Université de Neuchâtel
Faculté des Sciences
Institut d'Informatique

Efficient support for confidentiality-preserving publish/subscribe systems

par

Emanuel Onica

Thèse

présentée à la Faculté des Sciences
pour l'obtention du grade de Docteur ès Sciences

Acceptée sur proposition du jury:

Prof. Pascal Felber, Directeur de thèse
Université de Neuchâtel, Suisse

Dr. Hugues Mercier, Co-Directeur de thèse
Université de Neuchâtel, Suisse

Dr. Etienne Rivière, Co-Directeur de thèse
Université de Neuchâtel, Suisse

Dr. Christian Cachin
IBM Research Zürich, Suisse

Dr. Zbigniew Jerzak
SAP AG Dresden, Allemagne

Prof. André Schiper
EPF Lausanne, Suisse

Soutenue le 7 juillet 2014

UNIVERSITÉ DE
NEUCHÂTEL

Faculté des sciences
Secrétariat-décanat de Faculté
Rue Emile-Argand 11
2000 Neuchâtel - Suisse
Tél: + 41 (0)32 718 2100
E-mail: secretariat.sciences@unine.ch

IMPRIMATUR POUR THESE DE DOCTORAT

La Faculté des sciences de l'Université de Neuchâtel autorise l'impression de la présente thèse soutenue par

Monsieur Emanuel ONICA

Titre:

"Efficient support for confidentiality-preserving publish/subscribe systems"

sur le rapport des membres du jury composé comme suit:

- Prof. Pascal Felber, Université de Neuchâtel, directeur de thèse
- Dr Hugues Mercier, Université de Neuchâtel
- Dr Etienne Rivière, Université de Neuchâtel
- Prof. André Schiper, EPF Lausanne
- Dr Zbigniew Jerzak, SAP AG, Dresden, D
- Dr Christian Cachin, IBM Research, Zürich

Neuchâtel, le 10 juillet 2014

Le Doyen, Prof. P. Kropf

ABSTRACT

Publish/subscribe (pub/sub) is an attractive communication paradigm that offers efficient and decoupled information dissemination in distributed environments. Publishers generate the flow of information as publications, which are routed to subscribers based on their interests expressed as subscriptions. In the most common functional model, an infrastructure of brokers store the subscriptions, match incoming publications against stored subscriptions, and dispatch matching publications to the corresponding subscribers.

Early research on pub/sub mostly focused on improving performance, e.g., by maximizing the scalability of the pub/sub infrastructure and by minimizing dissemination latencies. The increase in popularity of pub/sub systems and externalized computing infrastructure lead to serious concerns about confidentiality preservation. Several techniques and mechanisms have been proposed to ensure confidentiality in pub/sub. However, these mechanisms come with performance costs. They also set new requirements that impede with the classical functional model of pub/sub systems. In this thesis, we present novel and innovative solutions to address these two aspects and make confidentiality-preserving pub/sub more practical and efficient.

Our first contribution is an overview of confidentiality-oriented research for pub/sub. We identify classes of solutions and highlight existing and future research directions. We observe the most important challenge for confidentiality-preserving pub/sub, which is to hide the content of publications and subscriptions from untrusted brokers, while allowing matching operations. Among the security models and solutions we identify in the existing work, encrypted matching schemes emerge as the most flexible solution.

Encrypted matching mechanisms allow untrusted brokers to match encrypted subscriptions against encrypted publications. However, these mechanisms have major performance overheads compared to non-encrypted matching. They may also prevent from using optimization techniques based on subscription containment. We propose a support mechanism that reduces the cost of encrypted matching, in the form of a prefiltering operator. This reduces the amount of encrypted subscriptions that must be matched against incoming encrypted publications. It leverages subscription containment information, but also ensures that containment confidentiality is preserved otherwise. We propose containment obfuscation techniques and provide a rigorous mathematical analysis to determine the amount of leaked information. We show that while there is a tradeoff between prefiltering efficiency and information leakage, it is practically possible to obtain good prefiltering performance in secure conditions.

Encrypted matching solutions require also appropriate key management support. Due to the use of encrypted subscriptions stored in untrusted domains, a key update may require all subscribers to re-encrypt and resubmit their subscriptions before publishers may use the new key. This is a costly and long operation. We introduce DynamiK, a lightweight key management architecture that takes into account the decoupled nature of pub/sub and allows updating encrypted subscriptions directly at the brokers. We present a security analysis and implementation of DynamiK for the ASPE encryption scheme, observing a minimal effect on the pub/sub service performance. We also extend the functionality and enhance the security of the original ASPE encrypted matching scheme, which we use for encrypted matching throughout our work.

Finally, we provide an overview of the current challenges implied by confidentiality preservation in content based pub/sub and discuss future research avenues.

Keywords: publish/subscribe, confidentiality, key management, encrypted processing

ACKNOWLEDGMENTS

I want to thank my advisor Professor Pascal Felber, as well as to my co-advisors Dr. Hugues Mercier and Dr. Etienne Rivière for their extremely valuable support during this thesis. I want to express my gratitude for their time availability and patience in long and fruitful discussions on the problems addressed. I was fortunate to receive exceptionally valuable input on both theoretical and practical grounds. Without their advice and feedback the results obtained in the presented work would not have been possible.

I want to thank the collective I worked with on the various implementations related to the contents in this thesis. I particularly want to express my appreciation for the advice received from Dr. Marcelo Pasin on the architecture of the library that concentrates most of the functionality described and evaluated in the presented work. I also want to mention Raphael Barazzutti, Stefan Weigert and Mascha Kurpicz with whom it was a pleasure to work on the practical side of the projects developed during my stay in Neuchâtel.

I have a lot of appreciation towards all my coleagues in the Institute of Computer Science, University of Neuchâtel for maintaining a great atmosphere, both at the office and in the activities organized outside work, as well as being very supportive whenever I needed their help.

I also want to thank some close friends, for their support, kind thoughts and encouragements throughout these years in moments when they where really needed. Credits go to Ciprian Amariei, Ancuța Rusu, Adrian Prodan and Anca Ozarchievici.

Finally, I want to address warm thanks to my family for their support during these years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Publish/subscribe systems

Publish/Subscribe (*pub/sub*) systems [Eugster et al., 2003] have been widely investigated in the past years as a convenient way to dynamically disseminate information in a distributed system from several sources (the *publishers*) to different subsets of interested users (the *subscribers*). Publishers produce data in the form of *publications*. Subscribers express their interests for receiving a subset of publications by issuing *subscriptions*. These subscriptions are composed of predicates, or constraints, that selectively filter the publications. Any publication *matching* a given subscription's constraints is delivered to the corresponding subscriber. The most common approach in pub/sub systems is to consider that the matching procedure is performed by a set of dedicated machines, the *brokers*. The brokers are organized in an overlay. They store the subscriptions received, filter incoming publications, and forward the matching ones towards the interested subscribers. Communications between publishers and subscribers are decoupled in time and space. Publishers do not need to know the identity of the interested subscribers and do not need to synchronize with them. The task of determining the subset of interested subscribers and routing the publications is the responsibility of the pub/sub system itself. A generic broker-based pub/sub system is illustrated in Figure 1.1.

Pub/sub paradigms

Pub/sub systems are typically classified according to the model they allow for subscriptions constraints. The two main models are *topic-based* and *content-based*.

In topic-based pub/sub, subscribers declare one or several topics of interest among a list of predefined topics. Topic-based pub/sub is equivalent to named (multicast) communication channels. A publication is tagged with a topic and propagated to all the subscribers that

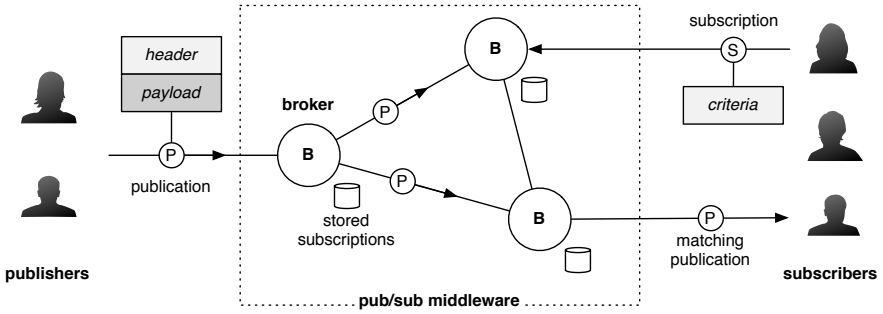

Figure 1.1: Generic broker-based pub/sub system.

registered a subscription on that topic. The main drawbacks of topic-based pub/sub are its low expressiveness and flexibility. If only a subset of the publications in one topic is of interest to a given subscriber, these must be filtered out by the subscriber itself. Examples of topic-based pub/sub systems include TIB/RendezVous [Oki et al., 1993], Scribe [Rowstron et al., 2001] or Bayeux [Zhuang et al., 2001].

In content-based pub/sub, publications are no longer classified according to predefined topics. The set of interested subscribers is determined at runtime based on the content of publications. The content is typically summarized or represented by a *header* that contains a collection of values over attributes. Subscriptions can filter publications of interest via a set of predicates, which are constraints over these attributes. The paradigm is strictly more expressive than topic-based pub/sub. A subscriber is no longer limited to a predefined topic and can combine different types of constraints on any of the attributes. Examples of content-based pub/sub include Elvin [Segall and Arnold, 1997, Segall et al., 2000], Gryphon [Strom et al., 1998, Astley et al., 2004], Hermes [Pietzuch and Bacon, 2002], JEDI [Cugola et al., 2001], PADRES [Li et al., 2005, Jacobsen et al., 2010], Rebecca [Mühl, 2001, Mühl, 2002], Siena [Carzaniga et al., 2001], X-NET [Chand and Felber, 2004], and StreamHub [Barazzutti et al., 2013].

The expressiveness and flexibility of content-based pub/sub has led to a broad range of applications. This includes the dissemination of stock quotes [Machanavajjhala et al., 2008], E-Health information systems [Ion et al., 2010a, Eze et al., 2010], network management systems [Martin-Flatin et al., 1999, Perera and Gannon, 2009], RSS feed monitoring [Rose et al., 2007], or algorithmic trading with complex event processing [Pietzuch et al., 2004, Adi et al., 2006]. Also, support for content-based pub/sub is present in industrial applications, like database management systems [Russell, 2002]. Therefore, considering the wider area of applicability, the contributions in this thesis are concentrated on the content-based pub/sub paradigm.

1.2 Motivation: the need for confidentiality

The focus of this thesis is on confidentiality-preserving pub/sub systems. In particular we develop new techniques that overcome drawbacks caused by confidentiality support. Pub/sub confidentiality is an aspect of great importance with the advent of externalized computing resources (*cloud computing*) or generally the trend to use pub/sub systems provided *as a service* by external parties. We provide two motivating examples that help identifying the context and need for confidentiality enforcement in pub/sub systems.

Stock exchange aplications. Our first motivating example is widely used in the literature (e.g., [Machanavajjhala et al., 2008]). It models a stock quote notification application. Publishers are stock markets issuing trading quotes. Subscribers are investors or other financial institutions wishing to receive quotes according to their various interests, such as all quotes above a certain volume of exchange, quotes with the highest daily variation, etc. A set of agencies provide pub/sub services by filtering quotes received from publishers according to constraints defined by subscribers. In this context, publications are public data, but subscriptions represent highly sensitive information. Indeed, leaking the subscriptions originating from a customer, Alice, would reveal intelligence about Alice's investment strategies, which can be used against her by competitors. We point out that encryption of subscriptions using standard techniques before submission to the broker is inadequate: the pub/sub system

must be able to route publications based on the constraints set in these subscriptions. If the constraints are encrypted such that they are completely opaque to the pub/sub system, routing is impossible and all filtering will have to be performed at Alice's side. Ensuring confidentiality in such a system is therefore a compromise between the ability to accurately route publications and the risk of leaking information.

