university of groningen

faculty of science and engineering

Master's Thesis

Securing Publish/Subscribe systems using Software Defined Networks

Distributed Systems — Bernoulli Institute
Department of Computing Science

Author:
C.S. Braams (s2969807)

Supervisors:
Prof. dr. B. Koldehofe (First Supervisor)
B. Boughzala, Msc
P. Agnihotri, Msc

November 21, 2022

Abstract

Software-Defined Networking (SDN) is an upcoming approach to networking characterised by the separation of the data plane and control plane. Implementing SDN for the communication of Pub/Sub systems leads to higher expressiveness and less latency in forwarding content. However, security issues like confidentiality, authentication and integrity for publish/subscribe communication in SDN are still relatively unexplored.

In this thesis, a secure system model for a Publish/Subscribe scheme using SDN is designed and implemented. This system covers cryptography using CP-ABE, efficient and secure routing using P4 switches and the decoupled implementation of pub/sub communication. A qualitative security analysis was performed on the proposed model through different use cases. Confidentiality of the content is well protected in these use cases, and unauthorised events are ignored. However, preservation of the integrity of the information remains a potential pitfall. No difference in computation overhead is measured for the Boolean operators *OR* or *AND*. However, numerical attribute comparison significantly increases the execution time of the KeyGen() algorithm. Since the controller that executes KeyGen() has extensive resources, this does not impose a direct problem.

Therefore, the proposed model provides a secure Content-based Publish/Subscribe (CBPS) communication without impairing the decoupled property or creating unacceptable overhead.

Table of Contents

Acronyms

ABE Attribute-based Encryption. 13, 14, 17, 29, 33, 34, 46, 53, 54

CBPS Content-based Publish/Subscribe. 1, 3, 4, 13, 19, 29, 31, 32, 34

CP-ABE Ciphertext attribute-based Encryption. 1, 2, 14, 18, 29, 31, 33–35, 44, 46–48, 50, 52–54

IBE Identity-based Encryption. 13, 17, 29

IoT Internet of Things. 3, 14, 29

KP-ABE Key-policy attribute-based Encryption. 14, 18, 29

MAC Media Access Control (address). 36, 38, 41, 44, 45

MoM Message Oriented Middlewares. 6

MST Minimal Spanning Tree. 41

OF OpenFlow. 9, 11, 12, 18, 19, 23, 27

P4 Programming Protocol-independent Packet Processors. 1, 4, 12, 19, 21, 23, 27, 39, 43, 54

Pub/Sub Publish/Subscribe. 1–4, 6–8, 13, 16–21, 23, 28, 29, 31, 34–37, 43, 46, 52–54

SDN Software-Defined Networking. 1, 4, 9–12, 14, 16, 18–21, 23, 28, 29, 31, 42, 52–54

TCAM Ternary Content-Addressable Memory. 26, 27

List of Figures

List of Tables

1 Introduction

In the present-day world, we deal with many modern applications, e.g. social networks, smart grids and IoT, that should connect and communicate with each other as fast as possible. People use social networks, entertainment streaming, and healthcare systems on a daily basis. For instance, people want instant updates on the COVID-19 pandemic news on their phones, the newest stock exchanges updates for their assets and streaming new movies on demand. In this regard, the publish/subscribe communication paradigm allows these distributed networks to communicate in such applications.

Publish/Subscribe Pub/Sub is a message pattern for content delivery in a distributed system that has increased in popularity for modern applications. This popularity is due to its asynchronous nature and its decoupling of distributed components. A Pub/Sub system has two roles; publishers who publish messages called events and subscribers who subscribe to show interest in certain events. Since publishers and subscribers are loosely coupled components, there is no direct connection between the two roles. Therefore, an middleware must handle communication between publisher and subscriber. An event service can be seen as a broker function that forwards messages.

In Pub/Sub communication, a publisher, who has no information about existing subscribers, passes data as an event to the network. The notification service receives events and notifies relevant subscribers matching their interests. This interest of the subscribers is called a subscription. Content-based Pub/Sub is a subscription model and is very popular in current systems. The dynamic nature of Pub/Sub interconnects many (un)trusted publishers and subscribers in the network which causes security concerns if no access control mechanisms are integrated. The protection of the content of the event against untrusted subscribers is important.

A Content-based Publish/Subscribe system takes routing decisions based on the current information and effectively establishes routing paths between publishers and subscribers only if there is a matching interest. The content-based routing imposes two new security challenges. Firstly, the system needs to ensure subscribers can only access the content they are authorized for. Secondly, the system should not reveal information about the interest of the subscriber or the event from the publisher for privacy. This thesis will focus on secure content-based routing, in which subscribers create expressive subscriptions that allow flexible and efficient filtering of events.

Related works for Pub/Sub solutions focus primarily on implementation in the application layer using a middleware layer [5–8]. Middleware solutions have an overlay network in which subscriptions are matched with publications. Routing via an overlay network is not bandwidth efficient since the overlay topology will differentiate from the physical topology. This results in an event being routed multiple times through the same physical link from a publisher to a particular subscriber [9]. Operating in the application layer brings latency to the processing time of a packet since it needs to be handled by the session and presentation layers first. Moreover, the overlay network is unaware of the underlying physical network.

Considering these disadvantages of using a middleware layer, a new research field has arised implementing content-based routing on the network layer [10]. However, this research area is restricted since it is unrealistic to change existing standard network protocols and hardware. More recent studies of network technologies like SDN raised new efforts in realising publish/subscribe middleware that supports network event filtering [10, 11].

SDN is a new network paradigm separating the network into control and data planes. The control plane has a centralised controller that manages the network devices and holds the network topology. The control plane is programmable, and it could thus change the network policies and control all data flows. In the data plane, data flows through programmable packet-forwarding devices called switches. This separation allows for flexible routing policies expression, which improves performance; it enables filtering directly in the data plane.

However promising SDNs are as an alternative to a middleware layer, a comprehensive system model with the implementation of Content-based Publish/Subscribe model on SDN still needs further exploration.

This thesis aims to look further than the efficient implementation of a Pub/Sub system on a SDN. Since Pub/Sub communication is used for information dissemination in distributed applications, it must also fulfil security needs. The security needs will be assessed by defining a threat model and fitting security and efficiency goals. The security of broker-less Pub/Sub systems has been described before [12]. However, a security assessment of Pub/Sub systems with the implementation of a SDN has not been covered. In addition, this thesis aims to investigate the trade-off using a common security framework for broker-less Pub/Sub systems, specifically for attribute-based encryption [12–14].

1.1 Novelty

The novelty of this thesis lies in the design of a secure Pub/Sub-system model on the network layer using SDN and programmable switches for the data plane. There are solutions available that implement Pub/Sub communication in the network layer [10, 15]. However, the security aspect of this implementation has not been described before. Security in middleware Pub/Sub implementations has been discussed before, however, these implementations need to be translated and improved to the new opportunities SDN brings.

Using SDN to build the Pub/Sub protocol allows for more flexibility and less complexity than traditional networks; Due to the logically centralised control plane architecture, the controller can have the key authority role for encryption schemes. Moreover, no transport-layer links between brokers in an overlay network are necessary for routing since switches in the SDN take over this role. Removing the brokers allows less end-to-end latency and bandwidth efficiency. This thesis takes the SDN concept further by allowing switches to be fully programmable and protocol-independent using Programming Protocol-independent Packet Processors (P4) language. Since P4 aims for open components, proprietary switches with potential backdoors and static protocols will be eliminated.

1.2 Outline

In Chapter 2, the Background information is represented.
Chapter 3 discusses relevant works from existing literature concerning Content-based Publish/Subscribe systems, with or without using a SDN and the security of such models.

Chapter 4 formulates the problem statement. It also describes the attack and security assumptions used to create a system model. Moreover, it defines the security and efficiency goals.
Chapter 5 lists all the design options for designing a final system model. It discusses which options were considered and compares the different options.
Chapter 6 shows the proposed theoretical system model, which includes the messages' design, the system's entities and the control application.
Chapter 7 covers the results of a security assessment and an overhead assessment of the proposed design. A qualitative security analysis was performed on the proposed model, discussing communication confidentiality, integrity and authentication employing different use cases. A proof-of-concept of the proposed solution is introduced by assessing computation overhead.
Chapter 8 concludes the final work, discusses the proposed solution's outcomes and places them in the context of related works.

2 Background

In this section, the background material is presented, which is required for the basic understanding of specific technologies.

2.1 Content-based pub/sub

Message Oriented Middlewares (MoM) are middleware architectures that support sending and receiving messages between distributed systems where much internal communication occurs. MoMs allow asynchronous communication in which the receivers of information do not wait for a response. The second property is that MoM decouples network components; the sender and receiver of messages do not know each other's address or location. The decoupled property is realised by brokers or intermediaries responsible for routing the packets to the correct receivers. Moreover, MoM allow for generic data type formats, which is beneficial for connecting different applications.

A Publish/Subscribe (Pub/Sub) is a pattern of MoM, an asynchronous service-to-service communication protocol. Pub/sub is mainly used in a distributed system in which information is disseminated dynamically. The data senders (publishers) and the data receivers (subscribers) communicate through messages (events). Communication is possible via an event service named the broker middleware.

A classical publish/subscribe system consists of publishers, subscribers and brokers. Brokers are the intermediates in the network that disseminate events sent by publishers to interested subscribers. Following are the components of a publish-subscribe system:

Publisher:

- Sender of events
- Producer of data
- Sends without knowing the destination address of the receiver(s)

Subscribers:

- Receiver of events
- Comsumer of data
- Receives events to which it is subscribed to
- Does not know the publisher of the event

Broker middleware:

- Intermediate service between the publishers and subscribers
- Forwards events from publishers to the interested subscribers
- Handles memberships of publishers and subscribers

- Responsible for keeping the addresses of publishers and subscribers
- Creates distributed nodes representing an overlay on the physical network for matching

In Figure 1, a basic Pub/Sub architecture is shown in which publishers can publish events, subscribers can subscribe by functions, and the middleware takes care of the routing.

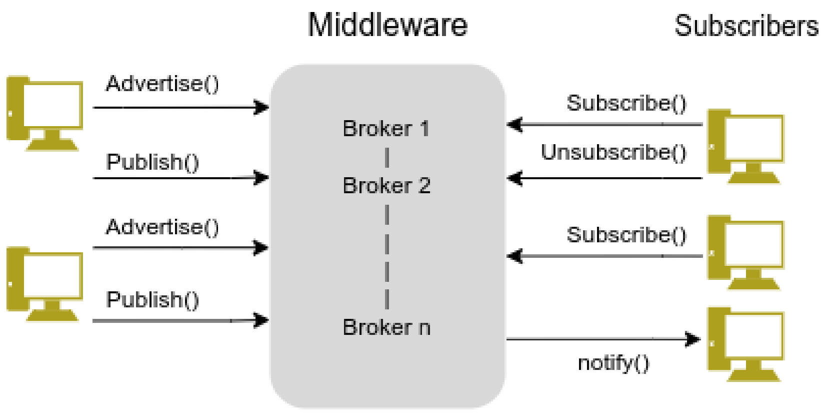


Figure 1: Basic architecture of a Pub/Sub system

In large-scale distributed systems, it is desirable to have a decoupled nature of the publishers and subscribers. It allows them to operate independently and creates a more scalable system. This decoupled nature can be implemented in the following dimensions:

- Space: Publishers and subscribers do not know each other or their location. Communication goes through an intermediate entity.
- Time: The publisher and subscriber do not need to wait on each other.
- Synchronisation: The communication is asynchronous, i.e., a publisher does not have to wait on all subscribers to receive an event to execute another task.

The routing method in a Pub/Sub-system depends on the architecture. Filtering of the events can happen on by metadata tags named topics (topic-based) or by the content of the packets (content-based).

Topic based

Topic-based events are published to channels (topics) based on the event's subject. A subscriber can subscribe to such a topic to receive all matched events. The publisher classifies the topics, and thereby the publisher classifies event content. A disadvantage of a topic-based system is that the subscribers have limited expressiveness by subscribing to such a topic; the subscriber will receive all the events classified to that topic.

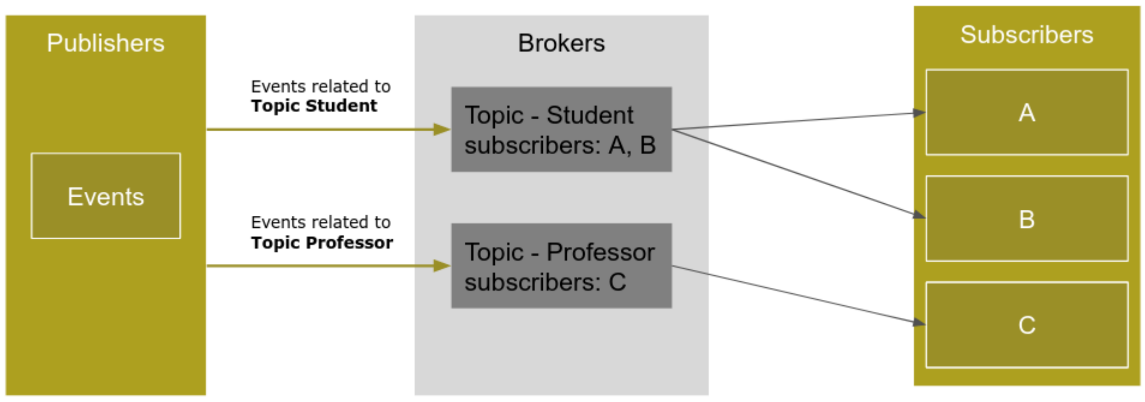

Publishers
Events
Events related to Topic Student
Events related to Topic Professor
Brokers
Topic - Student subscribers: A, B
Topic - Professor subscribers: C
Subscribers
A
B
C

Figure 2: Basic architecture of topic-based routing

Figure 2 illustrates a basic architecture of topic-based routing. In this example, publishers can publish events on the topic ‘Student’ or ‘Professor’, and subscribers can subscribe to these topics. Subscribers A and B are subscribed to the topic ‘Student’, while subscriber C is subscribed to the topic ‘Professor’. As a result, subscribers A and B will only receive events from the topic ‘Student’, and subscriber C will only receive events published in the topic ‘Professor’.

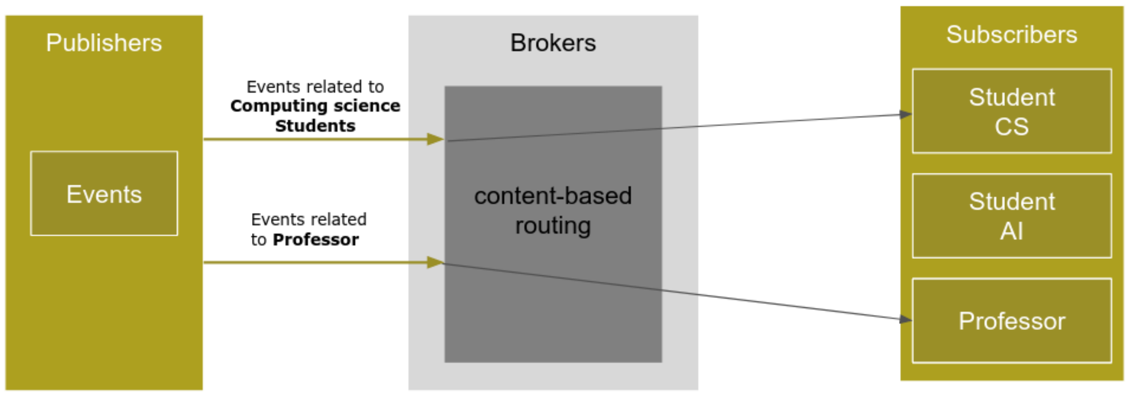


Figure 3: Basic architecture of content-based routing

Content based

In a content-based Pub/Sub system, filters are made using the content of the event, allowing subscribers to express their interests in more detail with constraints on the content. Subscribers can define a fine-grained filter to have more expressiveness in receiving events. In other words, events are not classified by a predefined topic but by using the properties of the events. A disadvantage of content-based Pub/Sub is that it is more expensive to execute than a topic-based match.

Figure 3 illustrates a basic architecture of content-based routing. Publishers can publish an event related to the content of ‘Computing Science’ and ‘Students’. A subscriber interested in ‘Computing Science’ and ‘Students’ will receive the event. However, a subscriber interested in ‘Artificial Intelligence’ (AI) and ‘Students’ will not receive the event.

2.2 Software-Defined Networking

In traditional networks, each device, for example, a router, has its control and data plane. Here, network devices have fixed policies and defined rules that are configured by the manufacturer. This fixed manner leads to proprietary software. Furthermore, changing protocols in traditional networks is not possible since reconfiguration is needed for the whole device.

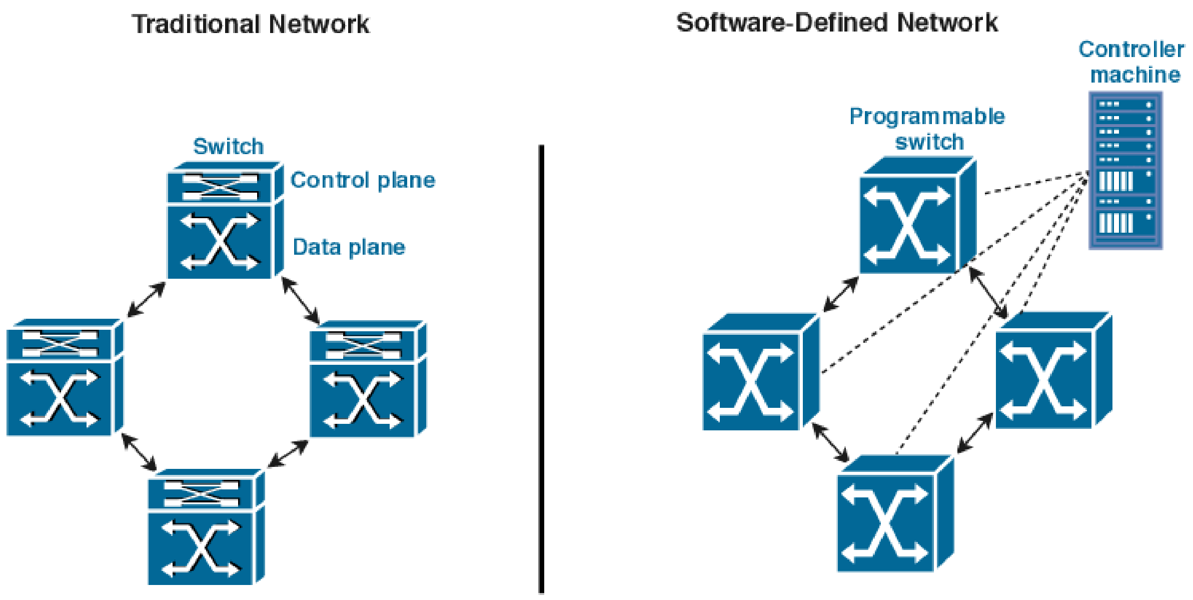


Figure 4: Traditional versus SDN architectures [1]

Software-defined networks came into existence to solve the challenges of traditional networks. SDN is a new paradigm for networking, in which the underlying network is abstracted as an entity that a central controller can control. The term SDN originated from a project at Stanford University, in which it was used as a definition to describe their OpenFlow project [16, 17]. The definition has later on expanded to a broader range of technologies in the networking industry. This thesis uses the four pillars of Kreutz *et al.* that define the SDN as a network architecture [17]:

1. The control and data plane are separated. Therefore, the controller logic is directly programmable since it is decoupled from the forwarding plane.

