timestamp = v_2$,TRUE))

t_1 = ($sensor_1$,$< A_1 = 15, A_2 = 20 >$,2:00:00AM)

t_2 = ($sensor_1$,$< A_1 = 10, A_2 = 20 >$,2:05:00AM)

Mapping o_1 to tuple t_1

1. The valuation from source description $(source = l, exp_l)$ to source l_k

   $\nu(sensor_1) = sensor_1$

   $\nu(exp_l) =$ TRUE

   Conditions in 1. are satisfied.

2. The valuation from data description (d, exp_d) to data instance d_k

   $\nu(v_1) = 15$

   $\nu(20) = 20$

   $\nu(exp_d) = \nu(v_1)<20$ AND $10<\nu(v_1) \rightarrow 15<20$ AND $10<15$

   $\nu(exp_d) =$ TRUE

   Conditions in 2. are satisfied.

3. The valuation from tuple description $(timestamp = ts, exp_{ts})$ to timestamp ts_k

   $\nu(v_2) =$ 2:00:00AM

   $\nu(exp_{ts}) =$ TRUE

   Conditions in 3. are satisfied.

All valuations are satisfied. t_1 is a member of o_1

Mapping o_1 to tuple t_2

1. The valuation from source description $(source = l, exp_l)$ to source l_k

   $\nu(sensor_1) = sensor_1$

   $\nu(exp_l) =$ TRUE

   Conditions in 1. are satisfied.

2. The valuation from data description (d, exp_d) to data instance d_k

   $\nu(v_1) = 10$

   $\nu(20) = 20$

   $\nu(exp_d) = \nu(v_1)<20$ AND $10<\nu(v_1) \rightarrow 10<20$ AND $10<10$

$\nu(exp_d)$ = FALSE

Condition in 2. is not satisfied.

3. The valuation from tuple description

($timestamp = ts, exp_{ts}$) to timestamp ts_k

$\nu(v_2)$ = 2:05:00AM

$\nu(exp_{ts})$ = TRUE

Conditions in 3. are satisfied.

Valuation 2. is not satisfied. t_2 is not a member of o_1

Definition 24. Security Label Assignments

Security label assignment is a mapping from a protected object o to a security label *Label*. Let λ a security mapping, a security label assignment is a mapping such that $\lambda : o \longrightarrow Label$. We denoted it as $\lambda(o) = Label$.

Definition 25. Instance Label Assignment

Let t be a data stream tuple and $N(o_1), ..., N(o_k)$ be satisfiable pattern mapping from protected objects $o_1, ..., o_k$ to t, where

$\lambda(o_1), ..., \lambda(o_k)$ are the security labels of $o_i | i \in \{1, ..., k\}$. We say that the security label of t is $LUB(\lambda(o_i)) | i \in \{1, ..., k\}$ where $LUB(\lambda(o_i))$ is the lowest upper bound of the security labels $\lambda(o_1), ..., \lambda(o_k)$.

Definition 26. Object Dominance

Let o_1 and o_2 be protected objects. We say that o_1 dominates o_2 if for every tuple t, there is a valuation $\nu(o_2) = t$, there must be a valuation $\nu(o_1)$ such that $\nu(o_1) = t$. We denote o_1 dominates o_2 as $o_1 \supseteq_d o_2$. Object dominance is reflective, transitive, and asymmetric, i.e., if $o_1 \supseteq_d o_2$ and $o_2 \supseteq_d o_3$ then $o_1 \supseteq_d o_3$.

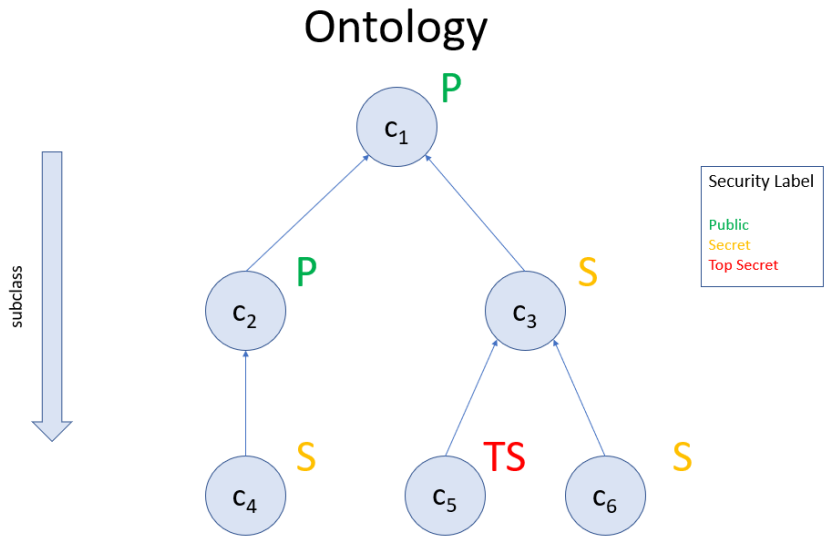

Figure 6.2 Ontology with Security Labels

Definition 27. Object Label Conflict Resolution

Let o_1 and o_2 protected objects such that $o_1 \supseteq_d o_2$ then $\lambda(o_1) \leq \lambda(o_2)$.