E-Health applications. E-Health information systems [Ion et al., 2010a, Eze et al., 2010] are another application that can leverage content-based pub/sub as a communication layer. Such systems link actors of public and private health sectors (physicians, hospitals, clinics, pharmacists). These actors share files about patients to ensure a timely dissemination of cases, tests, etc. A typical publisher could be an emergency unit receiving persons in critical condition. In this case publications include the identity of the patients along with the content of their medical file, which are to be disseminated to various hospital units, possibly geographically separated and in independent administrative domains, where the patient can be moved when his condition stabilizes or where tests must be performed and analyzed. Subscriptions can be submitted to the pub/sub system by these particular healthcare units to take act of new cases, organize and schedule the patient admission and treatment sessions. While part of the publication (part of the medical file) can be encrypted using end-to-end encryption, some other parts must be used for routing the publication between authorized and interested parties. The publication headers (name, address of the patient, nature of the test, etc.) are highly sensitive information. Subscriptions are also highly sensitive information: they can reveal for example which patient is treated by which clinic or for which type of ailment. The leakage of such information can lead to serious consequences: it is not difficult to imagine an insurance company observing such data and refusing to cover patients undergoing certain tests. Again, we note that there is a compromise between the ability to route based on some information and the potential leakage of that information to unauthorized parties. The situation gets more complex if the e-Health infrastructure interconnects with other systems through pub/sub communication. We can imagine for instance a secured law enforcement platform maintained on a private cloud, which needs access to information on accidents and potential victims or suspects. We further develop this scenario in Section 1.3 when we discuss the details of the threat model typically considered in such situations.

A characteristic of both scenarios is that the pub/sub service provider can be a third party. It belongs to an administrative domain that is different from the ones of publishers and subscribers. It is therefore not necessarily trusted to access sensitive data. Note that even if the service provider complies to a confidentiality policy and is allowed to access unencrypted publication headers and subscriptions constraints, the resources onto which the service is operated should not present intrinsic risks of leaking this information. This cannot be always guaranteed, in particular when the pub/sub service is operated using public cloud infrastructures. The use of virtualization and the lack of control on resource placement and communication in public clouds may prevent users from considering the pub/sub service secure in terms of confidentiality. Serious concerns have been expressed in respect to this [Nanavati et al., 2014]. For instance, recent research [Ristenpart et al., 2009, Somorovsky et al., 2011] has shown that exploits at the hypervisor level, or exploits based on virtual machines colocation, can be used to gather private information from a virtual machine running on a public cloud. This indicates that confidentiality cannot be based solely on agreements between the publishers and subscribers, or between these and the pub/sub service provider. Since the support infrastructure may be prone to attacks, confidentiality

should be provided by design such that unauthorized access to the information manipulated by pub/sub brokers cannot cause critical data leaks.

1.3 Confidentiality in the context of pub/sub

Confidentiality is the property for a communication system to prevent the disclosure of sensitive information carried in the exchanged messages. Confidentiality in the context of content-based pub/sub systems is approached in several ways, which we overview in this section. This overview provides the background on which this thesis' main contributions integrate.

In Subsection 1.3.1, we describe a generic functional content-based pub/sub system model. This system model is widely used as unsecured baseline for existing pub/sub security solutions. In Subsection 1.3.2, we detail the importance of trust assumptions over this model and we distinguish two main directions taken in research towards providing confidentiality in pub/sub systems. In Subsection 1.3.3 we propose a classification of pub/sub confidentiality properties and in Subsection 1.3.4 we discuss the related issue of key management.

1.3.1 Functional system model

As introduced in Section 1.1, a typical content-based pub/sub system is composed of a broker infrastructure that provides routing services, and two sets of clients: *publishers* that submit *publications* in the system and *subscribers* that submit *subscriptions* with the intent to receive the publications that match their constraints.

In our model, the structure of a publication includes a *header* that defines the *attributes* on which routing is based and their respective values (e.g., price = 300, name = "`ACME`", date = 2013/1/1), as well as a *payload* that contains the complete data to be delivered (e.g., a graph showing the daily variations of the stock value). A complete set of attribute names and corresponding value types define the *attribute schema*. Note that in practice a publication can be represented only by its header, the payload being optional.

The structure of a subscription consists in a set of *constraints* on the attributes (e.g., price > 300 and name = "`EMCA`"). In our work we also commonly refer at publication attributes and subscription constraints as message *fields* when discussing in a general context that contains both subscriptions and publications (e.g., encryption of both types of messages).

In addition to matching publications against stored subscriptions, a desired functionality for pub/sub systems is the ability to determine *containment* relations between subscriptions. A subscription contains another subscription if it is more general, i.e., if publications matching the contained subscription always match the container subscription (e.g., "S1 : stockquote > 100" contains "S2 : stockquote > 300"). Determining containment relations between subscriptions facilitates building efficient data structures that can considerably speed up the matching operation [Carzaniga et al., 2001, Chand and Felber, 2004, Jacobsen et al., 2010, Li et al., 2005].

The majority of the work on providing confidentiality in content-based pub/sub systems follows this base model. There is little research touching confidentiality aspects on pub/sub systems that depart from this. For instance, the case of *peer-to-peer* pub/sub systems, in which the publishers and subscribers self-organize in a brokerless overlay in order to perform the routing, is seldom considered in the field's literature. In our work, we naturally

focus on the most often used model. However, in Chapter 2 we also overview confidentiality solutions examples of cases that depart from the described base architecture.

1.3.2 Threat model and directions of research

Almost all research on preserving confidentiality in a pub/sub system naturally defines a *threat model* for the system. This can vary from very simple cases when the pub/sub system architecture is minimalistic and only confidentiality threats are considered, to complex situations when the pub/sub system spans multiple administrative domains. In the latter case other threats, like availability related, are sometimes also taken into account. In this section we identify common traits and differences in the threat models we encounter. We separate two directions of existing research on confidentiality in pub/sub systems.

The widely considered malicious behavior for confidentiality threats in content-based pub/sub systems is *honest-but-curious* (or *malicious passive*). The entities (publishers, subscribers, brokers) that are considered *untrusted* to access the *sensitive* information act correctly according to the system functionality specification. In particular they do not replay or delete messages. However, at the same time they try to collect information that can be used to access the restricted contents of the exchanged messages, or equivalently provide the facilities to do so to external entities, unauthorized to use the pub/sub system.

We observe that most of the elements used to define a *threat model* are similar in the majority of existing research on pub/sub confidentiality:

- the functionality – routing publications according to submitted subscriptions;
- the entity roles – subscribers, publishers and brokers;
- the data – publications and subscriptions;
- the malicious behavior – honest-but-curious.

Threat models considered in various solutions typically differ in the trust assumptions. These trust assumptions separate the subjects in the pub/sub system in trusted and untrusted ones to access particular sets of fields in pub/sub messages.

Relations between different administrative domains generally dictate trust assumptions. As an example we further detail the e-Health use case described in Section 1.2. In Figure 1.2, we display a sample message flow and marking of domains where data needs protection against access from untrusted nodes. The hospitals, which publish a patient's file or subscribe to take act of new cases, are trusted to access the publications or subscriptions in their local domain (e.g., sending the file between the same hospital departments). However, the pub/sub broker service used by the countrywide public health system to disseminate information between different hospitals can be hosted on a public cloud domain, which is untrusted. The threat posed to the data confidentiality by the untrusted brokers prohibits sending messages in clear. This affects the possibility of matching and routing.

At the same time a police-owned routing infrastructure can connect to the e-Health pub/sub service for collecting information such as the name, age and type of injury of filtered subsets of victims injured in specific cases: traffic accidents, gun shots, etc. Unlike the healthcare service, which is externalized to a public cloud, the nature of the police pub/sub service requires a privately maintained infrastructure. Therefore, the brokers are trusted to access the publications containing patient data. The police pub/sub service can be used afterwards by individual departments to track records for particular cases (e.g., in Figure 1.2 John Smith was monitored for future contraventions after having his driver license suspended).

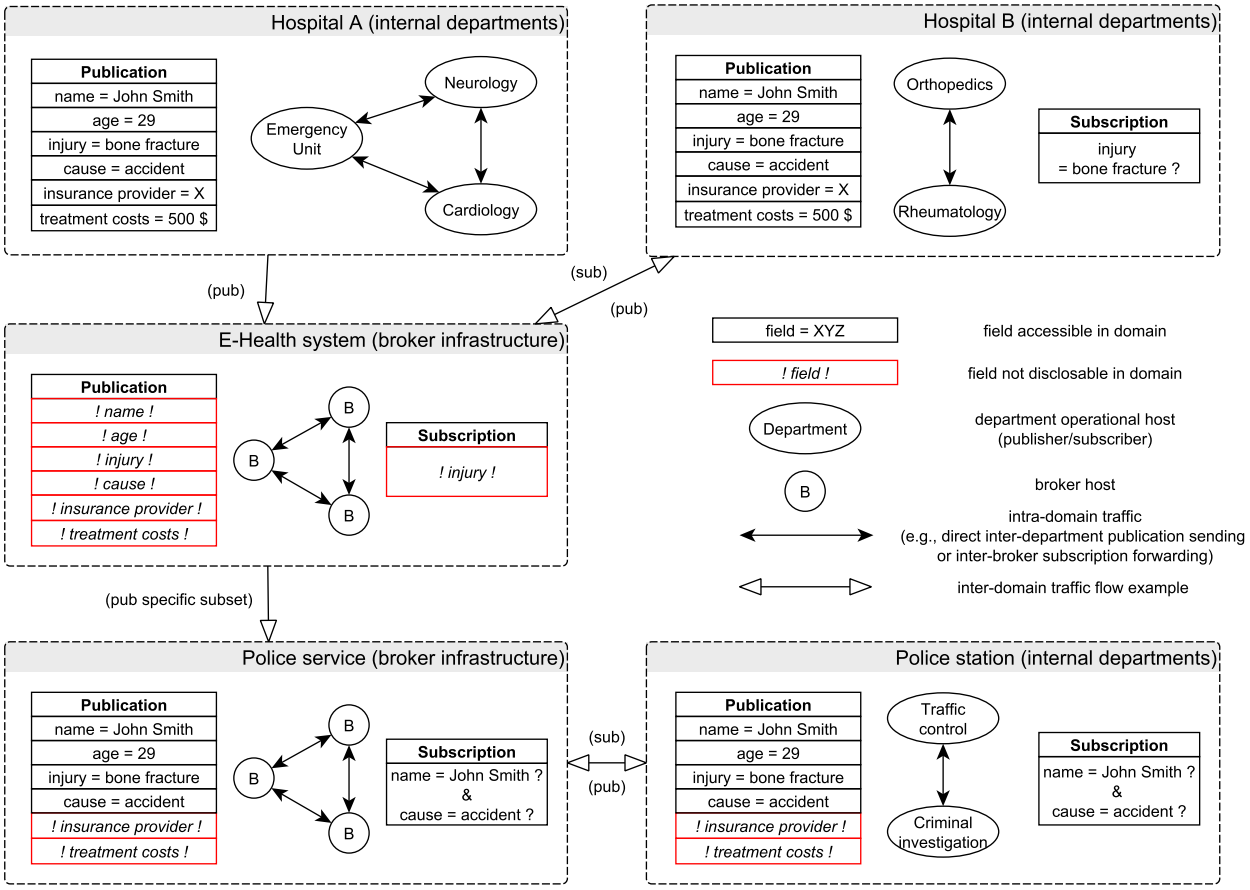

Figure 1.2: Example of message flow traversing multiple domains with different trust assumptions for accessing the data.

A confidentiality policy requires that extended information in the patient file like treatment costs and insurance provider to be kept private to the healthcare units, and not disclosed to the police unless mandated by legal actions. Such scenarios that imply interaction between multiple domains are often met in the literature (e.g., [Bacon et al., 2008, Bacon et al., 2010]).