2. Forwarding rules are flow based.

3. The network intelligence is centrally managed. The SDN controller has the control logic.

4. The SDN is, using software running on the controller in the control plane, programmable. The dynamic, automated SDN programs do not depend on proprietary software, which makes configuring, securing and optimising network resources relatively easy.

This SDN concept separates the traditional architecture of network devices and defines a new network abstraction into two separate entities, which is illustrated in Figure 4. The traditional network is a decentralised architecture in which every switch has its control layer; in an SDN, the control layer is centralised. Creating this separation comes with some benefits:

- The global topology and network configuration can be shared with all the network applications. Therefore, policy decisions and efficient routing rules can be made.
- Automation increases because of the ability to provision resources at will and not to reconfigure resources manually.
- It makes logic and configuration more programmable.
- Open standards like OpenFlow and P4Runtime interfaces and no vendor lock-in, since control logic is provided by the SDN controllers and not by multiple (different) closed-source, proprietary network devices.

However, using an SDN is still a new technology that also has some bottlenecks introduced:

- Since the controller manages the network centralised, it is a single point of failure. If the controller stops working it will bring the entire network down.
- Scalability of the control plane is a bottleneck in large networks with thousands of switches. The increase in network traffic does not scale with the performance of one controller.
- It is more prone to distributed denial-of-service attacks.

2.2.1 SDN planes

An SDN architecture consists of three layers and two interfaces between these layers. An example of a SDN architecture can be seen in Figure 5. The components of SDN will be briefly introduced.

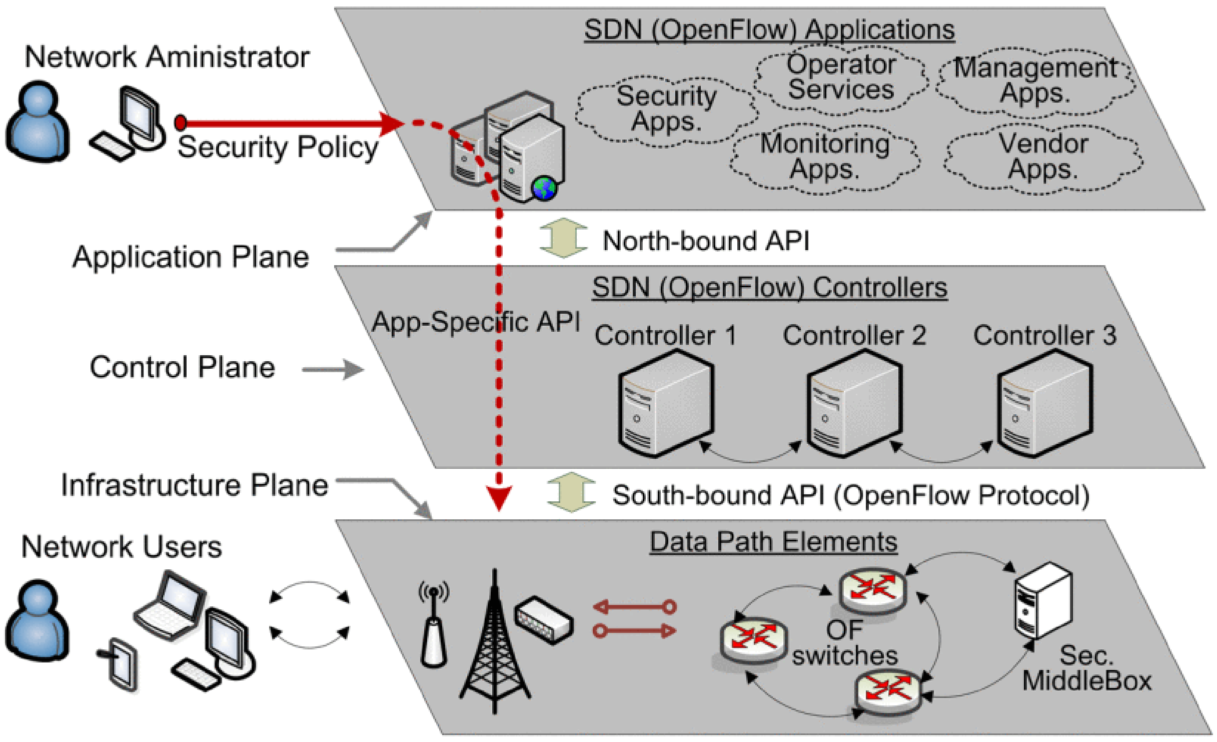


Figure 5: SDN reference architecture with network components [2]

- **Data plane** - The data plane is the network infrastructure where network elements are connected. In Figure 5, the data plane is presented as ‘infrastructure plane’. These elements can be traditional Ethernet switches, firewalls or routers. However, no full control layer is implemented in these devices but is offloaded to the control plane. Therefore, the southbound interface facilitates communication between the data and control plane. The OpenFlow communication protocol is an example of a southbound. Updates to the network devices can be done at any moment.

- **Control plane** - The control plane contains the network controllers that configure the network devices in the data plane. Controllers are responsible for services such as keeping track of statistics, keeping track of the overall network view and topology, routing and security. The northbound API are interfaces between the application plane with the control plane. The controller needs to communicate with the infrastructure layer to receive information about the networking devices and to create and edit the logic. Horizontally, the control planes have East- and Westbound interfaces; These interfaces enable connecting multiple controllers for scalability to bigger distributed networks.

- **Infrastructure plane** - The application plane, also known as the management plane, contains the overhead tasks that do not interact with the data plane. It has network applications that act as the intelligence of the entire network. The plane is connected via the Northbound API to the control plane.

2.2.2 Data plane protocols

The original SDN concept was to control the data flows based on existing Internet protocol headers such as IP, UDP and TCP. OpenFlow is a commonly used southbound interface that defines communication between the control plane and the data plane. OpenFlow is the standard protocol that can be used in the data plane architectures. However, when a new protocol header is introduced in a network, all routers need to know this new protocol before it can be processed. It is impossible to define new protocols when a network consists of static hardware switches since the hardware cannot be changed. A new technology, known as P4, Domain-Specific Language, solves this problem; it enables the programming of the packet forwarding behaviour of a switch.

P4 is flexible in defining new customised protocols and packet headers and specifying code for dynamically parsing these headers. Moreover, it allows custom forwarding tables and header field manipulation while forwarding packets. P4 can run on both software and hardware switches, and it has access to high-performance programmable packet processors [18]. Hence, switches do not need to wait for hardware updates by vendors. Network administrators can execute their own installations. This makes P4 completely protocol-independent and highly adaptable to new applications.

2.2.3 P4 workflow

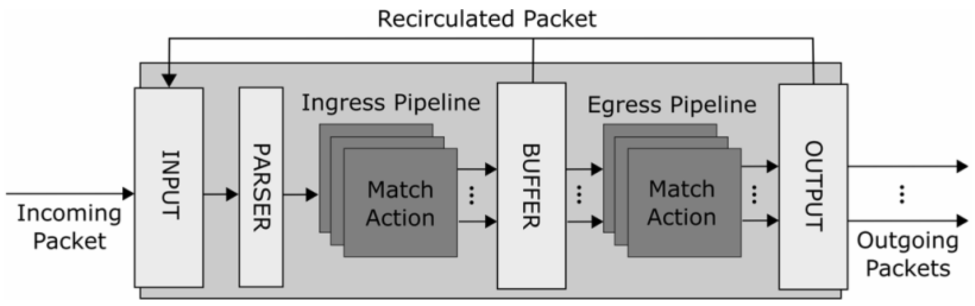


Figure 6: P4's abstract forwarding model [3]

Figure 6 shows a logical abstraction of how P4 switches forward packets through the programmable model. First, the incoming packet is parsed. The P4 program executes a finite state machine to rely on for parsing. Then, the ingress pipeline is entered, and the header field is matched using the match-action tables. Then it moves to the egress pipeline, where the rules are executed from the header fields. Finally, the packet will be deparsed using the final state for outputting the message again.

Each parser needs to know the number of fields it contains. Moreover, the size of fields is defined for the parser to recognise the transitions between the fields. The size of the header fields is specified in the number of bits used and can be written down in a P4 program.

2.3 Security

The main security focus in Content-based Publish/Subscribe systems is to secure the information communicated between the publishers and subscribers. In Pub/Sub access control and authorization of publishers and subscribersis more challenging due to their decoupled nature. Wang *et al.* [19] have pointed out security issues being authentication, integrity, and confidentiality in Pub/Sub systems. Authenticity in information security stands for verifying the identity of parties involved in the network. Integrity in information security is preventing the injection of other data by attackers. Confidentiality is needed to ensure that valuable information of Pub/Sub communication is private. Wang *et al.* [19] introduces the challenge of protecting the security of the data while having content-based routing and the decoupled property between publishers and subscribers. Therefore, confidential event distribution and key management are challenges for designing secure Pub/Sub systems.

To ensure security, encryption is needed in communication protocols. The basic understanding of encryption is that two parties want to share a secret without anyone else being able to understand this secret. For generating keys for a cryptographic scheme, a symmetric key can be generated that both parties use or an asymmetric key in which the encryption key is different from the decryption key.

In symmetric key cryptography, two parties share a unique key. However, the key distribution to both parties is rather complex. The key needs to be distributed securely to the parties that want to communicate with each other without revealing it to a third party. Moreover, it has more loose access control if encryption and decryption are executed with the same key.

In asymmetric key cryptography (public key cryptography), a party has two keys called a private key and a public key. Other users can encrypt a message with the public key. With the private key, it can be decrypted again due to a mathematical relation created between the public and private keys. A party can share his public key to a publicly accessible server or share it via an unsecured channel with another party. In general, this scheme is slower than a symmetric key scheme. But the distribution of keys is easier.
Public-key cryptography can be used to ensure confidentiality, like Attribute-based Encryption (ABE). ABE is an encryption technique that uses a set of attributes to encrypt and decrypt the data. It was derived from Identity-based Encryption (IBE) by Sahai *et al.* [20]. A user can have attributes, and the user's secret keys are based on these certain attributes. Implicitly, The key corresponds to attributes according to some policy rather than identities. A user can decrypt a ciphertext if his attributes match the access policy of the ciphertext. Three entities are needed in ABE schemes; users that can encrypt and decrypt messages and a key authority that generates the master keys and the attribute's private keys. The access policy is a collection of attributes and attribute value pairs. Most access policies have a data structure tree in which the nodes contain logical operators and the leaves the attributes.

If a cryptographic scheme is used, it is important to define who can create the keys and how they will be distributed. Key management schemes for CBPS schemes are:

- **Central authority** - A central authority can generate and distribute the keys for the cryptographic scheme. It is a beneficial architecture if a common key needs to be shared with multiple users. However, the network must trust this external service to realise this approach.
- **Publisher** - A publisher can generate and distribute a key needed for the decryption of his publication. The publisher has full access control of his data. Nevertheless, it increases the overhead of the publisher, and it can harm the decoupled property between the publisher and subscriber.
- **No key exchange** - No key exchange is needed if a cryptographic scheme is chosen in which no keys need to be exchanged. Public-key cryptography using only public keys (commutative) does not need management.

2.3.1 ABE

Attribute-based encryption is a cryptographic implementation in which encryption of a message using a key that is produced of attributes. Decryption in this system is done with a key that matches these attributes too. This cryptosystem is based on contextual information and the identity of certain attributes and was first proposed by Goyal *et al.* [21]. This paper introduced new asymmetric encryption, namely Key-policy attribute-based Encryption (KP-ABE). On this basis, Bethencourt *et al.* [22] came up with a slightly different ABE scheme which is Ciphertext attribute-based Encryption (CP-ABE).
The idea of ABE is to define access policies of the attributes with Boolean operators; if the receiver should be interested in 'weather' and is a 'student' or a 'professor', this can be written as; $Weather \wedge (Student \vee Professor)$. This Boolean formula is an access policy that should be converted into an access structure. This access structure, mainly represented as a tree, can be integrated into the ciphertext or the user's key that decrypts a message. With key-policy-based ABE, the ciphertext is described using attributes and the access policy is mapped into a decryption key. Since the policy is used for the private keys of the key issues, it is called a key-policy scheme. In CP-ABE, it is vice versa; the ciphertext is described with an access policy and the decryption key uses just attributes. Since the policy is used in the ciphertext by the encryptor, it is referred to as a cipher-policy. Therefore, the key generation algorithm defined in ABE schemes uses as input an access policy for CP-ABE and attributes for KP-ABE.
The expressive, fine-grained access control policies are the main advantage of using the ABE scheme. It fits the solution of using the SDN since it has no restrictions on the number of authorised entities, which makes it scalable. The heavy computation overhead is the main disadvantage of the ABE scheme. Especially the encryption operations take up a lot of memory and resources. In production, this is a major issue in IoT research. IoT networks have constrained devices with limited capacity.

2.3.2 Definition of CP-ABE

The CP-ABE scheme is based on four algorithms called Setup(), Encrypt(), KeyGen(), and Decrypt() [22]. The structure of these algorithms is as follows:

Setup()
Input: given a security parameter (S)
Output: a master secret key (MSK), a public key (PK) and public parameters

KeyGen()
Input: users attribute (Att) taken from the universe of attributes A
Output: decryption key for the user (SK_{Attr})

Encrypt()
Input: access policy (AP), message (M) and the public key (PK)
Output: ciphertext (C)

Decrypt
Input: ciphertext (CM), access policy (AP), the decryption key (SK_{Attr})
Output: message M only if the attributes of the decryption key (SK_{Attr}) satisfy the access policy (AP) of the generated ciphertext (C)

Each ABE scheme should meet some security requirements, which will be listed:

- Collusion resistance; users cannot share and combine their attributes to decrypt a ciphertext with an access policy that each user partially has. It can only be decrypted with the decryption key of one user.
- Data confidentiality; The ciphertext should not reveal information about the plain text.
- Fine-grained access control; The encryptor should be able to create multiple different access policies for their data.
- Policy updating; The encryptor should be able to update their access policies for the encrypting algorithm efficiently.
- Attribute revocation; If a decryptor does not have some of its attributes anymore, it should not be able to use old keys with these attributes for decryption in the future.

3 Related works

Much work has been done on Pub/Sub systems in recent years. Some of these works even focused on securing Pub/Sub systems. However, due to the upcoming networking paradigm SDN, new opportunities have been introduced for these researchers. Some recent works have looked into Pub/Sub systems integrated with SDN. This chapter will give a literature review on different research topics organised and discussed in the following sections.

3.1 Content-based pub/sub

Various literature presents approaches to implementing Pub/Sub broker-based systems with content-based filtering and routing of events [5–7]. Brokers collect from the subscriber's interests and design routing tables to satisfy the matching. More recent work focused on broker-less Pub/Sub systems, called peer-to-peer systems. These systems do not have a logical intermediary, but the publishers and subscribers handle forwarding.

The Scalable Internet Event Notification Architecture (SIENA) from Carzaniga et al. [7] is an example of a Pub/Sub-system capable of content-based routing. The system makes subscription spanning trees filter events to brokers that do not host interested subscribers. Shi et al. [5] introduced a topic-based solution for Pub/Sub systems. Since content-based Pub/Sub allows expressive defined subscriptions, it is a favourable solution for the routing scheme. Most content-based Pub/Sub systems work has focused on a good performance and achieving scalability [6]. However, these solutions use brokers attached to the network for performing filtering. This extra layer between the publisher and subscriber imposes a delay since an event must go via the broker to match against installed filters. None of these papers looked at a solution on the network layer that filters events and dispatches on the path. Srivatsa et al. propose Eventguard, an overlay network over a SIENA Pub/Sub system [8]. It uses the same in-network matching operators as SIENA. Nonetheless, this solution creates an overlay network that suffers from performance issues.

3.2 Security for content-based pub/sub

For developing a secure content-based Pub/Sub system, many works focus on the issues of confidentiality and privacy [12, 14, 23–26]. Ensuring confidentiality imposes new challenges in encrypting the data, routing this encrypted data in the system, and maintaining a key management system. For symmetric key encryption, a single key is needed; The one key solution is faster than asymmetric-key encryption since only one key needs to be generated and maintained for sender and receiver. Chen *et al.* [27] use symmetric encryption to implement content-based routing supporting range filtering. Every subscriber has a symmetric key. This key is then hashed in a one-way hash function to prevent brute force attacks. The main disadvantage is that range filtering using a shared key is vulnerable to plain text attacks, given that x greater than y is the same as x plus k greater than y plus k. In adaption to Chen *et al.* [27] proposed solution, Li *et al.* [28] propose a prefix-preserving encryption scheme that uses symmetric encryption but uses prefix filtering instead of range filtering. Content-based filtering needs fine-grained filtering based on the event's content; using the solution from Li *et al.* confidentiality is preserved against eavesdropping and to achieve range and prefix filtering. They propose a geometric representation of the prefix-preserving encryption function, in which each non-leaf node specifies a binary variable of the plaintext tree. So applying the encryption function on a plaintext tree results in rearranging the tree into a ciphertree. Raiciu *et al.* [29] use symmetric encryption by performing keyword matching using Bloom filters and supporting range matching using dictionaries.

The publishers and subscribers share security parameters to hash their messages. Keywords are divided into fixed-sized strings called words. When a subscriber is interested in a keyword, it sends the indices of the matching words hashed as a subscription. Publishers set the words in its publication to the value one using a dictionary data structure. Brokers use bloom filters to insert the received hashed words and send the events to interested subscribers. These bloom filters support equality filtering, fixed-sized range matching and keyword matching, which could be more granular to define the content expressively. In those mentioned above, symmetric key solutions require sharing a secret key between publishers and subscribers. Knowing the loosely coupled property of Pub/Sub systems, these solutions are not desirable.