Algorithm 4 Assign security label to stream tuple

Input: Stream tuple t, and security labels of all protected objects $\lambda(o_1), ..., \lambda(o_n)$
Output: Label assignment to a stream tuple

1: Initialization
2: Find all pattern mapping $N_1, ..., N_k$ from O (all protected objects) that is satisfied by t
3: Let $\{o_i, ..., o_k\} \in O$ be the protected objects that can be mapped by $N_1, ..., N_k$ to t
4: Let $\lambda(t) = LUB(\lambda(o_i), ..., \lambda(o_k))$

Example 6.4 Security Label Assignment to Stream Tuple

Security Labels: TopSecret(TS) > Secret(S) > Public(P)

t_1 = ((*sensor*$_1$, TRUE), $< A_1 = 20, A_2 = 20 >$, 2:00:00AM)

CASE 1:

o_1 = ((source=*sensor*$_1$,TRUE),($< A_1 = v_1, A_2 = 20 >, v_1 < 50$),

$(timestamp = v_2$,TRUE))

o_2 = ((source=$sensor_1$,TRUE) ,($< A_1 = v_1, A_2 = 20 >, v_1 < 10$) ,

$(timestamp = v_2$,TRUE))

$\lambda(o_1) = Secret$

$\lambda(o_2) = TopSecret$

Assign security label to t_1

t_1 is mapped to o_1.

Label of $t_1 = \lambda(o_1)$ = Secret.

CASE 2:

o_1 = ((source=$sensor_1$,TRUE) ,($< A_1 = v_1, A_2 = 20 >, v_1 < 50$) ,

$(timestamp = v_2$,TRUE))

o_2 = ((source=$sensor_1$,TRUE) ,($< A_1 = v_1, A_2 = 20 >, v_1 < 30$) ,

$(timestamp = v_2$,TRUE))

$\lambda(o_1) = Secret$

$\lambda(o_2) = TopSecret$

Assign security label to t_1

t_1 is mapped to both o_1 and o_2.

$LUB(\lambda(o_1), \lambda(o_2)) = \lambda(o_2)$

Label of $t_1 = \lambda(o_2)$ = Top Secret.

Next, we propose the concept of a complex protected object.

Definition 28. Complex Protected Object

Let $o_1, ..., o_n$ be the atomic protected objects. We say that data stream $S =< o_1, ..., o_n >$ or stream bundle $B = \{o_1, ..., o_n\}$ is a complex protected object O_c iff $\lambda(O_c) = \lambda(S) > LUB(\lambda(o_i))|i \in \{1, ..., n\}$ or $\lambda(O_c) = \lambda(B) > LUB(\lambda(o_i))|i \in \{1, ..., n\}$.

O_c is minimal, that is for any o_i, $\bar{O_c}$ = O_c - o_i then $\lambda(\bar{O_c}) \not> LUB(\lambda(o_j))|j \in (\{1, ..., n\} - \{i\})$. And there are no two objects o_i and o_j such that $o_i \supseteq_d o_j$

Theorem 4. Given a stream tuple t and security labels of all protected objects $\lambda(o_1), ..., \lambda(o_n)$. Algorithm 4 correctly assign a security label to a stream tuple t such that the assigned label l is the LUB of all labels of security objects that can be mapped to t.

Proof. By contradiction;
Let $\lambda(o_k)$ be a security label assigned to a stream tuple t. Assume that $\lambda(o_k)$ does not satisfied the security requirement of t.
Case 1: o_k cannot be mapped to t

1. From algorithm 4, o_k must be a protected object that can be mapped to t.
2. So t must be a member of o_k. This contradict the supposition that t is not a member of o_k.

Case 2: Security label $\lambda(o_k)$ is lower than the security requirement of t

1. There must exist a o_t such that $\nu(o_t) > \nu(o_k)$ and there is a pattern mapping from o_t to t, $\nu(o_t) = t$.
2. But the algorithm 4, $\lambda(t)$ is assigned by
   $LUB(\lambda(o_i), ..., \lambda(o_t), ..., \lambda(o_j))$ where $\{o_i, ..., o_j\}$ be the protected objects that can be mapped to t.
3. So $\lambda(o_k)$ must be $LUB(\lambda(o_i), ..., \lambda(o_j))$ which is the highest security labels of protected objects mapped to t. This contradict the supposition that $\lambda(o_k)$ is lower than the security requirement of t

Case 3: Security label $\lambda(o_k)$ is higher than the security requirements of t

1. From algorithm 4, $\lambda(t)$ is assigned by $LUB(\lambda(o_i), ..., \lambda(o_j))$ where $\{o_i, ..., o_j\}$ be the protected objects that can be mapped to t.

2. So $\lambda(o_k)$ must be $LUB(\lambda(o_i), ..., \lambda(o_j))$ which is the highest security labels of protected objects mapped to t. This contradict the supposition that $\lambda(o_k)$ is higher than the security requirement of t

□

6.2 Proposed Solution

6.2.1 Stream Data and Security Ontology

We expand the stream data ontology introduced by Kolozali et al. [53], adding data security. Traditional access control models consider security to be a passive property of data quality. However, the security is a complex entity which associates to many parts of stream data, e.g., security capabilities that devices can provide, security associated role of users, security property of data. We built an ontology containing six main modules; Stream Data, Device, Quality, User, Event, and Security.

