Stream Data Access Control Model: The Need for Data Semantics

Theppatorn Rhujittawiwat1, Ahmed Saaudi2 and Csilla Farkas1

1University of South Carolina, Columbia, South Carolina, U.S.A.
2University of Al-Muthanna, Samawah, Iraq

Keywords: Stream Data, Ontologies, Dynamic Access Control, Semantics-based Authorization.

Abstract: In this paper, we propose 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 Things. We propose an authorization model using ontologies and rules to express security requirements for stream data. Our model supports secure interoperation and it is independent from the data syntax. We propose security object patterns to express access control needs. These patterns are associated with ontological concepts. Data instances inherit the security protection needs via stream pattern mapping. Security object patterns can incorporate domain, context, and temporal security restrictions. We show that our model is resistant to attacks that aim to bypass security restrictions by reconstructing stream data.

1 INTRODUCTION

Stream data management is an important requirement for emerging computing environments, such as the Internet of Things (IoT). Initiatives, such as IoT-Lite (Bermudez-Edo et al., 2015)(Bermudez-Edo et al., 2016) and Stream Annotation Ontology (SOA) (Kolozali et al., 2014), aim to support intelligent interoperation. Along with the development of IoT applications, concerns about cybersecurity threats emerged. In this paper, we address the specific cybersecurity needs of stream data in the IoT environment. We argue that current stream data security models do not support dynamic security, thus, insufficient to support IoT applications.

During the last decade, several researchers addressed the need to provide data security for stream data. Most of these models address confidentiality protection for stream data. The proposed solutions range from encryption-based end-to-end communication security (Penrig et al., 2000)(Puthal et al., 2015)(Puthal et al., 2017)(Veltri et al., 2013)(Zhu et al., 2006) to proposing generic security models (Carminati et al., 2015)(Manogaran et al., 2017)(Mengke et al., 2016)(Nehme et al., 2008). Communication security for stream data is widely provided by end-to-end encryption during transmission. To further improve the performance of this method, many researchers introduced efficient end-to-end encryption techniques such as the dynamic key length based approach (Puthal et al., 2015) and approaches for grouping and sharing keys (Penrig et al., 2000)(Puthal et al., 2017)(Veltri et al., 2013)(Zhu et al., 2006). However, end-to-end encryption techniques provide confidentiality over the entire data stream and are unable to provide fine granularity authorization. These techniques are cumbersome to enforce advanced security requirements, such as role-based access control, context-aware security, and dynamic security policies.

Several researchers proposed access control policy models for stream data (Cao et al., 2009)(Carminati et al., 2010)(Carminati et al., 2007a)(Carminati et al., 2007b)(Thoma et al., 2019). These approaches adopt access control concepts of traditional database management systems (DBMSs) to stream data. However, these efforts do not sufficiently address the specific needs of data stream management systems (DSMSs). For example, a DSMS can access only incoming data while a DBMS can access the complete data. The operator scheduling in a DSMS significantly impacts system performance. Operator scheduling strategies have been proposed by Carminati et al. (Carminati et al., 2010) and by Thoma et al. (Thoma et al., 2019).

An enforcement of security requirements over the stream data instance has been proposed by Nehme et al. (Nehme et al., 2008). The security requirements are assigned to each attribute by their positions in the stream tuple. Their technique, called security punctuation, provides efficient access control for stream data. However, since the security punctuation is based on the syntax of the stream data, this approach provides limited interoperability. Moreover, modifica-
tion of the stream syntax may allow the attacker to bypass the access control restrictions.

Malicious corruption of the integrity of stream data was addressed by Guo et al. (Guo et al., 2007). The authors proposed a watermarking method to detect malicious modification of the stream data. The authors introduced the concept of data stream completeness. While their methods are able to detect modification of the data stream, they do not provide a model to express protection requirements.

