A Deontic Logic Model of Attribute-Based Information Flows in Database-Defined Networks with Application to Healthcare Monitoring

Benjamin Aziz[a], Ukamaka Oragwu[b] and Safa Tharib[c]
School of Creative and Digital Industries, Buckinghamshire New University, High Wycombe, U.K.

Keywords: Attribute-Based Policies, DDNs, Deontic Logic, Information Flows, SDNs.

Abstract: This paper addresses the increasing demand for robustness and reliability in modern software-defined networks, particularly in the context of critical business applications such as the healthcare domain. Traditional network architectures are vulnerable to both unintentional and intentional failures, leading to significant financial losses, especially in the healthcare sector. The paper propose a formal model for attribute-based information flow control in database-defined networks, which leverages attributes to evaluate compliance with desired network conditions, such as the quality of connectivity. Additionally, the paper employs deontic logic to define permissible, prohibited and obligatory changes to network configuration tables. The paper demonstrate how this model can enhance the management of body-area networks and ensure quality of service in healthcare monitoring. The findings suggest a promising direction for improving network reliability and security.

1 INTRODUCTION

The rapidly increasing number of critical business applications and high-volume data transmission over modern day communication networks brings up the demand for a high degree of robustness and reliability. Communication networks are prone to both unintentional (or unplanned) failures, e.g. human errors, natural disasters, overload and software bugs etc., as well as to intentional (or planned) failures caused by the process of maintenance or malicious interference (Markopoulou et al., 2004). All network elements (e.g. forwarding devices and links) are susceptible to failure incidents and leading to network facilities (like routers) being harmed. In addition, failures cause financial losses to service providers, e.g. cloud service providers. According to the statistics of 28 cloud providers from 2007 to 2013 (Cérin et al., 2013), financial losses were estimated at approximately \$285 million as a consequence of infrastructure and application failures. As networks evolve to address these challenges, Software-Defined Networking (SDN) has emerged as a promising solution to overcome the rigidity of traditional architectures, but it also introduces new security concerns that need to be addressed (Maleh et al., 2023), for example in relation to denial of service attacks (Sivarajan and Jeyalakshmi, 2024).

The success of OpenFlow (McKeown et al., 2008) in recent years as an SDN platform, brought much attention from both the academic and business communities and has driven the development of SDN research as a next generation of networking system architectures. More recently however, a new approach to the implementation of SDNs has emerged that aims to simplify the task of network administration through the introduction of further data-based abstractions of the control and data planes. This approach is called Database-Defined Networking (DDN) (Wang et al., 2016), in which the entire network is represented using a standard relational database. The communication network is abstracted as a database that can be managed like any other database. Such management is performed through a standard interface, like SQL, where the network state can be *queried* and its configuration *updated* using standard database languages. With this approach, it becomes straightforward, for example, to divide the network into multiple zones and enforce access control rules on those zones using access control lists as was demonstrated in (Glaeser and Wang, 2016).

In SDNs in general, network-wide policies and invariants are defined in high-level terms that are translated to device configuration. Eventually, the actual behavior of the network may violate policies

[a] https://orcid.org/0000-0001-5089-2025
[b] https://orcid.org/0009-0008-5213-9967
[c] https://orcid.org/0000-0002-0088-3363

Aziz, B., Oragwu, U. and Tharib, S.
A Deontic Logic Model of Attribute-Based Information Flows in Database-Defined Networks with Application to Healthcare Monitoring.
DOI: 10.5220/0013144900003899
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 11th International Conference on Information Systems Security and Privacy (ICISSP 2025) - Volume 2*, pages 330-341
ISBN: 978-989-758-735-1; ISSN: 2184-4356
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

(e.g., routing policies, security policies) and invariants (e.g., connectivity) due to some untested application bug, a network controller mistranslation, or a faulty protocol implementation in network devices. Therefore, network correctness must be constantly monitored in order to detect possible violations. Such problems may be encountered when network controllers are shared by different users or applications, or multiple controllers operate in the same domain, leading to conflicting rules, violation of policy, or network faults, such as loops, black holes, access control violations, etc. Malicious parties may even bypass security policies by defining strategic flow rules to relabel and redirect traffic.

In this paper, we outline a model of attribute-based information flow control for DDNs, which attempts to address the clarity of DDN (and generally, SDN) network policies and address the above problems. Every DDN has a configuration table that defines how packets are to be routed from their initial source to the final destination. We assume that nodes, composing such configuration tables, have *attributes*, which can be used to determine whether a configuration table satisfies certain desirable conditions or not. Additionally, we outline a model based on deontic logic (Von Wright, 1951) to determine whether changes to the configuration table are permissible, prohibited or obligatory. As far as we know, no previous works have addressed the problem of specifying SDN policies using this specific approach.

The rest of the paper is organised as follows. In Section 2, we discuss state-of-the-art literature related to the areas covered in this paper. In Section 3, we provide some background on DDNs and outline the theoretical model underlying DDNs, which we use in this paper. In Section 4, we define our model for attribute-based information flows in DDNs, and in Section 5 we consolidate this model using deontic logic operators to be able to express permission, prohibition and obligation policies. In Section 6, we demonstrate how this model can be applied to a case study involving body-area networks, and show how the configuration of such networks can benefit from our deontic logic-based information flow policies. In Section 7, we show how deontic policies, in particular obligation policies, can be used to implement quality of connectivity subscription levels in a healthcare monitoring environment. Finally, in Section 8, we conclude the paper and discuss future research.

2 RELATED WORK

We review in this section some state-of-the-art works in literature, which are of relevance to our work presented in this paper, along four main strands: The use of deontic logic for the specification of security policies, non-policy-based security mechanisms for SDNs, security policy frameworks for SDNs with application to other (non-healthcare) domains and securing healthcare networks.