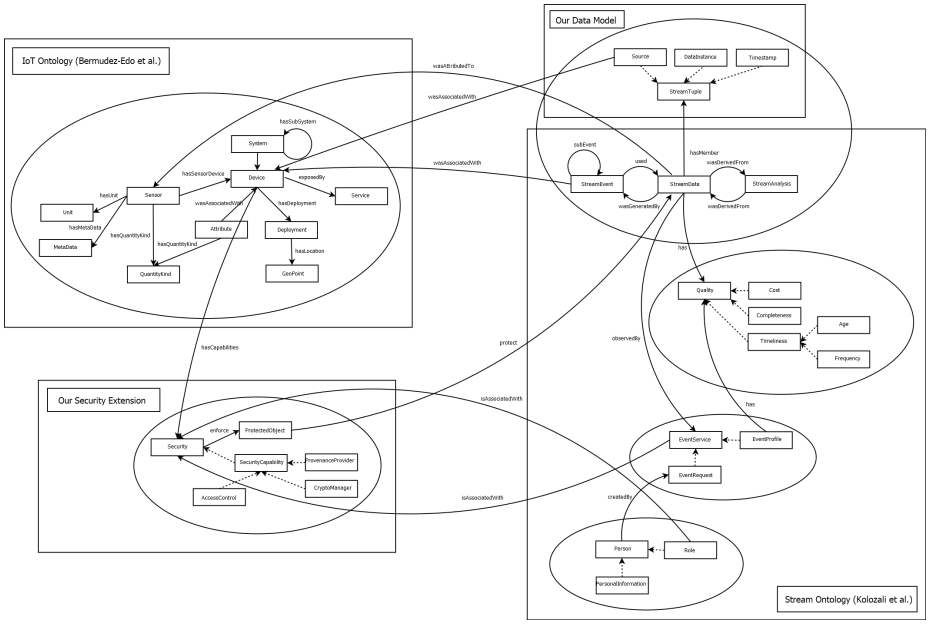

Figure 6.3 Stream Ontology

Stream ontology is generalized so it can represent stream data environment. We extend the stream ontology to represent the context of the stream data. The transmitted data may have different meanings in different contexts. We developed additional specific context ontologies for the data domain of each individual system. We further extend the stream ontology with the context ontology.

We present an extensible context ontology for modeling context in stream data environments. We follow the context model from CONON [54] and divide our context

ontology into upper ontology and specific ontology. Our upper ontology contains all basic contextual entities from CONON with additional entities from CoDAMoS [55] and Smart Space [56]. The upper ontology describes 6 basic concepts; CompEntity, Location, Person, Activity, Environment, and Time. The specific ontology represents the details of general concepts in each sub-domain.

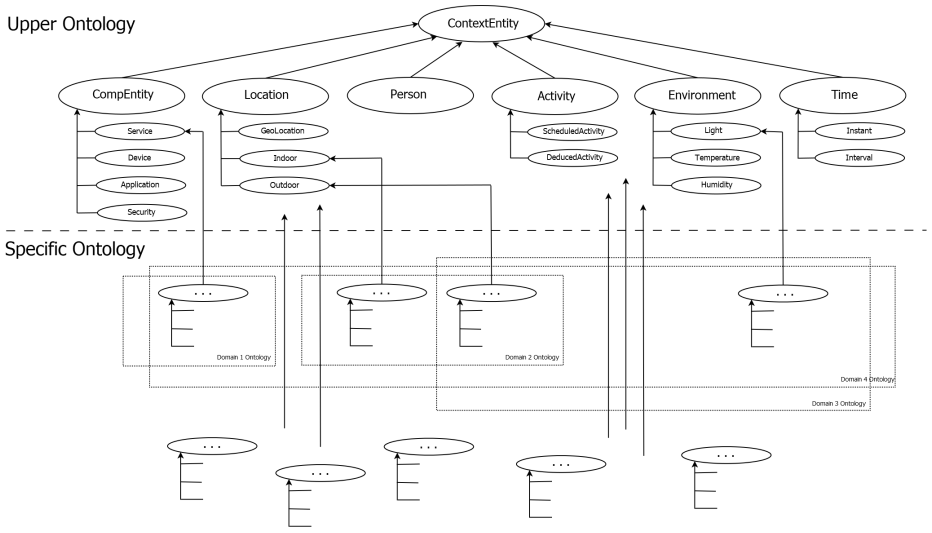

Figure 6.4 Context Ontology

6.2.2 Context Reasoning

To illustrate the logical reasoning mechanism, we present a scenario in which the data generated from sensors can have different level of security policies depending on the current situation. For example, when people are sleeping, their oxygen saturation levels will naturally decrease because their body needs less oxygen. Their bodies will reduce the respiration rate and heart rate during sleep. While the oxygen saturation will also decrease during exercise, this happens due to the body consuming more oxygen. Despite having low oxygen saturation, the respiration rate and heart rate will increase during exercise. An oxygen saturation value during rest that is below 90% is considered a medical condition called hypoxemia. However, an oxygen saturation

value 88% can be considered normal during intense exercise. In this case, the oxygen saturation alone cannot determine the condition of hypoxemia; the respiration rate and heart rate must also be considered. We want to provide dynamic security policy to protect patient A's privacy by limiting access during normal situations but allowing access during emergencies. By using the context and ontology rules, we can derive that the current oxygen saturation is normal or is indicating an emergency situation. The security label assignments can be defined as $\lambda(o_1) = TopSecret$ and $\lambda(o_2) = Public$ where o_1 is an oxygen saturation during a normal situation and o_2 is an oxygen saturation during an emergency situation. The protected objects can be separated into 2 cases; rest and exercise and be defined as the following example;

Example 6.5 Protected Objects in Different Situations

A protected object o_{1a} representing a normal situation during resting

o_{1a} = ((source=$patient_A$,TRUE), ($<$ $oxygenSaturation = v_1$, $respirationRate = v_2$, $heartRate = v_3 >$, $90 < v_1$), $*$)

A protected object o_{1b} representing a normal situation during exercising

o_{1b} = ((source=$patient_A$,TRUE), ($<$ $oxygenSaturation = v_1$, $respirationRate = v_2$, $heartRate = v_3 >$, $88 < v_1$ AND $40 < v_2$ AND $150 < v_3$), $*$)

A protected object o_{2a} representing an emergency situation during resting

o_{2a} = ((source=$patient_A$,TRUE), (<$oxygenSaturation = v_1$, $respirationRate = v_2$, $heartRate = v_3$ >, $v_1 < 90$ AND $v_2 < 40$ AND $v_3 < 150$), *)