In this paper, we present a security model to express security needs for stream data. Our work is based on the intersection of stream data semantics modeling and semantics-based authorization models. Data semantics is widely used to support stream data integration and data analytics (Le-Phuoc et al., 2011)(Whitehouse et al., 2006). Our solution was also influenced by work to support semantics-based authorization models (Sabelfeld and Myers, 2003)(Choi et al., 2014) and context-based decision support (Celdrán et al., 2014)(Chen et al., 2003)(Montanari et al., 2005).

The main contributions of our work are to 1) express security requirements via stream data semantics and 2) support dynamic and flexible policy management. Our focus is on the concept of protected object patterns. These patterns represent (partial) stream tuples and their combinations. Values for the tuple attributes may be variables or constants, thus allowing for the expression of security needs over varying levels of granularity. We model the mapping from the protection objects to the data instances via the corresponding ontologies. A protection object may be applicable to multiple ontology concepts and a concept may have multiple protection objects mapped to it. Stream data instances inherit the security requirements from the security object patterns. We show that our security label assignment provides strong confidentiality protection. Syntactically different but semantically same data items will have the same security protection assignment. In this paper, we present our functional model. We also give proof of formal properties of our security label assignment.

The organization of the paper is as follows: Section 3.2 discusses the motivating situations and problems which drive the direction of the solution. Section 4 defines the stream data model and security assignment. Section 5.1 describes the concept of stream data ontology. Section 5.2 illustrates logical reasoning and how to incorporate it into security policy. Section 6 presents an example of security policy implementation using ontology. We conclude in Section 7.

2 RELATED WORK

In this section, we give an overview of current research on stream data security, dynamic access control, and the limitations of these works.

2.1 Stream Data Security

Current work on stream data security focuses on confidentiality (Carminati et al., 2015)(Manogaran et al., 2017)(Mengke et al., 2016). The first priority of stream data design is performance. To provide security capabilities with the least effect on performance, the major methods of providing confidentiality are based on end-to-end cryptography (Pen-rig et al., 2000)(Puthal et al., 2015)(Puthal et al., 2017)(Veltri et al., 2013)(Zhu et al., 2006). 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. (Nehme et al., 2008) 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 the security needs which are suitable for privacy sensitive data domains such as medical information.

The growing field of IoT has led to increasing number of stream data applications. Many of these applications also collect sensitive user data. To protect confidentiality and privacy of users, researchers propose access control for IoT stream data in different context. For example, Shafagh et al. (Shafagh et al., 2020) proposed a decentralized access control based on blockchain technology. Their approach provides an extra layer of anonymity for users but has a limited security granularity. Namibiar et al. (Namibiar et al., 2020) extended Apache Storm to protect privacy of the data in distributed systems. Their approach supports a variety of access control models, such as Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role-Based Access Control (RBAC).

Semantic approaches have been used in stream
data since the field was established. However, their main focus is on how to interpret data and use it for data analytics (Le-Phuoc et al., 2011)(Whitehouse et al., 2006)(Sejdija et al., 2021). The security community has demonstrated the benefit of semantics-based security, such as Sabelfeld et al. (Sabelfeld and Myers, 2003)(Choi et al., 2014)(Celdrán et al., 2014)(Chen et al., 2003)(Montanari et al., 2005). However, the potential of these approaches for supporting stream data security and privacy has not been fully investigated.

Moreover, stream data requires dynamic security policies because it is time sensitive. The same attribute in stream data may require different security levels at different times such as healthcare information in emergencies. Lu et al. (Lu et al., 2012) proposed a framework to provide privacy on mobile health monitoring devices. This approach uses mathematical models on monitored information to detect emergency situations and to release information to qualified helpers. Other mathematical models to express emergency situations were also proposed by many researchers (Jeong et al., 2014)(Lee et al., 2013)(Yu et al., 2017).

2.2 Dynamic Security Policy

Privacy is the major security concern related to commercial IoT devices. Recent research shows that current IoT devices focus on having secure connections between devices and the server (Alharbi and Aspinall, 2018)(Janes et al., 2020)(Ren et al., 2019). However, 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) (Ren et al., 2019). End users, such as smart home users, are often left with little or no control over device security. Moreover, security policies may be dependent on the context. Semantics-based approaches can be used to express the context and situations, and adjust the security policy accordingly.