Deontic Logic for Security Policy Specifications. The use of deontic logic for security policy specifications is a concept that has been explored for many years. For example, Ortalo (Ortalo, 1996) looked into this particular problem by addressing the difficulties of ensuring consistent adherence to security policies. Their work demonstrated how deontic logic, which formalises permissions, obligations and prohibitions can help overcome these challenges. Ortalo also introduced a graphical method to represent these policies, making it easier to visualize and manage complex security rules. Similarly, in 2012, Amini et al. (Amini et al., 2012) applied a variant of deontic logic, called MASL (Multi-Authority Security policy Language), to specify and enforce security policies within Multi-Security Domain (MSD) environments. Their approach facilitated the coordination and administration of security policies across multiple domains, ensuring consistency and compliance, even in the face of differing administrative boundaries. In (Cheng and Miura, 2006), Cheng and Miura introduced the use of deontic relevant logic as a foundational framework for reasoning about information security and assurance, allowing for precise specification of security policies, enabling systems to enforce rules and detect violations more effectively. More recently, Deb et al. (Deb et al., 2024) explored declarative logic-based decision-making, which can complement deontic logic in optimising agent-based security decisions by formalising conditions under which actions are obligatory or prohibited. Olszewski et al. (Olszewski et al., 2024), on the other hand, explore the application of deontic logic by attributing specific characteristics, or “attributes” to network nodes, which then makes it possible to assess whether configurations comply with certain desired conditions. Their use of deontic logic offers a formal framework to determine if changes to network configurations are permissible, prohibited or obligatory, which is especially valuable in preventing policy violations, routing errors, or access control breaches. This approach enhances security compliance and fault tolerance. Aside from the above, Olivieri et al. (Olivieri et al., 2024) introduce the concept of deontic meta-rules, which provide a higher-order structure for interpreting and resolving conflicts between network policies. This is particularly relevant in environments where multiple controllers or users interact with the same network infrastructure, as meta-rules help prioritise and harmonise potentially conflicting flow rules. Finally, Makin (Makin, 2024) further explores the foundational aspects of deontic operators, emphasising their applicability in formalising ethical and procedural rules within complex systems.

Non-Policy-Based Security Mechanisms for SDNs. Beyond deontic logic and security policy-based frameworks, other mechanisms have also been proposed to secure SDNs. For example, Chaudhary et al. (Chaudhary et al., 2018) developed an SDN-enabled, multi-attribute secure communication model for Industrial Internet of Things (IIoT) environments, particularly smart grids. Their model combines a cuckoo-filter-based (Fan et al., 2014) forwarding scheme, attribute-based encryption (Goyal et al., 2006) and a Kerberos-based (Steiner et al., 1988) authentication system, enhancing both security and network efficiency. Vyas and Shyamasundar (Vyas and Shyamasundar, 2021) introduced SecSDN, a new architecture that addresses SDN vulnerabilities stemming from weak authentication between controllers and switches. SecSDN uses repetitive hashing to verify flow tables and detect malicious switches, offering enhanced security without imposing overhead on network performance. Shi et al. (Shi et al., 2017) proposed an enhanced SDN security framework targeting threats at multiple layers—application, control, resource and interface. Their architecture emphasises attribute-based encryption to strengthen access control and defend against known vulnerabilities. While SDN's popularity continues to rise, the authors highlight that security research in this areas remains in its infancy. Smith et al. (Smith et al., 2016) presented an information-theoretic approach to partitioning networks, incorporating node and link attributes. Their method improves on existing algorithms by identifying network modules that reflect real-world community structures, though further research is needed to address highly correlated attributes. Finally, smart contracts and the blockchain technology have also been suggested as a possible approach for securing the SDN. The authors in (Sivarajan and Jeyalakshmi, 2024) address this by proposing an attribute-based access control mechanism using blockchain smart contracts to prevent unauthorised traffic and mitigate denial of service risks in SDNs. This approach enhances security by ensuring only authorised traffic reaches the control plane.

Security Policies for SDNs in Other Domains. Security policies have also been employed to enhance SDN security and reliability properties within domains of application other than the healthcare domain. For example, Zhu et al. (Zhu et al., 2020) introduced Attribute-Guard, an attribute-based flow authentication and fine-grained access control framework for SDNs. This framework addresses security gaps by verifying the legitimacy of data flows through an attribute-based authentication protocol, effectively rejecting malicious flows without impacting network performance. Another relevant work to ours is that of Al-Haj and Aziz (Al-Haj and Aziz, 2019) introduced a method for enforcing multilevel security policies in DDNs using Row-Level Security (RLS) in relational database systems. Their approach provides fine-grained control over data flow by applying upward and downward information flow policies at the individual table row level, simplifying network management while enhancing security. This mapping from high-level security policies onto low-level rules also suggests a method for the refinement of security policies in the context of SDNs/DDNs. The authors in (Varadharajan et al., 2019) explored a policy-based architecture addressing inter-domain security essential for environments like healthcare, finance and critical infrastructure, where information can flow accross multiple domains. Finally, (Karmakar et al., 2020) proposed a dynamic policy-enhanced SDN architecture, with the ability to adjust policies in real-time, in domains such as Cloud computing.

Securing Healthcare Networks. The last literature group we consider here are works relevant to the securing of healthcare networks. We use the term "healthcare networks" to refer to any network technology (including SDNs) used to carry health-related information, where sensitive patient data and critical medical systems are interconnected. We mention here only a few such relevant examples. Esteves and Rodríguez-Doncel (Esteves and Rodríguez-Doncel, 2024) explore ontologies and policy languages within the GDPR framework (European Parliament and Council of the European Union, 2016), which is highly relevant in healthcare, where stringent data privacy regulations must be adhered to. Their work emphasises the need for well-defined policies to govern how sensitive data are to be handled and shared across healthcare networks. This work represents a high-level approach to ours, nonetheless, of interest as future approach. On the other hand, taking a low-level perspective, Walid et al. (Walid et al., 2024) compare various attribute-based encryption schemes that secure healthcare systems by controlling access to sensitive data. Their work underscores the importance of "attribute-based approaches" in safeguarding patient information, enabling only authorised users to access critical data based on defined attributes. Zhang et al. (Zhang et al., 2024) further enhance this approach with a secure attribute-based dynamic data sharing scheme specifically designed for the Internet of Medical Things, which not only hides access policies but also efficiently updates them. Both of these two works represent a future direction for our work in refining our approach to the actual system level.

3 A MODEL OF DDNS

In the model of DDNs adopted in (Alhaj and Aziz, 2019), it is possible to define a configurable network in terms of three tables of data: First, a *network topology* table, which defines how nodes are physically connected, second, a *configuration table*, which defines flows through the network that packets can be routed through and finally, a table that defines flow *reachability* attributes. For the purpose of this paper, we only need to focus on configuration tables.

Definition 1 (Configuration Table). *Define a DDN configuration table as the following function (Alhaj and Aziz, 2019):*

$cf : \mathbb{N} \to ((\mathbb{N} \times \mathbb{N} \times \mathbb{N}) \hookrightarrow \mathbb{N})$ □

The assumption here is that all nodes and flows are identified as numbers. *cf* takes as input a flow's number, $f_{id} \in \mathbb{N}$, and returns as output a partial injective function. This function provides a sequence of triples representing the identity numbers of the nodes forming the flow path. These nodes are defined as the number of the current node (switch), $s_{id} \in \mathbb{N}$, the previous node, $p_{id} \in \mathbb{N}$, and the next node, $n_{id} \in \mathbb{N}$. Each triple is an expression of the configuration of the various switches on the network that the flow path traverses. The fact that each triple is itself mapped to a number renders the triples as a finite sequence (or a tuple). We call the set of all configuration tables $\mathcal{C}$. In order to simplify the model later, we also refer to each flow element of a configuration table *cf* as $flow_i$, where $cf = \{flow_1, \ldots, flow_n\}$, $i \in \{1, \ldots, n\}$ and $n = \mid cf \mid$. Sometimes we drop index *i* and use *flow* to refer to any flow in *cf*.