A protected object o_{2b} representing an emergency situation during exercising

o_{2b} = ((source=$patient_A$,TRUE), (<$oxygenSaturation = v_1$, $respirationRate = v_2$, $heartRate = v_3$ >, $v_1 < 88$ AND $40 < v_2$ AND $150 < v_3$), *)

We will use Protégé[51] and Semantic Web Rule Language (SWRL) to derive the security policies. SWRL allows us to write reasoning rules to assign security labels where protected object $o = ((source = l, exp_l), (d, exp_d),$ $(timestamp = ts, exp_{ts}))$ with label assignment $\lambda(o) = Label$ in the form as following;

$$StreamTuple(?\,t) \wedge hasSource(?\,t, N(source = l, exp_l)) \wedge hasDataInstance(?\,t, N(d, exp_d)) \wedge hasTimestamp(?\,t, N(timestamp = ts, exp_{ts})) \rightarrow hasLabel(?\,t, Label)$$

Where $N(source = l, exp_l)$, $N(d, exp_d)$, and $N(timestamp = ts, exp_{ts})$ can be derived from;

$$ProtectedObject(?\,o) \wedge source(?\,o, ?\,source) \wedge dataInstance(?\,o, ?\,data) \wedge timestamp(?\,o, ?\,timestamp) \wedge StreamTuple(?\,t) \wedge hasSource(?\,t, ?\,source) \wedge hasDataInstance(?\,t, ?\,data) \wedge hasTimestamp(?\,t, ?\,timestamp)$$

6.3 Implementation

In our previous work [57], we implemented our context-aware security policy engine. We implemented our framework using Protégé [51] and Semantic Web Rule Language (SWRL) on an open-source home automation called Home Assistant [52]. Figure 6.5

shows a simplified ontology containing StreamTuple and ProtectedObject and figure 6.6 shows a protected object with simple elements describing the source, data instance, timestamp, and security label. SWRL allows us to write reasoning rules to assign security labels where protected object $o = ((source = l, exp_l), (d, exp_d), (timestamp = ts, exp_{ts}))$ with label assignment $\lambda(o) = Label$ in the form as following;

$$\begin{aligned} &StreamTuple(?\,t) \wedge hasSource(?\,t, N(source = l, exp_l)) \\ &\quad \wedge hasDataInstance(?\,t, N(d, exp_d)) \\ &\quad\quad \wedge hasTimestamp(?\,t, N(timestamp = ts, exp_{ts})) \rightarrow hasLabel(?\,t, Label) \end{aligned} \tag{6.1}$$

Where $N(source = l, exp_l)$, $N(d, exp_d)$, and $N(timestamp = ts, exp_{ts})$ can be derived from;

$$\begin{aligned} &ProtectedObject(?\,o) \wedge source(?\,o, ?\,source) \\ &\quad \wedge dataInstance(?\,o, ?\,data) \wedge timestamp(?\,o, ?\,timestamp) \\ &\quad \wedge StreamTuple(?\,t) \wedge hasSource(?\,t, ?\,source) \\ &\quad \wedge hasDataInstance(?\,t, ?\,data) \wedge hasTimestamp(?\,t, ?\,timestamp) \end{aligned} \tag{6.2}$$

We show the SWRL rule to assign a security label in figure 6.7 and the result where the label is assigned to a stream tuple in figure 6.8