From our example we can observe a bivalent relation between the pub/sub entities in a specific domain and the exchanged data. On one hand we have a functional requirement dictated by the role of the entity: brokers need to perform matching between subscriptions and publications in order to route, publishers and subscribers do not. On the other hand we have trust assumptions made about the visibility of pub/sub data that may vary per domain: entities may not be trusted to read any of the message content (e.g., the brokers in the countrywide health system infrastructure) but the same message can be partially accessed as it enters a different domain (e.g., the police broker infrastructure). Based on these observations we can split publications or subscriptions contents as following:

- *routable* and *non-routable* fields based on the functional necessity, in a specific domain, to use the header fields in routing the message or the absence of this requirement (the publication payload is naturally non-routable);
- *sensitive* and *non-sensitive* parts based on the need for protection against confidentiality threats in a domain, according to the threat model trust assumptions for the pub/sub entities (e.g., the age of the patient is a sensitive field in the health system infrastructure

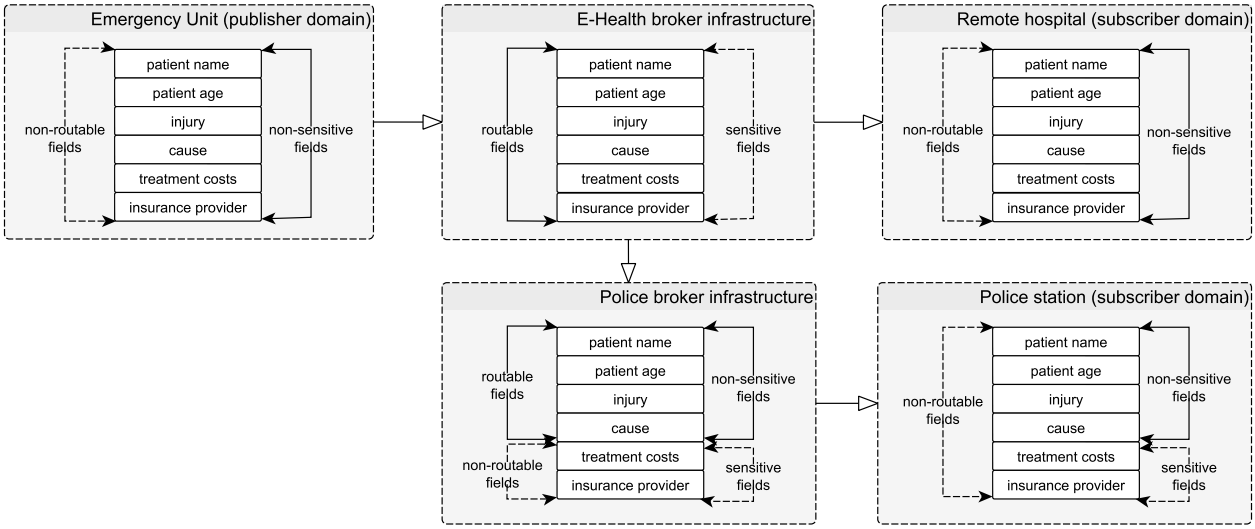

Figure 1.3: Example of publication field traits variation in respect to traversed domain.

where the brokers are not trusted to access it, but it is non-sensitive in the police broker infrastructure).

We illustrate in Figure 1.3 how publication field traits vary as the message passes through various domains, characterized by different levels of trust, in our e-Health example. Note that the purpose of this example is not to be exhaustive or optimal as an effective solution, but general enough to cover cases met in most of the existing research and altogether simple to understand. For instance the pub/sub entities in a domain have the same role, which is a simplifying assumption but not uncommon in the literature.

We observe two directions in the work that targets confidentiality in pub/sub systems, directions that tightly relate to the discussion in this subsection. We would like to emphasize that the two directions are not exclusively separated in the literature, although published work tends to focus on either one of them.

- **(A) - Research focused on encrypted matching.** In order to deal with confidentiality threats a form of encryption can be used for the *sensitive* data. This seems not feasible when the functionality requires the sensitive information to be also *routable* (e.g., in the e-Health broker infrastructure). This case implies the specific need for a scheme that permits *matching over encrypted data*, which we generically name *encrypted matching* throughout this thesis. There is an important amount of work concerned with devising such schemes. We overview the most significant examples in Section 2.1. Given the very specific case handled, the prevalent threat model considered is rather simplistic. Typically it is assumed that *all* brokers are untrusted and the *entire* content of pub/sub messages is sensitive to confidentiality threats while being routed in the broker infrastructure (e.g., considering exclusively the e-Health brokers case in Figure 1.2). The focus lies specifically on devising cryptographic mechanisms and on a much lesser extent on variations in the threat model or underlying trust assumptions.
- **(B) - Research focused on security models.** The work done in this area is geared towards finding an optimal architecture or the best security model for a pub/sub infrastructure that has to respond to confidentiality threats. The effective cryptographic mechanisms are typically of less concern. The threat model varies from case to case, with different trust assumptions set for the system components in regard to the data

accessed (e.g., the difference between the e-Health and police brokers in Figure 1.2). We can easily observe from our previous discussion, specifically on the message sensitivity, that this suggests the possibility to derive an access control matrix in which the pub/sub entities are the subjects, the pub/sub messages constitute the objects, and the trust assumptions could be actually formalized as access rights. Several articles in this area actually define an access control model for complex pub/sub systems. We review the area of work focused on security models in Section 2.2.

The work presented in this thesis aims at providing efficient support for existing solutions on pub/sub confidentiality, in that purpose extending them. We mostly focus on the encrypted matching schemes direction. The techniques we devise, presented throughout Chapters 3, 4 and 5 have a direct applicability in this area. However, part of our work can successfully integrate with security models covered in the second category above. We further detail our contribution in Section 1.4.

1.3.3 Flavors of pub/sub confidentiality

Confidentiality in pub/sub systems can be categorized in different flavors. We categorize it with respect to the sensitive content of the exchanged messages, that can be split in *subscription confidentiality*, *publication confidentiality* and *payload confidentiality*.

- *Subscription confidentiality* is defined as the property of hiding information in the subscriptions (constraints) submitted to the pub/sub system. When the threat model assumes untrusted brokers, this property breaks the functional system model. Brokers cannot perform the matching anymore over constraints supposed to be used in routing. In such cases, ensuring the property requires encryption mechanisms that make the matching action possible. However, encrypted matching support is not required for providing *subscription confidentiality* in cases where routing is not necessary. Any encryption scheme would suffice in such cases.
- *Publication confidentiality* is the property of hiding information in the header of a publication submitted to the pub/sub system. As in the case of subscriptions, enforcing this property may seem to break the functional system model, impairing the ability of untrusted brokers to match and route the publications. As mentioned above, encrypted matching algorithms are used to overcome this issue. Similary, when routing is not necessary, the scheme does not need to support matching over encrypted data. As an example this applies in domains where there are no brokers and only subscribers have restrictions in accessing the data.
- *Payload confidentiality* is the property of hiding the payload part of the publications submitted to the pub/sub system. Note that the payload is not used for routing purposes, only the header is. Therefore, enforcing this property is completely orthogonal to the specifics of pub/sub communications. It is feasible to use classical encryption techniques that apply also to other distributed systems.

To our knowledge, the most cited work that provides a particular separation of *confidentiality* properties in pub/sub systems is [Wang et al., 2002]. We have several differences in our classification. For instance *publication confidentiality* property is defined in [Wang et al., 2002] in two forms. "Information confidentiality" ensures the lack of disclosure against the broker infrastructure. "Publication confidentiality" ensures that only a restricted

Property	**Threat model assumptions**	**Scheme supporting encrypted matching**	**Classic encryption scheme**	**Categorization in** [Wang et al., 2002]
Subscription confidentiality	domain includes untrusted brokers	required for routable fields	for non-routable fields	Subscription confidentiality
Publication confidentiality	domain includes untrusted brokers	required for routable fields	used for non-routable fields	Information confidentiality
	only untrusted subscribers	possible but not necessary	typically used	Publication confidentiality
Payload confidentiality	any untrusted pub/sub parties	not applicable to payload	always used	payload separation not considered

Table 1.1: Overview of confidentiality properties.

set of subscribers are able to access the publications including their headers. We assume that publication confidentiality is supposed to be provisioned against *any* (possible different) untrusted system components. This makes the property independent of the threat model.

We consider confidentiality properties decoupled from the trust assumptions. However, a property can apply only in cases that require it (e.g., it does not make sense to enforce subscription confidentiality for a threat model that assumes only subscribers as not trusted, since subscriptions do not reach other subscribers domain except the one generating them). What depends on the trust assumptions, and also on the need to perform routing over encrypted data, is the mechanism used to enforce the property. Our property descriptions mention the need to use encrypted matching or not. Table 1.1 summarizes our properties classification.[1]

We observe that encrypted matching schemes are required for both subscription and publication confidentiality when the pub/sub system is deployed over a domain that includes untrusted brokers. This situation resembles closely the nowadays typical externalization of processing tasks on public clouds, bringing an additional argument for our thesis contributions focus on the encrypted matching area. Also, in particular, one of the support techniques we develop relates directly with preserving subscription confidentiality, as we will discuss in Chapter 3.

1.3.4 Key management in secured pub/sub systems

In section 1.3.2 we have diferentiated two main existing research directions: one targeted on innovating techniques of encryption that permits matching and another focused on developing complex security models in respect to threats considered in different interworking domains. We observe as a common point that for preserving one or more of the confidentiality properties a form of cryptography is used. Independent of the fact that the cryptographic solution is an encrypted matching scheme or not, it often requires the exchange of secret information or negotiating parameters between the participating pub/sub system entities. In other words the presence of a *key management* solution is necessary. In essence the functionality of a key management solution for secure communication in a generic distributed system case can be briefly summarized to:

- Allow the system components to obtain or negotiate in a secure manner the initial keys to start the secure communication;
- Deal with situations when the key information has to be refreshed at the different communicationg parties: when a member is evicted from the system and is not trusted

[1]In this table we consider only the sensitive pub/sub data. Cases for the messages or parts of the messages that are not sensitive do not make the subject of ensuring any confidentiality property.

anymore, when a member joins the system and is not trusted to decrypt previous communicated information, or cases dictated by the specific encryption scheme for other security reasons (e.g., counter brute force attacks, etc).

Pub/sub systems introduce an important additional challenge for key management. Unlike in point-to-point or named channels communication models, publishers are not aware in advance of subscribers that will eventually receive their message. A key management solution has to know who are the communicating parties that need to receive related key information. Therefore, it inherently restricts the natural pub/sub decoupled communication. Along Sections 2.1 and 2.2 we refer to existing work that offers a pub/sub-specific approach. As directions observed, we can identify the following:

- Most of work dealing with cryptographic schemes used in pub/sub systems considers key management as an orthogonal problem, decoupled from the cryptographic mechanisms that use the keys. These cases usually simply refer to the need of an external solution that has to ensure key provisioning to the communicating parties.
- A particular case of cryptographic schemes, relying on attribute based encryption, associates the encryption key information with the actual contents of the pub/sub messages, rather than with particular communicating parties. This reduces the coupling imposed by key management. We overview such schemes in Section 2.1.
- Finally, there are some solutions that modify the pub/sub functional model towards a less decoupled model, therefore allowing more straightforward designs for key management. We also cover some examples in Section 2.2.

A major contribution in our thesis is providing a new approach for key management in pub/sub systems. Our solution addresses both the pub/sub coupling problem, as well as a consequent issue brought by a key update, which is stored subscriptions invalidation. We thouroughly detail our work in Chapter 5.

1.4 Contributions and structure summary

This thesis is focused on techniques that tackle problems met in confidentiality-preserving pub/sub systems. A first major issue addressed are the performance drawbacks inflicted by secure mechanisms. Another critical aspect for which we also contribute is the key management aspect introduced in 1.3.4.

Our work is mostly applicable as support for optimizing or filling gaps in existing confidentiality-preserving pub/sub solutions. In particular, from the research directions described in section 1.3.2, we mostly target *encrypted matching* schemes. This is the newest and most flexible paradigm in secure pub/sub systems that has concentrated the interest in the latest years as we can observe from the published solutions we overview in our study. Also, it is the area which in our opinion is the most unexplored, and presents multiple open directions for future research. However, multiple designs for securing pub/sub can also benefit from our developments. In order to offer a thorough image of the existing confidentiality-preserving pub/sub solutions that might benefit of our developed techniques, in Chapter 2 we give an in-depth overview of the most important work that follows the two main directions of research: encrypted matching schemes and pub/sub security models.