The general form of an *n*-flow configuration table looks like the following:

$$cf = \left\{ \begin{array}{ll} (f_1, & \{((p_{11}, s_{11}, n_{11}), 1), \\ & \quad \vdots \\ & \qquad ((p_{1k_{f_1}}, s_{1k_{f_1}}, n_{1k_{f_1}}), k_{f_1})\}), \\ \vdots & \\ (f_n, & \{((p_{n1}, s_{n1}, n_{n1}), 1), \\ & \quad \vdots \\ & \qquad ((p_{nk_{f_n}}, s_{nk_{f_n}}, n_{nk_{f_n}}), k_{f_n})\}) \end{array} \right\}$$

We use the notation $flow_i.j$ to refer to the specific $j^{th} \in \{1, \ldots, k_{f_i}\}$ entry in the table for $flow_i$. For example, $flow_1.1$ refers to the entry that is (p_{11}, s_{11}, n_{11}) in *cf* above.

Consider further the following example of a configuration table mentioned in (RAVEL, 2024):

```
 fid     pid      sid      nid
------  -------  ------   ------
  1        5        1        3
  1        1        3        4
  1        3        4        6
```

We simply model this as:

$$\{(1, \{((5,1,3),1), ((1,3,4),2), ((3,4,6),3)\})\}$$

where $flow_1.1 = (5,1,3)$, $flow_1.2 = (1,3,4)$ and $flow_1.3 = (3,4,6)$.

Naturally, a DDN, as a variation of a software-defined network, should be capable of changing its flow configurations. We model this change in terms of the following relation:

$\rightsquigarrow : \mathcal{C} \to \mathcal{C}$

Formally, $\rightsquigarrow$ is defined by the rules of Figure 1.

Rule ($R1$) adds a flow, that currently does not exist, to a configuration table. Rule ($R2$) removes a flow from a configuration table. Rule ($R3$) replaces a previous node in an entry in a flow in some configuration table with a new previous node, such that the new node is different from the replaced one. Every other entry in the flow remains intact. Rule ($R4$) replaces the current node in an entry in a flow in some configuration table with a new current node, such that the new node is different from the replaced one. Every other entry in the flow remains intact. Finally, rule ($R5$) replaces the next node in an entry in a flow in some configuration table with a new next node, such that the new node is different from the replaced one. Every other entry in the flow remains intact.

The $\rightsquigarrow$ relation only expresses what changes are possible starting from some configuration table, and not that such changes actually will take place. In other

$(R1)$	$\mathit{cf} \rightsquigarrow (\mathit{cf} \cup \mathit{flow})$, *where* $\mathit{flow} \notin \mathit{cf}$
$(R2)$	$\mathit{cf} \rightsquigarrow (\mathit{cf} \setminus \{\mathit{flow}\})$, *where* $\mathit{flow} \in \mathit{cf}$
$(R3)$	$\mathit{cf} \rightsquigarrow (\mathit{cf} \setminus \{\mathit{flow}\}) \cup \mathit{flow}'$, *where* $\mathit{flow} \in \mathit{cf}$ *and* $\mathit{flow}.j = (p_{id}, s_{id}, n_{id}) \wedge \mathit{flow}'.j = (p'_{id}, s_{id}, n_{id}) \wedge p'_{id} \neq p_{id} \wedge \forall i \neq j : \mathit{flow}'.i = \mathit{flow}.i$
$(R4)$	$\mathit{cf}[\mathit{flow}] \rightsquigarrow \mathit{cf}[\mathit{flow}']$, *where* $\mathit{flow} \in \mathit{cf}$ *and* $\mathit{flow}.j = (p_{id}, s_{id}, n_{id}) \wedge \mathit{flow}'.j = (p_{id}, s'_{id}, n_{id}) \wedge s'_{id} \neq s_{id} \wedge \forall i \neq j : \mathit{flow}'.i = \mathit{flow}.i$
$(R5)$	$\mathit{cf}[\mathit{flow}] \rightsquigarrow \mathit{cf}[\mathit{flow}']$, *where* $\mathit{flow} \in \mathit{cf}$ *and* $\mathit{flow}.j = (p_{id}, s_{id}, n_{id}) \wedge \mathit{flow}'.j = (p_{id}, s_{id}, n'_{id}) \wedge n'_{id} \neq n_{id} \wedge \forall i \neq j : \mathit{flow}'.i = \mathit{flow}.i$

Figure 1: Rules of the $\rightsquigarrow$ relation.

words, it expresses change semantics, and not a transitional one (Van Benthem and Bergstra, 1994).

The above rules model every fundamental change required on the configuration table. We ignore cosmetic changes such as the changing of the flow id, f_{id}, or the range of indices (e.g. j variables in Figure 1) used to order entries for each flow, as such changes would result in congruent tables. We also write $\rightsquigarrow^*$ to denote the reflexive transitive closure of $\rightsquigarrow$. In other words, cf in $\mathit{cf} \rightsquigarrow^* \mathit{cf}'$ can arrive at cf' in zero or more finite number of $\rightsquigarrow$ steps.

4 ATTRIBUTE-BASED INFORMATION FLOWS

Attribute-based information flow policies (Yuan and Tong, 2005) are a framework for managing the flow of sensitive information in computing systems based on a set of defined attributes. Unlike traditional access control models that grant or deny access based solely on roles or identities, attribute-based policies consider a wider range of attributes such as user credentials, contextual data, time of access and the sensitivity levels of the data themselves. This allows for more fine-grained and dynamic control over how information is shared, processed and restricted.