Nabeel *et al.* [26] use a public key cryptosystem by Paillier. The authors use this cryptosystem to encrypt each attribute of an event. Subscribers have an extra step by multiplying their interest with value -1 before encryption. Using the Paillier homomorphic property, $E(M1) \doteq E(-M2) = E(M1 - 2)$. A broker will have multiple attributes of an event with the subscription. Decrypting E(M1) or E(M2) can perform range filtering while not knowing both values of M1 and M2. However, this Pub/Sub system can only match equality.

Anusree *et al.* [12] ensure confidentiality, authentication, integrity and non-repudiation in the broker-less publish/subscribe service using the elliptic curve identity-based signcryption. This variant of IBE performs the encryption function and digital signature. The security depends on the elliptic curve discrete logarithm problem, making the keys smaller and, therefore, less computational cost and communication cost than non-EC cryptography. The publisher can decide who should or not access their data. However, no routing mechanism or matching schemes describe how the events are disseminated.

More literature can be found on functional encryption. Functional encryption is a public-key encryption scheme that uses different decryption keys, allowing users to learn the encrypted data's specific functions. Techniques applied are versions of ABE and IBE. In Tariq *et al.* [14], they design a broker-less content-based Pub/Sub system by using ABE to ensure security in the Pub/Sub. They introduce fine-grained key management in this approach. Anusree *et al.* [12] have a solution in which filtering is done by subscribers instead of intermediate brokers. In literature, this is called broker-less solutions in which publishers and subscribers organise themselves forming an event overlay [14]. Maithili *et al.* [13] follow Tariq et al. [14] in this approach to use a broker-less Pub/Sub system and construct a secure system by using a fuzzy variant of identity-based encryption. Srivatsa *et al.* [8] use ABE in their solution; however, they distinguish between insensitive and non-sensitive attributes. Sensitive attributes will be encrypted, and non-sensitive attributes will be used for routing the events via the brokers. This weaker implementation of confidentiality reduces the number of keys but will not work in all use cases if all the data is defined as sensitive. Ion *et al.* [23] uses attribute-based encryption in the variants KP-ABE and CP-ABE to support confidentiality-preserving content-based forwarding. Combining multi-user searchable data encryption for matching events with interests supports advertisement and subscription privacy. However, sharing private keys between the publisher and subscriber is not met.

Scalable key management must be met to ensure the use of encryption techniques by the publishers and subscribers. Keys must be disseminated while keeping the loosely coupled property between publishers and subscribers. Most of the papers [23, 24, 28–30] rely on a third party to maintain the keys and disseminate information. Yoon *et al.* [31] propose a fully distributed solution to key management implementing Shamir's secret sharing scheme. This scheme supports the distributed solution by sending the secret into multiple pieces into the network. When subscribers have the appropriate number of pieces, they can know the secret value. Unfortunately, the support for using sensitive data for expressive content-based routing is lacking in this solution.

While the paper, as mentioned earlier, focuses primarily on information security attributes, confidentiality, integrity, and authenticity, more security issues can occur. In Dahlmanns *et al.* [32] is argued that the security in the pub/pub system is strongly tied to the security of the brokers mediating all data flows. They introduce two attack vectors on the Pub/Sub communication. First, a misconfigured broker in which an attacker could inject commands into the communication flow and, second, the malicious/compromised broker is not a trusted entity that violates confidentiality. They introduce five requirements for a Pub/Sub system to mitigate the introduced attack vector. Firstly, confidentiality, integrity and authenticity need to be ensured. In addition, the detection, dropping, and duplication of messages need to be notified by the subscribers, the system should be designed in a deployable manner, and no significant communication latency and data latency must occur.

3.3 Content-based pub/sub using SDN

The literature mentioned above is distributed Pub/Sub systems supporting content-based routing between publishers and subscribers. However, the systems solutions are implemented in the application layer. Application layer solutions cause poor performance compared to communication protocols in the network layer. Koldehofe *et al.* [10] expose this problem and state that content-based routing implemented on the network layer would be highly effective; the routing is more bandwidth efficient, and the routing could perform line-rate forwarding of packets. They show that the SDN provides an abstraction for configuring the publish/subscribe middleware. By mapping the matching process of the pub/sub system with the OpenFlow entries, a controller can disseminate routing information to the switches in an SDN. The paper of Tariq *et al.* [4] introduced a content-based middleware network layer solution named PLEROMA.

The controller establishes a communication channel between the hosts, and events are forwarded to the network layer. Zhang *et al.* [11] argue that there is a more fundamental study of SDN that will benefit the Pub/Sub-system implementation. They introduce the upcoming networking paradigm SDN and the impact on Pub/Sub-middleware solutions. They propose separating the traditional roles of the publishers and subscribers to fulfil the requirement to separate the data and control plane in an SDN. The control plane with the advertisements and subscription roles. The data plane with the publication production and consumption role. Moreover, Bhownik *et al.* [15] assessed the solution of content-based Pub/Sub middleware from Tariq *et al.* [4] very extensively, making it one of the most modern middleware implementations using the SDN approach.

In all the literature mentioned above, [4, 10, 11] are using the OpenFlow protocol 1.0 in which the flow rules are a one-to-one communication rule, and no multicasting is implemented. These flow rules can fill up a flow table quite fast. Hung *et al.* [33] uses the OpenFlow 1.3 protocol to introduce multicasting, which reduces the flow table size and involves no extra multicast switch or multicast addresses.

These works show significant steps in implementing Pub/Sub systems in the network layer. Some Pub/Sub systems are designed to work with the new SDN paradigm. However, the previous works focus more on efficiency than on the security of such implementations. To secure Pub/Sub systems, many security solutions have been designed [34]. The next section will elaborate on secure implementation for CBPS using SDN.

3.4 Security for content-based pub/sub using SDN

Using the decoupling property of SDN to separate the control plane and data plane raises the level of decoupling and can introduce powerful functionality. Zhang *et al.* [11] present an SDN-like Pub/Sub model which borrows properties from SDN. They separate the role of the Pub/Sub model. They deliver an architecture of a logically centralised pub/sub controller that can be implemented on top of an existing Pub/Sub-engine. This paper addresses the new design SDN brings along for message-oriented middleware. However, no security guarantees are introduced. Pub/sub systems that use brokers for communication cause a significant impact on bandwidth costs. Tariq *et al.* [4] designed PLEROMA which is an SDN-based high-performance Pub/Sub middleware, and evaluated this middleware solution. Hungyo *et al.* [33] introduce an SDN-based implementation of Pub/Sub systems using OpenFlow. The solution entails a better implementation of the traditional IP multicast, namely OpenFlow multicast. However, this paper's solution is based on IP-level port forwarding rules and IPv4 addresses; security mechanisms are not discussed. Wernecke *et al.* [3] uses P4 programming language to define their solution in the data plane using programmable switches. The solution introduces a hybrid approach using forwarding rules in the switches and routing information in the events. Security mechanism remains not discussed.

The literature content-based routing for a Pub/Sub system over a SDN is rare; considering security by designing such systems is less focused on. Most of the literature focuses on OpenFlow implementations and does not consider the benefits of a programmable data plane using the P4 language. To fill this literature gap, there is a need to analyse security mechanisms in a Pub/Sub-system model that allows in-network event dissemination over software-defined networks.

4 Problem Statement

Due to the loosely coupled property of SDN, the Pub/Sub-system is a great communication protocol to integrate into the implementation of SDN. Most of the literature has been focused on the scalability and expressiveness of such a Pub/Sub model in SDN. Nevertheless, less focus is set on the security of such an implementation. In an ideal world, switches in SDN would be trustworthy compared to the traditional broker-based Pub/Sub middlewares. However, in a more realistic model, the switches in the network are not trusted, and the dissemination of the published event cannot be labelled as secure. Providing subscription confidentiality in a Pub/Sub system without weakening the decoupled property is still an open issue [14].

This thesis aims to evaluate the implementation of security mechanisms in Pub/Sub systems using SDN architecture. To reach this aim, some secondary objectives need to be met. The first objective is the understanding of the security goals in the Pub/Sub model using SDN. The second objective is to propose a new approach and design for this Pub/Sub model. It should provide confidentiality of publications and subscriptions. This approach includes a mechanism to match (encrypted) events with (encrypted) filters for event dissemination. Furthermore, the security implementation should strive for a key management system that manages the keys of publishers and subscribers without losing the decoupled property. These security mechanisms must be efficient and should not significantly burden the network or cause too much overhead. Therefore, this thesis will analyse the impact of the security mechanisms on communication for the overall implementation.

Research question: Can an SDN provide secure content-based Pub/Sub communication without weakening the decoupled property? Is it possible to route encrypted events from a publisher to a subscriber? And finally, what are the trade-off of implementing security mechanisms in a content-based Pub/Sub-SDN scheme?

4.1 System model

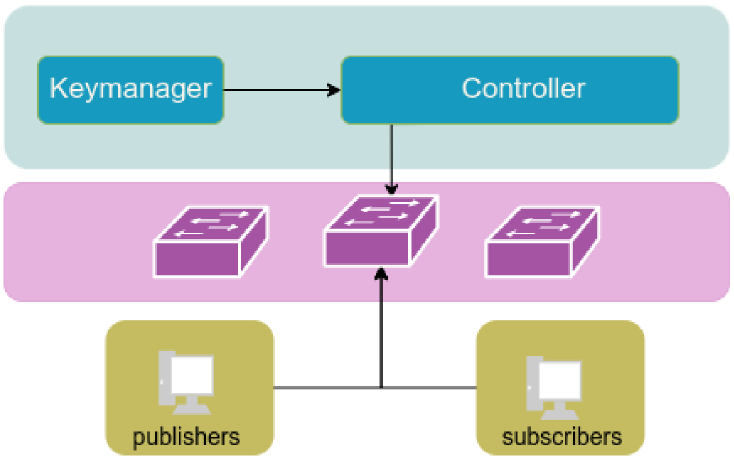


Figure 7: System model of the Pub/Sub using SDN

The system model introduces a Pub/Sub system using an SDN that can be seen in Figure 7. It illustrates the component's control plane (blue), the data plane (purple) and the publishers and subscribers (green). The control plane has the components controller and the Key manager. The controller keeps track of the advertisements and subscriptions while the Key manager functions as the trusted authority (TA) needed for public key encryption schemes. The data plane has all the switches integrated into the architecture.

4.2 Threat model

Following the literature, concerning the security of such models [35], the assumption is that the control plane is fully trusted. The switches in the data plane are assumed to be honest but curious. The switches will route the events following the flow tables honestly, but are also curious and try to infer information from the events by eavesdropping on messages and learning about the content of the events or filters. On the other hand, the centralised controller is fully honest and will not be a curious party. The publishers and subscribers will stay within their roles and follow their protocols. However, they are curious about other data that can break the subscribers' privacy and will collect more information if possible. The communication paths between these end-users and the control plane are assumed to be secure.

We consider the following threat model:

- Honest but curious publishers that are hosts to the SDN network.
- Honest but curious subscribers that are hosts to the SDN network.
- Fully trusted controller, including centralised key management.
- Honest but curious P4 switches.

4.3 Design goals

To set our security goals for the privacy-by-design implementation of the pub/sub system, confidentiality, integrity and authentication regulations need to be analysed. The first goal is that subscribers only accept events that authorised publishers have produced. Secondly, the goal is to prevent unauthorised subscribers from revealing information content if the interest does not match the access policy. To preserve the decoupled property between publishers and subscribers, they should refrain from communicating with each other directly or knowing their location or identity. The security goals are summarised as follows;

Security Goals:

- Publication confidentiality; the secret of a published event can not be known by non-authorised subscribers, even if they have an interest that matches the published event.
- Subscription confidentiality; The interest of a subscriber should not be known by other subscribers, publishers, switches, or outsiders.
- Integrity; unauthorised parties cannot modify or delete the event's content.
- Publication authentication; Only authorised publishers can publish an event in the system.
- Subscription authentication; Only authorised subscribers will receive and decrypt events.

It is crucial to have a secure model. However, the scalability efficiency of the network should be maintained. For example, a model will not be scalable if the model's keys are dependent on the number of subscribers. Moreover, the communication costs are high if the switches have to communicate with the controller in every step to process a subscription or route an event. Therefore, the efficiency goals are summarised as follows;

Efficiency Goals:

- Communication overhead; The number of messages exchanged for key management should not be too high.
- Computation overhead; The keys' computation, generation, and forwarding should be as low as possible. Publishers and subscribers can have limited computation capabilities.
- Usability and scalability: The designed system should support expressive publications and subscriptions, while the control plane can handle incoming requests.

5 Design

The following chapter will explain and highlight designs that could be implemented to pursue the security goals for the Pub/Sub system implementation in SDN. After considering different design alternatives, only one final design is proposed to scope the project in Chapter 6.

5.1 SDN configuration

In Chapter 4 'Problem statement', the new architecture for the Pub/Sub communication model is illustrated in Figure 7. The SDN replaces the broker functionality, and instead, there is a central control plane that allows for central key management. However, what type of switches are used and how the events for advertisement, subscription and publication are expressed will be detailed in this chapter.

5.1.1 OpenFlow versus P4 switches

Using the OpenFlow (OF) standard in switches has enabled the decoupling of the data and control planes. OF is therefore widely supported in the academic field and industry. In the work as mentioned earlier by Koldehofe *et al.* [10], the authors focus on content-based filtering in Pub/Sub middlewares from middleware boxes to OF switches for line-rate filtering. However, implementing OF as the communication protocol for the data plane results in predefined pipelines and non-flexible packet header parsers and matching.

Alternatively, packet headers can be created using programmable P4 switches to interconnect the publishers and subscribers. Creating new packet headers allows for more expressiveness and options to route events. Importantly, distribution trees of subscribers can be encoded in the header. Therefore no controller information is needed to forward the packet [36]. This decision is crucial for the rest of the proposed solution model since the routing possibilities depend on which headers contain routing information and whether potential new headers require a new design of the parsing pipeline.

5.1.2 Expressiveness of subscriptions and publications

A publisher's goal is to publish content as an event message, while the subscriber's goal is to receive the events it is interested in based on the content. However, the expressiveness of the content presents a challenge to the matching process. In literature [10], flexible attribute/values pairs are represented as $< name, value >$. An event is a set of attribute-value pairs, e.g. $[< energy, 200 >, < price, 20 >]$. The set could also be extended with an extra comparison operator $(<, >, =, = / =, \leq, \geq)$, which results in a representation $< name, value, operator >$, e.g. $< energy, 200,' <' >$. For subscribers to receive events, they need to register their interest within the data plane through a filter. A filter can be written as a logical expression of predicates between attribute values, e.g. filter $= [energy < 150\ \&\ energy \geq 50]$. Possible combinations of matching subscription interests and advertisement access policies are shown in Table 1.

Publisher	**Subscriber**
Attr1, Attr2	Attr1 and Attr2
Attr1 and Attr2	Attr1, Attr2
Attr1 = 5	Attr1 <10 or Attr2 >13
Attr1 <10 or Attr2 >13	Attr1 = 5

Table 1: Combinations how a subscription interest and advertisement access policy can be configured

After expressing interests and publications using attributes, the attributes need a mapping to the packet header fields of an event. An event consists of a payload with the actual data and packet header fields, which allows for the insertion of routing information that the switches in the data plane will read.

PLEROMA solution from Tariq *et al.* [4] uses an event space to express the content of the subscription's interest and advertisement's access policy. Event- and subscription data is defined with the decomposition space in the PLEROMA middleware. PLEROMA defines the whole n-dimensional event as ϵ in which the event exists. Figure 8 shows an example of such an event space. The binary representation of this space is called a DZ and can indicate the event subspace to which an event belongs or in which a subscriber is interested.

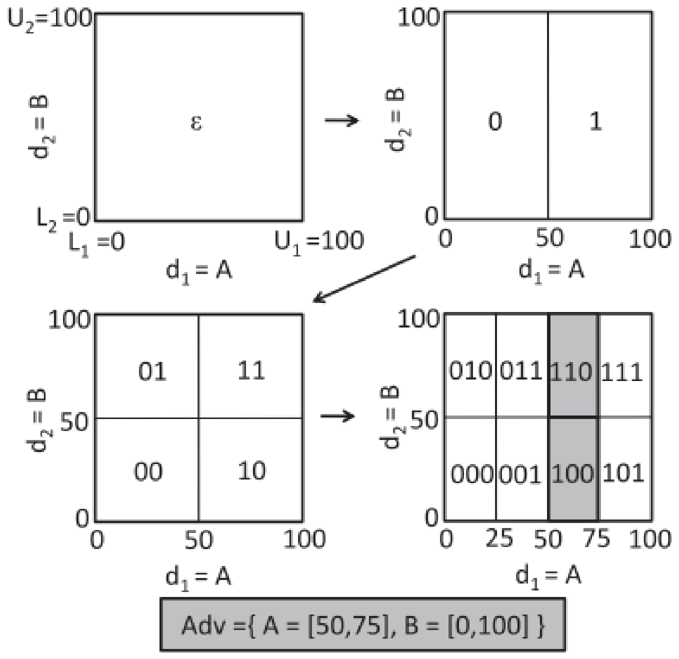


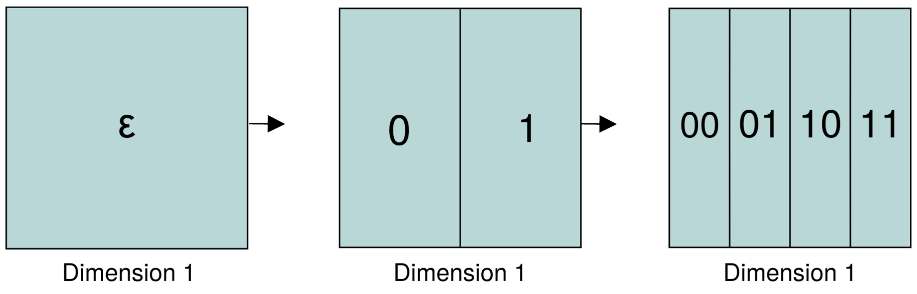
Figure 8: Decomposition of an example event space of two attributes A and B using the definition of PLEROMA [4]

The main advantage of DZ strings is the covering relationship for prefix-based routing. A short DZ can cover a long DZ if the shorter DZ is a prefix of the longer DZ. For example, the $DZ\_a = 0$ covers $DZ\_b = 01$ and $DZ\_c = 1010$ covers $DZ\_d = 101010101$. Moreover, the DZ strings are granular, which makes it a very expressive solution for the expressive content-based model. The only limitation is the event space range maximum; The maximum range size depends on the implementation in the code.