Carminati et al. (Carminati et al., 2015) propose an approach to provide privacy while working around the current static model, such as using statistic models to detect anomalies in data and granting access to all data if they are requested during the presence of an anomaly. Semantics-based security policies can provide needed dynamic capabilities to protect privacy without hindering emergency management. An architecture that incorporates semantics-based security policy into a stream data model has the potential to provide a dynamic security policy that provides functionalities in emergencies without sacrificing privacy.

2.3 Vulnerabilities of Stream Access Control Enforcement via Syntactic Policy Enforcement

Nehme et al. (Nehme et al., 2013)(Nehme et al., 2008) proposed a security policy for stream data called security punctuation (sp). The security punctuations are inserted by the source of the stream to indicate the protection needs of the stream data. A security punctuation contains the following: punctuation type (pt), data description part (ddp), security restriction part(sp), sign, timestamp (ts), and enforcement (et). The 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 the security needs which are suitable for privacy sensitive data domains, such as medical information.

Example 1: Security Punctuation Policy Examples.

1. Data Security Punctuation

\[ < d d p ( e n f ( t s ) = 1 / 1 : 00 : 00 , t s ) > \]

Only queries registered by role \( R_1 \) can query the stream \( S_1 \).

2. Query Security Punctuation

\[ < q u e r y ( R_1 ) = 1 / 0 : 00 , t s ) > \]

Query acquires a role \( R_1 \).

3. Policy Enforcement

\[ < p o l i c y ( e n f ( t s ) = 1 / 0 : 00 , t s ) > \]

Query acquires a role \( R_2 \) after IPM and the role \( R_1 \) is permitted to only access stream \( S_2 \).

The security punctuation is subject to attacks that can destroy or delay the sp. This is a common vulnerability of syntax-based security policies. Malicious attackers can potentially disclose unauthorized data by changing the position of the target tuple or attribute, thus making the system interpret the security level incorrectly. For example, due to the nature of DSMS, the data can be accessed without authorization if the security policy is not updated with the correct time. The new sp can apply to only tuples which are still in the sliding window and tuples which arrive after sp. However, it cannot apply to tuples which are finished querying and not in the sliding window anymore. In the case that sp is destroyed during transmission, the policy will not be updated. This gives opportunities for attackers to access the unauthorized data with the outdated policy. In the case that sp is delayed during transmission, attackers can access the unauthorized data with the outdated policy until sp arrives. Moreover, if the data items are shuffled, the security punctuation that refers to the data
position in the tuple, may indicate the incorrect secu-
rity requirements. Our work, that describes security
for stream tuples based on data semantics, is indepen-
dent of the actual stream syntax and can avoid these
attacks. Moreover, our approach can be used with the
security punctuation-based implementation to verify
that the security has not been compromised.

3 ACCESS CONTROL
ARCHITECTURE AND
MOTIVATING EXAMPLE

The main focus of this work is to represent syntax-
independent access control requirement for stream
data. In this section, we present the system architec-
ture for our stream data access control and a motivat-
ing example demonstrating the need for a semantics-
based security model.

3.1 Architecture

We propose the use of ontologies to model domain
and context semantics. Security requirements are ex-
pressed over the ontologies and dynamically updated
according to context changes. Figure 1 shows the
overview of our system architecture. The policy part
of architecture contains domain and context ontolo-
gies, the reasoning engine, and the list of security ob-
jects. The policies will be applied on data instances of
the stream. The reasoning engine will derive security
labels from ontologies and assign them to security ob-
jects. The instances matching the security object de-
scriptions will be assigned by security labels of those
security objects.