In our case, we start by first defining for each node (switch or host) in the network, $e \in \mathbb{N}$, its set of attributes drawn from $\mathcal{A}$, the set of all possible attributes, using the following function:

$$\mathit{attr} : \mathbb{N} \rightarrow \wp(\mathcal{A})$$

such that, $\mathit{attr}(e) = \{a_1, \ldots, a_n\}$. We assume that each node has at least one attribute as a minimum, e.g. its human-readable name or its IP address. In (Alhaj and Aziz, 2019), the authors considered another example of such attributes relevant to multi-level security policies (Bell and LaPadula, 1973), namely security levels, which could be drawn from a lattice structure (Dilworth, 1950). Attributes can be resolved to their values using the function:

$$\mathit{attrValue} : \mathcal{A} \rightarrow \mathbb{Z}$$

where we assume for simplicity that all our attributes have values that are drawn from the set of integer numbers. If not, then it is easy to imagine that there is a mapping from the integer value to another value of different type. Perhaps the most interesting aspect of attribute values is that they are *mutable*; they can change values in a dynamic system. Therefore, we assume the presence of a change function, $\hookrightarrow : (\mathcal{A} \rightarrow \mathbb{Z}) \rightarrow (\mathcal{A} \rightarrow \mathbb{Z})$, defined as follows:

$$\mathit{attrValue}[x \mapsto y] \hookrightarrow \mathit{attrValue}[x \mapsto y']$$

We consider that a node's attribute is a more fundamental concept than a link's attribute, as the latter may be expressed in terms of the attributes of its two linked nodes. For example, to express that a link is either up or down, we can define this as a predicate, *LinkUp*:

$$\mathit{LinkUp} : \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{B}$$

which takes as parameters two attributes representing a node's point of view of the link's availability with another node and returns a Boolean expressing whether the link between the two nodes is up or down:

$$\mathit{LinkUp}(\mathit{connection2e2}, \mathit{connection2e1}) = \begin{cases} \mathbf{T}, & \mathit{if}\ \mathit{attrValue}(\mathit{connection2e2}) = \\ & \quad \mathit{attrValue}(\mathit{connection2e1}) = 1 \\ \mathbf{F}, & \mathit{otherwise} \end{cases}$$

The *connection2eX* attribute has a value 1 if the connection to node X is up and running from the point of view of a node e, and 0 otherwise. In our case, $\mathit{connection2e2} \in \mathit{attr}(e1)$ and $\mathit{connection2e1} \in \mathit{attr}(e2)$.

In general, we observe that for any two adjacent nodes in a flow, it may be possible to state something about how their corresponding attributes relate. Hence, for any two nodes, e_1 and e_2, there may be some predicate, $P(\{a_1^{e1}, \ldots, a_n^{e1}\}, \{a_1^{e2}, \ldots, a_m^{e2}\})$, defined over some of their attributes, whereby

$\{a_1^{e1},\ldots,a_n^{e1}\} \subseteq attr(e_1)$ and $\{a_1^{e2},\ldots,a_m^{e2}\} \subseteq attr(e_2)$. We call the set of all predicates on attributes of e_1 and e_2 that we are interested in, $\mathcal{Q}(e_1,e_2)$, and we range these over by variables P,Q etc.

We define now an information flow condition as the following logical formula:

$$C : \wp(\mathcal{Q}(p_{id},s_{id})) \times \wp(\mathcal{Q}(s_{id},n_{id})) \rightarrow \mathbb{B}$$

which takes any number of predicates defined on attributes of adjacent nodes in a configuration table flow j^{th} entry, $(p_{id},s_{id},n_{id}) \in flow.j$, and returns a Boolean to signify whether the condition is True or False. Note that this condition is general in that it describes logically how the sets of predicates on attributes of all adjacent nodes should be logically related. For example, using the *LinkUp* predicate, we could define the condition, *EntryUp*, as the conjunction of the status of consecutive links in a flow's entry:

$$EntryUp = \\ LinkUp(connectionpid2sid, connectionsid2pid) \ \wedge \\ LinkUp(connectionsid2nid, connectionnid2sid)$$

therefore testing whether the two links in a configuration table flow's entry are up and running. As expected, $connectionpid2sid \in attr(p_{id})$, $connectionnid2sid \in attr(n_{id})$ and $connectionsid2pid, connectionsid2nid \in attr(s_{id})$. On the other hand, the predicate:

$$LinkAnalysis(e2\_analysis, e1\_analysis) = \begin{cases} \mathbf{T}, & \text{if } attrValue(e2\_analysis) = 1 \ \vee \\ & \quad attrValue(e1\_analysis) = 1 \\ \mathbf{F}, & otherwise \end{cases}$$

tests for whether either node in a link deploys some packet analysis algorithm. We can now define the condition:

$$PacketAnalysed = \\ LinkAnalysis(pid\_analysis, sid\_analysis) \ \vee \\ LinkAnalysis(sid\_analysis, nid\_analysis)$$

which ensures that a packet is analysed at least once in each flow's entry.

From this, we say that a particular flow's j^{th} entry in a configuration table, *flow.j*, where $flow \in cf$, satisfies a condition C, and write:

$C \vdash flow.j$, if and only if:

$$C(\{P_1(attr(p_{id}),attr(s_{id})),\ldots, \\ P_k(attr(p_{id}),attr(s_{id}))\}, \{Q_1(attr(s_{id}),attr(n_{id})),\ldots, \\ Q_{k'}(attr(s_{id}),attr(n_{id}))\}) = \mathbf{T}$$

where $flow.j = (p_{id},s_{id},n_{id})$. We use the variables P_i to range over predicates applying to previous and current nodes, and Q_i for predicates applying to current and next nodes. Conversely, we say that the condition is not satisfied, and we write:

$C \nvdash flow.j$, if and only if:

$$C(\{P_1(attr(p_{id}),attr(s_{id})),\ldots, \\ P_k(attr(p_{id}),attr(s_{id}))\}, \{Q_1(attr(s_{id}),attr(n_{id})),\ldots, \\ Q_{k'}(attr(s_{id}),attr(n_{id}))\}) = \mathbf{F}$$

Definition 2 (Attribute-based Information Flow Policies)**.**
Define an attribute-based information flow policy simply as a set of k-number of information flow conditions:

$$\theta = \{C_1,\ldots,C_k\} \qquad \square$$

We call the set of all policies, $\Theta : \wp(\wp(\mathcal{Q}(p_{id},s_{id})) \times \wp(\mathcal{Q}(s_{id},n_{id})) \rightarrow \mathbb{B})$. Setting up any policy for the network must satisfy the following feasibility property.

Property 1 (Policy Feasibility)**.**
Defining an attribute-based information flow policy, $\theta \in \Theta$*, must satisfy the following feasibility condition:*

$$\exists cf \in \mathcal{C}, \forall C \in \theta, flow \in cf, j \in dom(cf(flow)^{-1}) : \ C \vdash flow.j \qquad \square$$

We next use the definition of attribute-based information flow policies in conjunction with deontic logic operators to state the conditions under which changes in network configurations are permitted, prohibited and obligated.

5 DEONTIC LOGIC OPERATORS

In this section, we suggest the use of deontic logic operators as first formalised by von Wright in (Von Wright, 1951) to ensure that configuration tables are changed in a correct manner. These operators are *permissions* ($\mathcal{P}$), *prohibitions* ($\mathcal{F}$) and *obligations* ($\mathcal{O}$). They reflect situations where changes are permitted, prohibited or obligated, respectively, when applied to $\rightsquigarrow$.