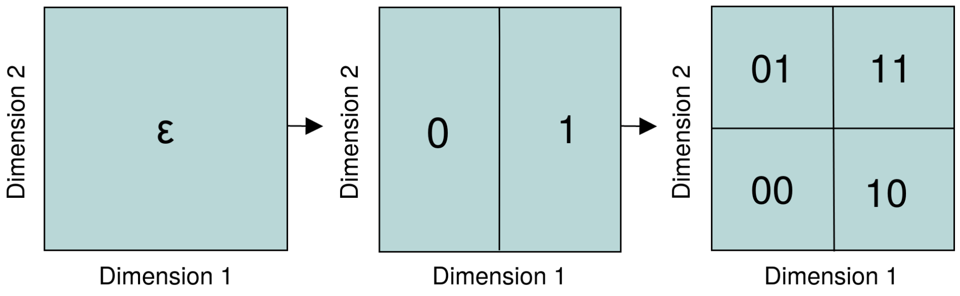
Pub or Sub	**A**	**B**	**Raw DZ**	**final DZ**
Sub 1	50-75	50-75	{100, 110} && {010,011,110,111}	{110}
Sub 2	50-75	50-75	{100, 110} OR {010,011,110,111}	{100,010,011,110,111}
Sub 3	0-100	70	{0, 1} && {110}	{110}
Sub 4	0-100	70	{0, 1} OR {110}	{0,1,110}
Pub 1	40	40	{001}	-
Pub 2	35	80	{011}	-
Pub 3	90	80	{111}	-

Table 2: Examples of attribute values and the final DZ string based on the event space in Figure 8

Table 2 shows a publisher or subscriber interested in attributes A and B but with varying ranges. Column 'Raw DZ' translates these spaces into a DZ string following the event space in Figure 8. To deliver an event from Publisher 1 (Pub 1) shown in Table 2, all subscribers mapped to these subspaces should receive the event. Subscribers subscribed to subspace {001} should receive the event, and subscribers interested in the subspaces 00, 0 and ϵ. When more attributes are defined in the network, the event space becomes more extensive; therefore, scalability is a concern. Tariq *et al.* [14] defines an alternative for this problem by creating event spaces for each attribute separately. Therefore, every attribute will have its decomposition tree and thus, its own DZ string. Figure 9 shows two possible representations of how the events space can be mapped to the binary representation. In Figure 9a, a one-dimensional example represents an event space for a single attribute, while in Figure 9b an event space for two attributes can be seen. With these designs, the filtering will be more fine-grained on each dimension when the binary string gets longer.



(a) Event space indexing with one dimension



(b) Event space indexing with two dimensions

Figure 9: Content representation using spatial indexing with en event space ϵ

While converting events and filters into a DZ string that can be used for prefix-based routing is highly beneficial, it is not a secure routing design if the DZ space is publicly known; the length of the DZ string can say something about the granularity of the interest or publication. This information leak can be prevented if the controller only knows the event space. However, an anonymous event space requires sharing the event space with every subscriber and publisher if they want to send or receive an event. Hence, private event spaces can cause much overhead for the controller.

5.1.3 Data plane and control plane limitations

In the data plane, a filter's expressiveness also impacts the system model's bandwidth efficiency. The bandwidth overhead increases if a published event is sent to a subscriber who is not interested. Therefore, it is essential to use effective routing to prevent needle bandwidth increases in the network and a bad design would increase the communication overhead. In addition, flow entries and tables are limited. In one clock cycle, switches can perform line-rate forwarding using its Ternary Content-Addressable Memory (TCAM). However, switches using TCAM are limited by a maximum of 150.000 possible entries [10]. In addition, each flow entry individually has a limited number of bits available to express matching rules.

In the control plane, the controller needs to process all the events of publishers and subscribers in the system and react to them by installing or modifying flow rules. A controller can only process one incoming event simultaneously to preserve consistency and can not deal with a lot of computation overhead. The controller overhead will greatly increase if the proposed system model involves controller communication in every event. Therefore, preventing the controller's involvement in every event would be favourable while designing the solution.

5.1.4 Routing

Besides securing the data, it is also essential to efficiently forward events from the publishers to certain subscribers based on the advertisements and the subscriptions. Forwardig rules are installed on the OpenFlow switches to forward events in the data plane. However, the security aspect in the routing schemes and the benefits of P4 implementations require re-evaluating the forwarding of events.

A simple solution of content-based routing is flooding. Flooding is a broadcast pattern; a published event is flooded to all the subscribers. Unfortunately, flooding creates much overhead, and subscribers receive events that they are not interested in. Unwanted events subscribers receive, increase the false positive (FP) rate enormously. Moreover, the decryption cost will increase if the subscriber will decrypt every received event.

IP multicasting is seen as a better approach to content-based routing than flooding. It is a communication pattern in which a single event can be sent to a group of receivers. Multicasting with only one multicast group with all receivers can be seen as the broadcast pattern. Multicasting is a good solution for topic-based communication in which one multicast group is defined for each topic. Content-based interests are more expressive and will not fall into static multicast groups. In other words, any combination of subscribers could be possible for each event due to the expressive nature of the subscriber's interest. Many multicast groups should then be established to cover all combinations in more extensive networks. Depending on the design decision where these multicast trees should be stored, it can take a lot of storage space in, for example, a switch. Moreover, the switches' routing table will be overloaded if the flow rule for every expressive multicast group is defined. As mentioned above in section 5.1.3, an abundance of flow rules is undesirable due to TCAM memory limitations [15]. In brief, the drawback of this approach for content-based filtering is limited scalability and complexity of management.

The DZ expressions solution uses this multicast approach by translating the DZ expressions into IPv6 multicast addresses. The covering relationship of DZ realises multicasting. However, honest-but-curious switches can see the plain DZ expressions as binary strings revealing information. A randomization design could help to make the plain DZ expressions less obvious. Design examples could be shifting the string between switches, multiplying it with a secret binary string, or creating a randomize function. This design aims to make the switches less knowledgeable while the events will be forwarded using the correct forwarding rules. Securing the content-based routing will be discussed further in the next section.

5.2 Security

Providing secure communication must be achieved in multiple ways. The message content from a publisher to a subscriber should be secured, and the message needs to be matched and forwarded securely. It is important to know how the publishers and subscribers receive keys, what kind of cryptography scheme and key management is suitable, and how to securely the routing events from a publisher to a subscriber.

5.2.1 Secure communication between the pub/sub and the controller

Publishers and subscribers are hosts of the SDN network. No direct connection to the control plane and controller is set by default. This scope makes it difficult to authenticate the publishers and subscribers and receive keys from the trusted authority. In an unsecured Pub/Sub SDN, a publisher can send a new event in the data plane; if the switches do not have a rule for this event, they will send it to the controller. The communication flow of this situation can be seen in Figure 10 inside the SDN. This use case has been elaborated in multiple studies on Pub/Sub communication on SDN [4, 10, 11, 15, 33]. In a secure Pub/Sub communication, the publisher and subscriber first need to obtain keys from the key manager before secure communication is possible. If the key manager is situated in the control plane, a secure channel should be established between each publisher/subscriber and the control plane. No plain keys can be sent via the SDN to maintain the secure communication. This secure communication channel, visualised with a red arrow in Figure 10, allows an authentication protocol or a key exchange between the controller and the publisher/subscriber.

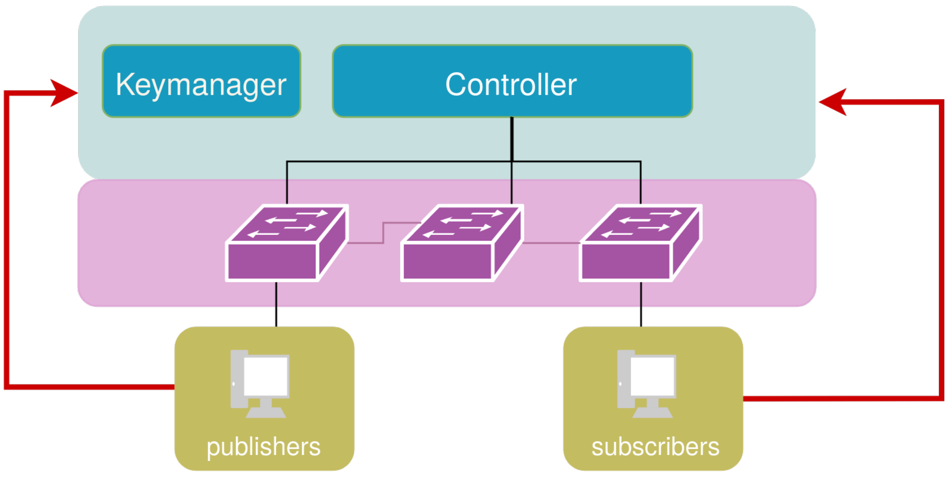


Figure 10: Communication possibilities for the publishers and subscribers with the controller. The red communication line is outside the SDN network.

5.2.2 Key management

A suitable key management for this solution is a central key authority. The solution architecture uses SDN, so a central authority is already incorporated into the network. Moreover, if the publisher is responsible for distributing the keys, it can conflict with the decoupled property between the publishers and subscribers. In a CBPS model, every publication event can have different interested subscribers, which makes key management challenging. One idea is to create subscriber groups and assign a group key to the group subscribers. This static approach can lead to managing up to 2^n keys, where n = subscribers, since each publication event can go to a different group. All keys are generated by the key manager, disseminated by the switches and managed by the controller. Assigning keys to different subscriber groups is not scalable when the number of subscribers increasesand not a solution for the dynamic nature of a CBPS model [37].

Another solution for key management is to encrypt each event using individual subscribers' keys. Since the event is encrypted with only the desired subscriber's key, this option is secure. However, it will weaken the decoupled property between the publisher and the subscriber since the publisher needs to know the key of the receiving subscribers and is not scalable.

A better approach would be to generate keys based on specific properties, such as attributes, so the solution does not depend on the number of subscribers. An example of an attribute-based cryptography scheme is ABE. Since attribute-based schemes the user can create his key using its attributes, sharing keys is not challenging anymore. However, if the user is not authorised anymore by the system to have the key, key management is needed to support key revocation.

5.2.3 Event encryption

Encrypting the payload of the message ensures information confidentiality. The easiest solution would be to encrypt the message on the publisher side and send it via the switches to the subscribers, who can decrypt the event on their side. There is a need for a design that allows authorised subscribers to decrypt events without contacting the source publisher for keys.
Foremost, the encryption key that the publisher needs for encrypting its event should be decryptable for the interested subscriber. Since this implies fine-grained data access control over encrypted data, an Attribute-based encryption scheme would be well-suited. The expressive, fine-grained access control policies are the main advantage of using the ABE scheme. In addition, ABE scheme fits the SDN solution since it has no restrictions on the number of authorised entities, which makes it scalable. However, there are many options within ABE schemes. Choosing the KP-ABE scheme allows the key issuer in our solution, i.e. the controller, to define an access policy. On the other hand, when choosing CP-ABE, the responsibility of defining access control lies with the publisher. In addition, different implementations of both schemes exist.

The heavy computation overhead is the main disadvantage of the ABE scheme. Especially the encryption operations take up a lot of memory and resources. This overhead is a significant issue in IoT research, where IoT devices are constrained with limited computational and memory capacity. The variation in ABE schemes implementations is extensive, each having its optimisations and specific use cases. However, optimising the choice of ABE scheme for the proposed solution model falls outside the scope of this thesis; as a proof-of-principle, the original ABE scheme will be assessed [22].

Importantly, only authorised subscribers and publishers can send events to the network in the proposed solution model. For generating key pairs for publishers and subscribers without sharing keys, Identity-based Encryption is a promising solution. With IBE, any identifier which is unique to the user can be used as a public key. The main disadvantage of this encryption design is that the publisher should know the ID of the subscribers to perform the encryption, and subscribers should know the publisher's public key to decrypt. Knowing each other's ID compromises the loosely coupled nature of the Pub/Sub system. An improved design would involve the controller handling the authorisation; the controller knows its network and does not have a decoupled nature with the publisher or subscriber. The controller will then reject an advertisement or subscription event if the subscriber or publisher is unauthorised by the controller. An authorisation design is, therefore, essential and helps with access control in the network. A disadvantage of centralised entity is the single point of failure.

5.2.4 Secure routing

Routing events in the data plane is done by looking at the packet header fields of the event. These headers can be standard fields, e.g. the EtherType header or IPv4 header, or customised headers, e.g. P4, where new headers can be added. However, since content-based routing reveals information about the content, information confidentiality should be protected when forwarding events. For example, if a binary string is used for forwarding, which reveals multicast group information, a hashed version could be used in the data plane. The controller must oversee this mapping and install suitable flow rules. A more secure design will be if there is a distinction between subscription groups, defined by the controller, and local subscription groups, installed in the switches. Local subscription groups can result in the anonymity of the actual groups, which the controller oversees. For the routing a randomization function can be made. Likewise, a change to the packet header field could be made every time it leaves an output port of the switch to a new switch.

5.2.5 Decrypting events

The subscriber will decrypt events it receives. The efficiency of decrypting events depends on the design of the system model. If the subscriber receives an event, it will try to decrypt the event's payload. However, the subscriber can have many saved keys and multiple subscriptions. The naive solution would be to try all the keys the subscriber owns and see if one succeeds. If the event does not match the subscriber's interests, the decryption calculations are unnecessary. For example, if flooding is used for routing the events, many false positive events will reach the subscriber leading to many unnecessary decryption attempts. A more fine-grained approach is applied when an event is delivered to a certain cluster. However, it depends on the clustering method of these groups if the received event is an event the subscriber is interested in. Overhead can be created if a subscriber needs to decrypt many messages in which it is not interested or when it needs to try many keys. The most effective design would lead to the subscriber-only receiving events of interest and knowing what key to use for decrypting the payload.

6 Solution

The CBPS model using SDN is proposed in this chapter based on the design considerations described in Chapter 5. First, the model architecture is described. Second, the cryptographic setup is explained; the solution uses CP-ABE to secure the publishers' data. Finally, content-based Pub/Sub communication over SDN needs secure routing, which is explained in the routing section.

6.1 Model architecture

6.1.1 System model and entities

A new system model is proposed for securing a content-based Pub/Sub-system over a SDN. In this system, the traditional roles of the publisher and subscriber are split into advertisers, producers, interest managers, and consumers, respectively [11]. The advertisers and interest managers will operate in the control plane of the SDN, while the producer and the consumer will be connected as hosts via the SDN. For implementation purposes, the definitions 'publisher' and 'subscriber' will still be used for the producers and consumers. The functions of advertisement and subscription will fulfil the roles of advertisers and interest managers. In an extension of Figure 7, a simplified overview of the system model can be seen in Figure 11 where the control plane (blue), the data plane (purple) and the publishers/subscribers (green) are visualised. These publishers and subscribers can interact with switches from the data plane.

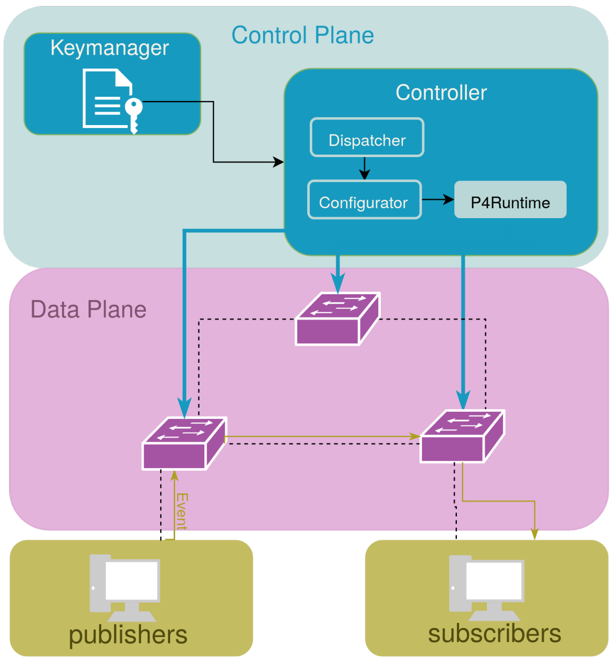


Figure 11: The new system model of the Pub/Sub using SDN

The solution entails a secure connection between the publishers/subscribers, the control plane, and its key manager. This connection is required for the authentication step, which will be explained later. This design is chosen to maintain loosely coupled communication of the CBPS system.

In our system model, we have the end devices, which we call publishers or subscribers. The publishers and subscribers are denoted by $P = \{P_1, P_2, \dots P_n\}$ and $S = \{S_1, S_2, \dots S_n\}$, respectively. These publishers and subscribers are connected to the switches in the data plane. Packets flowing through the data plane are called events, and the source of this event can be the publisher, the subscriber or a reply from the controller. A subscriber can only send a subscription to the controller, which we call a subscription request. A subscriber first needs to send out a subscription before receiving any messages. On the other hand, a publisher must send an advertisement before it can send publications. The first communication step with the controller is needed because the controller does not know the publisher's and subscribers' attributes by default.

A subscriber can be interested in multiple subscriptions, and a publisher can send multiple advertisements. Moreover, subscribers can have overlapping subscriptions with other subscribers and the same holds for an advertisement. Publishers and subscribers are unique and have a unique identity (ID).

6.1.2 Attribute space

To design the model, the definition of the attribute space is needed. Publishers can advertise and publish. Subscribers can consume the data and can subscribe. Both need a formal definition of how they can express their publication and interest. The proposed solution is based on the following structure for the publication and subscription data:

Attribute - An attribute $attr\_0$ is a $< name >$ with the name being a string which should be unique. The simplest form is to match the attribute string, e.g., attribute is "Student".
Attribute value - An attribute value $< value >$ with value being an integer or string. The attribute value belongs to an attribute name, e.g. "Course, CS".
Attribute pair - An attribute pair looks like $< name, value, operator >$ with the name being the attribute name, value the attribute value and the operator a chosen comparison operator from the set $(<, >, =, = / =, \leq, \geq)$.
Interest - An interest is a Boolean expression of attribute pairs and attributes concatenating by the Boolean operators $\wedge$ and $\vee$, e.g. $attr_0 = 5 \wedge attr_1, > 100$.