3.2 Motivating Example

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

In stream data, only a part of the data can be pro-
cessed at any given time via continuous queries. The
data will be processed with the same queries over
and over again. A different order of the same stream
tuples may result in different computational results.
Similarly, missing tuples will impact the results of
the continuous queries. Incorrect ordering can be de-
tected in case the timestamps are correct. However,
malicious users may modify the timestamp to achieve
a desired evaluation schedule. Moreover, the com-
pleteness of the stream may be difficult to detect if
there is no predetermined schedule for the arrival of
the stream tuples. Next, we present a hypothetical
scenario that may occur in the presence of malicious
users.

Incorrect Oxygen Saturation Measurement.
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 the
blood. The normal value is between 95-100%. An
oxygen saturation value below 90% is referred to as
hypoxemia. Patients generally show signs of mental
impairment at oxygen saturation below 85% and
completely lose consciousness at below 75%(Powell
et al., 1996). 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. The oxygen saturation level is generally
observed every 4 hours in admitted patient care but
constantly observed in critically ill patient cases.

Figure 2 shows an example of oxygen saturation
monitoring results. In this example, as soon as hy-
poxemia was detected, treatment was initiated. The
timely treatment prevented long-term complications
for the patient.
4 STREAM DATA MODEL

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

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

Definition 1. 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 \( A_i \) is an attribute name.

Definition 2. 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 \( a_i \in \{1, ..., n \} \) is the value of attribute \( A_i \) such that \( a_i \) is in the domain of attribute \( A_i \), i.e., \( a_i \in \text{Dom}(A_i) \).
We also denote data instance \( d \) as \( d = < A_1 = a_1, ..., A_n = a_n > \) to explicitly identify the attribute values.

Definition 3. 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 2: A stream tuple \( t_1 \) from source \( \text{sensor}_1 \) at 1:00AM
\[ t_1 = (\text{sensor}_1, < A_1 = 10, A_2 = 20, A_3 = 30 >, 1:00:00AM) \]

The above stream tuple \( t_1 \) was generated by source \( \text{sensor}_1 \) at 1:00:00AM. It contains a data instance \( d = < A_1 = 10, A_2 = 20, A_3 = 30 > \) 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 4. Data Stream
A data stream \( S \) from source \( l \) during a time period from \( ts_j \) to \( ts_k \) is denoted as \( S_l = < t_1, ..., t_k > \), 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 \( S_l[ts_j, ts_k] = < t_1, ..., t_k > \) or \( S_l[ts_j, ts_k] = < (l, d, ts), ..., (l, d, ts) > \).

Definition 5. Stream Bundle
Let \( ts \) denotes a timestamp and \( t_1, ..., t_n \) denote stream data sources. A stream bundle at the time \( ts \), denoted as \( B_{ts} = \{(l_1, d_1, ts_1), ..., (l_n, d_n, ts_n)\} \), 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 6. Atomic Protected Object Pattern
An atomic protected object pattern, also referred to as 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 \( o = (l, \text{exp}_1, (d, \text{exp}_2), (ts, \text{exp}_3)) \) where each element is as follows.

Let \( v \) denotes a variable, \( c \) denotes a constant, \( A_i \) an attribute name in \( d \), and \( o \) denotes the operation \( = \) or \( \lt \). Note, we require that all attributes of the expressions are bounded, i.e., they must appear in the corresponding element \( l, d \) or \( ts \).

1. A source description \( (l, \text{exp}_1) \) where
   (a) \( l \) is a data source \( c \), such that \( c \) is a valid source, or \( v \);
   (b) \( \text{exp}_1 \) is a Boolean expression of the form Component\(_1\) AND ... AND Component\(_n\), where Component\(_i\) (\( i = 1, ..., n \)) is TRUE or \( (v \text{ op } c) \).