We now define the $\vDash$ relation to denote the fact that a deontic logic operator, $op \in \{\mathcal{P},\mathcal{F},\mathcal{O}\}$, holds under a specific policy θ when applied to $\rightsquigarrow$:

$$\theta \vDash op(cf \rightsquigarrow cf')$$

For example, the relation $\theta \vDash \mathcal{F}(cf_1 \rightsquigarrow cf_2)$ is True, meaning it prohibits changing the configuration table from its state cf_1 to state cf_2 under the θ policy. We now expand on the definition of the $\vDash$ relation for each of the deontic logic operators, to include the ability to express the logical conditions under which such operators apply.

5.1 Permissions

We start first by expressing what changes are permitted under the presence of a policy θ. We write such permission as follows:

$$\theta \vDash \mathcal{P}(cf \rightsquigarrow cf')\ \textit{such that}\ \forall C \in \theta : C \vdash cf'$$

Note here that the change is not necessarily triggered by changes in node attributes in the original *cf*, rather this could be any reason why *cf* requires changing. The permission expresses purely authorised conditions under which such changes *may* be applied to network configurations.

5.2 Prohibitions

Prohibitions express policies that are the opposite of permitted policies in which case they define the conditions prohibited for configuration tables to be changed to. This is defined as follows:

$$\theta \vDash \mathcal{F}(cf \rightsquigarrow cf')\ \textit{such that}\ \exists C \in \theta : C \nvdash cf'$$

The following two-way implication holds naturally (a change is not permitted if prohibited explicitly and vice versa):

$$\theta \vDash \mathcal{F}(cf \rightsquigarrow cf') \quad \Leftrightarrow \quad \neg\,(\theta \vDash \mathcal{P}(cf \rightsquigarrow cf'))$$

However, the following implication also holds naturally (a change is not prohibted if permitted explicitly):

$$\theta \vDash \mathcal{P}(cf \rightsquigarrow cf') \quad \Rightarrow \quad \neg\,(\theta \vDash \mathcal{F}(cf \rightsquigarrow cf'))$$

but not vice versa. Nonetheless, a *less secure* variation might allow the opposite implication to also hold (i.e. that a change is implicitly permitted if not explicitly prohibited):

$$\neg\,(\theta \vDash \mathcal{F}(cf \rightsquigarrow cf')) \quad \Rightarrow \quad \theta \vDash \mathcal{P}(cf \rightsquigarrow cf')$$

5.3 Obligations

Obligation policies state that once an entry in a configuration table violates a policy condition, that entry must be changed to one that satisfies the violated condition. Such change is necessary and obligatory, to maintain adherence to whatever good properties are sought of the network. More formally:

$$\theta \vDash O(cf \rightsquigarrow cf')\ \textit{such that}\ \exists C \in \theta : C \nvdash cf$$

Note that, in this most primitive form of an obligation policy, we do not require that $\forall C \in \theta : C \vdash cf'$. This is because an obligation policy states that the change *must* take place, but not necessarily to a *good* configuration. If we want such good configuration, then we must keep on trying until we reach it, or we must prohibit transitioning to a bad configuration.

The following implication holds naturally (an obligated change needs to be permitted explicitly):

$$\theta \vDash O(cf \rightsquigarrow cf') \quad \Rightarrow \quad \theta \vDash \mathcal{P}(cf \rightsquigarrow cf')$$

Obviously, the opposite implication is not true.

6 EXAMPLE: RECONFIGURING A BODY AREA NETWORK

The case study we adopt in this paper focuses on Body Area Networks (BANs) as an example of healthcare monitoring systems. A BAN (Movassaghi et al., 2014) is a wireless network of wearable or implantable devices that monitor and transmit physiological data from a person's body, typically for health-related purposes. These devices, such as sensors, actuators and communication modules, interact to track vital signs like heart rate, blood pressure or glucose levels in real time enabling personalised healthcare and remote monitoring by medical professionals. BANs often use short-range wireless communication protocols like Bluetooth or ZigBee to ensure secure, low-power transmission of data within the human body's vicinity. However, some of the more advanced recent BAN devices also use mobile networks (3G/4G/5G) and WiFi.

A BAN could consist of the following elements:

- A low functionality health monitoring watch device (e.g. health smart watch, glucose monitoring device etc.). This is called D1
- A high functionality health monitoring device (e.g. home ECG monitor). This is called D2

- A device configuration app called APP, which is used to configure the settings of D1 and D2
- A back-end server called SRV, which is used to collect data from D1 and D2 and analyse them

Figure 2 illustrates an example of a BAN, with the example elements mentioned above, as well as the various connectivity modes (5G, 4G, 3G, WiFi). We assume that D1 can only be reached via 3G and WiFi connectivity, whereas D1 can be reached via 4G, 5G and WiFi connectivity. The APP runs on a standard smart phone, with 4G and WiFi connectivity, and finally, the back-end server is reachable via any type of connection (3G, 4G, 5G and WiFi).

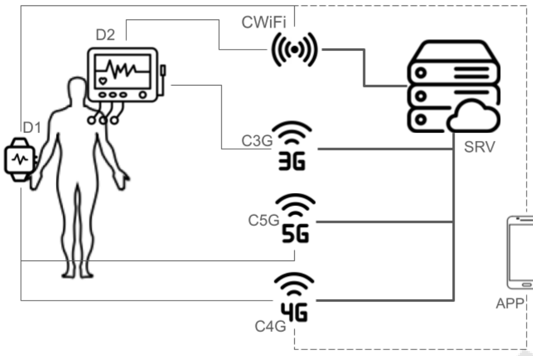

Figure 2: An Example of a BAN with connectivity modes to a server and an app.

The network in Figure 2 is defined through the following instance of a DDN configuration table, which we call cf_{ban}:

fid	pid	sid	nid
1	--	APP	CWiFi
1	APP	CWiFi	D2
1	CWiFi	D2	--
2	--	D2	CWiFi
2	D2	CWiFi	SRV
2	CWiFi	SRV	--
3	--	D2	C3G
3	D2	C3G	SRV
3	C3G	SRV	--
4	--	APP	CWiFi
4	APP	CWiFi	D1
4	CWiFi	D1	--
5	--	APP	C4G
5	APP	C4G	D1
5	C4G	D1	--
6	--	D1	CWiFi
6	D1	CWiFi	SRV
6	CWiFi	SRV	--
7	--	D1	C4G
7	D1	C4G	SRV
7	C4G	SRV	--
8	--	D1	C5G
8	D1	C5G	SRV
8	C5G	SRV	--

One of the requirements in such networks as stated in (Gama et al., 2007), for example, is that the body area network must continue to function and operate during reconfiguration (including network topology reconfiguration), and a second requirement is that the reconfiguration must not fail (i.e. it should result in a functioning new system). Here, we focus more on the idea a reconfiguration must result in a sound (i.e. functional or exhibiting desirable non-functional properties) network topology.