In Chapter 3 we introduce our first technical contribution. We develop a generic technique that consistently alleviates performance drawbacks inflicted by encrypted matching

schemes in pub/sub communication. Our mechanism consists in a *prefiltering* stage that is executed by the brokers in a pub/sub system before the effective encrypted matching. This stage reduces the amount of publications passed through the actual encrypted matching filter by apriori discarding part of the non-matching cases. The normal overhead that encrypted matching schemes bring is substantially reduced. We perform a thorough analysis of our technique that evidentiates the existing trade-offs between performance and achievable security.

In our work we extensively use Asymmetric Scalar Product-preserving Encryption (ASPE). The scheme was introduced in the field of secure databases [Wong et al., 2009], being afterwards adapted for pub/sub in [Choi et al., 2010]. We develop and test our support techniques for secure pub/sub schemes using ASPE as use case scenario. In that purpose we extend the existing scheme, by adding significant changes to its original form. We also perform a thorough analysis on both the security and design of the scheme. We identify existing flaws and we fix these. We present a summary of the original ASPE scheme as part of the overview included in Chapter 2. Chapter 4 describes our primary functional extensions of ASPE: multidimensional matching and subscription containment, and presents our position on the possibility of adapting other existing schemes for encrypted matching purposes.

Chapter 5 introduces our third contribution. We introduce an architecture that deals with the major issue of key management in secure pub/sub systems. Our solution presents three major advances compared to the reduced number of existing alternatives. First, we tackle the inherent coupling problem that a key exchange protocol naturally presents, and which contradicts the pub/sub decoupled communication paradigm. Second, we are the first to deal with the significant problem posed by invalidating stored subscriptions. In a typical scenario, when a key is updated, all subscriptions stored by brokers have to be re-encrypted and resubmitted by subscribers. This can cause critical problems in the pub/sub quality of service. Thirdly, our solution is instantiated by extending the ASPE scheme, obtaining basically a novel enhanced version.

Finally, in Chapter 6 we conclude our work. We summarize our contributions and we identify open paths for further research.

Chapter 2

Overview of existing solutions providing pub/sub confidentiality

2.1 Encrypted matching and pub/sub confidentiality

In this chapter we cover the most representative solutions that follow the two directions of research on pub/sub confidentiality identified in Section 1.3.2. In this section we overview encrypted matching, and in Section 2.2 we cover security models.

Our work is mainly centered on providing support techniques that directly target encrypted matching solutions in this area. Such schemes are needed to encrypt sensitive fields in pub/sub messages when the threat model assumes the brokers untrusted to access these fields, but the functional model still requires the fields to be routable and used in matching. This situation occurs in both the e-Health and stock exchange scenarios previously described in Chapter 1. Encrypted matching is a paradigm that differs significantly from classic encryption algorithms. To the best of our knowledge, no comprehensive analysis and review exists in the literature. Therefore, prior to summarizing details of the most representative solutions published in the area, we consider useful centralizing some general characteristics that apply to encrypted matching schemes.

2.1.1 General considerations on encrypted matching schemes

Main class of cryptographic algorithm

A broad classification for cryptographic schemes splits them based on the algorithm class they fit into: *secret-key encryption* and *public-key encryption*. In secret-key encryption algorithms (also known as symmetric-key) the same private key information is used in both encryption and decryption. In public-key encryption algorithms (asymmetric) a public key is used in encryption and a different private one in decryption.

One important characteristic for encrypted matching algorithms used in pub/sub systems is that the final target is the matching operation. The sensitive data these schemes protect does not necessarily need to be decrypted. Encrypted matching schemes are normally used to encrypt subscriptions and publication headers. The subscription flow ends at the brokers, which store the messages without decrypting. The publication flow ends at the subscribers, which are normally interested in the publication payload and not in the encrypted headers (if the headers are of interest, these can be integrated also in the payload).

We observe that many existing encrypted matching solutions rely on, or adapt, an existing cryptographic scheme (e.g., already used in a normal point-to-point communication

or in secure database scenarios). The classic definitions of both private and public key schemes imply though the existence of a decryption algorithm. The lack of need for such an algorithm is a significant difference that would make improper categorizing an encrypted matching scheme as a private or public-key one. Moreover, in case of solutions inspired from established public-key schemes, the encryption phase always uses a private key component. In some cases this takes the form of an additional secret-key blinding layer added on top of the public-key scheme. Other designs use the algorithm in a somehow similar manner with the classic digital signatures case: the private key is used in encryption and the public key in the matching process. Even if it is cumbersome to define a distinct categorization for encrypted matching schemes, we can refer to one as *based on* a secret-key or public-key algorithm.

Types of constraints handled

A subscription's constraints can be classified based on two criteria:

- *type of field* - numerical (including both integer and floating point representations) or character strings
- *type of operator* - equality or range for numerical types, and identity, prefix, suffix and substring for strings

Most of the existing pub/sub encrypted matching schemes handle only numerical values. Even more restrictive, some of them do not support evaluating ranges over encrypted data. This can be acceptable for topic-based scenarios, where topic matching could be expressed by evaluating numerical equalities. However, in case of content-based pub/sub systems this severely limits the subscription expressiveness. Also, filtering optimizations based on subscription containment are limited only to equality based relations. Equality constraints on numerical data are still the predominant type, being available in virtually all encrypted matching schemes as the most simple constraint type.

Matching algorithm technique

We observe distinct manners to obtain the final encrypted matching result in the existing schemes:

- Based on a *exact relation preserving isomorphism*. Let us consider a function $\Diamond$ applied on a publication attribute a and a constraint value c in a subscription, such that a matching relation between a and c can be determined. For instance the difference between values: $\Diamond(a, c) = a - c$, can determine a "greater than" relation and whether or not the attribute matches the constraint. Let us then consider the encryption algorithm E and another function $\Box$ applied on the ciphertexts $E(a)$ and $E(c)$. If a correspondence can be established between the results of $\Diamond$ and $\Box$ then based on the result of $\Box$ we can determine the exact relation and matching between a and c. In different words the encryption E gives an isomorphism such that:

$$\Diamond(a, c) \Leftrightarrow \Box(E(a), E(c))$$

- Based on a *pre-mapped equality comparison.* In these solutions it is typical to first perform a pre-mapping of a publication attribute a and a constraint c in a different domain, which permits to evaluate the matching based strictly on equality comparisons. One basic example is to consider the bit prefixes that correspond to c and split a in a similar set of bit prefixes. Then if one prefix in the a associated set is equal to one corresponding to c the matching can be evaluated as positive. An encryption scheme E is applied afterwards over the pre-mapping results, such that the semantical equality can still be evaluated over the resulting ciphertexts.

It can be argued that the latter case is a particularization of the first, when the relation preserved by the encryption is semantical equality. However, there are cryptographic schemes which permit evaluating equality *only* and can follow exclusively this particular method, requiring the preliminary mapping phase.

Our separation is meant to emphasize different generic design patterns that can be applied when devising an encrypted matching scheme. In practice, two well-established classes of encryption schemes typically follow one of these directions when they are adapted for pub/sub encrypted matching.

- *Homomorphic encryption.* Such schemes have the property that, given two plaintext terms t_1 and t_2 and an operation α applied over these, encrypting the result of α corresponds to the result of an operation β applied over the individual encryptions of t_1 and t_2:

$$E(\alpha(t_1, t_2)) = \beta(E(t_1), E(t_2))$$

  Let us assume that α can be used to determine matching between an attribute and a constraint in a pub/sub context. Then, we can easily observe a correlation between α and β and the functions $\Diamond$ and respectively $\Box$ in the *exact relation preserving isomorphism* generic design. The difference here is that the α result is encrypted. Therefore giving it to a broker is not enough to determine a relation between the terms (unlike the case of $\Diamond$), unless the possibility of decryption is also provided. However, an untrusted broker cannot be allowed to decrypt encrypted messages. To overcome this issue, a classic homomorphic scheme normally needs to be adapted for an encrypted matching scenario. Examples of solutions overviewed we overview in Section 2.1.2 are [Nabeel et al., 2012, Nabeel et al., 2009].
- *Attribute based encryption (ABE).* The schemes in this category permit decryption only if a policy is satisfied by a set of defined attributes. The typical idea in a pub/sub scenario is to model the publication header as the attributes, and the subscription constraints as a"key" access structure (the policy). However, in ABE, the attributes or at least the access structure typically remain in cleartext. The target of the scheme is to encrypt a payload associated with the attributes that can be decrypted only when these match the access structure. This applies for *payload confidentiality.* To hide also the attributes (publication headers) and the access structure (subscription constraints), and to peform encrypted matching over them, typically an extra encryption layer must be added. To support both ABE and this extra layer, publication headers and subscription constraints are modeled into the ABE attributes and access structure, process which often follows the *pre-mapped equality comparison* design. For instance, the previous given example of pre-mapping to bit prefixes is used in conjunction with ABE in [Ion et al., 2012, Ion et al., 2010b]. Other examples that we overview in Section 2.1.2 are [Tariq et al., 2010, Srivatsa and Liu, 2007].

Note that the solutions based on homomorphic encryption or based on ABE do not cover completely the more generic design paths of *exact relation preserving isomorphism* and respectively *pre-mapped equality comparison*. We point out the use of these design paths also in other schemes overviewed in section 2.1.2.

Matching performance

Encrypted matching operations typically imply evaluating a function over a pair formed by an encrypted constraint and an encrypted publication attribute. The schemes we overview in Section 2.1.2 differ greatly in terms of supported constraints and matching technique. Some use various flexible cryptographic mechanisms, which can be chosen from a given class of algorithms (e.g., a commutative cryptosystem [Shikfa et al., 2008]). Therefore, comparing the performance of the various solutions is cumbersome. However, it is possible to observe the cases where the encrypted matching between a constraint and an individual attribute implies computation over a larger set of values than only the evaluated pair (e.g., the complete set of publication attributes). This typically implies a significant load, which can make a scheme less usable for some scenarios (e.g., workloads with numerous attributes per publication). We point out such situations in our overview. The *prefiltering* support technique that we develop in Chapter 3 is specifically designed to reduce such performance drawbacks, and make adequate to use any encrypted matching scheme, independent of its particular characteristics.

Cryptanalytic attack model considerations

A cryptographic scheme has to withstand a range of cryptanalysis attacks in which the purpose is to obtain the plaintext or the encryption key. There are four *basic* attack models considered when evaluating the security of classic cryptographic schemes, based on the attacker power:

- *Ciphertext-only attack (COA)* when the attacker can only observe the ciphertexts;
- *Known-plaintext attack (KPA)* when the attacker can observe the exact correspondence between ciphertexts and plaintexts;
- *Chosen-plaintext attack (CPA)* when the attacker has the power to obtain the ciphertexts corresponding to plaintexts of its choice;
- *Chosen-ciphertext attack (CCA)* when the attacker has the power to obtain the decryption of specific chosen ciphertexts.

Typically the analysis of security towards CPA or CCA is formalized through an indistinguishability game, where the attacker has to make a distinguishment decision over ciphertexts, with a non-neglictible probability. The schemes allowing encrypted matching in pub/sub scenarios present however, a different case from typical encrypted communication. [Raiciu and Rosenblum, 2006] expresses the idea that achievable indistinguishability is limited. By the possibility to match encrypted publications with encrypted subscriptions, an untrusted broker can draw conclusions about similarities between publications. Moreover, the schemes that allow to determine if one encrypted subcription contains (is more general) than another, inherently permit also distinguishing encrypted subscriptions. When functional requirements permit establishing such relations between encrypted subscriptions and encrypted publications, the security analysis of existing encrypted matching schemes does not usually

consider indistinguishability decisions at the level of complete encrypted messages. The assumed malicious behavior is honest-but-curious. The *chosen* attack models are rarely considered. The published work on encrypted matching most often either defers to the weaker KPA or COA or defines other more particular attack models, and respectively more particular security proofs.

The contributions in this thesis complement encrypted matching schemes. We focus in keeping the existing level of security, while adding functionalities that facilitate the practical deployment by increasing performance. The prefiltering solution we introduce in Chapter 3 takes in account the containment issue. We offer a thorough probabilistic analysis that proves interesting tradeoffs between performance and security. The key management solution we propose in Chapter 5 builds upon an existing encrypted matching scheme, enhances it, and proves its resilience to an attack equivalent to a KPA, more powerful than the one that was considered in the original scheme [Choi et al., 2010].