In the attribute space definition, expressiveness is defined as how fine-grained content can be. An attribute alone is less expressive than an attribute pair.
For the cryptographic protocol described in section 6.2, a monotonic access tree is needed for encryption. The monotonic access tree has leaves which represent the attribute pairs or attributes. These attributes (pairs) are connected in the tree with inner nodes that function as Boolean operators $\wedge$ and $\vee$. The formal definition follows:

Monotone Access Structure. Let $A = \{attr_1, attr_2, \dots attr_n\}$ be a set of attributes. In set $\Gamma = 2^{(attr_1 \cdot attr_2 \cdot attr_*)}$, for $\forall M, N$: if $M \in A$ and $M \subseteq N$, then $N \in \Gamma$, we say Γ is an monotonic access tree.

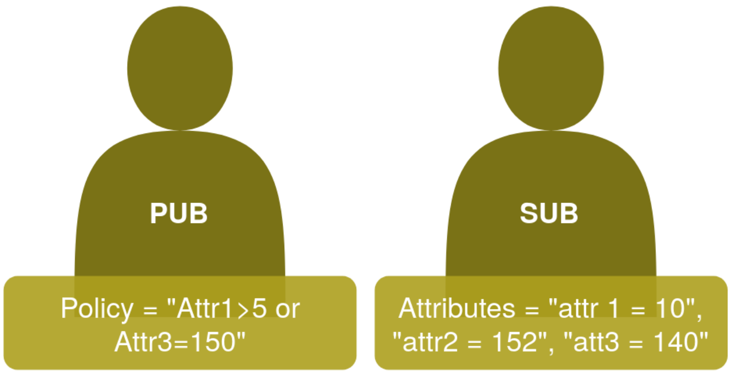

PUB
SUB
Policy = "Attr1>5 or Attr3=150"
Attributes = "attr 1 = 10", "attr2 = 152", "att3 = 140"

Figure 12: Example of a policy and an interest

In Figure 12 you can see an example of an access policy of the publisher and the interest of the subscriber.

6.2 Cryptographic protocol

In the proposed model, the concept of ABE is used, which was first introduced by Shai *et al.* [20]. This ABE scheme is used instead of IBE since the publisher cannot know the identities of the interested subscribers beforehand. The ciphertext and keys are labelled with attributes in the ABE construction. A key can decrypt a ciphertext if at least k attributes match between the key and ciphertext. In Tariq *et al.* [14], unique ciphertexts are created using the CP-ABE scheme for each of the subspaces that match with the event. An example, if publisher P has for $attr_0$ a DZ-string $=$ 00 and for $attr_1$ a DZ-string$=$ 1, credentials; Cred($attr_0$, 0), Cred($attr_0$, 00), Cred($attr_1$, 1) should be obtained. Publisher P can then encrypt its event separately with the given credentials and put this in the event's payload. However, this has the disadvantage that a publisher should duplicate the event as much as his DZ string has subspaces which create overhead. Duplicate events will cause some flooding of the same event with different encryption policies.

To take advantage of the ABE scheme, the event is encrypted with the full attribute policy of the publisher. Subscribers will only receive the packet if their interest string matches with the publisher access policy. This is taken care of in the routing logic. So, the subscriber can decrypt encrypted events when it matches the publisher's access policy which is the purpose of CP-ABE. The original idea of PLEROMA DZ strings will be used for creating subscriber groups in the proposed solution to create overlapping groups. However, the original attribute policy without DZ strings will be used for the cryptography for the payload of events. This results in encrypting the event once with the access policy and not multiple times used in PLEROMA.

6.2.1 CP-ABE scheme

The CP-ABE scheme is based on four algorithms called Setup(), Encrypt(), KeyGen(), and Decrypt() [22]. Some mathematical definitions need to be set to start executing the algorithm; Let λ be the security parameter and G a bilinear group of order p, where p is a prime number of at least λ bits. A non-degenerate bilinear map $e : G \times G \rightarrow G_T$ will be defined and set $S \subseteq Z_p$. For this thesis, the ABE scheme of Bethencourt *et al.* [22] is used. This scheme is the same scheme that is used by Tariq *et al.* [14] for securing a broker-less Pub/Sub system and is also a widely known CP-ABE scheme that has proven in literature [13, 22, 23, 38] to help secure CBPS systems.

Definition of Bethencourt *et al.* [22] follows:

— **Setup**(λ) **:** The initialisation algorithm chooses a bilinear group G of order p with generator g. It chooses a random α and $\beta \in Z_p$, so that the master **secret** key (MSK) is composed of (β, g^{α}).

— **KeyGen(MSK, S):** The KeyGen algorithm takes as input the set of attributes S and outputs a key that corresponds to that set. The algorithm picks a random $r \in Z_p$ and for each attribute $j \in S$ it randomly obtains $r_j \in_R Z_p$. Then the key is computed as

$$\mathrm{SK} = \left(S, D = g^{(\alpha+r)\beta'}, \{D, = g^r \mathcal{H}(i)^r, D' = g^r\}_{j \in S}\right).$$

- **Encrypt(MPK, T, M):** The Encrypt() algorithm encrypts message $M \in G_r$ under the access policy T by generating polynomials P_x for each node x of T.

— **Decrypt(CT, SK):** The Decrypt() algorithm will be performed recursively. Let $\mathcal{T}$ be the access policy corresponding to the ciphertext (CT), and S is the set of attributes corresponding to the secret key (SK). Assume that $\mathcal{T} \models \mathcal{S}$. Let Y' be the set of leaf nodes of $\mathcal{T}$ such that at $(y) \in \mathcal{S}$ for all $y \in Y'$. Then for all $y \in Y'$ defined is $i = \operatorname{att}(y)$ and computed:

$$\frac{e\left(D_i, C_y\right)}{e\left(D_i', C_y'\right)} = \frac{e\left(g^r \mathcal{H}(i)^{r_i}, g^{P_y(0)}\right)}{e\left(g^{r_i}, \mathcal{H}(i)^{P_u(0)}\right)} = e(g, g)^{r P_y(0)}.$$

For these leaf nodes, defined is the output of the function DecryptNode(CT, SK, y) as $e(g, g)^{r P_\nu(0)}$. Otherwise, the output is defined as $\perp$. Then for each non-leaf node z the output is defined for DecryptNode(CT, SK ,z) as F_z. This value is computed recursively by looking at the values of its children. By invoking the DecryptNode function on the root node r, computed is $A = \text{DecryptNode}\ (\text{CT}, \text{SK}, r) = c(g, g)^{rs}$. The plaintext M can be retrieved by computing

$$\frac{\hat{CA}}{e(C, D)} = \frac{Me(g, g)^{as} e(g, g)^{rs}}{e\left(h^x, g^{(\alpha+r)/\beta}\right)} = \frac{Me(g, g)^{as+rs}}{e(g, g)^{(a+r)s}} = M$$

Cryptographic interaction between the components

In Figure 13, the interaction diagram for the cryptographic protocol is visualised. The controller can execute the Setup() algorithm and the Keygen() algorithm; it outputs the master private key (PK) and the master secret key (MSK). This key pair sets the foundation for all the other cryptographic algorithms models using a centralised authority. The controller is represented as the controller and the key manager in this case. Both lie in the control plane and have a secure connection. The Setup() algorithm is called once by the controller. Keygen() is executed for every subscriber that requests a subscription with a given interest. Keygen() outputs a secret key that represents the subscriber's interest. Encrypt() is executed when a publisher wants to send an event and needs to encrypt the payload of that event. The publisher has the message M it wants to send, its access policy A, but it needs the public key from the controller. The output of Encrypt() is a ciphertext (CP). At last, the subscriber calls Decrypt () for each event it wants to decrypt. For the Decrypt() algorithm, the input is the received ciphertext (CT), a secret key (SK) that represents its interest and the public key (PK) from the controller. The output will result in the plaintext message (M) again or fail when the subscriber's interest does not match the access policy (A) of the published event.

Before interaction between the publishers/subscribers and the controller via the data plane, an authentication step is first executed. This authentication step is done at the controller, generating a key pair for each authorised publisher or subscriber. This step is needed to provide secure communication for advertisement and subscriptions.

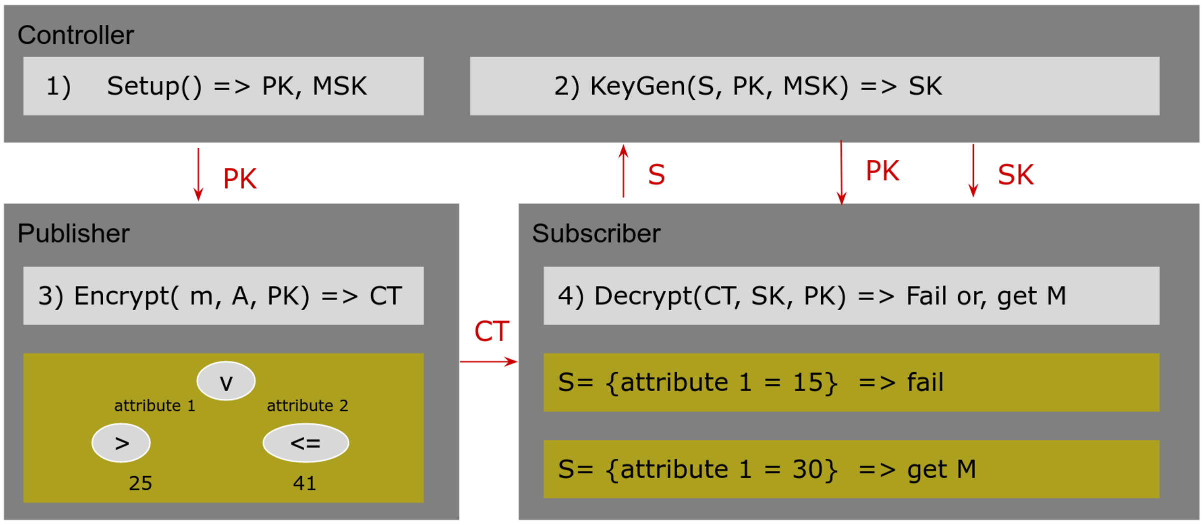

Controller
1) Setup() => PK, MSK
2) KeyGen(S, PK, MSK) => SK
PK
S
PK
SK
Publisher
3) Encrypt(m, A, PK) => CT
V
attribute 1
attribute 2
>
<=
25
41
CT
Subscriber
4) Decrypt(CT, SK, PK) => Fail or, get M
S= {attribute 1 = 15} => fail
S= {attribute 1 = 30} => get M

Figure 13: The overview of the CP-ABE scheme embedded in proposed solution

6.3 Routing

In a Pub/Sub model, advertisement, subscription and publication events must be forwarded between the entities. Considering the chosen CP-ABE protocols, the following section will describe the creation of packets and the secure forwarding between the entities.

6.3.1 Packets creation

Three packets will be sent in the data plane for advertisement, subscription and publication. In Figure 14 the structure of these packets is shown. For this solution, the EtherType values EtherType=0x9003 and EtherType=0x9004 are used as a reference for the switches to know the match in the match/action table for parsing the Ether header. These EtherType values are unused in existing protocols.

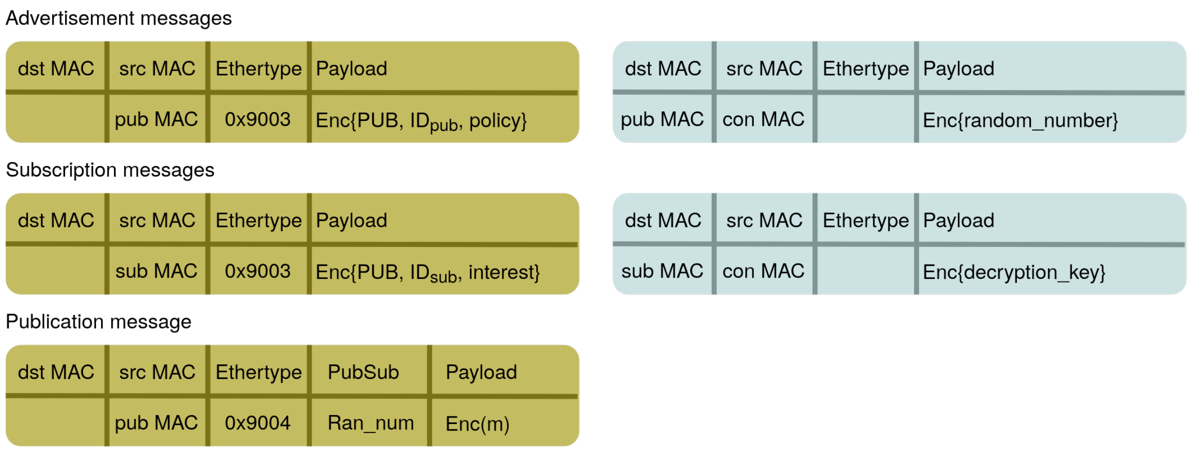


Figure 14: Format of the messages for Advertisement, Subscription and Publication. The green-coloured messages are from Pub/Sub to the controller, and the blue-coloured messages are the replies from the controller.

6.3.2 Publisher

Publishers are connected to switch(es). However, their identity has yet to be known to the key manager. Before creating advertisements and events, a publisher (idem the subscriber) should execute an authentication step as preparation for the Pub/Sub communication. The control plane consists of a key manager connected to the controller.

When the key manager knows the publisher, we can define the publisher as authenticated publisher and thus trusted. The publisher is then able to follow the protocols for advertisement and publication.

Advertisement Procedure:

- Publisher P has a set of attributes/attribute pairs $A = A1, A2...A_n$.
- Create an access structure (AP) out of the attributes (A) as access policy (AP) for message M.
- Encrypts message M = Encrypt(type, ID, AP) with algorithm Encrypt().
- It creates an advertisement packet following the structure of Figure 14 with his MAC address, EtherType 0x9003 and in the payload message M.
- Send the packet to its connected switch and wait for a response.
- Receive advertisement response from the switch following the structure of Figure 14. The destination MAC is his own MAC address, and the payload has an encrypted message.
- Decrypt the payload message with its private key and tries to succeed.
- Has decrypted a random number that belongs to the access policy (AP).

A diagram of this procedure can be seen in Figure 15. Once the advertisement has been completed, the publisher can send, as many times as possible, events containing a message M under the access policy (AP). This procedure is called publication.

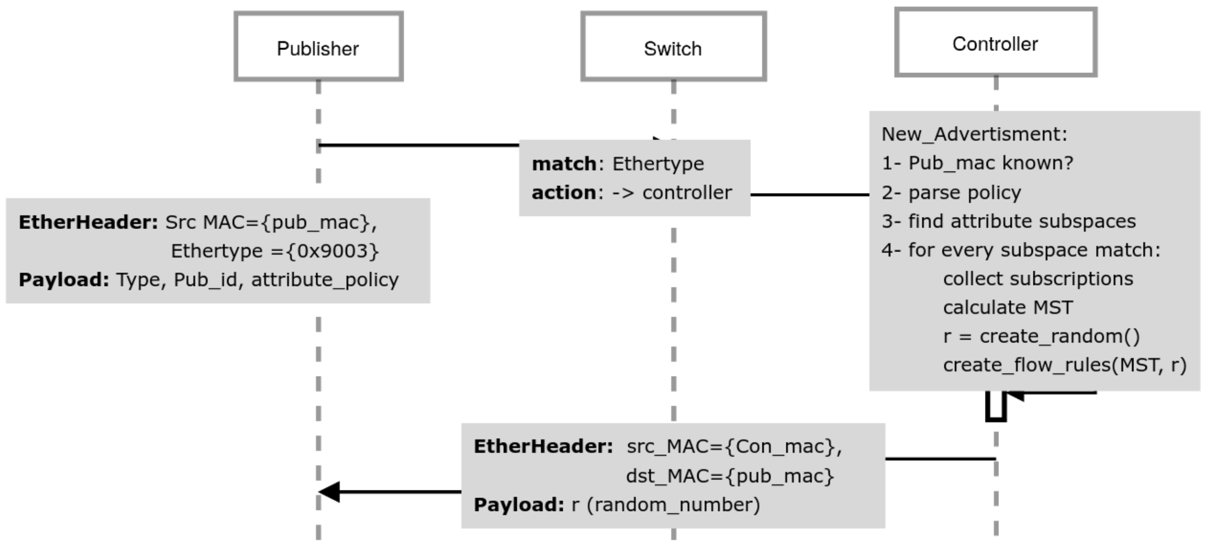

Publisher
Switch
Controller
match: Ethertype
action: -> controller
EtherHeader: Src MAC={pub_mac},
Ethertype ={0x9003}
Payload: Type, Pub_id, attribute_policy
New_Advertisment:
1- Pub_mac known?
2- parse policy
3- find attribute subspaces
4- for every subspace match:
collect subscriptions
calculate MST
r = create_random()
create_flow_rules(MST, r)
EtherHeader: src_MAC={Con_mac},
dst_MAC={pub_mac}
Payload: r (random_number)

Figure 15: The sequence diagram of advertisement

```
header_type routing_header_t {
    random_number: 16;
    }
```

Listing 1: New Pub/Sub header field Definition in P4-14

A diagram of this procedure can be seen in Figure 16, and the procedure steps of the publication algorithm are described below.

Procedure **Publication**:

- Encrypt following the encrypt algorithm message (M) using access policy (AP) and MPK resulting in ciphertext (CT).
- Create a publication packet following the format of Figure 14 with EtherType=0x9004, the random number as Pub/Sub header and as payload the encrypted message CT.
- Send it to the connected switch.