In the following, we discuss examples of permitted, prohibited and obliged deontic-based policies.

6.1 Permitted Policies

As an example of permitted policies, let's consider the following policy:

$$\theta \vDash \mathcal{P}(cf_{ban} \rightsquigarrow cf_{smallban})$$

whereby θ is defined by the following soundness conditions:

$C1 : \exists flow \in cf_{smallban} : flow.2 = (p,s,n) \Rightarrow (first\_is\_device\_D1(p,s) \wedge second\_is\_server(s,n))$

$C2 : \exists flow \in cf_{smallban} : flow.2 = (p,s,n) \Rightarrow (first\_is\_device\_D2(p,s) \wedge second\_is\_server(s,n))$

$C3 : \exists flow \in cf_{smallban} : flow.2 = (p,s,n) \Rightarrow (second\_is\_device\_D1(s,n) \wedge first\_is\_app(p,s))$

$C4 : \exists flow \in cf_{smallban} : flow.2 = (p,s,n) \Rightarrow (second\_is\_device\_D2(s,n) \wedge first\_is\_app(p,s))$

These conditions essentially ensure that both devices D1 and D2 are connected (through some means) to both the back-end server and the configuration app. We call these conditions informally the *soundness* conditions, as they ensure that the BAN has the correct and necessary connectivity among all its components. The check in each condition is applied to the second entry for each flow in the configuration table.

The definition of the operators in the above conditions is given as follows:

$$first\_is\_device\_D1(p,s) = \begin{cases} \mathbf{T}, & if\ attrValue(identity.p) \in \{D1\} \\ \mathbf{F}, & otherwise \end{cases}$$

$$first\_is\_device\_D2(p,s) = \begin{cases} \mathbf{T}, & \textit{if } attrValue(identity.p) \in \{D2\} \\ \mathbf{F}, & \textit{otherwise} \end{cases}$$

$$second\_is\_device\_D1(s,n) = \begin{cases} \mathbf{T}, & \textit{if } attrValue(identity.n) \in \{D1\} \\ \mathbf{F}, & \textit{otherwise} \end{cases}$$

$$second\_is\_device\_D2(s,n) = \begin{cases} \mathbf{T}, & \textit{if } attrValue(identity.n) \in \{D2\} \\ \mathbf{F}, & \textit{otherwise} \end{cases}$$

$$second\_is\_server(s,n) = \begin{cases} \mathbf{T}, & \textit{if } attrValue(identity.n) \in \{SRV\} \\ \mathbf{F}, & \textit{otherwise} \end{cases}$$

$$first\_is\_app(p,s) = \begin{cases} \mathbf{T}, & \textit{if } attrValue(identity.p) \in \{APP\} \\ \mathbf{F}, & \textit{otherwise} \end{cases}$$

each of which in turn checks the identity of the first (p) and third (n) node in a second entry of a flow in some configuration table (In all of these definitions, $identity.e$ as an attribute of node e carrying its identity, e.g. IP/MAC address, name, number etc.).

We can now give a definition of the θ permission policy in terms of the above soundness conditions:

$$\theta = \{C1, C2, C3, C4\}$$

Using the above θ, we can optimise a BAN network configuration from one which is overly connected (i.e. cf_{ban}) to one which is more compact but is still a permitted configuration (i.e. it maintains the soundness conditions above). For example, the following $cf_{smallban}$ table is one such compact network:

```
 fid     pid      sid      nid
------  -------  ------   ------
  1       --      APP     CWiFi
  1      APP     CWiFi     D2
  1     CWiFi     D2       --
  2       --      D2      CWiFi
  2      D2      CWiFi     SRV
  2     CWiFi     SRV      --
  4       --      APP     CWiFi
  4      APP     CWiFi     D1
  4     CWiFi     D1       --
  6       --      D1      CWiFi
  6      D1      CWiFi     SRV
  6     CWiFi     SRV      --
```

which maintains the connectivity between both D1 and D2 with their app and the back-end server, however only using the WiFi access point, and hence producing a smaller network than that produced by cf_{ban}. Such a *smaller BAN* can be considered to be more optimum in the sense of its energy consumption; both D1 and D2 will consume less energy as they are not required to operate 3/4/5G connectivity.

6.2 Prohibited Policies

We turn now to the case of prohibition policies. Let's consider the scenario where neither device D1 nor device D2 are connected to the server for a prolonged period of time (set by the health carers at the environment where person being monitored is placed). A policy to prohibit this scenario can be expressed as follows:

$$\theta \vDash \mathcal{F}(cf_{ban} \rightsquigarrow cf'_{ban})$$

where $cf'_{ban} = cf_{ban} \backslash \{2,3,6,7,8\}$. In other words, we are prohibited from changing the configuration table cf_{ban} above to a new one where flows 2, 3, 6, 7 and 8 are removed and no new flows added to ensure some form of connectivity between either D1 or D2 and SRV. The definition of θ would be the following:

$$\theta = \{C5\}$$

where $C5 = C1 \vee C2$, and $C1$ and $C2$ are as defined in the previous section. In other words, we are prohibited from changing a configuration table into one which violates either condition $C1$ or condition $C2$, meaning that both devices are disconnected from the server. A stronger prohibition condition would require that both devices be connected to the server. This would be expressed as $C5 = C1 \wedge C2$ (which in this case is different from setting $\theta = \{C1, C2\}$).

In real terms, what the above prohibition policy gives us is a fail-safe criticality condition (i.e. $C5$), which ensures that no new configuration tables are constructed such that all/some health monitoring devices become disconnected from the back-end server.

6.3 Obliged Policies

In our third example, we demonstrate how an obligation policy can ensure that a desired status of the network connectivity is eventually reached. Let's assume the network was initially configured using the table of $cf_{smallban}$ defined earlier. In other words, that both D1 and D2 are reachable only via the WiFi access point. However, consider a new situation where D1 becomes suddenly unreachable via WiFi (e.g. due to some internal D1 malfunction or an issue with the WiFi access point). This would result in a new configuration table, call it $cf'_{smallban}$, which the network transitions to as a correct reflection of the current network state:

```
 fid     pid      sid      nid
------  -------  ------   ------
  1       --      APP     CWiFi
  1      APP     CWiFi     D2
  1     CWiFi     D2       --
  2       --      D2      CWiFi
  2      D2      CWiFi     SRV
  2     CWiFi     SRV      --
```

Again, we may have a fail-safe obligation policy, defined as follows:

$\theta = \{C1, C2, C3, C4\}$

where we can see that $C1 \nvdash cf'_{smallban}$ as well as $C3 \nvdash cf'_{smallban}$, therefore triggering the obligation policy via either of these two violations.