   We also denote a source description \( (v, \text{TRUE}) \), representing any source, as \( \ast \).
2. A data description \((d, \text{exp}_d)\) where
(a) \(d\) is a data instance on data schema \(D\), denoted as \(d = \langle A_1 = k_1, ..., A_n = k_n \rangle\), where \(k_i (i = 1, ..., n)\) is \(c\), such that \(c \in \text{Dom}(A_i)\) or \(v\);
(b) \(\text{exp}_d\) is a Boolean expression of the form \(\text{Component}\_1 \text{AND} ... \text{AND} \text{Component}_n\), where \(\text{Component}_i (i = 1, ..., n)\) is \(\text{TRUE}\), \((v_i \text{op} c)\), \((v_i \text{op} v_j)\), \((A_i \text{op} c)\), \((A_i \text{op} v_j)\), or \((A_i \text{op} A_j)\).

We also denote a data description \(\langle A_1 = v_1, ..., A_n = v_n \rangle\), \(\text{TRUE}\), representing any data instance on data schema \(D\), as \((\ast)\).

3. A timestamp description \((ts, \text{exp}_s)\) where
(a) \(ts\) is a timestamp, such that \(ts\) is \(c\) or \(v\);
(b) \(\text{exp}_s\) is a Boolean expression of the form \(\text{Component}\_1 \text{AND} ... \text{AND} \text{Component}_n\), where \(\text{Component}_i (i = 1, ..., n)\) is \(\text{TRUE}\), \(\text{op} c\), \(\text{op} v\).

We also denote a timestamp description \((v, \text{TRUE})\), representing any timestamp, as \((\ast)\).

**Definition 7. Simple Object Valuation**

Let \(v\) be a symbol mapping, such that:

1. \(v\) maps a constant \(c\) to itself, i.e., \(v : c \rightarrow c\).
2. \(v\) preserves equality of variables, i.e., \(v : v \rightarrow v\) and \(v : v \rightarrow c_1\) then \(c_1 = c_2\).
3. \(v\) maps special symbols to themselves, i.e., \(v : \langle \rightarrow \rightarrow \rangle \rightarrow v\).
4. \(v\) maps an expression \(v(x, op, y) = v(x) \text{op} v(y)\), where \(x, y\) are any of the symbols above.

**Definition 8. Stream Pattern Mapping**

Let \(o = (\langle l, \text{exp}_l \rangle, \langle d = \langle A_1 = k_1, ..., A_n = k_n \rangle, \text{exp}_d \rangle)\) be an atomic protection object, and \(v\) a symbol mapping. A stream pattern mapping \(N\) is defined as:

1. \(N(l, \text{exp}_l)\) is defined as \(N(l, \text{exp}_l) = \langle v(l), v(\text{exp}_l)\rangle\).
2. \(N(< A_1 = k_1, ..., A_n = k_n \rangle, \text{exp}_d)\) is defined as \(N(< A_1 = k_1, ..., A_n = k_n \rangle, \text{exp}_d) = \langle v(A_1) = v(k_1), ..., v(A_n) = v(k_n), v(\text{exp}_d)\rangle\) and \(N(A_i) = N(k_i) = v(k_i)\).

3. \(N(ts, \text{exp}_s)\) is defined as \(N(ts, \text{exp}_s) = \langle v(ts), v(\text{exp}_s)\rangle\).

**Definition 9. Stream Pattern Mapping Satisfiability**

Let \(o\) be an atomic protected object. We say that the stream tuple \(t\) satisfies \(o\) if there is a pattern mapping \(N\) from \(o\) to \(t\) denoted as \(N(o) = t\), such that \(N(\text{exp}_l) = \text{TRUE}\), \(N(\text{exp}_d) = \text{TRUE}\), and \(N(\text{exp}_s) = \text{TRUE}\).

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

**Example 3: Protected Object Mapping.**

**Definition 10. Object Dominance**

Let \(o_1\) and \(o_2\) protected objects. We say that \(o_1\) dominates \(o_2\) if for every tuple \(t\), there is a valuation \(v(\text{exp}_2) = t\), there must be a valuation \(v(\text{exp}_1)\) such that \(v(o_1) = t\). We denote \(o_1\) dominates \(o_2\) as \(o_1 \geq_d o_2\). Object dominance is reflexive, transitive, and asymmetric, i.e., if \(o_1 \geq_d o_2\) and \(o_2 \geq_d o_3\) then \(o_1 \geq_d o_3\).