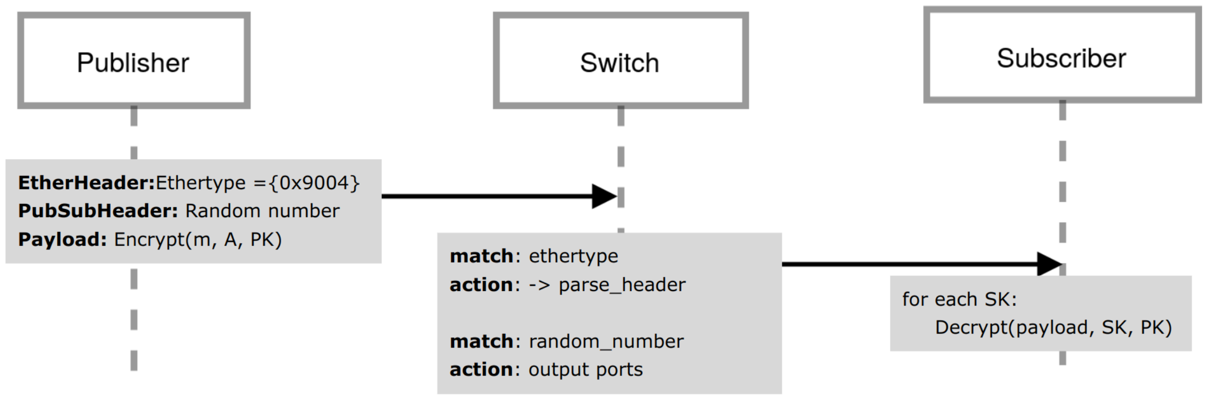


Figure 16: The sequence diagram of publication

6.3.3 Subscriber

Subscribers are connected to switch(es). However, their identity has yet to be known to the key manager. Subscribers must follow the same authorisation protocol as the publishers to be trusted subscribers. See the publisher protocol as mentioned above for the steps. When the key manager knows the subscriber, it can follow the protocol for subscription to show its interest. The procedure for a subscriber to create a subscription is described below.

Subscription Procedure:

- Subscriber S has an interest shown in section 6.1.2.
- It creates a subscription packet following the structure of Figure 14 with his MAC address, EtherType 0x9003 and in the payload his interest.
- Send the created subscription packet to the controller and wait for a response.
- Receive subscription response from the switch following the structure of Figure 14. The destination MAC is his own MAC address, and the payload has a decryption key.
- Decrypt payload message with its private key and tries to succeed
- Stores the decrypted decryption key (SK) concerning the matching interest.

The subscription procedure is executed to receive the decryption key for a giving interest. A diagram of this procedure can be seen in Figure 17. Procedure Receive is there to receive events from switches and try to decrypt them using the decryption keys.

Receive Procedure:

- Receive event (E) from a connected switch (SW).
- For every decryption key (SK) subscriber (S) has, try decrypting the payload of the event (E).
- if successful, obtain message (M).

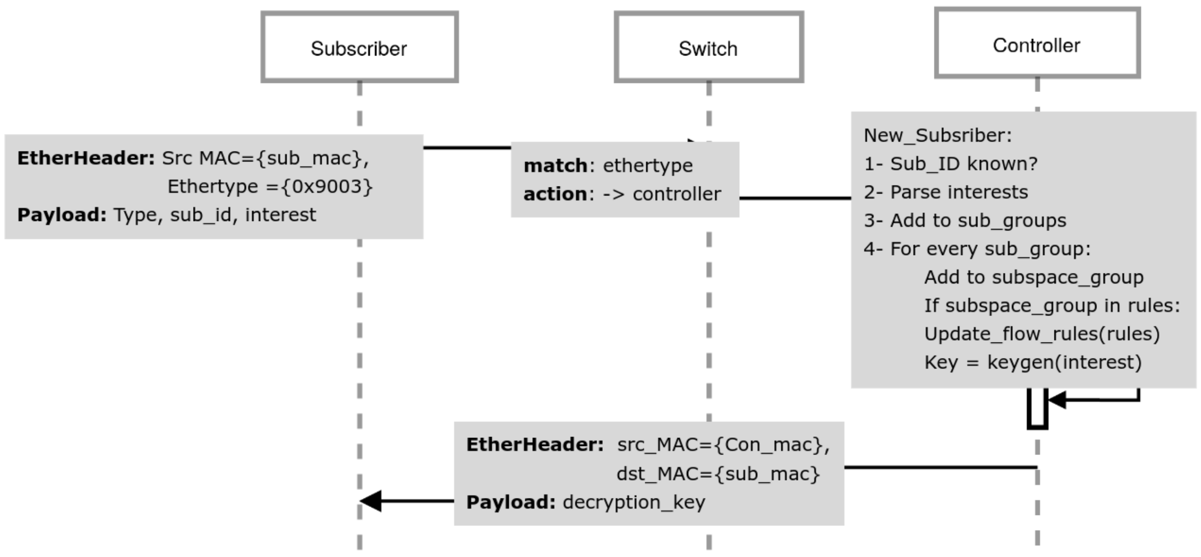


Figure 17: The sequence diagram of subscription

6.3.4 Control plane

The control plane consists roughly of a key manager and a controller. The key manager is a service that manages, distributes and stores the keys. In this section, the key manager and controller are discussed.

Controller

The system model is based on the PLEROMA solution proposed by Bhowmik [15]. Figure 18 illustrates the design of the proposed controller, which is two-tiered architecture including a dispatcher and a configurator. This new solution exposed a P4runtime API to the controller to control the programmable switches in the data plane:

dispatcher - The dispatcher collects events from publishers and subscribers in the system. The dispatcher's role is to serve as an entry point and collects all data plane controller requests. When a packet arrives at the controller, the dispatcher reads the payload values and identifies whether it is an advertisement or a subscription. Afterwards, the subscription or the advertisement is then forwarded to the configurator.
configurator - receives subscriptions and advertisements and performs network updates accordingly; Calculations are executed to calculate the relevant area for the installation and switches. Therefore, it needs to read the current status of the network and decide to update and make changes to the network to fit the new advertisement or subscription.
P4Runtime API - This control plane software is used in the controller to control the forwarding plane of the switches. In this proposed solution, we use programmable switches in the forwarding plane.

The controller acquires a global view of the network which can be used by tools to configure the network. With this solution, the control communicates with the switches programmed with P4 in the data plane. This allows for modifications to the flow table during runtime.

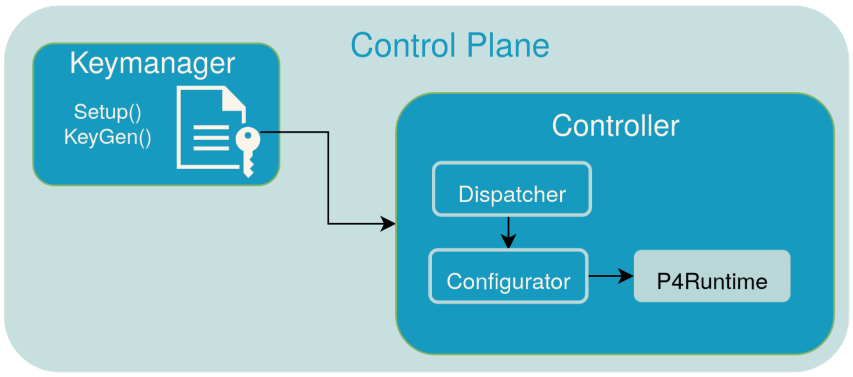
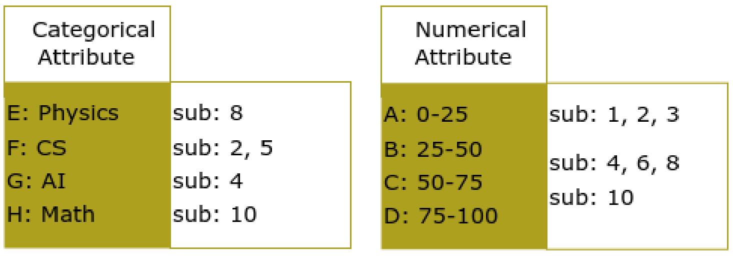


Figure 18: Control plane components overview

Attribute trees

For the proposed solution, every attribute has its range tree in which subscribers that fit in this range will be stored. The data structure allows for expressive subscriptions, including the Boolean operator OR, since subscribers can be stored on more than one leaf node in this tree. The maximum integer value or amount of categories for non-numerical attributes can be provisioned dynamically. Defining the max value of the composition tree is not needed using this data structure. Figure 19 shows an example of an attribute data structure. On the left is an example of a hash map in which the attribute value is non-numerical. On the right side is an example of a ranging tree in which the attribute values are numerical.



Categorical Attribute	
E: Physics	sub: 8
F: CS	sub: 2, 5
G: AI	sub: 4
H: Math	sub: 10

Numerical Attribute	
A: 0-25	sub: 1, 2, 3
B: 25-50	
C: 50-75	sub: 4, 6, 8
D: 75-100	sub: 10

Figure 19: Data structures used for storing the subscriber's interest for each attribute. On the left is a hash map for categorical data, and on the right is a range tree for numerical data.

The subscriber's ID is stored as a tuple with a random number. The tuple combination allows a subscriber to have multiple interests. Multiple interest results in multiple decryption keys. In the controller, while matching advertisements with subscriptions, multiple interests from a unique subscriber can be mixed up. If the interest has a unique ID, this can be prevented.

Spanning tree in the configurator

To embed paths between the publishers and subscribers in a network, a single spanning tree is made by the configurator of the controller. The spanning trees allow low latency paths between the publishers and subscribers. Finding the optimal path between a publisher and the interested subscribers can be seen as an optimisation problem in graphs.

Problem:
Let Graph G = <V, E> a coherent non-directed graph, where E is the set of edges and V is the set of vertices. Each edge will have a weight. A subset of these edges (E) should be defined in which all vertices (V) are connected and the sum of these weights is minimised. This MST is a Minimal Spanning Tree (MST).

For this, the Kruskal algorithm can obtain an MST [39]. The graph would be the network topology, and the traversed switches are then known as well as the input and output ports of the switches. In this way, the number of times events are forwarded can be reduced by optimised routes.

Handling publications and subscriptions

A publisher can send an event following the structure advertisement message in Figure 14. The EtherType for this kind of message is set to 0x9003, which is an unused EtherType so far that is used to send an advertisement or subscription. Switches have a standard flow installed that requires all sent with this EtherType to forward to the controller. The advertisement algorithm steps can be seen in Algorithm 1. First, the controller decrypts the payload using its private key. When this succeeds, the controller knows if it is a PUB or SUB, the ID and the access policy. It checks if the controller knows the publisher by looking into the list of authenticated publishers. If this check succeeds, it will parse the policy included in the advertisement. The parsing procedure makes sure it highlights all the potential subscriber groups and boils down the final subset of subscribers using the Boolean logic provided in the policy.

When the final set of subscribers is defined that match the policy of the advertisement, we create a random number and store this random number with the publisher's ID and final subscribers set in the list routingLogic. The controller is tracking and updating this list to know which former advertisement should change when handling a new subscription.

The MST between a publisher and each subscriber in the final subset will be calculated using the configurator's main MST. When knowing the efficient path, the procedure createFlowRules can be executed to send the proper rules to the switches. Finally, the controller will send this random number created to the publisher via the message format in Figure 14. In the payload, the encrypted random number will be created using the public key from the publisher. The MAC address of the publisher can be copied in the new message and then sent to the data plane.

A subscriber can send a subscription event following the structure subscription message in Figure 14. The Ether type for this kind of message is the same as the above-mentioned advertisement message. The subscription algorithm steps can be seen in Algorithm 2. Like an advertisement, the controller decrypts the payload using its private key. When this succeeds, the controller knows if it is a PUB or SUB, the ID and the access policy. It checks if the controller knows the subscriber by looking into the list of authenticated subscribers. If this check succeeds, it will parse the interest in the subscription. The parsing procedure for subscription ensures that the subscriber is added to all the subscriber groups to which it belongs. This subscription is added as a tuple with a random number to ensure the

Algorithm 1 New Advertisement

```
Ensure: upon event receive(PUB, ID, AP) do
1: function ADVERTISEMENT()(...)
2:     if $ID \subset authorised\_pubs$ then
3:         $subscribers\_groups \leftarrow parsePolicy()$
4:         $subscribers \leftarrow booleanParser(subscribers\_groups)$
5:         $r \leftarrow createRandom()$
6:         $routingLogic \leftarrow add(r, PUB, subscribers)$
7:     for subscriber in subscribers do
8:         $subscribers\_groups \leftarrow calculatePath(MST, PUB, SUB)$
9:     createFlowRules(r, mstSwitches)
```

subscribers' interests are combined in the matching process.

The procedure updateFlowRules can be executed to send rules changes to the switches. This procedure will look into the controller's list routingLogic to decide if subscribers need to be added to older advertisement rules. Finally, it can create a decryption key using KeyGen() and send it to the subscriber.

Algorithm 2 New Subscription

```
Ensure: upon event receive(SUB, ID, interest) do
1: function SUBSCRIPTION()(...)
2:     if $ID \subset authorised\_subs$ then
3:         $parsed\_interest = parseInterest(interest)$
4:         $matches \leftarrow Match(parsed\_interest, subscriber\_groups$
5:         $r \leftarrow createRandom()$
6:         $addToGroup(SUB, subscriber\_groups, random\_number)$
7:     for routing_rule ⊂ routingRules do
8:         updateFlowRules(routing_rule)
9:     $key = KeyGen(interest)$
```

6.3.5 Data plane

The security requirements rely mainly on the solution of the cryptographic scheme; however, routing is an essential part of the solution to create access control and confidentiality. The messages are routed in the data plane of the SDN. The publisher can send two message types to the data plane: advertisement and events. The publisher will send the messages to the connected switch. Switches are then responsible for forwarding the messages to the right subscribers. Furthermore, subscribers can send a subscription message. The task and behaviour of the switch-receiving events will be explained in further detail.

Switch

In order to create P4 switches that can interact with the designed controller, a new packet processing pipeline needs to be designed. In Figure 20 the proposed pipeline can be seen, which starts with link layer headers parsing depending on the content in the headers. First, the Ethernet header is parsed and passed into the ingress pipeline. Since this solution relies on the EtherType of the packet for further parsing, it passes this into the ingress pipeline. The match-action table will have entries for the EtherType 0x9003 and 0x9004 for this solution. Once the Ethernet header is extracted, the next parser can be invoked depending on the value of the EtherType, the Pub/Sub parser or the IPv4 parser. An incoming packet with a successfully parsed Pub/Sub header is passed to the forwarding table of Pub/Sub packets. The controller fills this table with the random number in Algorithm 1 for matching and the correct output ports of the switch as action.

In order to drop a message that is irrelevant for the Pub/Sub communication, the switches can drop a message in the ingress pipeline. This dropping mechanism of the switch allows bandwidth efficiency in the network since it avoids events that do not match any rules. This situation is possible if it is an event from an unauthorised publisher that the controller does not know.

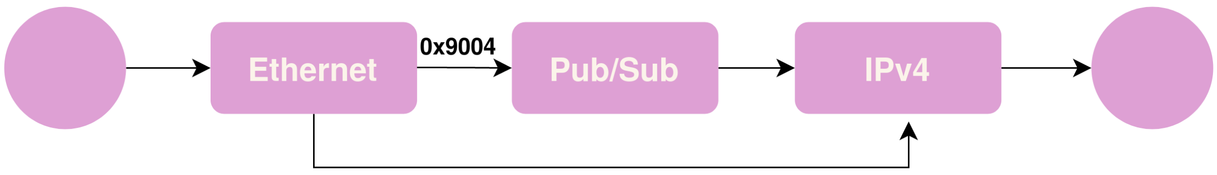


Figure 20: Graph of the P4 parsing pipeline

7 Results

7.1 Security analysis

In this chapter, a qualitative security analysis of the proposed solution in section 6 is performed by discussing the communication's confidentiality, integrity and authentication in different use cases.

7.1.1 Confidentiality

Confidentiality in information security is the ability to protect sensitive information or messages from access by unauthorised parties. Only the expected receivers and sender should know the content of the encrypted information.

Use case 1: Switch is curious about the information in the payload

Following the treated model described in the problem statement in section 4, the switches are honest in forwarding packets following their protocol. However, they are curious about the data that is in the payload. However, all messages forwarded in the data plane are encrypted, and switches will not have keys nor the computation power to execute cryptographic algorithms.

Use case 2: Subscribers will collude with each other to reveal information

A known collusion attack is called an accessor attack, where different parties try to combine their keys to decrypt messages they cannot decrypt alone. Fortunately, the CP-ABE scheme is collusion-resistant [22]. Using a random number for the algorithms Setup() and Keygen(), you need the same random number in the private key to decrypt a ciphertext. A combination of random number is not allowed; therefore, an accessor attack cannot be successful.

Use case 3: Compromised switches will send data to unauthorised paths

Packets in the data plane should traverse from the source to the destination along the path authorised by the controller. However, switches might get compromised or misconfigured intentionally and forward packets via unauthorised routes. If the packet reaches a switch that does not belong to the route, this switch will not have a rule for the packet. Consequently, the packet will be dropped in this switch.

On the other hand, if a packet is wrongfully forwarded to a publisher, the publisher will not recognise the MAC address of this packet, and no further decryption of the payload will follow. Publishers are usually not receivers but this can happen by a flood decision of the switch.

In another scenario, a packet, either an advertisement packet, subscription packet or publication packet, is incorrectly sent to a subscriber. In this case, no information on the payload is revealed since the payload is encrypted with a private key of the sending publisher. A subscriber will nonetheless try to decrypt the payload but will not succeed. The subscriber can prevent this fruitless decryption attempt by checking whether the ethertype is correct; this measure is not implemented in the current solution but is a good extension for efficiency. It can create extra decryption overhead for subscribers but does not cause security risks.

7.1.2 Integrity

In information security, integrity is the ability to protect the message from alterations by unauthorised parties.

Use case 4: External malicious parties performs man-in-the-middle attack

If an unauthorised party wants to alter a payload, it will eavesdrop on the communication between the publisher and the controller. This man-in-the-middle party might alter the payload, even though it is encrypted, and resend it to the switch. The switch will not recognise this alteration in the payload, and the packet continues its intended path. When the package arrives at the subscriber, the payload will not be decrypted successfully, as the ciphertext is modified. In this scenario, there is no influence on the confidentiality of the payload, but the integrity is at risk. Packet integrity verification methods to detect unauthorised alterations of the payload have not been implemented, but it would be very interesting to explore further. One could keep track of statistics of individual packet flows, such as changes in payload bit size.

7.1.3 Authenticity

Authenticity in information security stands for verifying the identity of parties involved in the network. Authenticating identities means a receiving party can check whether the package is sent from a trusted party.

Use case 5: Unauthenticated party sends a message to a trusted party

This use case consists of sub-use cases in which the sending party can be the subscriber, publisher or controller, and the receiving party idem. If a controller receives events from unauthorised publishers or subscribers, it will not know them and will drop the event. The controller will only act if the receiver has done the authentication step beforehand. Due to the decoupled property between the publishers and subscribers, there are not able to authenticate each other. Authentication is the indirect task of the controller, and the forwarding rules its sent to the switches.

Publishers only receive a message from the controller as a reply to their advertisement message. In the authentication step with the controller, publishers know the MAC address of the controller, which they can trust. So initially, publishers can ignore messages from sources other than the controller.