Now, in order to rectify this violation, we must reconfigure the network to a safe configuration that does not further trigger our obligation policy. One example of such safe configuration is the following, which refer to as $cf''_{smallban}$:

```
 fid     pid      sid      nid
------  -------  ------   ------
  1       --      APP     CWiFi
  1      APP     CWiFi     D2
  1     CWiFi     D2       --
  2       --      D2      CWiFi
  2      D2      CWiFi     SRV
  2     CWiFi     SRV      --
  5       --      APP     C4G
  5      APP      C4G     D1
  5      C4G      D1       --
  7       --      D1      C4G
  7      D1       C4G     SRV
  7      C4G      SRV      --
```

This new configuration does not trigger the obligation policy θ since $\forall C \in \theta : C \vdash cf''_{smallban}$. The connectivity to D1 is restored via the 4G connection point.

7 POLICIES FOR QUALITY OF CONNECTIVITY

In addition to the examples we mentioned in the previous section, it is also possible to use deontic policies to manage network configurations based on quality of connectivity (e.g. transmission bandwidth) levels. Let's start first by refining the four soundness conditions $C1$–$C4$ we defined earlier in Section 6.1:

$C1^c : \exists flow \in cf_{smallban} : flow.2 = (p,s,n) \Rightarrow (first\_is\_device\_D1(p,s) \wedge second\_is\_server(s,n) \wedge (attrValue(identity.s) = attrValue(identity.c)))$

$C2^c : \exists flow \in cf_{smallban} : flow.2 = (p,s,n) \Rightarrow (first\_is\_device\_D2(p,s) \wedge second\_is\_server(s,n) \wedge (attrValue(identity.s) = attrValue(identity.c)))$

$C3^c : \exists flow \in cf_{smallban} : flow.2 = (p,s,n) \Rightarrow (second\_is\_device\_D1(s,n) \wedge first\_is\_app(p,s) \wedge (attrValue(identity.s) = attrValue(identity.c)))$

$C4^c : \exists flow \in cf_{smallban} : flow.2 = (p,s,n) \Rightarrow (second\_is\_device\_D2(s,n) \wedge first\_is\_app(p,s) \wedge (attrValue(identity.s) = attrValue(identity.c)))$

The new variations $C1^c$–$C4^c$ are parameterised by the variable c. This variable is defined as $c \in \{C3G, C4G, C5G, CWiFi\}$. In other words, the new *stronger* conditions perform an extra check as to whether the connection node, s, is the one supplied by the condition. Informally, this means that the condition holds if and only if the original soundness condition is true in addition to having the correct type of connectivity (i.e. 3/4/5G/WiFi).

This strengthening of the soundness conditions allows us to set up a *quality of connectivity ordering* relation, $\prec$, defined as follows:

$C3G \prec C4G \prec C5G \prec CWiFi$

which essentially says that 3G is lower quality than 4G, 4G lower quality than 5G and 5G lower quality than WiFi. It is worth noting here that this definition of the quality of connectivity ordering is only one example. One may define the ordering in a different manner, depending on the definition of the business scenario at hand.

Based on the above definition of $\prec$, we can define another order on the possible instances of our varied soundness conditions $C1^c$–$C4^c$ above, as follows:

$(Cn^c \prec_C Cn^{c'}) \Leftrightarrow (c \prec c')$

where $\prec_C$ is a quality of connectivity ordering on the soundness conditions and $n = 1 \ldots 4$. For example, we consider that D1's connection to the server would be of higher quality over 5G than over 4G:

$C1^{C4G} \prec_C C1^{C5G}$

since $C4G \prec C5G$. This in turn leads to the emergence of deontic policies, which are entirely based on the ordering on these conditions.

Table 1 illustrates an example of different *subscription packages* that a healthcare monitoring system might be set up with as offered by a healthcare provider in association with a telecommunication company, and the mapping of these packages to their associated obligation policies.

Table 1: Different subscription packages and their mapping to obligation policies.

Subscription Package	**Obligation Policy** θ
Gold	$\{(C1^{C5G} \prec_C C1^c),$ $(C2^{C5G} \prec_C C2^c),$ $(C3^{C5G} \prec_C C3^c),$ $(C4^{C5G} \prec_C C4^c)\}$
Silver	$\{(C1^{C4G} \prec_C C1^c),$ $(C2^{C4G} \prec_C C2^c),$ $(C3^{C4G} \prec_C C3^c),$ $(C4^{C3G} \prec_C C4^c)\}$
Bronze	$\{(C1^{C3G} \prec_C C1^c),$ $(C2^{C3G} \prec_C C2^c),$ $(C3^{C3G} \prec_C C3^c),$ $(C4^{C3G} \prec_C C4^c)\}$

From these, we can see that a *Gold* subscription ensures that the quality of connectivity of the BAN at the patient's environment is better than 5G connectivity. On the other hand, *Silver* subscription ensures that that quality is better than 4G. Finally, a *Bronze* subscription guarantees only better than 3G quality of connectivity.

8 CONCLUSION

In this paper, we presented an attribute-based information flow framework for the specification of deontic policies (authorisation, prohibition and obligation policies) in DDN systems. We demonstrated how such a formal framework can be used to specify soundness properties in BAN networks based on deontic logic, and provide richer semantics for describing the quality of connectivity in such systems.

Our research will facilitate the next step in investigating how risk can be modelled in such networks, as well as other healthcare inter-connected systems. Another area of future work will be to integrate machine learning techniques to enhance the potential of SDN/DDN systems, especially in optimising network configurations dynamically (e.g. as in (Arya et al., 2024)). We plan to build datasets of historical DDN configuration tables and use these to learn to predict future tables, guided by our soundness properties, as well as other properties that would be more difficult to express, e.g. non-functional properties. Finally, we also plan to focus on the role of the secure and trusted-by-design smart contracts as applied to the healthcare sector, e.g. (Dargaye et al., 2024), where we can use such contracts to enforce compliance with policies related to patient data access by ensuring that historical configuration tables adhere to such policies as recorded in past transactions (i.e. configurations), and where the smart contract will be used to reward normal and punish abnormal configurations leading to higher quality SDN management.

REFERENCES

Al-Haj, A. and Aziz, B. (2019). Enforcing multilevel security policies in database-defined networks using row-level security. In *2019 International Conference on Networked Systems (NetSys)*, pages 1–6. IEEE.

Alhaj, A. and Aziz, B. (2019). Enforcing multilevel security policies in Database-Defined networks using Row-Level security. In *2019 International Conference on Networked Systems (NetSys'19)*, Garching b. München, Germany. IEEE.

Amini, M., Jalili, R., Ehsan, M. A., and Faghih, F. (2012). Cooperative security administration in multi-security-domain environments using a variant of deontic logic. *Scientia Iranica*, 19(3):635–653.