2.1.2 Encrypted matching solutions

In the following we overview existing content-based pub/sub encrypted matching schemes. In our discussion we generically use the notation $E(x)$ for a ciphertext obtained using an encryption algorithm E applied to the plaintext x, and $D(y)$ for a decryption algorithm D applied to a ciphertext y. If the context is not general and implies a particular key K, the notation used is $E_K(x)$, respectively $D_K(y)$. We give a summary of the surveyed work in this section in Table 2.1.

Solution	**Confidentiality pub/sub/payload**	**Containment support**	**Key management support**	**Encrypted matching technique design**
[Choi et al., 2010]	yes/yes/no	yes	external solution required	exact relation preserving isomorphism
[Raiciu and Rosenblum, 2006]	yes/no[1]/no (with false positives)	yes	external solution required	pre-mapped equality comparison
[Nabeel et al., 2009]	yes/yes/no	yes	direct between peers (coupled system)	exact relation preserving isomorphism (adapted homomorphic encryption)
[Ion et al., 2012]	yes/yes/yes	no	managed by central trusted authority	pre-mapped equality comparison (adapted attribute based encryption)
[Li et al., 2004]	yes/yes/no	no	external solution required	pre-mapped equality comparison
[Tariq et al., 2010]	yes/yes/yes (with false positives)	yes	managed by central trusted authority	pre-mapped equality comparison (adapted attribute based encryption)

Table 2.1: Overview of the encrypted matching solutions.

[Choi et al., 2010] describes a solution using *asymmetric scalar product-preserving encryption* (ASPE). ASPE was initially introduced by [Wong et al., 2009] in the field of database security. The solution is applied for any numerical constraints.

We use ASPE extensively in our work. We both consider it as an use case scenario to test the performance of the encrypted matching performance enhancing solution we introduce in Chapter 3, and also we directly modify it to provide a key management architecture in Chapter 5. In the current section we just cover the basics of the scheme, focused on

[1]Subscriptions are encrypted but the keyed procedure lacks randomness and the subscriptions are admitted as semantically unsecure by the authors. Nevertheless, the encoding is needed to provide matching with the encrypted publication.

the unidimensional case, as presented in [Choi et al., 2010]. In Chapter 4 we describe the multidimensional case, filling the gaps in the original form, and we also correct flaws in the containment support.

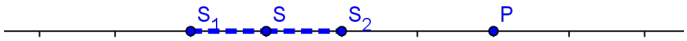

Figure 2.1: The ASPE subscription split.

Mechanism. Publication attributes and subscriptions constraints are represented as coordinates of multidimensional points. The idea is illustrated, without loss of generality, for the simple 1-dimension case in Figure 2.1. Consider a subscription point S and a publication point P that we are interested to match. The subscriber chooses two reference subscription points S_1 and S_2, placed symmetrically at equal distance from S. The *distance difference*: $D_1 = dist(S1, P) - dist(S2, P)$, can be compared to zero to determine the relation between S and P:

$$dist(S1, P) - dist(S2, P) > 0 \Rightarrow dist(S1, P) > dist(S2, P) \Rightarrow S < P$$

The reason to use the distance difference instead directly the distance between S and P is caused by a potential security issue. In [Wong et al., 2009], the authors prove that any encryption scheme preserving *distance recoverability* between two points is vulnerable to known-plaintext attacks (KPA).

The distance difference D_1 can be expressed as a sum of scalar products:

$$D_1 = dist(S_1, P) - dist(S_2, P) = \| S_1 \|^2 - \| S_2 \|^2 + 2(S_2 - S_1) \cdot P$$

ASPE offers a solution to encrypt points S_1, S_2 and P, such that the scalar product is preserved and the distance difference result can still be obtained from the encrypted data. The algorithm is based on the properties of matrix multiplication. It uses as keys on the subscriber side an invertible matrix M and on the publisher side the inverse M^{-1}. The matching phase relies on the natural reduction of the key when the ciphertext encrypted with M is multiplied with the one encrypted with M^{-1}. The obtained result is the same distance difference multiplied with a positive random value: $D_2 = D_1 * q$ $(q > 0)$. We can observe that the scheme follows the *exact relation preserving isomorphism* design defined in Section 2.1.1. The initial exact relation between points S and P can be determined based on the correspondence between results D_2 and D_1, obtained from the ciphertexts and respectively from the plaintexts. Subscription containment is optionally supported by adding extra reference points to the split subscription, and following a similar distance comparison technique as for matching. We detail the accurate containment mechanism, including our corrections, in Chapter 4.

Security considerations. The security evaluation in [Choi et al., 2010] is rather shallow, basically considering only a particular attempt for a ciphertext-only attack (COA). To guarantee *subscription* and *publication confidentiality*, the paper relies on more conclusive security proofs given in the original database scenario [Wong et al., 2009], which handle the case of a KPA. However, to obtain security in such case [Wong et al., 2009] enhances the original scheme by introducing artificial dimensions and splitting the original ones. These additions are not discussed in [Choi et al., 2010], but they are potentially adaptable to the

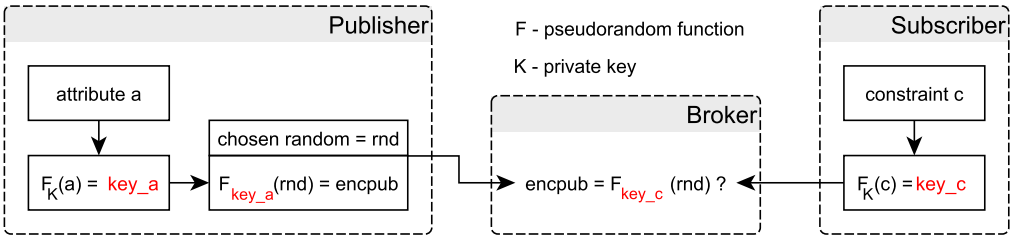

Figure 2.2: Equality encrypted matching building block in [Raiciu and Rosenblum, 2006].

pub/sub case. We use a different approach to enhance the scheme, and prove its resilience to a KPA as part of developing a key management architecture in Chapter 5.

Practical aspects. ASPE is a secret-key algorithm, which naturally needs that the key information be distributed among participants. However, the paper does not give any information about key management. As main contribution of this thesis, we develop a key management architecture, that we integrate with ASPE in Chapter 5.

Another limitation is the multidimensional case (multiple constraints and attributes), which is only briefly discussed. Our own analysis on this indicates that the encrypted matching depends on the size n of the schema defining the set of publication attributes, the complexity being O(n) per evaluated constraint. Therefore, the solution is less appropriate for workloads with a large number of attributes.

[Raiciu and Rosenblum, 2006] presents a set of encrypted matching schemes that use different encryption mechanisms according to the type of the values (integers, strings) and the nature of subscriptions constraints (equalities, ranges).

Mechanism. For equality filtering, the paper presents a simple scheme initially proposed by [Song et al., 2000]. This scheme is used as a base for the other cases, and is actually a building block also in other published work [Shikfa et al., 2008, Srivatsa and Liu, 2007]. The mechanism is depicted in Figure 2.2. Given a publication attribute a and a constraint c, after applying a K-keyed pseudorandom function F, the results $F_K(a)$ and respectively $F_K(c)$ are used as keys to encrypt a common random value (using the same function F). If the ciphertexts are similar then the matching can be considered positive.

The authors use this simple scheme as building block for more complex cases. One of them deals with string matching. String matching refers here strictly to constraints of keyword inclusion (e.g., a constraint could be: "includes ACME", where "ACME" is the keyword). The algorithm considers a dictionary with possible words. First, the dictionary indexes of constraint and attribute keywords are given to a keyed pseudorandom permutation. The obtained permuted indexes are used essentially as a and c in the first scheme to form keyes that encrypt a random value. However, the scheme requires for publications to encode every dictionary index in this manner, and not only the attribute indexes. The values resulted exclusively from attribute indexes are XORed with 1, to differentiate from the rest. When matching a constraint resulted value (also using a XOR), the result is positive if the same 1 is obtained.

The other flavors of encrypted matching are similary built, by defining a dictionary fit to the characteristics of the constraints evaluated. For instance, for numerical inequalities, a set of reference points associated with a smaller or greater comparator are chosen as the dictionary (e.g., $\{<1, <3, <5, >1, >3, >5\}$). In such cases, an initial approximation of

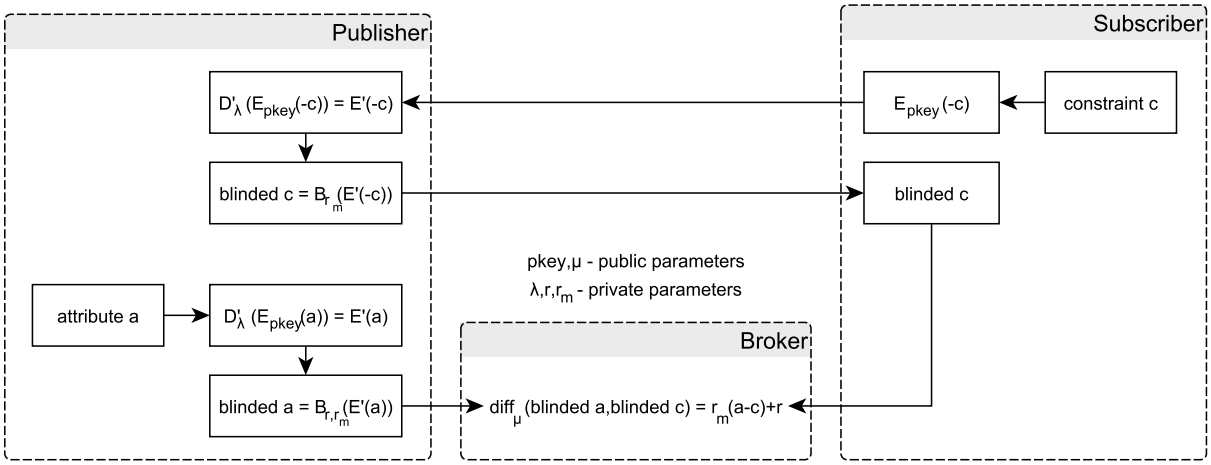

Figure 2.3: Encrypted matching mechanism in [Nabeel et al., 2012].

the values is required before encryption to pick the closest reference point in the dictionary. Basically, the scheme follows a *pre-mapped equality comparison* design as defined in Section 2.1.1. Subscription containment is determined in a relatively similar manner. This implies adding an additional reference vector with approximation points for each constraint.

Security considerations. The article points out the indistinguishability limitation in encrypted matching schemes. In case of subscriptions, this comes naturally as effect of the distinguishability resulted from containment. The authors admit that in such case the subscription encryption is not semantically secure (although containment support is optional). Therefore, they focus on *publication confidentiality*. In particular we can notice that only the publication encryption includes a random factor. The particular attack model considered is focused on the broker's ability to distinguish between publications. For this, the security evaluation ensures that the only information leaked is the positive or negative matching result.

Practical aspects. All schemes are based on secret-key encryption algorithms. The article does not propose any key management solution to distribute the needed information between peers.

The approximation used for numerical constraints generates false positives. Their rate increases as the granularity of the dictionary decreases. It is desirable, however, to have a small dictionary for performance purposes because the size of the encrypted messages depends directly on the dictionary size. Our scheme study revealed that the approximation phase can also produce false negatives in the containment tests. This does not have any implication in the matching of individual subscriptions, but it prevents some useful optimizations at the broker level.

[**Nabeel et al., 2012, Nabeel et al., 2009**] present a solution based on the public-key Paillier scheme [Paillier, 1999] applied for any constraints on numerical values. The Paillier cryptosystem benefits of homomorphic properties (e.g., $E(x) \cdot E(y) = E(x+y)$).

Mechanism. The pub/sub system architecture in the paper is adapted in order to support the scheme's specific design. It is assumed that subscribers know in advance the publishers from whom they wish to receive publications, and that the publishers are also aware of this intention.