Furthermore, a subscriber can receive a publication from an unauthorised publisher. However, this implies that there were rules installed on the switches that match this publication. Nevertheless, the unauthorised publisher has never communicated with the controller; thus, the switches never receive logical forwarding rules of publication from this publisher. The only possibility is that an unauthorised party guessed a random number as a pub-sub header. To prevent this guess, the bit size of the Pub/Sub header could be increased. Then the chance of correctly guessing is decreased.

7.2 Prototype simulation

In this section, a quantitative security analysis of the proposed solution in section 6 is performed by implementing a proof-of-concept. The performance of CP-ABE is dependent on the capacity of the component's hardware in the architecture, the desired security level, and the chosen expressiveness of the access policy and interests. Attributes greatly influence the computations; the encryption algorithm needs two exponentiation calculations for every attribute in an access policy. This proof-of-concept simulation of CP-ABE analyses the computation overhead of the communication between publisher and subscriber. Outside the scope of this implementation is the data plane implementation with routing efficiency. The controller logic is implemented and revised in terms of the extra computation overhead the cryptographic scheme introduces. First, the experimental setup is presented, and then the experimental results are shown.

7.2.1 Experimental setup

ABE based on the Bettencourt *et al.* [22] implementation is used for this thesis. This CP-ABE implementation is implemented in the Advanced Crypto Software collection. The cpabe toolkit provides a set of programs implementing a ciphertext-policy attribute-based encryption scheme [1] written in C++ code language. More conveniently, the Charm framework is used for implementing the ABE scheme in Python [2]. Internally, it runs C modules and existing libraries for mathematical operations, which is faster than any implementation in Python code. The C implementation benefits rapid prototyping with fast mathematical calculations, while the implementation can be done in Python. However, this framework does not allow for numerical expressions in the attributes [40]. Since this thesis wants to investigate the expressiveness of publications and subscriptions, this framework is not used. Instead, CP-ABE of Bethencourt *et al.* that uses bash commands is implemented with Python's subprocess function to allow for external program executions [3]. This CP-ABE toolkit mainly uses the GNU Multiple Precision arithmetic library (GMP), a cryptography toolkit. Moreover, the PBC library [4] is used in this toolkit to allow algebraic operations, e.g., element operations of finite groups. These operations are

[1]https://acsc.cs.utexas.edu/cpabe/

[2]https://www.cs.purdue.edu/homes/clg/files/Charm.pdf

[3]https://docs.python.org/3/library/subprocess.html

[4]https://crypto.stanford.edu/pbc/

needed in Bettencourt *et al.* CP-ABE scheme for the bilinear map. Integers up until 264 − 1 are supported. For the algorithm Encrypt() the access policy can have mathematical operators: $=, <, >, \leq, \geq$ for numerical attributes. For Keygen(), only the equal operator is implemented. Using one operator in the interest of a subscriber is less expressive than the original solution of this thesis.

To assess the performance of the proposed model, the computational overhead is evaluated based on benchmark experiments for the KeyGeneration, Encryption, and Decryption algorithms. The proof-of-concept included the full CP-ABE algorithm implementations and was therefore used for the simulations.

With the computation overhead results, a discussion about the efficiency goal 'communication overhead' can also be analysed. In combination with the communication use-case diagrams of the proposed solution, the overhead for the controller, subscriber and publisher can be highlighted individually. In section 8.1 Discussion, the usability of this overhead will be discussed.

The benchmarks are performed on a laptop with an Intel Core i5 (7th Gen) CPU and 16 GB Ram running Xubuntu 18.10 operating system. In table 3 the system specification is displayed.

System Configuration	**Specification**
OS	Xubuntu 18.10 x64
CPU	Intel core i5 (7th Gen)
Compiler	bash=4.4.20(1)-release
RAM	16
Language	C

Table 3: Local system specifications

Data creation using CP-ABE toolkit

A subscriber key is created using a set of attributes with attribute strings or numerical attribute values. Using the CP-ABE toolkit, it is important not to use the white spaces around the equal sign. The documentation specifies no whitespaces, but in practice, the decryption will not work. In addition, using (double) quotes in bash, it should be written as '\"input attribute"\', to make sure the single quotes are printed around one or more attribute parameters. Only then can it be parsed as a whole. For the python implementation, this is covered by the sub-process library.

A publisher with an access policy will encrypt a message. A subscriber can only decrypt this message if his interest matches the access policy of the encrypted message. A simple example is provided in Listing 2 on the usage of CP-ABE using the bash script for executing the algorithms KeyGen(), Encrypt() and Decrypt().

7.2.2 Experiment 1: Difference between OR and AND operators in policies

The first experiment will look at the trade-off between the expressiveness of the Boolean logic in the access policy and the execution time of the three cryptographic algorithms

```
cpabe-keygen -o priv_key pub_key master_key "attr1 = 2" "attr3 = 100"
cpabe-enc -k pub_key file1.txt  "attr1 = 2 or attr2 > 10"
cpabe-dec pub_key priv_key file1.txt.cpabe
```

Listing 2: Example bash code of CP-ABE toolkit

(encryption, decryption and key generation). The encryption and decryption will be done on an empty text file named file.txt containing no content. A script will run the KeyGen() at the controller, the publisher's Encryption() and the subscriber's Decryption(). A new attribute will be added to the subscriber's interest in a loop. When a new interest is made by adding an extra attribute, this attribute will also be added to the access policy, concatenating this attribute with an *OR* or *AND* operator. There is no overhead of end-to-end communication, only the components' runtime.

The setup contains one controller, one switch that connects 50 publishers to 50 subscribers. The experimental setup runs five times, and the average of these measurements will be compared. The Real time, User time , and system process times are outputted. Since the user time is the amount of CPU time spent in user mode and system time is the amount of CPU time spent in the kernel within the process, the sum will tell how much CPU time the process has used. This sum of the user time and system time for each run is used.

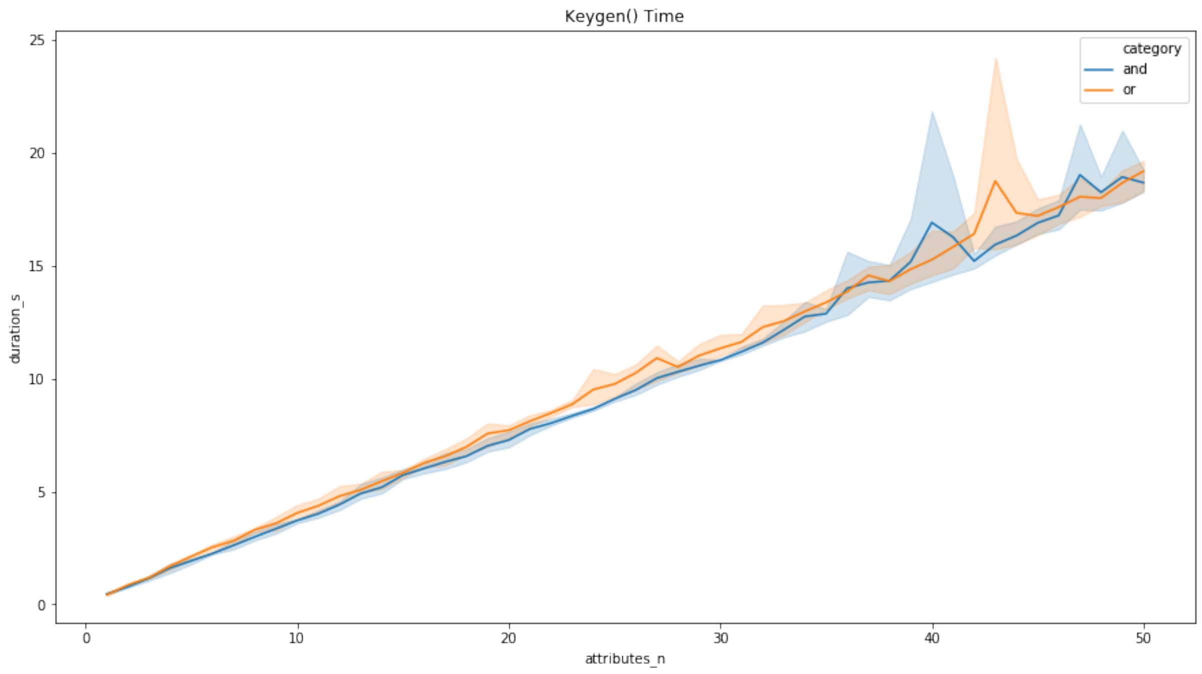


Figure 21: Time for Key Generation using exclusively OR in the access policy or only AND

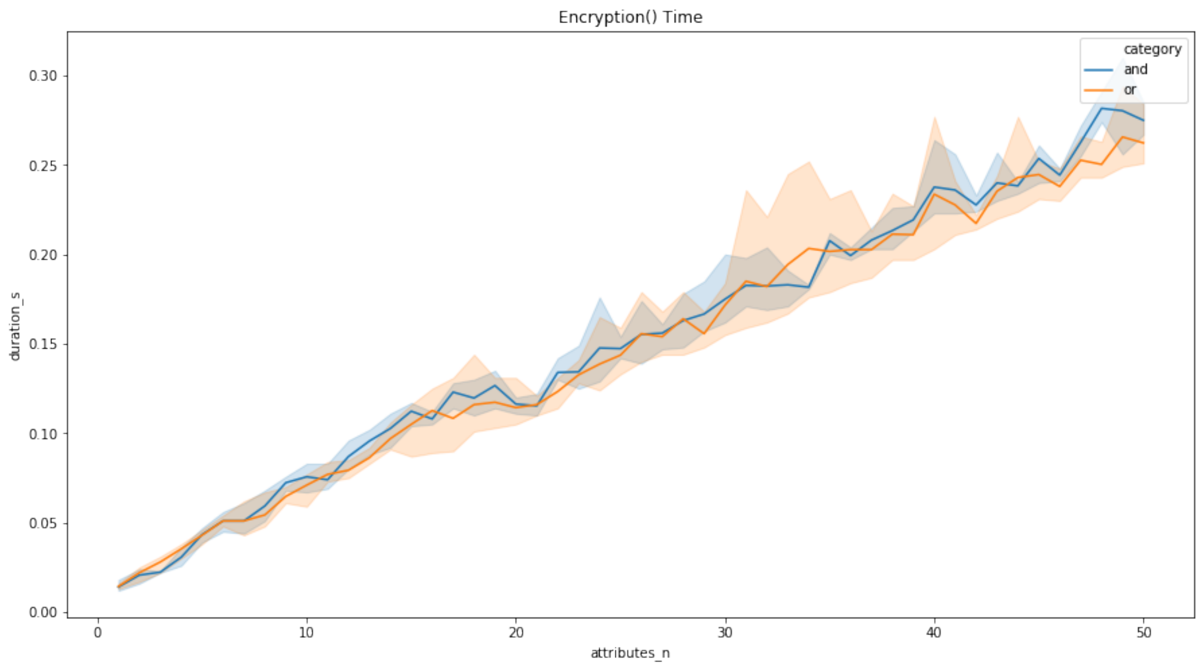


Figure 22: Time for Encryption using exclusively OR in the access policy or only AND

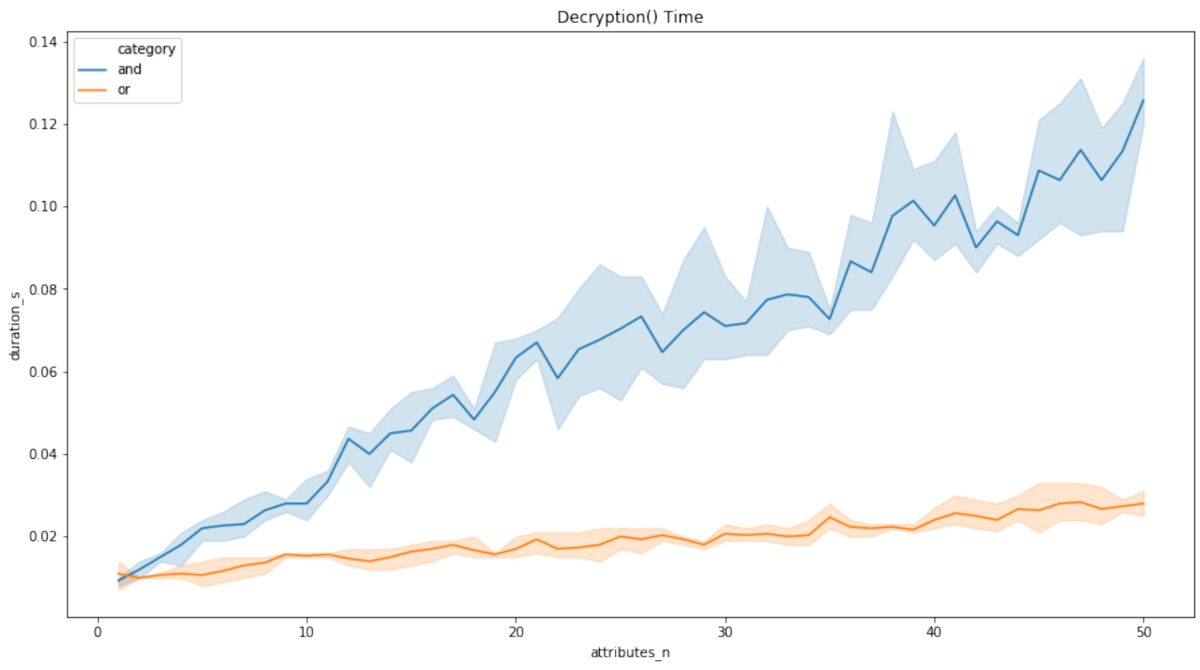


Figure 23: Time for Decryption using exclusively OR in the access policy or only AND

Figure 21 and 22 show that the difference between using AND or OR is insignificant. However, in Figure 23 the difference between using the AND or OR Boolean for the access policy of the publisher is bigger. Moreover, we see in all three algorithms that the number of attributes increases linearly to the execution time. Comparing the execution time of the algorithms, we see that the decrypt() algorithm has the shortest execution time, following Encrypt() and the KeyGen() algorithm. However, the KeyGen() algorithm has a significantly higher execution time compared to the Encrypt() and Decrypt() algorithms. This high execution time was not demonstrated by the original CP-ABE work [22].

7.2.3 Experiment 2: Comparison of numerical attributes

The second experiment will compare the execution time of the three cryptographic algorithms (encryption, decryption and key generation) using attribute strings or numerical attribute values. A mix of Boolean operators *OR* or *AND* expression is used to concatenate the attributes in the access policy. Using the CP-ABE toolkit, it is necessary to specify an exact length of k-bits for an integer. To ensure they all have the same possible bits, we choose 32 bits for every integer, which is the average possible length but still expressive enough to store our chosen attribute values. Bethencourt *et al.* highlights the possibility of using numerical attributes and the ability of the implementation to compare integers using a policy tree [22]. However, no results of this feature are shown, and no other work makes this comparison. The setup contains one controller, one switch that connects 100 publishers to 100 subscribers.

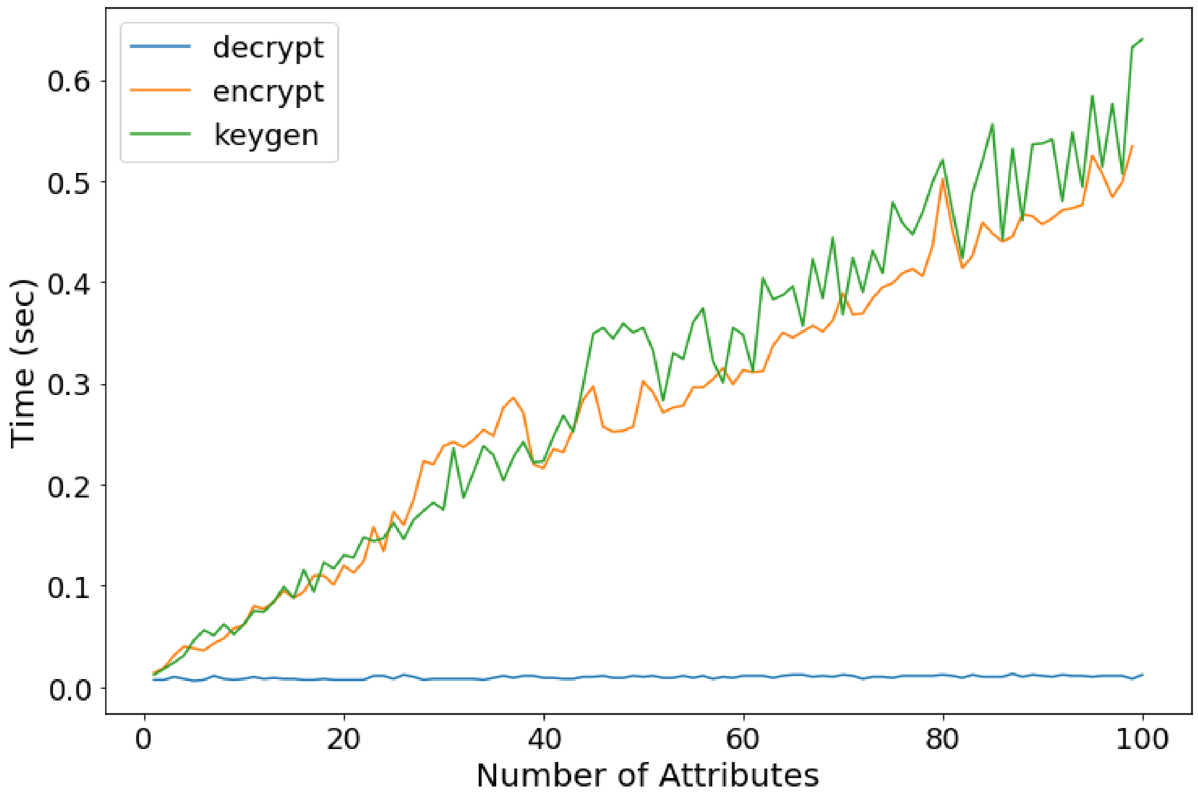


Figure 24: Execution time having only attribute strings in the interest and in the publication