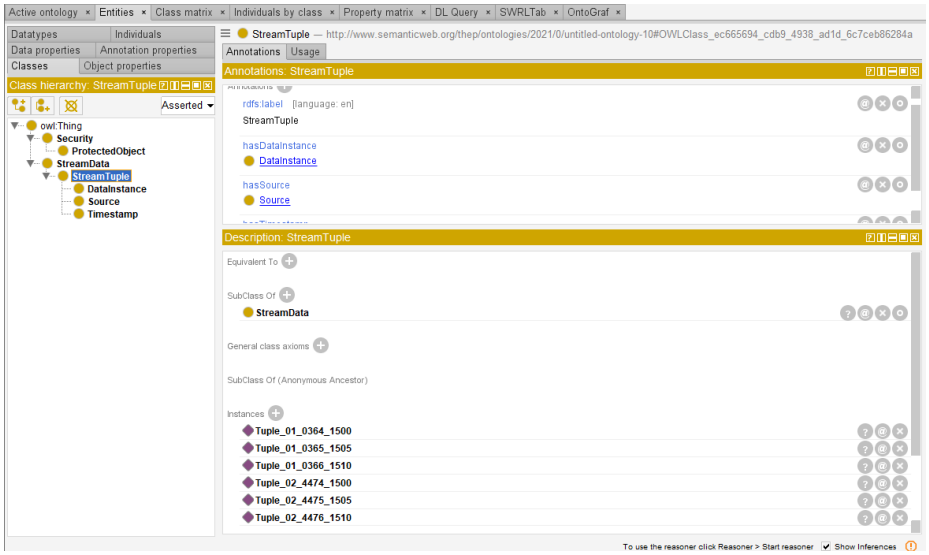

Figure 6.5 The Stream Ontology in the Protégé

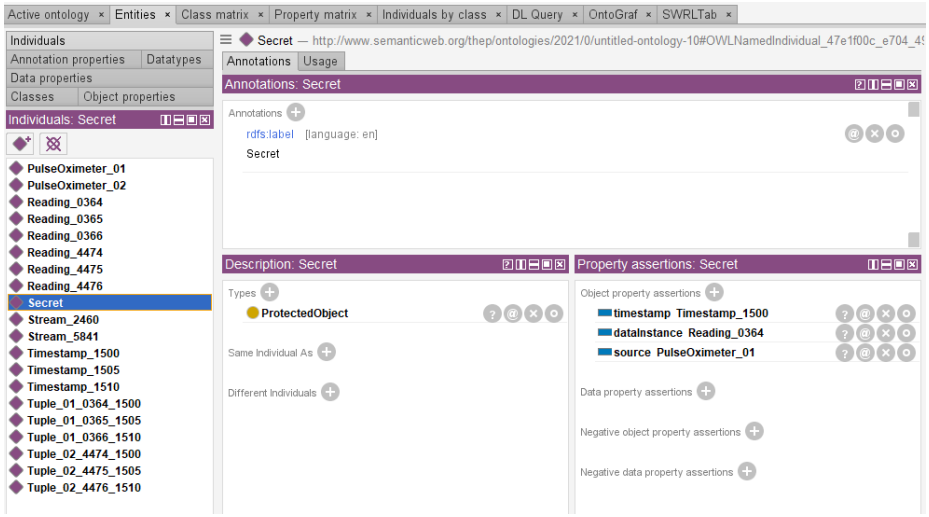

Figure 6.6 The Protected Object in the Protégé

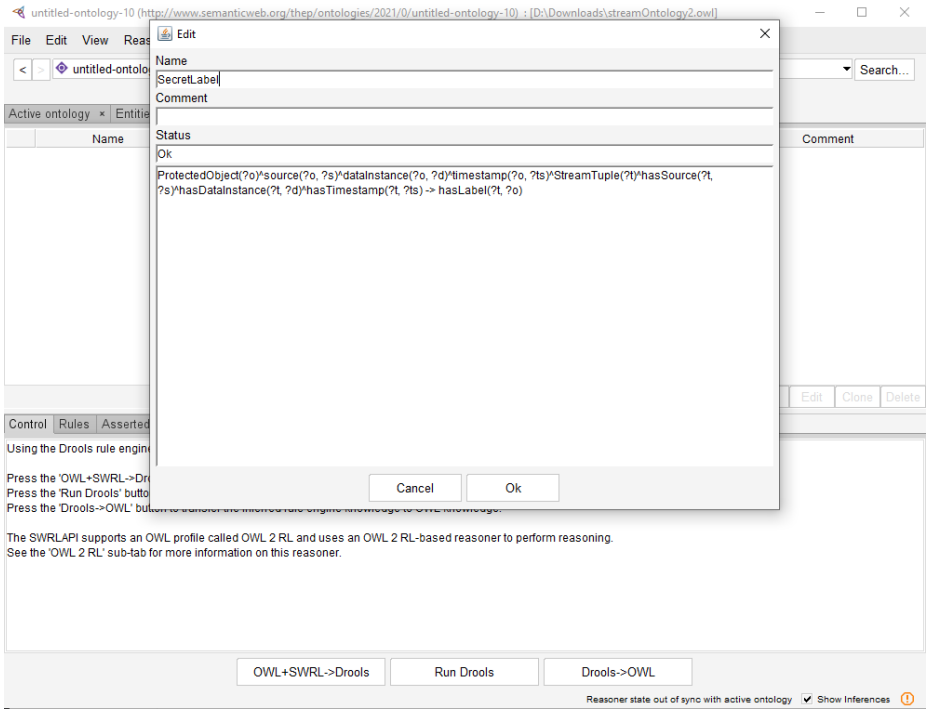

Figure 6.7 Label assignment with SWRL in the Protégé

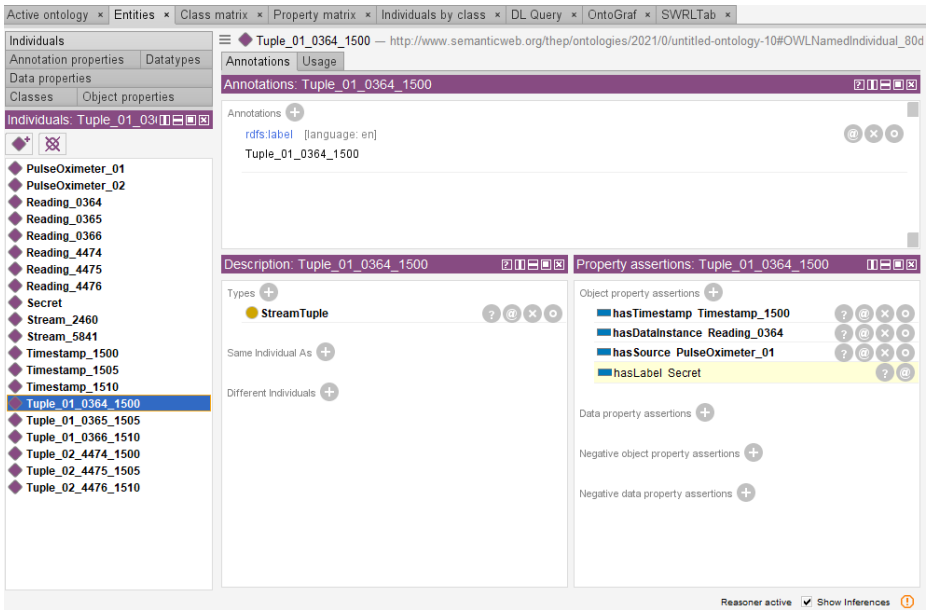

Figure 6.8 Reasoning result from label assignment

We use XML to facilitate data exchange between Home Assistant and Protégé. This approach makes it possible to extend data sharing for future applications. Figure 5.6 outlines how the data is extracted from the Home Assistant database, converted to the XML schema, and shared with Protégé. Similarly, the decision reached by Protégé is converted to the XML format, and incorporated into the Home Assistant database.