A sketch summarizing the encrypted matching is shown in Figure 2.3. For the following discussion it is essential to mention that the original Paillier cryptosystem private key has

two components used in decryption: λ and μ. In the initial phase the subscriber encrypts the complement to each constraint value c using the public key *pkey* obtaining $E_{pkey}(-c)$. These values are sent to the publisher of interest. The publisher applies an additional layer of blinding, on top of the received Paillier encryption. The blinding scheme can be described as follows:

- pre-applying part of the Paillier decryption operation, which we note as D', using the secret λ parameter: $D'_\lambda(E(-c)) = E'(-c)$
- applying an additional encryption B, which makes use of a private random component r_m, and obtaining the blinded value: $B_{r_m}(E'(-c))$

The blinded encrypted constraints are sent back to the subscriber, which registers the encrypted subscriptions with the broker. The publisher follows a rather similar path for first encrypting and then blinding each attribute a, using the same separate private r_m, but also an extra random parameter r, finally obtaining $B_{r,r_m}(E'(a))$. For matching, the broker is given the second part of the Paillier private key: μ. Using this, and the homomorphic properties of the Paillier cryptosystem the broker is able to perform a computation *diff* over the blinded attribute and constraint. This computation completes the Paillier decryption, obtaining:

$$diff_{\,\mu}(B_{r,r_m}(E'(a)), B_{r_m}(E'(-c))) = r_m(a-c)+r$$

The randoms r_m and r are chosen in an interval that is limited according to the values domain, such that the *diff* result gives the relation between a and c. Optionally, containment is supported through a similar technique.

Security considerations. The scheme achieves *subscription confidentiality* and *publication confidentiality* according to the security assumptions it considers. However, the article does not define a specific attack model. The scheme's security analysis is merely based on the fact that the primitives used are secure on their own ground. Due to the change of the private μ parameter in the Paillier cryptosystem into a public one, we believe that a more extensive analysis of the implications would be useful, but this exceeds the limits of the current overview. Further, the r_m random parameter chosen when encrypting a publication cannot differ from the one used when encrypting the subcription that is supposed to be matched. This means that the publisher must reuse the same value of r_m when sending other publications. [Nabeel et al., 2009] acknowledges the possibility that eventually this could be brute forced and the actual differences $a-c$ could be determined, therefore the addition of the extra random parameter r done in [Nabeel et al., 2012]. Note that the difference $a-c$ would give the exact distance between the two values, which according to the proofs in [Wong et al., 2009] would make the scheme a *distance recoverable encryption*, which is unsecure to a known plaintext attack (KPA).

Practical aspects. In our opinion, a main drawback is the tight coupling considered in the design. The initial subscriber-publisher communication seems to be advantageous however, in regard to the issue of key management. Nevertheless, the model is not equivalent to an initial key exchange. Every subscription must be sent first to the publisher of interest before being registered to a broker. This will cause both a communication overhead and a processing overhead at the publisher side. We think that a basic optimization idea would be to delegate these operations to a trusted third party instead of a publisher.

[Ion et al., 2012, Ion et al., 2010b] present a scheme (also referred in [Ion et al., 2010a]) that is using *attribute based encryption* (ABE) and multi-user *searchable data encryption* (SDE). The scheme is applied for any numerical constraint and string equalities.

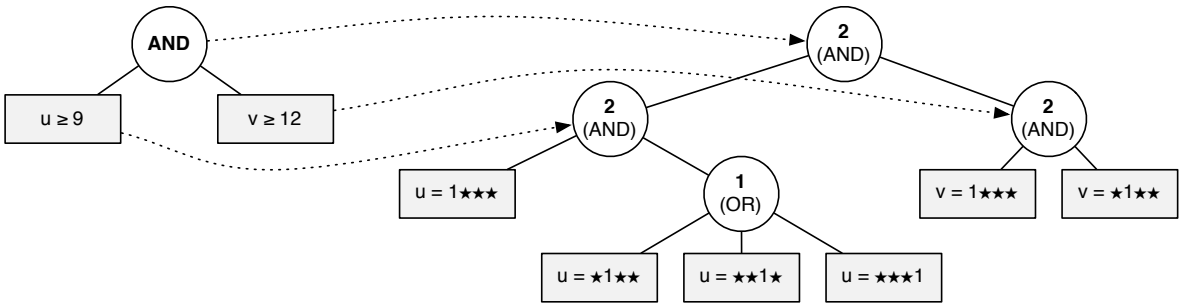

Figure 2.4: ABE resulted subscription representation (four bits).

Mechanism. We discussed the generalities of ABE in Section 2.1.1. In this scheme the main use of the paradigm is in encrypting the publication payload. The publications can be decrypted only if an access policy associated with the subscriber that receives them is satisfied. The access policy sets a specific structure for the subscription constraints. It also dictates the publication attributes format. The constraints are encoded into an access tree in which the non-leaf nodes are threshold gates specifying the number of subtrees that need to be satisfied. Consider the case of a subscription with two constraints depicted in Figure 2.4. This will result in a root node with a value of 2, meaning that each of the two subtrees needs to be satisfied (equivalent with a logical AND between the constraint subtrees). Further, a constraint is expanded following the representation in [Bethencourt et al., 2007]. Following, one leaf-node token per bit is required in order to take a decision. For instance, for a predicate like "$v \geq 12$" in a 4-bit representation, the constraint subtree will have a leaf-node with $v = \texttt{1***}$ ("bit 1 of v must be 1") and a leaf-node with $v = \texttt{*1**}$ ("bit 2 of v must be 1") joined by a parent "AND" node. For publications, the attributes are also split into 1-bit similar tokens (e.g., $v = 12$ will be split in $v = \texttt{1***}$, $v = \texttt{*1**}$, $v = \texttt{**0*}$, $v = \texttt{***0}$, which matches the example constraint subtree in the subscription representation).

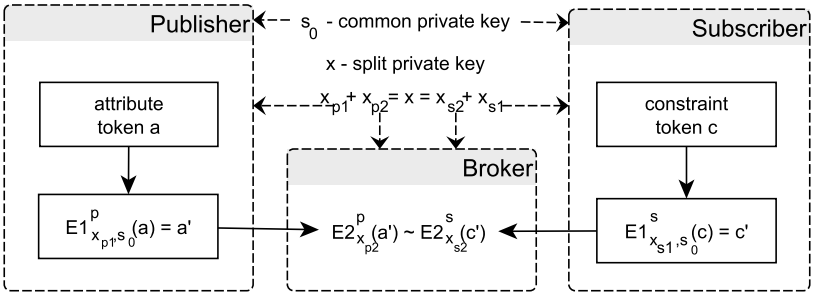

Figure 2.5: SDE private key usage applied in pub/sub context.

The 1-bit tokens obtained from publications and subscriptions are encrypted using multi-user SDE [Dong et al., 2008]. This technique adapts the public key El Gamal scheme [El Gamal, 1985] to a proxy encryption context. A trusted authority generates the public key parameters and a private key formed by two components x and s_0. The idea is to randomly split the private x in two pieces $x = x_{i1} + x_{i2}$ for each pair i formed by an end node (publisher or subscriber) and the edge broker through which the end node sends the messages. The sketch in Figure 2.5 summarizes the technique. The end node performs an initial encryption $E1$ in which his private part x_{i1} and s_0 are used. The broker performs a re-encryption $E2$ using his part of the private secret x_{i2}. The algorithms used for publications and subscriptions differ. However, it is provided that for any split configuration of the private parameter x, the obtained ciphertexts can be compared to determine the matching. The paper does not give

any information about performing containment evaluations. Due to different random values used by subscribers in the encryption process we conclude that the brokers cannot establish any relation between two encrypted constraints.

Security considerations. The security for providing *subscription confidentiality* is analysed using a chosen plaintext attack (CPA) model. In case of *publication confidentiality* the scheme is evaluated on a model which has similarities with the one in [Raiciu and Rosenblum, 2006] the goal being that nothing be leaked to an adversary besides the results observed from matching history traces. The analysis uses the fact that the employed ElGamal mechanism is proven CPA secure on its own ground. Finally, *payload confidentiality* provided by ABE is proven secure under a stronger threat model, which takes in account the possibility of collusion between malicious subscribers, publishers and brokers.

Practical aspects. The paper elaborates also on an access control model that can be applied in more complex scenarios such as the e-Health platform taken as use case. This relies on restrictions that can be imposed on various groups of subscribers through ABE. In that regard [Ion et al., 2012] is one of the articles which touches in sufficient extense both areas of research we distinguished in Section 1.3.2.

The article considers the lack of need for key exchange as a decoupling advantage. However, this does not eliminate the need for a common key material set at the communicating peers. The difference is that this key material is not exchanged between participants, but retrieved from a trusted authority.

The scheme is integrated with PADRES [Jacobsen et al., 2010] pub/sub middleware, for evaluation. [Ion et al., 2012] mentions that matching time increases linearly with the number of comparisons done between constraint and attribute 1-bit tokens. A constraint subtree can have at most n leaf-nodes, and a 1-bit attribute set has n elements where n is the number of bits used to represent a value. The worst case scenario is that each 1-bit attribute must be compared with each 1-bit leaf-node. The maximum number of comparisons per constraint is n^2.

[Li et al., 2004] presents a scheme that applies a technique based on prefix preservation to evaluate any numerical constraints. The idea has some conceptual elements in common with [Ion et al., 2012].

Mechanism. The scheme considers subscription constraints as always expressed through a closed interval (e.g., $x \in [32, 111]$). The principle is that the membership to the closed interval can be mapped into a finite set of prefixes (e.g., for the example above, in case of an 8-bit value representation: `001*`, `010*`, `0110*`). A publication attribute that matches the interval will naturally match one of the associated prefixes (e.g., $x = 64 =$ `01000000` matches `001*`).

The next step is to encrypt such sets obtained for interval constraints and as well the publication attributes values in a manner that preserves the significant prefix. The solution makes use of a simple pseudorandom permutation of the bits. The plaintext prefixes can be represented on a binary tree in which each node under the root corresponds to a bit as displayed in Figure 2.6. A permutation is obtained by flipping any sequence of bit values in the non-leaf nodes and switching the two child subtrees accordingly. Each of all possible prefix-preserving ciphertext domains is generated through such a bit flip permutation. The matching on the ciphertexts simply consists of comparing the prefixes as in the plaintext case. We observe that the technique fits in the *pre-mapped equality comparison* design. The paper does not specify anything about containment support.

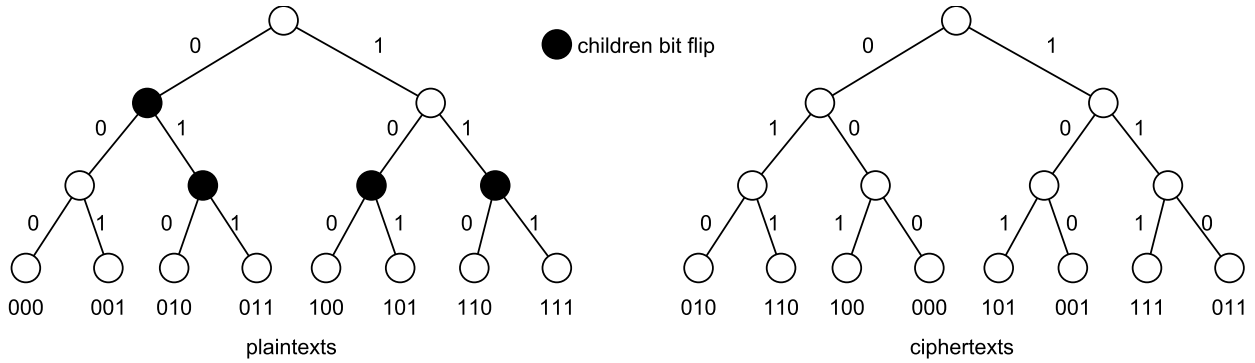

Figure 2.6: The bit-flip based prefix preserving permutation.

Security considerations. The authors admit that the scheme has limited resistance under a KPA attack model. *Subscription confidentiality* and *publication confidentiality* are therefore achievable only on a weaker ciphertext-only-attack (COA).