5 SECURITY MODEL

We assign the default security labels for each entity on ontologies. Those default security labels represent
security labels in the normal situation. We use a context reasoner to decide what is the current situation and assign an appropriate security label according to that situation. In this section, we give a brief overview of the ontologies and context reasoner.

5.1 Stream Data and Security Ontology

We expand the stream data ontology introduced by Kolozali et al. (Kolozali et al., 2014), adding data security. Traditional access control models consider security to be a passive property of data quality. However, security is a complex entity associated with many parts of the stream data, e.g., security capabilities that devices can provide, security associated role of users, and security property of data. We built an ontology containing six main modules: Stream Data, Device, Quality, User, Event, and Security.

Stream ontology is generalized so it can represent the 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 develop 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 (Wang et al., 2004) 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 (Preuveneers et al., 2004) and Smart Space (Qin et al., 2007). 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.

5.2 Context Reasoning

To illustrate the logical reasoning mechanism, we present a scenario in which the data generated from sensors can have different levels 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. 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 of 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 a 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 \( \lambda(o_1) = \text{TopSecret} \) and \( \lambda(o_2) = \text{Public} \) 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 4: Protected Objects in Different Situations.

A protected object \( o_1 \) representing a normal situation during resting
\[ o_1 = ((\text{source} = \text{patient}, \text{TRUE}), (\text{oxygenSaturation} = v_1, \text{respirationRate} = v_2, \text{heartRate} = v_3) > 90 < 20), +) \]

A protected object \( o_2 \) representing a normal situation during exercising
\[ o_2 = ((\text{source} = \text{patient}, \text{TRUE}), (\text{oxygenSaturation} = v_1, \text{respirationRate} = v_2, \text{heartRate} = v_3) > 88 < 91 \text{ AND } v_2 < 40 \text{ AND } v_3 < 150), +) \]

A protected object \( o_3 \) representing an emergency situation during resting
\[ o_3 = ((\text{source} = \text{patient}, \text{TRUE}), (\text{oxygenSaturation} = v_1, \text{respirationRate} = v_2, \text{heartRate} = v_3) < 20 \text{ AND } v_2 > 40 \text{ AND } v_3 < 150), +) \]

A protected object \( o_4 \) representing an emergency situation during exercising
\[ o_4 = ((\text{source} = \text{patient}, \text{TRUE}), (\text{oxygenSaturation} = v_1, \text{respirationRate} = v_2, \text{heartRate} = v_3) < 10 \text{ AND } v_2 < 40 \text{ AND } v_3 < 150), +) \]

Definition 11. Ontology Mapping

Let \( O \) denote the ontology entity, and \( o = ((l, \text{exp}_1), (d, \text{exp}_2), (t, \text{exp}_3)) \) be an atomic protection object. An ontology mapping \( N \) is defined as:

1. If \( O \) is a \( \text{CompEntity} \) concept entity, \( N(O) = o = ((l, \text{exp}_1), (d, \text{TRUE}), (t, \text{TRUE})) \) where \( (l, \text{exp}_1) \) is the source corresponding to \( \text{CompEntity} O \) with its associated location concept, and both \( (d, \text{TRUE}), (t, \text{TRUE}) \) can be any given data description and timestamp description.

2. If \( O \) is either person, activity, or environment concept entity, \( N(O) = o = ((l, \text{TRUE}), (d, \text{exp}_2), (t, \text{TRUE})) \) where \( (d, \text{exp}_2) \) is the attributes corresponding to either person, activity, or environment concept associated with \( O \), and both \( (l, \text{TRUE}), (t, \text{TRUE}) \) can be any given source description and data description.

3. If \( O \) is a time concept entity, \( N(O) = o = ((l, \text{TRUE}), (d, \text{TRUE}), (t, \text{TRUE})) \) where \( (t, \text{exp}_3) \) is the timestamp corresponding to time concept \( O \), and both \( (l, \text{TRUE}), (d, \text{TRUE}) \) can be any given source description and data description.