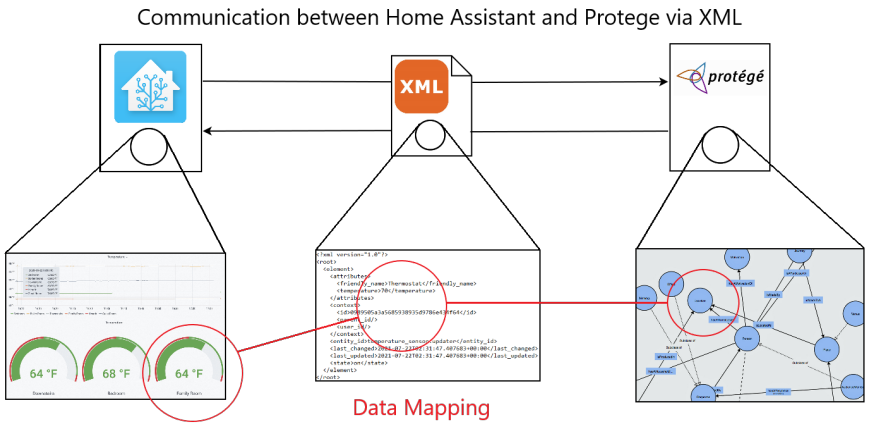

Figure 6.9 Communication between Home Assistant and Protégé via XML [57]

To get data from Protégé into Home Assistant, we converted the CSV file that was exported from Protégé. We created bash scripts to convert data from each side to XML and convert XML to respective data formats for each sides. Our ongoing work addresses the security label enforcement using security punctuation.

Chapter 7

Conclusions & Future Work

In this dissertation, we studied how to prevent an attack targeting databases using malicious transactions. We proposed a recovery method incorporating data provenance. We also studied how to express security and privacy needs of data in the Internet of Things (IoT) environments such as a smart home environment. We proposed a dynamic access control model based on context-aware architecture and stream data model. We demonstrate our proof-of-concept implementation using a commercial and open-source home automaton software.

To extend our work further, we will focus on how to incorporate multiple systems together. As our work focuses on securing a smart home. We can extend our approach by integrating smart homes together. Our goal is developing a system for a smart city. Many researchers have been focused on developing semantics-based privacy protection for a smart city[58][59]. However, they are the top-down models. In most works, the main system decides the security policy without getting feedback from each nodes. In reality, you and your neighbors can have different security requirements despite being in the same neighborhood at the same time. For example, you want to go travel and prefer to have polices observe your house during that time. You would like to share your security camera footage to polices while your neighbor would not want to share it. We can see that the top-down models which the main system decide all the policies are not flexible enough to support this requirements. Our future work will focus on developing a smart city framework using a bottom-up model. In this way, each nodes can express their security requirements. To achieve our goal, we have

to develop techniques to merge different ontologies together while preserving security requirements of each ontologies. To achieve it, we plan to develop 1) policy integration of multiple systems, 2) verification of consistent integrated policy, 3) practical analysis of the proposed approach in the context of city development.

Bibliography

[1] IBM Security, *Cost of a data breach report 2020*. [Online]. Available: https://www.ibm.com/security/digital-assets/cost-data-breach-report.

[2] P. Ammann, S. Jajodia, and P. Liu, "Recovery from malicious transactions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 5, pp. 1167–1185, Sep. 2002, ISSN: 2326-3865. DOI: 10.1109/TKDE.2002.1033782.

[3] P. Liu, P. Ammann, and S. Jajodia, "Rewriting histories: Recovering from malicious transactions," in *Security of Data and Transaction Processing*. Springer, 2000, pp. 7–40.

[4] B. Panda and K. A. Haque, "Extended data dependency approach: A robust way of rebuilding database," in *Proceedings of the 2002 ACM symposium on Applied computing*, ser. SAC '02, New York, NY, USA: Association for Computing Machinery, Mar. 2002, pp. 446–452, ISBN: 978-1-58113-445-2. DOI: 10.1145/508791.508875. [Online]. Available: http://doi.org/10.1145/508791.508875 (visited on 02/06/2021).

[5] T. Kim, R. Chandra, and N. Zeldovich, "Recovering from intrusions in distributed systems with DARE," in *Proceedings of the Third ACM SIGOPS Asia-Pacific conference on Systems*, ser. APSys '12, USA: USENIX Association, Jul. 2012, p. 10. (visited on 02/06/2021).

[6] R. Chandra, T. Kim, M. Shah, N. Narula, and N. Zeldovich, "Intrusion recovery for database-backed web applications," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, ser. SOSP '11, New York, NY, USA: Association for Computing Machinery, Oct. 2011, pp. 101–114, ISBN:

978-1-4503-0977-6. DOI: 10.1145/2043556.2043567. [Online]. Available: http://doi.org/10.1145/2043556.2043567 (visited on 02/06/2021).

[7] J. Powell, D. Menon, and J. Jones, "The effects of hypoxaemia and recommendations for postoperative oxygen therapy," *Anaesthesia*, vol. 51, no. 8, pp. 769–772, 1996.

[8] L. He, P. Yue, L. Di, M. Zhang, and L. Hu, "Adding geospatial data provenance into sdi—a service-oriented approach," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 2, pp. 926–936, 2014.

[9] O. Q. Zhang, R. K. Ko, M. Kirchberg, C. H. Suen, P. Jagadpramana, and B. S. Lee, "How to track your data: Rule-based data provenance tracing algorithms," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, IEEE, 2012, pp. 1429–1437.

[10] M. Backes, N. Grimm, and A. Kate, "Data lineage in malicious environments," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 178–191, 2015.