Practical aspects. The scheme requires a secret key to determine a common permutation at the publisher and subscriber sides. Key exchange is considered however as an orthogonal problem. Performance-wise, the authors prove that the maximum size of a subscription prefix set is $2(n-1)$ for any interval in which the bounds are represented on n bits. This is also the maximum number of comparisons needed to determine a match, which is lower than quadratic as in [Ion et al., 2012] solution that also uses bit prefixes. This limit is set by the number of ciphertexts resulted following publication encryption. In the current scheme there is one ciphertext per publication attribute, compared to one per attribute bit in [Ion et al., 2012]. However, the heavy security drawbacks limit severely the possible usage of this prefix preserving solution.

[Tariq et al., 2010] is one of the very few solutions considering confidentiality provisioning in a *peer-to-peer* pub/sub architecture. Therefore we find useful discussing it and pointing some general traits that can apply in other cases. The article elaborates a secure design using *attribute based encryption* (ABE), which potentially supports any numerical or string constraints.

Mechanism. The peers in the system have both publisher and subscriber roles. The subscription constraints and publication attributes are first mapped into bit strings using domain decomposition. The domain is decomposed by gradually spliting in half for the numerical case, as displayed in Figure 2.7, or considering prefix and suffix trees for strings. A containment relation can be established between the split subspaces based on their bit prefixes.

The bit strings obtained are formalized as *credentials* for a publication or subscription. Publishers and subscribers self-generate public keys for each credential, which are used in encryption. A publication is encrypted using every public key associated with the credentials of a publication. This enables decryption for every peer who subscribed on a matching criteria which covers the publication's credentials. Decryption is done with the private keys. A private key for a subscription credential is obtained from a centralized key server. The credentials themselves are not included in the encrypted publications. A subscriber tries to decrypt a ciphertext under its subscription credentials and knows if it was successful based on a hashed checksum. The ABE schemes algorithms used in the scheme are [Bethencourt et al., 2007] and [Goyal et al., 2006].

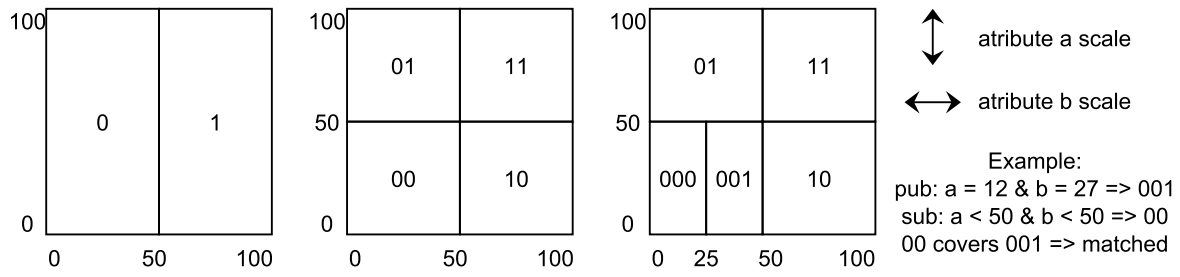

Figure 2.7: The domain decomposition mechanism.

Practical aspects. The fact the pub/sub system is formed from equivalent peers brings some significant changes from the broker-based model. First, the subscriptions are not effectively disseminated in the system. Consequently, the cryptographic scheme does not provide subscriptions encryption. Peers self-organize in a hierarchical overlay based on subscription containment when they connect to the pub/sub architecture. A peer has links with parent peers which have subscriptions containing the peer's own ones. The same peer has links with children that store subscriptions contained in the peer's own ones. To establish the position in this overlay a peer issues a *connection request*. The connection request message is similar to a publication encrypted under the credentials of the joining peer's subscription. As the usual publication case this connection request is also encrypted under all the other containing credentials. The request is forwarded and decrypted by peers having matching subscriptions in the overlay, gradually, until it successfully reaches the closest subscribers in terms of selectivity who will become parents of the joining node. An actual publication is essentially disseminated the same way. Since the publication routing is based on the containment relation derived from the domain decomposition, the solution admits false positives.

Security considerations. Encrypted matching does not require preserving *publication confidentiality* against the peer that performs the operation if the result is positive. If a peer matches a publication with one of its subscriptions, then the peer is also a destination for the publication. This is a fundamental difference from the broker based model where the encrypted matching operation is not supposed to disclose the publication content after a successful match. The solution defines a specific notion of *subscription confidentiality*. It is assumed that any leaks on information about subscription containment are acceptable between neighbor peers in the overlay, which comes as a consequence of a subscriber's connection request.

The first two of the following solutions permit exclusively matching equality constraints but present interest in regard to the key management technique. Finally, the last two use ABE to perform encrypted matching. However, these do not provide neither *publication confidentiality* nor *subscription confidentiality*, which makes them more suitable as building blocks in specific use cases. Therefore, we just briefly describe some of their specific mechanisms and functionality.

[**Srivatsa and Liu, 2005, Srivatsa and Liu, 2007, Srivatsa et al., 2011**] cover the development of a full-fledged pub/sub securing framework named *EventGuard*. [Srivatsa et al., 2011] details the most recent development on a set of functionality designs named *guards*, which provide an extensive set of security properties, but their functionality is limited to a topic-based scenario. However, the key management scheme used can be applied in a

content-based pub/sub architecture. This scheme is mainly discussed in [Srivatsa and Liu, 2007] for the case of numerical constraints (under the name of *PSGuard*) and extended in the later work.

An encryption key K_p is used to encrypt the publication payload. An authorization key K_s is associated with a subscription such that K_p can be derived from K_s by a subscriber only if the received publication matches. The mechanism is based on a key tree in which the root is a key associated with the entire domain range for an attribute. Each child of a tree node is a key associated with a partition of the parent's node domain and is obtained by hashing the parent key. A trusted centralized key distribution point is in charge with the keys dissemination in the system.

The idea resembles the usage of ABE as previously described in other schemes. The design has similarities with the one in [Tariq et al., 2010] (which references EventGuard). However, the schemes differ in the individual building blocks like the key generation procedures and encryption. Another design having some resemblance is the one in [Ion et al., 2010b]. Nevertheless, both the encryption techniques and the value mapping in 1-bit tokens are different. EventGuard is focused mostly on preserving *payload confidentiality*. With respect to *publication confidentiality* and *subscription confidentiality* the paper uses a tokenization technique similar to [Song et al., 2000] (previously overviewed in [Raiciu and Rosenblum, 2006]). This restricts encrypted matching to equality comparisons.

[Shikfa et al., 2008] presents a scheme based on multiple layer commutative encryption (MLCE). This implies using several layers of encryption, which can be applied in different order independently of the action performed, encryption or decryption, as permitted by the commutative property. For instance, for two keys, k_1 and k_2, and plaintext d, we have: $E_{k_2}(E_{k_1}(d)) = E_{k_1}(E_{k_2}(d))$. The scheme supports only equality comparisons.

The brokers are organized in a chain, each one sharing a different key with each other broker from a set of r ancestors and descendants. Assume for instance the chain of: $S{-}B_1{-}B_2{-}B_3{-}P$ with $r = 2$. B_1 will share k_{S,B_1} with his subscriber predecessor and k_{B_1,B_2}, k_{B_1,B_3} with his broker successors. When a message is received, a broker is able to remove the cryptographic layer given by the set of keys it shares, but not further. After doing this and performing the equality matching, the brokers change one layer of encryption from the packet. In the given example B_1 will receive from S a message d encrypted as $E_{k_{S,B_1},k_{S,B_2}}(d)$. It will remove the layer for which it knows the key: k_{S,B_1}, and will add a layer excluding the following broker: k_{B_1,B_3}, before routing the message to it. Using this technique the message "travels" through the broker overlay being covered at all moments by an encryption layer. This layer does not allow the brokers to access the plaintext. However, it permits to evaluate the equality matching. This design potentially fits any cryptographic algorithm that has commutative properties, the paper considering the application of the Pohlig-Hellman scheme [Pohlig and Hellman, 1978].

The scheme presents interest due to the simplified key management requirements directly related with the design used. A broker has basically the need to maintain shared secrets with the neighbors only up to degree r. One thing to note is that the scheme will work with an r as small as 2, but the value of r also reflects the minimum number of colluding neighbor nodes to obtain the plaintext. Therefore, this parameter should be set according to the compromise between the security requirements of the system and key management costs.

[Shi et al., 2007] is not specifically related to providing confidentiality in pub/sub infrastructures, but discusses a scheme closely resembling other ABE solutions [Tariq et al., 2010, Ion et al., 2012]. This can be adapted for a pub/sub system where brokers are selectively

allowed to access subscriptions based on the matching result. The paper describes the use case of a stock trading scenario. The proposed mechanism offers the possibility to encrypt a query formalized as a hyper-rectangle B, and to decrypt only if a point $X \in B$. The use case considered is of an investment broker assigned to execute a transaction order: buy or sell stock for an investor when an event matches the query B. The investor does not trust the broker enough for revealing his query *before* the order is executed. Basically, a positive matching result implies decrypting the query. As long as the matching fails (the order is not executed) the broker does not learn anything about the client's subscription.

[Pal et al., 2012] presents a secure middleware pub/sub solution that departs from the broker based pub/sub model. Instead of brokers, publishers and subscribers rely for publication delivery on *dissemination* and *repository* servers. The publisher encrypts the publication payload using ABE [Bethencourt et al., 2007] and publication headers using hidden vector encryption (HVE) [Boneh and Waters, 2007], a technique which is conceptually close to ABE. HVE effectively hides the publication attributes, which in case of an ABE scheme as [Bethencourt et al., 2007] must be revealed to decrypt the payload. Encrypted publications are sent to the dissemination server. This forwards the payload to the repository server and the header to the subscribers. The subscribers perform the matching at their own site over encrypted publications headers. Subscriptions are not sent into the system, since HVE does not support also encrypting these. In case of an unsuccesful match the subscriber does not obtain anything. If the matching is positive the subscriber gets an ID which is afterwards used to retrieve the payload from the repository server, this being decrypted using the ABE scheme.

2.2 Security models and access control in pub/sub

Solution	**Security model design**	**Confidentiality enforcement**	**Message routing**	**Base test platform**
[Bacon et al., 2008]	RBAC-based: nodes are granted rights according to specific roles in their domains	messages encrypted through a symmetric scheme (e.g., AES)	restricted to decryptable attributes based on node rights	Hermes [Pietzuch and Bacon, 2002]
[Zhao and Sturman, 2006]	RBAC-based: nodes are granted rights according to an associated *principal* identifying a role	publication oriented, through distribution restriction caused by limited principal subscription rights	normal filtering on accessible messages	Gryphon [Strom et al., 1998]
[Khurana, 2005]	system-wide separation between sensitive and non-sensitive fields	publication-oriented, encrypted sensitive fields (secure XML document dissemination [Bertino and Ferrari, 2002])	based on accessing exclusively the non-sensitive fields	secure XML document dissemination [Bertino and Ferrari, 2002]
[Fiege et al., 2004]	access control policies at the level of *scopes* (grouping structures)	publication-oriented, through distribution restriction caused by limited scope subscription rights	normal filtering on accessible messages	Rebeca [Parzyjegla et al., 2010]
[Wun and Jacobsen, 2007]	post-matching policies set at the broker filtering level	publication-oriented, through distribution restriction based on post matching rules	normal filtering on accessible messages	PADRES [Jacobsen et al., 2010]

Table 2.2: Overview of pub/sub security models.

The second area of published work we overview considers security models research. The articles we cover in this section consider different trust assumptions in their threat models. For instance it may be possible that (some) brokers are allowed or trusted to access

the content of publications and subscriptions, given that these brokers are granted a certain level of authorization. An example is the police routing infrastructure presented in Chapter 1 in Figure 1.2. Also, the issue of unauthorized subscribers and publishers that might try to access the pub/sub service and gain access to sensitive data is taken sometimes in account. The security architecture design often targets the case of multiple interconnected domains in which the threat model and consequently the trust assumptions on data access are different. In the overviewed security models this often results in developing an access control solution.