4. If \( O \) is an ontology entity associated with more than one concept, \( N(O) = o = ((l, \text{exp}_1), (d, \text{exp}_2), (t, \text{exp}_3)) \) where \( (l, \text{exp}_1), (d, \text{exp}_2), (t, \text{exp}_3) \) are the source, attributes, and timestamp corresponding to ontology entity \( O \).

Definition 12. Protection Object Security Label Assignment

Let \( O_{\text{Dom}} \) denote the domain ontology, and \( \Lambda \) the mapping of security labels to the concepts in \( O_{\text{Dom}} \). We denote \( \Lambda(O_{\text{Dom}}) \) the assignment of security labels to all the concepts in the ontology, and \( \lambda(o_i) \) to denote the security label of the concept \( o_i \). We require that \( \lambda(o_i) \geq \lambda(o_j) \) if \( o_i \) is a descendent of \( o_j \). The set of Core Security Objects (CSO) is the set of all object patterns corresponding to the ontological concepts. That is, for \( \text{CSO} = (c_{t_1}, \text{TRUE}), (c_{t_2}, \text{TRUE}), (c_{t_3}, \text{TRUE}) \), \( \Lambda(\text{CSO}) = \Lambda(c_{t_1}), \Lambda(c_{t_2}), \Lambda(c_{t_3})) \).

Additional security restrictions due to context-aware or temporal needs, may be expressed over more specific protection objects. Let \( o_1 \geq_d o_2 \), then \( \Lambda(o_1) \leq \Lambda(o_2) \).

Note, that emergency scenarios may require modification of core security object classification and, in our current model, can only be handled by reassigning labels to the ontological concepts.

Definition 13. 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 \( LUB(\lambda(o_1)) \) where \( LUB(\lambda(o_i)) \) is the lowest upper bound of the security labels \( \lambda(o_1), ..., \lambda(o_k) \).

Algorithm 1: Assign security label to stream tuple.

Require: Stream tuple \( t \), and security labels of all protected objects \( \lambda(o_1), ..., \lambda(o_k) \).

Ensure: Label assignment to a stream tuple
1: Initialization
2: Find all pattern mappings \( N(o_1), ..., N(o_k) \) from \( O \) (all protected objects) that are satisfied by \( t \)
3: Let \( \{o_1, ..., o_k\} \subseteq O \) be the protected objects that can be mapped by \( N(o_1), ..., N(o_k) \) to \( t \)
4: Let \( \lambda(t) = LUB(\lambda(o_1), ..., \lambda(o_k)) \)

Example 5: Security Label Assignment to Stream Tuple.

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

\[ \lambda_1 = ((\text{sensor} = \text{TRUE}), (\text{oxygenSaturation} = v_1, \text{respirationRate} = v_2, \text{heartRate} = v_3) > 20, 20 < v_1 < 50), +) \]

\[ \lambda_2 = ((\text{sensor} = \text{TRUE}), (\text{oxygenSaturation} = v_1, \text{respirationRate} = v_2, \text{heartRate} = v_3) > 20, 20 < v_1 < 10), +) \]

\[ \lambda_3 = \text{Secret} \]

\[ \lambda_4 = \text{TopSecret} \]

Assign security label to \( t \).
Next, we propose the concept of a complex protected object. This allows to express protection needs over aggregates of stream tuples, while allowing maximum availability of the individual tuples.

**Definition 14. Complex Protected Object**

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

$O_c$ is minimal, that is for any $o_i, O_c = O_c - o_i$, then $\lambda(O_c') \neq LUB(\lambda(o_i)) | j \in \{1, ..., n\} - \{i\}$. And there are no two objects $o_i$ and $o_j$ such that $o_i \geq o_j$.

**Theorem 1.** Given a stream tuple $t$ and security labels of all protected objects $\lambda(o_1), ..., \lambda(o_n)$. Algorithm 1 correctly assigns 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 satisfy the security requirement of $t$.