[11] I. T. Foster, J.-S. Vöckler, M. Wilde, and Y. Zhao, "The virtual data grid: A new model and architecture for data-intensive collaboration.," in *CIDR*, 2003.

[12] S. Osborn, "Mandatory access control and role-based access control revisited," in *Proceedings of the second ACM workshop on Role-based access control*, 1997, pp. 31–40.

[13] S. Osborn, R. Sandhu, and Q. Munawer, "Configuring role-based access control to enforce mandatory and discretionary access control policies," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 2, pp. 85–106, 2000.

[14] M. J. Covington, W. Long, S. Srinivasan, A. K. Dev, M. Ahamad, and G. D. Abowd, "Securing context-aware applications using environment roles," in *Proceedings of the sixth ACM symposium on Access control models and technologies*, 2001, pp. 10–20.

[15] D. Kulkarni and A. Tripathi, "Context-aware role-based access control in pervasive computing systems," in *Proceedings of the 13th ACM symposium on Access control models and technologies*, 2008, pp. 113–122.

[16] B. Carminati, E. Ferrari, and M. Guglielmi, "Detection of unspecified emergencies for controlled information sharing," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 6, pp. 630–643, 2015.

[17] G. Manogaran, C. Thota, D. Lopez, and R. Sundarasekar, "Big data security intelligence for healthcare industry 4.0," in *Cybersecurity for Industry 4.0*, Springer, 2017, pp. 103–126.

[18] Y. Mengke, Z. Xiaoguang, Z. Jianqiu, and X. Jianjian, "Challenges and solutions of information security issues in the age of big data," *China Communications*, vol. 13, no. 3, pp. 193–202, 2016.

[19] A. Penrig, D. Song, and D. Tygar, "Elk, a new protocol for efficient large-group key distribution," in *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*, IEEE, 2000, pp. 247–262.

[20] D. Puthal, S. Nepal, R. Ranjan, and J. Chen, "A dynamic key length based approach for real-time security verification of big sensing data stream," in *International conference on web information systems engineering*, Springer, 2015, pp. 93–108.

[21] D. Puthal, X. Wu, S. Nepal, R. Ranjan, and J. Chen, "Seen: A selective encryption method to ensure confidentiality for big sensing data streams," *IEEE Transactions on Big Data*, 2017.

[22] L. Veltri, S. Cirani, S. Busanelli, and G. Ferrari, "A novel batch-based group key management protocol applied to the internet of things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2724–2737, 2013.

[23] S. Zhu, S. Setia, and S. Jajodia, "Leap+ efficient security mechanisms for large-scale distributed sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 4, pp. 500–528, 2006.

[24] R. V. Nehme, E. A. Rundensteiner, and E. Bertino, "A security punctuation framework for enforcing access control on streaming data," in *2008 IEEE 24th International Conference on Data Engineering*, IEEE, 2008, pp. 406–415.

[25] D. Le-Phuoc, M. Dao-Tran, J. X. Parreira, and M. Hauswirth, "A native and adaptive approach for unified processing of linked streams and linked data," in *International Semantic Web Conference*, Springer, 2011, pp. 370–388.

[26] K. Whitehouse, F. Zhao, and J. Liu, "Semantic streams: A framework for composable semantic interpretation of sensor data," in *European Workshop on Wireless Sensor Networks*, Springer, 2006, pp. 5–20.

[27] A. Sabelfeld and A. C. Myers, "Language-based information-flow security," *IEEE Journal on selected areas in communications*, vol. 21, no. 1, pp. 5–19, 2003.

[28] C. Choi, J. Choi, and P. Kim, "Ontology-based access control model for security policy reasoning in cloud computing," *The Journal of Supercomputing*, vol. 67, no. 3, pp. 711–722, 2014.

[29] A. H. Celdrán, F. J. G. Clemente, M. G. Pérez, and G. M. Pérez, "Secoman: A semantic-aware policy framework for developing privacy-preserving and context-aware smart applications," *IEEE Systems Journal*, vol. 10, no. 3, pp. 1111–1124, 2014.

[30] H. Chen, T. Finin, and A. Joshi, "An ontology for context-aware pervasive computing environments," *The knowledge engineering review*, vol. 18, no. 3, pp. 197–207, 2003.

[31] R. Montanari, A. Toninelli, and J. M. Bradshaw, "Context-based security management for multi-agent systems," in *IEEE 2nd Symposium on Multi-Agent Security and Survivability, 2005.*, IEEE, 2005, pp. 75–84.

[32] R. Lu, X. Lin, and X. Shen, "Spoc: A secure and privacy-preserving opportunistic computing framework for mobile-healthcare emergency," *IEEE transactions on parallel and distributed systems*, vol. 24, no. 3, pp. 614–624, 2012.

[33] Y.-S. Jeong, S.-H. Lee, and S.-S. Shin, "Access control protocol based on privacy property of patient in m-healthcare emergency," *Wireless personal communications*, vol. 79, no. 4, pp. 2565–2578, 2014.

[34] C.-C. Lee, C.-W. Hsu, Y.-M. Lai, and A. Vasilakos, "An enhanced mobile-healthcare emergency system based on extended chaotic maps," *Journal of medical systems*, vol. 37, no. 5, p. 9973, 2013.

[35] W. Yu, Z. Liu, C. Chen, B. Yang, and X. Guan, "Privacy-preserving design for emergency response scheduling system in medical social networks," *Peer-to-Peer Networking and Applications*, vol. 10, no. 2, pp. 340–356, 2017.