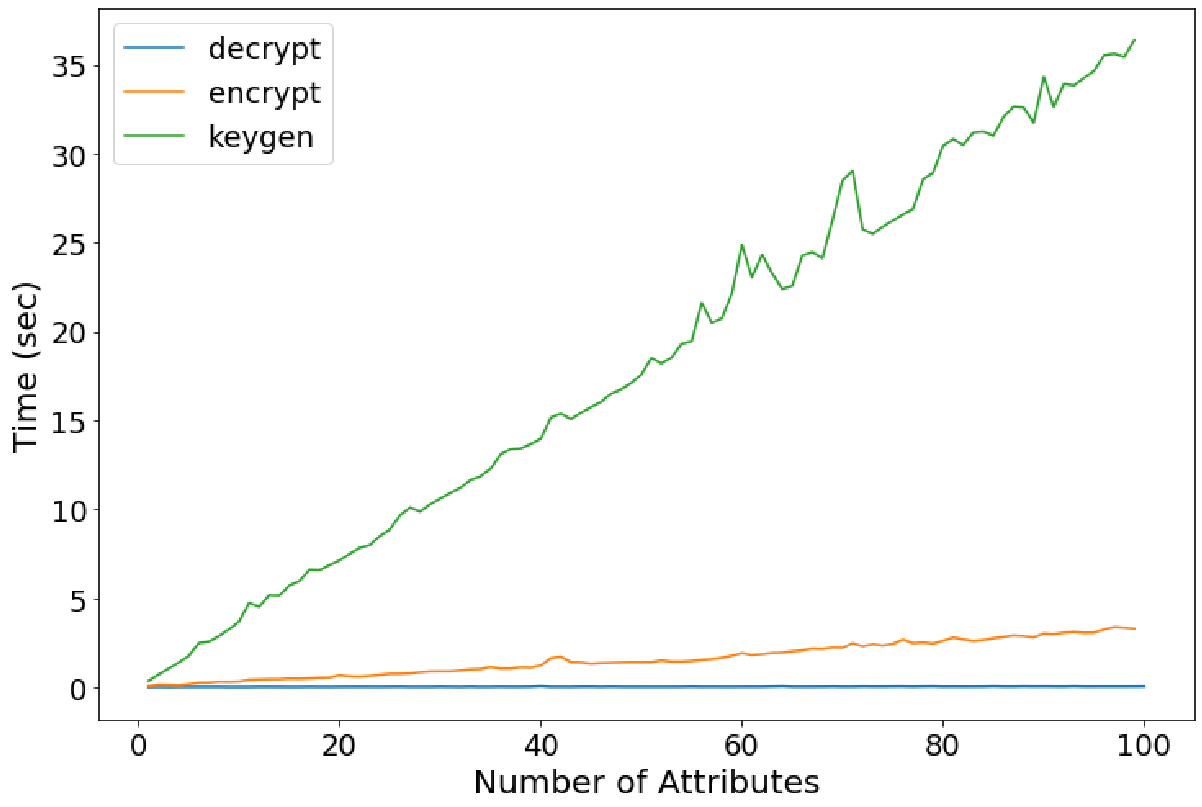


Figure 25: Execution time having attribute values in the interest and in the publication

Figure 24 shows the execution time having access policies and interest without numerical attributes. Figure 25 shows the execution time having access policies and interest with numerical attributes. The Figures demonstrate that the decrypt() algorithm creates the shortest execution time for both use cases. Figure 24 shows that Keygen() and Encrypt() algorithms grow linear and have the same significant execution time. However, this insignificant difference is not the same in Figure 25 in which the Keygen() algorithm results in a higher execution time.

Another finding was the key size of the keys generated by the KeyGen() algorithm. The key size goes up by approximately 24kb per attribute.

8 Discussion and Conclusion

8.1 Discussion

Compared to a traditional broker-based Pub/Sub middleware, the focus has shifted from middleware implementations to network layer implementations of Pub/Sub communication, such as SDN. Since SDN has a decoupled nature, the burden on the traditional broker is lessened and data delivery delays are reduced. A few designs using SDN with a Pub/Sub communication have been described; however, the security of content-based Pub/Sub and the impact on the decoupled property have not been assessed yet.

In this study, a secure Pub/Sub communication model is designed to adopt cryptographic algorithms used by the controller, publishers and subscribers. The design contains the model's architecture and the model routing logic.
A secure channel outside the SDN stands out when looking at the model's architecture. This extra channel is designed for authorisation between the publishers/subscribers and the controller. Communication outside the SDN is unavoidable for authorising identities of publishers and subscribers but potentially creates vulnerabilities.
When securing content-based Pub/Sub communication, fine-grained access control is necessary. Fortunately, CP-ABE ensures fine-grained access control over the message even when the content is encrypted. Another benefit of CP-ABE is that access control lies with the publisher. Therefore, no involvement of the key manager is needed for the encryption algorithm. Importantly, this meets the efficiency criterion to have as little controller communication as possible. Moreover, CP-ABE does not reveal someone's identity, which complements the decoupled property of Pub/Sub systems.
Unique to the routing of secure content-based communication in this model is the introduction of a novel packet header. As a result, no existing headers, e.g. IPv6 packet header, are used, which can be implemented for other purposes. Another beneficial design choice of the model is the exclusion of the controller communications for publications. In other words, the controller has communication overhead for every advertisement and subscription events but not for publications events. This exclusion is beneficial for the scalability of the model.

A qualitative security analysis of the proposed model was performed, discussing the communication confidentiality, integrity and authentication by means of different attack use cases. Confidentiality is assured in the proposed system, and authentication is guaranteed by the controller logic of handling advertisements and subscriptions. However, the integrity of the event is vulnerable in case of a man-in-the-middle attack alterations in the payload will not be prevented or recognised at an early stage. Compromised switches forwarding events to unauthorised paths is a possible scenario. However, non-compromised switches will drop packets that do not match any rule, so the system is still secure.
A proof-of-concept of the proposed solution is realised which allowed analysing computation overhead. No difference in computation overhead is measured for the Boolean operators *OR* or *AND*. Therefore, when creating access policies, no limitations should be considered between the use of operators *OR* or *AND*.
Figures 21 and 25 demonstrate that numerical attribute comparison significantly increases the execution time of the KeyGen() algorithm. Fortunately, the controller performs Key-

Gen () in the proposed solution, since the controller has unrestrained resources. However, it shows a direct trade-off between the expressiveness of the interest versus the computation overhead. Even though the authors of the KeyGen() algorithm describe that the compilation and comparison of numerical attributes are handled, the effect on execution time has not been shown [22]. In future implementations, more CP-ABE schemes that handle numerical attributes should be compared, to see whether this increased execution time is a general problem.

In line with previous work [22], Keygen() (without numerical attributes) and Encrypt() algorithms demonstrate linear execution times when the number of attributes increases. This performance is quite acceptable as computational overhead. The decrypt algorithm has the lowest execution time compared to the Keygen() and Encrypt() algorithms. This result is favourable in the designed model, since a subscriber has to perform a decryption algorithm for every incoming event.

In this model, the number of attributes influences the size of the decryption key in a linear fashion, as was designed by Bethencourt *et al.* [22]. However, in other existing CP-ABE schemes, the key size remains constant [41]. Choosing an optimal CP-ABE scheme depends on the use case of the system model.

One limitation of SDN is the need for a centralised controller; if the controller is compromised, the whole system is not secure anymore as the 'private' master key is public. Also, the availability of the controller can be affected by a DDOS attack. In that case, publishers and subscribers can not communicate with the controller and Pub/Sub communication is not possible.

A point of attention is the false positive rate of subscription clustering. Subscribers can receive events that do not match their interests if the clusters are not sufficiently fine-grained. A good clustering algorithm is hard to make, especially the expressiveness of the attribute makes this challenging. Since the controller is responsible for subscription clustering, subscriber's overhead depends on the implementation of the controller logic.

In addition, the false positive rate also influences subscriber's overhead by unnecessary decryption attempts. Due to the event's anonymity, the subscriber can not know whether it matches their interest before decryption. The privacy of the access policy is a burden when the subscriber owns many keys; it would try to decrypt the event with every key it possesses until one succeeds.

ABE is an expensive cryptography scheme in general. For this thesis, the overhead is analysed with the CP-ABE implementation of Bethencourt *et al.*, which is the first CP-ABE scheme but not the most efficient and fine-grained scheme currently available.

8.2 Future Work

Building on the results presented in this study, the prototype model of the proposed solution can be extended on different levels. Firstly, it would be interesting to investigate more ABE schemes for the proposed system model that generate less computational overhead.

Another idea to decrease computational overhead, is to allow a weaker notation of the full access policy security. Lai *et al.* [42] calls this partially hidden access policy, in which certain parts of the access policy are public.

In addition, it is worth examining whether external parties can do the most expensive cryptography calculations, and if this helps reduce overhead. For example, in the study of Touati *et al.*, [43], the implementation of CP-ABE on resource-constrained devices is feasible by outsourcing costly operations to a set of assistant nodes while keeping the data confidential.

The key refreshment remains a challenge in ABE schemes; implementing an expiry date as a numerical attribute could be investigated as a solution to key refreshment. Related to key refreshment, the key revocation requirement can be assessed. Implementing key revocation with a simple EPOCH value in the packet header would be possible.
In extension to the preliminary assessment of overhead computation in this thesis, data plane latency and overall end-to-end latency could be quantified. Computation overhead could likely be decreased by investigating different subscription clustering algorithms, for example by looking at false positive rates of subscribers receiving events.

To address hardware limitations of the data plane, flow-optimisation algorithms could be researched. Also, it would be interesting to quantify the limit of the amount of flow rules that can be installed for the Pub/Sub communication.

8.3 Conclusion

In this study, for the first time, a comprehensive model is proposed for content-based Pub/Sub communication on an SDN using P4 switches while implementing a CP-ABE security scheme. This model can provide a secure content-based Pub/Sub communication without weakening the decoupled property. Furthermore, it can route encrypted events from a publisher to a subscriber. One of the trade-offs of implementing a CP-ABE security scheme in this content-based Pub/Sub model is the possibility of compromising the integrity of the communication. In addition, expressiveness is expensive when looking at the computation overhead of the three cryptographic algorithms (encryption, decryption and key generation).

References

[1] Ali Malik, Benjamin Aziz, and Chih-Heng Ke. THRIFTY: Towards High Reduction In Flow Table memorY. In Edoardo Pirovano and Eva Graversen, editors, *2018 Imperial College Computing Student Workshop (ICCSW 2018)*, volume 66 of *OpenAccess Series in Informatics (OASIcs)*, pages 2:1–2:9, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[2] Ijaz Ahmad, Suneth Namal, Mika Ylianttila, and Andrei Gurtov. Security in software defined networks: A survey. *IEEE Communications Surveys Tutorials*, 17(4):2317–2346, 2015.

[3] Christian Wernecke, Helge Parzyjegla, and Gero Muhl. Implementing Content-based Publish/Subscribe on the Network Layer with P4. *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN 2020 - Proceedings*, pages 144–149, 2020.

[4] Muhammad Adnan Tariq, Boris Koldehofe, Sukanya Bhowmik, and Kurt Rothermel. Pleroma: A sdn-based high performance publish/subscribe middleware. In *Proceedings of the 15th International Middleware Conference*, Middleware '14, page 217–228, New York, NY, USA, 2014. Association for Computing Machinery.

[5] Yulong Shi, Jonathon Wong, Hans Arno Jacobsen, Yang Zhang, and Junliang Chen. Topic-Oriented Bucket-Based Fast Multicast Routing in SDN-Like Publish/Subscribe Middleware. *IEEE Access*, 8:89741–89756, 2020.

[6] Haiying Shen. *Content-Based Publish/Subscribe Systems*, pages 1333–1366. Springer US, Boston, MA, 2010.

[7] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19(3):332–383, aug 2001.

[8] Mudhakar Srivatsa, Ling Liu, and Arun Iyengar. EventGuard: A system architecture for securing publish-subscribe networks. *ACM Transactions on Computer Systems*, 29(4), 2011.

[9] Muhammad Adnan Tariq, Boris Koldehofe, and Kurt Rothermel. Efficient content-based routing with network topology inference. In *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems (DEBS)*, pages 51–62. ACM Press, 2013. DEBS '13 : The 7th ACM International Conference on Distributed Event-Based Systems ; Conference date: 29-06-2013 Through 03-07-2013.

[10] Boris Koldehofe, Frank Dürr, Muhammad Adnan Tariq, and Kurt Rothermel. The power of software-defined networking: Line-rate content-based routing using openflow. In *Proceedings of the 7th Workshop on Middleware for Next Generation Internet Computing*, MW4NG '12, New York, NY, USA, 2012. Association for Computing Machinery.

[11] Kaiwen Zhang and Hans-Arno Jacobsen. Sdn-like: The next generation of pub/sub. *CoRR*, abs/1308.0056, 2013.

[12] P. Anusree and Sreela Sreedhar. A security framework for brokerless publish subscribe system using identity based signcryption. In *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, pages 1–5, 2015.

[13] Maithily B and Swathi Y. Securing Broker-less Publish / Subscribe System using Fuzzy Identity-Based Encryption. 6(3):2823–2826, 2015.

[14] Muhammad Adnan Tariq, Boris Koldehofe, and Kurt Rothermel. Securing Broker-Less Publish / Subscribe Systems Using Identity-Based Encryption. 25(2):518–528, 2014.

[15] Sukanya Bhowmik, Muhammad Adnan Tariq, Boris Koldehofe, Frank Dürr, Thomas Kohler, and Kurt Rothermel. High performance publish/subscribe middleware in software-defined networks. *IEEE/ACM Transactions on Networking*, 25(3):1501–1516, 2017.

[16] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The road to sdn: An intellectual history of programmable networks. *Queue*, 11(12):20–40, December 2013.

[17] Fellow Ieee, Christian Esteve Rothenberg, Member Ieee, Siamak Azodolmolky, Senior Member Ieee, Steve Uhlig, and Member Ieee. Software-Defined Networking : A Comprehensive Survey. 103(1), 2015.

[18] Bochra Boughzala and Boris Koldehofe. Accelerating the performance of data analytics using network-centric processing. pages 192–195, 06 2021.

[19] Chenxi Wang, A. Carzaniga, D. Evans, and A.L. Wolf. Security issues and requirements for internet-scale publish-subscribe systems. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 3940–3947, 2002.

[20] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, pages 457–473, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[21] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, page 89–98, New York, NY, USA, 2006. Association for Computing Machinery.

[22] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-Policy Attribute-Based Encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, Berkeley, France, May 2007. IEEE.

[23] Mihaela Ion, Giovanni Russello, and Bruno Crispo. Providing confidentiality in content-based publish/subscribe systems. *SECRYPT 2010 - Proceedings of the International Conference on Security and Cryptography*, pages 287–292, 2010.

[24] Mihaela Ion, Giovanni Russello, and Bruno Crispo. Supporting publication and subscription confidentiality in pub/sub networks. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, 50 LNICST:272–289, 2010.

[25] Emanuel Onica, Pascal Felber, Hugues Mercier, and Etienne Rivière. Confidentiality-preserving publish/subscribe: A survey. *ACM Computing Surveys*, 49(2):1–41, 2016.

[26] Mohamed Nabeel, Stefan Appel, Elisa Bertino, and Alejandro Buchmann. Privacy preserving context aware publish subscribe systems. In *International Conference on Network and System Security*, pages 465–478. Springer, 2013.

[27] Weifeng Chen, Jianchun Jiang, and Nancy Skocik. On the privacy protection in publish/subscribe systems. *Proceedings - 2010 IEEE International Conference on Wireless Communications, Networking and Information Security, WCNIS 2010*, pages 597–601, 2010.

[28] Jun Li, Chenghuai Lu, and Weidong Shi. An efficient scheme for preserving confidentiality in content-based publish-subscribe systems. 02 2004.

[29] Costin Raiciu and David S. Rosenblum. Enabling confidentiality in Content-Based Publish/Subscribe infrastructures. *2006 Securecomm and Workshops*, 2006.

[30] Mihaela Ion, Giovanni Russello, and Bruno Crispo. Design and implementation of a confidentiality and access control solution for publish/subscribe systems. *Computer Networks*, 56(7):2014–2037, 2012.

[31] Changhoon Yoon, Taejune Park, Seungsoo Lee, Heedo Kang, Shin Seungwon, and Zonghua Zhang. Enabling security functions with sdn: A feasibility study. *Computer Networks*, 85, 05 2015.

[32] Markus Dahlmanns, Jan Pennekamp, Ina Berenice Fink, Bernd Schoolmann, Klaus Wehrle, and Martin Henze. *Transparent End-to-End Security for Publish/Subscribe Communication in Cyber-Physical Systems*, volume 1. Association for Computing Machinery, 2021.

[33] Misha Hungyo and Mayank Pandey. SDN based implementation of publish/subscribe paradigm using OpenFlow multicast. *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems, ANTS 2016*, 2017.

[34] Anton V. Uzunov. A survey of security solutions for distributed publish/subscribe systems. *Computers and Security*, 61:94–129, 2016.

[35] Qixu Wang, Dajiang Chen, Ning Zhang, Zhe Ding, and Zhiguang Qin. PCP: A Privacy-Preserving Content-Based Publish-Subscribe Scheme with Differential Privacy in Fog Computing. *IEEE Access*, 5:17962–17974, 2017.

[36] Christian Wernecke, Helge Parzyjegla, M Gero, Peter Danielis, and Dirk Timmermann. Realizing Content-Based Publish / Subscribe with P4. 2018.

[37] Lukasz Opyrchal and Atul Prakash. Secure distribution of events in Content-Based publish subscribe systems. In *10th USENIX Security Symposium (USENIX Security 01)*, Washington, D.C., August 2001. USENIX Association.

[38] Cristian Borcea, Arnab “Bobby” Deb Gupta, Yuriy Polyakov, Kurt Rohloff, and Gerard Ryan. PICADOR: End-to-end encrypted Publish–Subscribe information distribution with proxy re-encryption. *Future Generation Computer Systems*, 71:177–191, 2017.

[39] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. 1956.

[40] Sebastian Zickau, Dirk Thatmann, Artjom Butyrtschik, Iwailo Denisow, and Axel Küpper. Applied attribute-based encryption schemes. 03 2016.

[41] Keita Emura, Atsuko Miyaji, Akito Nomura, Kazumasa Omote, and Masakazu Soshi. A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In Feng Bao, Hui Li, and Guilin Wang, editors, *Information Security Practice and Experience*, pages 13–23, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[42] Junzuo Lai, Robert H. Deng, and Yingjiu Li. Expressive cp-abe with partially hidden access structures. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ASIACCS ’12, page 18–19, New York, NY, USA, 2012. Association for Computing Machinery.

[43] Lyes Touati, Yacine Challal, and Abdelmadjid Bouabdallah. C-cp-abe: Cooperative ciphertext policy attribute-based encryption for the internet of things. 06 2014.