**Case 1:** $o_k$ cannot be mapped to $t$

1. From algorithm 1, $o_k$ must be a protected object that can be mapped to $t$.
2. So $t$ must be a member of $o_k$. This contradicts the assumption 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 an $o_i$ such that $\nu(o_i) > \nu(o_k)$ and there is a pattern mapping from $o_i$ to $t$.
2. But the algorithm 1, $\lambda(t)$ is assigned by $LUB(\lambda(o_1), ..., \lambda(o_i), ..., \lambda(o_j))$ where $\{o_i, ..., o_j\}$ be the protected objects that can be mapped to $t$.
3. So $\lambda(o_k)$ must be $LUB(\lambda(o_1), ..., \lambda(o_j))$ which is the highest security labels of protected objects mapped to $t$. This contradicts the assumption 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 1, $\lambda(t)$ is assigned by $LUB(\lambda(o_1), ..., \lambda(o_j))$ where $\{o_i, ..., o_j\}$ be the protected objects that can be mapped to $t$.
2. So $\lambda(o_k)$ must be $LUB(\lambda(o_1), ..., \lambda(o_j))$ which is the highest security labels of protected objects mapped to $t$. This contradicts the supposition that $\lambda(o_k)$ is higher than the security requirement of $t$.

**6 IMPLEMENTATION**

In our previous work (Rhujiattawiat et al., 2021), we implemented our context-aware security policy engine. We implemented our framework using Protégé (Stanford, 2022) and Semantic Web Rule Language (SWRL) on an open-source home automation called Home Assistant (Home Assistant, 2021). Figure 8 shows a simplified ontology containing StreamTuple and ProtectedObject and figure 9 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 $o = (\langle source = l, exp \rangle, (d, exp_d), (timestamp = ts, exp_{ts}))$ with label assignment $\lambda(o) = Label$ in the form as following:

\[
\text{StreamTuple}(\langle t \rangle) \\
\land \text{hasDataInstance}(\langle t, N(d, exp_d) \rangle) \\
\land \text{hasTimestamp}(\langle t, N(timestamp = ts, exp_{ts}) \rangle) \\
\land \text{hasLabel}(\langle t, Label \rangle)
\]

(1)

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

\[
\text{ProtectedObject}(\langle o \rangle) \\
\land \text{source}(\langle o, ?source \rangle) \\
\land \text{dataInstance}(\langle o, ?data \rangle) \\
\land \text{timestamp}(\langle o, ?timestamp \rangle) \\
\land \text{StreamTuple}(\langle t \rangle) \\
\land \text{hasSource}(\langle t, ?source \rangle) \\
\land \text{hasDataInstance}(\langle t, ?data \rangle) \\
\land \text{hasTimestamp}(\langle t, ?timestamp \rangle)
\]

(2)

We show the SWRL rule to assign a security label in figure 10 and the result where the label is assigned to a stream tuple in figure 11

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 12 outlines how the data is extracted
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.

7 CONCLUSION

In this paper, we proposed a semantics-based access control model for stream data. Our solution eliminates the dependency on stream syntax and provides 1) syntax independent expression of security requirements, 2) high assurance of security compliance over the IoT network, and 3) dynamic and adaptable security policy representation. We proposed the concept of a stream data protection object that allows a user-friendly and inter-operable expression of the protected data. We developed techniques so stream data instances can inherit security requirements from the protection object. In our model, we support strong security by enforcing the most restrictive security requirement that an instance inherits. We also coupled our security model with context-based security. The security restrictions of the protection objects may change based on the current context. Our work provides an approach to support data security for the rapidly evolving IoT environment. We applied technologies that are widely used for IoT applications to support data integration and intelligent analysis, thus our model is compliant with the technologies already used by the industry.

REFERENCES


Puthal, D., Nepal, S., Ranjan, R., and Chen, J. (2015). A dynamic key length based approach for real-time security verification of big sensing data stream. In Inter-
national conference on web information systems engineering, pages 93–108. Springer.