[36] D. Harwell, "Police can keep ring camera video forever and share with whomever they'd like, amazon tells senator," *The Washington Post*, Nov. 19, 2019. [Online]. Available: `https://www.washingtonpost.com/technology/2019/11/19/police-can-keep-ring-camera-video-forever-share-with-whomever-theyd-like-company-tells-senator/` (visited on 02/17/2020).

[37] R. Alharbi and D. Aspinall, "An iot analysis framework: An investigation of iot smart cameras' vulnerabilities," in *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, IET, 2018.

[38] B. Janes, H. Crawford, and T. OConnor, "Never ending story: Authentication and access control design flaws in shared iot devices," in *2020 IEEE Security and Privacy Workshops (SPW)*, IEEE, 2020, pp. 104–109.

[39] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi, "Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach," in *Proceedings of the Internet Measurement Conference*, 2019, pp. 267–279.

[40] A. Lounis, A. Hadjidj, A. Bouabdallah, and Y. Challal, "Healing on the cloud: Secure cloud architecture for medical wireless sensor networks," *Future Generation Computer Systems*, vol. 55, pp. 266–277, 2016.

[41] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, IEEE, 2017, pp. 468–477.

[42] D. K. Tosh, S. Shetty, X. Liang, C. Kamhoua, and L. Njilla, "Consensus protocols for blockchain-based data provenance: Challenges and opportunities," in *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, IEEE, 2017, pp. 469–474.

[43] H. M. Kim and M. Laskowski, "Toward an ontology-driven blockchain design for supply-chain provenance," *Intelligent Systems in Accounting, Finance and Management*, vol. 25, no. 1, pp. 18–27, 2018.

[44] Y. Mowafi, D. Abou-Tair, T. Aqarbeh, M. Abilov, V. Dmitriyev, and J. M. Gomez, "A context-aware adaptive security framework for mobile applications," in *Proceedings of the 3rd International Conference on Context-Aware Systems and Applications*, ser. ICCASA '14, Dubai, United Arab Emirates: ICST (In-

stitute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014, pp. 147–153, ISBN: 9781631900051.

[45] S. Yi, "Situation-aware security for wireless ad hoc networks," AAI3199187, Ph.D. dissertation, USA, 2005, ISBN: 0542449064.

[46] L. Chen, C. Nugent, and A. Al-Bashrawi, "Semantic data management for situation-aware assistance in ambient assisted living," ser. iiWAS '09, Kuala Lumpur, Malaysia: Association for Computing Machinery, 2009, ISBN: 9781605586601. DOI: 10.1145/1806338.1806394. [Online]. Available: https://doi.org/10.1145/1806338.1806394.

[47] M. Tavakolifard, "Situation-aware trust management," ser. RecSys '09, New York, New York, USA: Association for Computing Machinery, 2009, ISBN: 9781605584355. DOI: 10.1145/1639714.1639802. [Online]. Available: https://doi.org/10.1145/1639714.1639802.

[48] B. Cheng, M. Wang, S. Zhao, Z. Zhai, D. Zhu, and J. Chen, "Situation-aware dynamic service coordination in an iot environment," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2082–2095, Aug. 2017, ISSN: 1063-6692. DOI: 10.1109/TNET.2017.2705239. [Online]. Available: https://doi.org/10.1109/TNET.2017.2705239.

[49] G. O'Driscoll, *Essential Guide to Smart Home Automation Safety & Security: Use Home Automation to Increase Your Families Safety Levels (Smart Home Automation Essential Guides) - Volume 3*, 1st. North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2015, ISBN: 150870127X.

[50] R. Leitão, "Anticipating smart home security and privacy threats with survivors of intimate partner abuse," in *Proceedings of the 2019 on Designing Interactive Systems Conference*, ser. DIS '19, San Diego, CA, USA: Association for Computing Machinery, 2019, pp. 527–539, ISBN: 9781450358507. DOI:

10.1145/3322276.3322366. [Online]. Available: https://doi.org/10.1145/3322276.3322366.

[51] Stanford, *Protégé*, 2022. [Online]. Available: https://protege.stanford.edu/.

[52] Home Assistant, *Open source home automation*, http://www.home-assistant.io/, 2021.

[53] S. Kolozali, M. Bermudez-Edo, D. Puschmann, F. Ganz, and P. Barnaghi, "A knowledge-based approach for real-time iot data stream annotation and processing," in *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*, IEEE, 2014, pp. 215–222.

[54] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung, "Ontology based context modeling and reasoning using owl," in *IEEE annual conference on pervasive computing and communications workshops, 2004. Proceedings of the second*, Ieee, 2004, pp. 18–22.

[55] D. Preuveneers, J. Van den Bergh, D. Wagelaar, *et al.*, "Towards an extensible context ontology for ambient intelligence," in *European Symposium on Ambient Intelligence*, Springer, 2004, pp. 148–159.

[56] W. Qin, Y. Shi, and Y. Suo, "Ontology-based context-aware middleware for smart spaces," *Tsinghua Science and Technology*, vol. 12, no. 6, pp. 707–713, 2007.

[57] T. Rhujittawiwat, C. Anderson, D. Keen, *et al.*, "Making smart platforms smarter: Adding third party applications to home automation platforms," *Journal of Computing Sciences in Colleges*, vol. 37, no. 5, pp. 43–53, 2021.

[58] M. Gheisari, H. E. Najafabadi, J. A. Alzubi, *et al.*, "Obpp: An ontology-based framework for privacy-preserving in iot-based smart city," *Future Generation Computer Systems*, vol. 123, pp. 1–13, 2021.

[59] N. Seydoux, K. Drira, N. Hernandez, and T. Monteil, "Iot-o, a core-domain iot ontology to represent connected devices networks," in *European knowledge acquisition workshop*, Springer, 2016, pp. 561–576.