Arya, S., Xiang, Y., and Gogate, V. (2024). Deep dependency networks and advanced inference schemes for multi-label classification. In *International Conference on Artificial Intelligence and Statistics*, pages 2818–2826. PMLR.

Bell, D. E. and LaPadula, L. J. (1973). Secure computer systems: Mathematical foundations. Technical report, MITRE CORP BEDFORD MA.

Cérin, C., Coti, C., Delort, P., Diaz, F., Gagnaire, M., Gaumer, Q., Guillaume, N., Lous, J., Lubiarz, S., Raffaelli, J., et al. (2013). Downtime statistics of current cloud solutions. *International Working Group on Cloud Computing Resiliency, Tech. Rep.*

Chaudhary, R., Aujla, G. S., Garg, S., Kumar, N., and Rodrigues, J. J. (2018). Sdn-enabled multi-attribute-based secure communication for smart grid in iiot environment. *IEEE Transactions on Industrial Informatics*, 14(6):2629–2640.

Cheng, J. and Miura, J. (2006). Deontic relevant logic as the logical basis for specifying, verifying, and reasoning about information security and information assurance. In *First International Conference on Availability, Reliability and Security (ARES'06)*, pages 8–pp. IEEE.

Dargaye, Z., Gürcan, Ö., Kirchner, F., and Tucci-Piergiovanni, S. (2024). Towards secure and trusted-by-design smart contracts. *arXiv preprint arXiv:2403.16903*.

Deb, T., Jeong, M., Molinaro, C., Pugliese, A., Li, A. Q., Santos, E., Subrahmanian, V., and Zhang, Y. (2024). Declarative logic-based pareto-optimal agent decision making. *IEEE Transactions on Cybernetics*.

Dilworth, R. P. (1950). Review: G. birkhoff, lattice theory. *Bull. Amer. Math. Soc.*, 56(2):204–206.

Esteves, B. and Rodríguez-Doncel, V. (2024). Analysis of ontologies and policy languages to represent information flows in gdpr. *Semantic Web*, 15(3):709–743.

European Parliament and Council of the European Union (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council. https://data.europa.eu/eli/reg/2016/679/oj.

Fan, B., Andersen, D. G., Kaminsky, M., and Mitzenmacher, M. D. (2014). Cuckoo filter: Practically better than bloom. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 75–88.

Gama, O., Figueiredo, C., Carvalho, P., and Mendes, P. (2007). Towards a reconfigurable wireless sensor network for biomedical applications. In *2007 International Conference on Sensor Technologies and Applications (SENSORCOMM 2007)*, pages 490–495. IEEE.

Glaeser, N. and Wang, A. (2016). Access control for a database-defined network.

Goyal, V., Pandey, O., Sahai, A., and Waters, B. (2006). Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98.

Karmakar, K. K., Varadharajan, V., Tupakula, U., and Hitchens, M. (2020). Towards a dynamic policy enhanced integrated security architecture for sdn infrastructure. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE.

Makin, L. T. (2024). The logic of deontic operators. *Oxford Studies in Metaethics, Volume 19*, 19:349.

Maleh, Y., Qasmaoui, Y., El Gholami, K., Sadqi, Y., and Mounir, S. (2023). A comprehensive survey on sdn security: threats, mitigations, and future directions. *Journal of Reliable Intelligent Environments*, 9(2):201–239.

Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C.-N., and Diot, C. (2004). Characterization of failures in an ip backbone. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2307–2317. IEEE.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

Movassaghi, S., Abolhasan, M., Lipman, J., Smith, D., and Jamalipour, A. (2014). Wireless body area networks: A survey. *IEEE Communications surveys & tutorials*, 16(3):1658–1686.

Olivieri, F., Governatori, G., Cristani, M., Rotolo, A., and Sattar, A. (2024). Deontic meta-rules. *Journal of Logic and Computation*, 34(2):261–314.

Olszewski, M., Parent, X., and Van der Torre, L. (2024). Permissive and regulative norms in deontic logic. *Journal of Logic and Computation*, 34(4):728–763.

Ortalo, R. (1996). Using deontic logic for security policy specification. *LAAS report*, 96380.

RAVEL (2024). Ravel Walkthrough. Last accessed: 21 June, 2024.

Shi, Y., Dai, F., and Ye, Z. (2017). An enhanced security framework of software defined network based on attribute-based encryption. In *2017 4th International Conference on Systems and Informatics (ICSAI)*, pages 965–969. IEEE.

Sivarajan, E. and Jeyalakshmi, V. (2024). Attributes based access control mechanism using smart contracts against denial of service attacks in sdn. In *AIP Conference Proceedings*, volume 3149:1. AIP Publishing.

Smith, L. M., Zhu, L., Lerman, K., and Percus, A. G. (2016). Partitioning networks with node attributes by compressing information flow. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(2):1–26.

Steiner, J. G., Neuman, B. C., and Schiller, J. I. (1988). Kerberos: An authentication service for open network systems. In *Usenix Winter*, pages 191–202.

Van Benthem, J. and Bergstra, J. (1994). Logic of transition systems. *Journal of Logic, Language and Information*, 3:247–283.

Varadharajan, V., Karmakar, K., Tupakula, U., and Hitchens, M. (2019). A policy-based security architecture for software-defined networks. *IEEE Transactions on Information Forensics and Security*, 14(4):897–912.

Von Wright, G. H. (1951). Deontic logic. *Mind*, 60(237):1–15.

Vyas, P. and Shyamasundar, R. (2021). Secsdn: A novel architecture for a secure sdn. In *SECRYPT*, pages 587–594.

Walid, R., Joshi, K. P., and Choi, S. G. (2024). Comparison of attribute-based encryption schemes in securing healthcare systems. *Scientific Reports*, 14(1):7147.

Wang, A., Mei, X., Croft, J., Caesar, M., and Godfrey, B. (2016). Ravel: A database-defined network. In *Proceedings of the Symposium on SDN Research*, SOSR '16, pages 5:1–5:7, New York, NY, USA. ACM.

Yuan, E. and Tong, J. (2005). Attributed based access control (abac) for web services. In *IEEE International Conference on Web Services (ICWS'05)*. IEEE.

Zhang, L., Xie, S., Wu, Q., and Rezaeibagha, F. (2024). Enhanced secure attribute-based dynamic data sharing scheme with efficient access policy hiding and policy updating for iomt. *IEEE Internet of Things Journal*.

Zhu, X., Chang, C., Xi, Q., and Zuo, Z. (2020). Attributeguard: Attribute-based flow access control framework in software-defined networking. *Security and Communication Networks*, 2020(1):6302739.

APPENDIX

The research presented in this paper is partly funded by Innovate UK/Eureka CELTIC project ADA (award number 10098971).