Unlike the work described in the previous section, the articles we overview here are far less focused on the cryptographic mechanisms and their internals. Cryptography is still used for preserving confidentiality. However, what is required or used in the implementation of the models are well-known schemes, which do not feature specific functionalities like encrypted matching. In the following we focus on such research that does not imply matching over encrypted data. Our main thesis contributions do not apply directly to this area of confidentiality related work, being tightly coupled to encrypted matching schemes. However, for completeness, we consider of interest a short overview of this material. Moreover, we believe solutions ensuring confidentiality through encrypted matching in pub/sub, which are usually presented in a simplistic security model context can integrate with more complex designs, like the ones we survey in this section. Actually, some of the work surveyed in Section 2.1.2 overlaps with the current overviewed area. In particular articles like [Nabeel et al., 2009], [Ion et al., 2012] or [Tariq et al., 2010] go beyond the encrypted matching technique, touching issues such as access control.

[Bacon et al., 2008] describes a pub/sub security solution that uses a role-based access control model. The initial form of the proposed architecture is described in [Belokosztolszki et al., 2003], with extensions developed in [Bacon et al., 2005, Pesonen and Bacon, 2005, Pesonen et al., 2007a, Pesonen et al., 2007b]. Most of the presented work builds upon OASIS [Bacon et al., 2002], an architecture providing role-based access control in the general area of distributed systems. The work in the referenced papers is integrated on top of the Hermes [Pietzuch and Bacon, 2002] pub/sub platform. We concentrate our presentation on the development status presented in [Bacon et al., 2008], which specifically focuses on *confidentiality*. This work is followed by [Bacon et al., 2010], where the authors present a general look upon security in multi-domain pub/sub systems. [Singh et al., 2011] also takes a further step from the initial solution into the area of disclosure control.

Architecture organization. The system participants (publishers, subscribers, brokers) are not trusted to access the pub/sub data until they obtain authorization for their assigned role, which grants specific access rights. [Bacon et al., 2008] considers multiple domains of administration. In each domain an *access control manager* is responsible to grant rights related to pub/sub operations according to predefined policies. *Key group managers* administrate a distinct organization of *key groups* in a similar manner for rights related to cryptographic operations.

Functionality description. The model functionality is based on a role-based access control (RBAC) system. Clients that have the right to define message types (e.g., the right to register a "patient data" publication type) become the *owners* of the types they register. The *owners* of a message type establish the access control policy for the respective type. A policy associates different roles with specific pub/sub action rights on a particular message type: advertise before publication, publish, subscribe, and extend (modify) the type. The *access control managers* enforce the pub/sub rights policy in their domain: after authenticating, a system node will benefit of the specific set of rights associated with his role. *The access*

control manager grants the access to the pub/sub system functionality and implicitly to pub/sub data in its plaintext form. For providing fine-grained *confidentiality* the system design further considers the usage of cryptography. Initial encryption and final decryption of pub/sub messages before delivery are carried out by edge brokers (brokers that are directly connected to the publishers and subscribers). The generic assumption is that edge brokers are trusted to access the pub/sub data in the domain of the clients they are connected to. Besides this assumption, the type owners establish the access rights in case of non-edge brokers through *policies*. These policies are enforced by *key group managers*, which authorize brokers to join *key groups*. A *key group* is formed of brokers, which share the same level of access to keys required to encrypt or decrypt pub/sub data. [Pesonen et al., 2007a] and [Bacon et al., 2008] suggest the usage of OFT (One-way Function Trees) [Sherman and McGrew, 2003] for key management in the groups formed. [Pesonen et al., 2007b] mentions AES [Daemen and Rijmen, 2002] in EAX mode [Bellare et al., 2003] as the preferred encryption scheme.

Confidentiality provisioning. The level of protection for pub/sub data depends on the encryption granularity. Two cases are considered: encryption of the whole message and encryption per attribute. The first case permits any broker to route messages based on the type, which is always accessible, similar to a topic based routing scenario. Authorized brokers can decrypt the complete message and perform content based routing. In case of encryption per attribute, an independent encryption key is associated to each attribute. A broker routing capacity depends on how many attributes it can decrypt according to its given rights. This allows to model fine-grained trust assumptions by the type owners but also implies a level of overhead that grows proportionally with the number of accessible attributes.

[Zhao and Sturman, 2006] presents a service model providing access control in a pub/sub system with specific focus on the dynamic character of the solution. This refers to the effects caused by changing the rules enforced at runtime. The pub/sub system used to implement the service model is Gryphon [Strom et al., 1998].

Architecture organization. The access to pub/sub data traversing multiple interconnected domains is regulated through access control rules. Neverheless, the article limits the discussion mostly to the context of a single domain. Subscribers and publishers are not trusted to perform actions on pub/sub data unless specifically authorized. A *security administrator* is the authority that has the responsibility of adding or performing changes to the rights granted to the pub/sub system clients.

Functionality description. Pub/sub clients obtain action rights as connect, publish or subscribe, if they authenticate successfully as a *principal*. This is similar to a role, multiple clients being able to run on behalf of the same principal. The service model formalizes access control rules into rights using an extended format for pub/sub messages: [*Principal, Type of Access, Filter*]. The filter can specify, depending on the *type of access*, either a simple boolean value (e.g., in case of connection right) or an extended filter similar to a subscription (for setting restrictions in subscriptions and publications). The rights are stored into a database by the *security administrator*, which also handles rights changes. Such changes are processed in atomic batches distributed in the broker network. A broker receiving such a batch of access control changes chooses a starting point in the message stream routed, from where to enforce the new rights. This starting point is communicated to the *security administrator*. Afterwards, new rules are propagated and enforced in a consistent manner throughout the whole system.

Confidentiality provisioning. The property is considered with respect to subscribers untrusted to access specific publications. Confidentiality is provided by restricting publication distribution. This is achieved by limiting granted subscription rights.

[Khurana, 2005] presents a model relying on proxy encryption. Messages are represented by XML documents. The actual pub/sub functionality is ensured through a solution targeting secure XML document dissemination [Bertino and Ferrari, 2002].

Architecture organization. The threat model assumes that brokers are not trusted to access sensitive parts in pub/sub messages. Routing is done over non-sensitive fields. The secure pub/sub system requires the presence of trusted *servers* that will host a proxy security and accounting service (PSAS).

Functionality description. From the publications that are sent to the system, only the part that is considered to be sensitive and non-routable is encrypted. The technique used is the secure XML document distribution [Bertino and Ferrari, 2002] by the means of symmetric encryption. The key used for this purpose is further encrypted with a public key belonging to the publisher and sent as part of the encrypted publication. In order to distribute the symmetric key the system relies on the PSAS provided by the set of trusted servers. The PSAS acts on broker requests, and transforms the packets encrypted with the publisher public keys into packets encrypted with the subscribers' public keys, therefore avoiding the issue of the necessary key exchange between the source and destination. The scheme is based on [Jakobsson, 1999]. It is assumed that a quorum formed by a given number of PSAS servers will be always available.

Confidentiality provisioning. The property is enforced by protecting the publication attributes that are not routable. Since publications are represented entirely through XML documents, the payload case is not specifically referred, but one can assume that this is protected through the same mechanism as the headers.

[Fiege et al., 2004] introduces a security model based on *scopes* as the underlying support for creating groups of trust. The pub/sub platform used as test bed for the model implementation is Rebeca [Parzyjegla et al., 2010].

Architecture organization. Common groups of trust cover the entities in the system (publishers, subscribers, brokers). Communication in these groups is separated from the outside world. The admission of new members is based on credential authentication, according to group acceptance criteria. In order to support the creation of trust groups, the paper introduces the notion of *scopes*, which we detail below.

Functionality description. Mutual trust relationships between different administrative domains lead to the creation of *scopes*. A scope is a group that encapsulates a number of system nodes. A scope can recursively be a member of other scopes. Scopes are used to limit the visibility of a message to their members. This is achieved through access control policies. The enforcement of the rules in these policies is based on checks performed on *attribute certificates* [Farrell and Housley, 2002], which represent credentials consisting of a signed identity associated to a set of attributes. Such *attribute certificates* are issued either by an *attribute authority* or by the owner of the broker network in the scope of interest.

Nodes obtain the credentials and use them when advertising a type of publication (the functional model requires this step before publishing) or when they try to submit subscriptions. The edge brokers enforce the checking of credentials and further permit the dissemination inside the scope. A broker tries to match the attributes in a subscriber's

certificate to the ones in the publisher's advertisements previously received. If the match fails, the subscriptions will not be processed against the type of publications corresponding to the respective advertisement.

Confidentiality provisioning. The property is not specifically addressed in the article. However, the access control mechanisms provide confidentiality against any node that falls outside a scope of trust, although this does not imply modifying the messages content for protection, but merely restricting their dissemination.

[Wun and Jacobsen, 2007] presents a policy management framework dedicated to content-based pub/sub architectures. This framework offers general policy management services, not necessary related to security. However, the use case proposed in the paper is that of specifying security policies. The framework was implemented on top of the PADRES [Jacobsen et al., 2010] pub/sub system.

Architecture organization. The article considers a simple setting of *trust groups* in the security policies use case, in which the entities present in groups (publisher, subscriber, broker) interact as discussed in the following.

Functionality description. The design is centered on a post-matching policy model. In this model, a certain rule defined in a policy is triggered immediately after or even during matching. The main purpose is to avoid testing the rules *a priori*, which would duplicate the matching functionality in case of policy rules that are semantically based on the publication content.

In the use case considered for security policies, it is assumed as a precondition that each *trust group* is associated with a shared group secret. This secret information is used by the pub/sub entities for authentication and encryption using specific group related protocols. The effective contribution brought by the policy management framework to confidentiality preservation in the system comes under the form of *authenticated event scopes* and *security zones*.

Confidentiality provisioning. The *authenticated event scopes* are associations of an authentication policy to rules in the post matching model. For instance, if a message must be routed to a certain broker after it was matched, an authentication of the receiver might be performed first according to the policy. *Security zones* extend the functionality of the authentication policies. The attributes in the matched publications can be either pruned or encrypted before forwarding, based on criteria given by established zones with different policy settings.

2.3 Summary

In the current chapter we have overviewed existing solutions that target confidentiality in publish/subscribe systems. In Section 2.1 we have discussed first in detail encrypted matching as a cryptographic paradigm. The particularities and the novelty of this approach among cryptographic solutions make cumbersome to characterize solutions that follow this design. We have identified several characteristics of encrypted matching, that helped further classifying the schemes. A practical deployment of these schemes can benefit directly of our main contributions in this thesis, presented in Chapters 3 and 5.

Further, in Section 2.2 we surveyed the most cited work on security models, that targets scenarios with more complex threat assumptions, and which does not rely or reference

encrypted matching as effective tool for confidentiality preserving. In our view, although some common approaches exist, there is just a minor gap between the two areas of research surveyed. We believe that encrypted matching schemes can be integrated with security models based on other paradigms like access control, which often rely on classic cryptographic algorithms that lack flexibility, but are nevertheless needed for the model's confidentiality requirements.

Chapter 3

Pub/sub prefiltering: alleviating performance drawbacks in encrypted matching

3.1 Motivation, problem description and contributions

Matching and routing publications over subscriptions in untrusted domains is an issue in pub/sub systems. Encrypting the pub/sub data provides the required confidentiality. Schemes that permit performing comparison operations over the resulting ciphertexts are the most flexible alternative to maintain the pub/sub filtering functionality. This is the reason why, as overviewed in the previous chapter, encrypted matching is lately under focus and seemingly the preferred paradigm to use. Despite the latest advancements in this direction, content-based pub/sub is not very often deployed as the communication mechanism of choice for applications requiring confidentiality and spanning multiple administrative domains. We believe one of the main reasons is the performance penalty associated with the privacy-preserving filtering.

As an example, one of the existing solutions we overviewed in Section 2.1 is ASPE [Choi et al., 2010]. The scheme permits both equality and ranged comparisons over encrypted data, making it interesting in terms of functionality. As a drawback, the filtering costs have a worst-case quadratic increase with the number of attributes. In comparison, the plaintext matching cost increases at most linearly. Figure 3.1 presents the average time required to filter one publication against one subscription, measured over 1,000 publications and 1,000 subscriptions. ASPE is compared against plaintext matching and uses a naive approach where each constraint is evaluated separately and no optimization is used for the filtering process. The workload used is composed of simple multi-dimensional uniformly distributed ranges. Note that the y axis represents the number of constraints actually evaluated. We clearly observe that the cost of evaluating one subscription is higher and evolves in a quadratic manner in the number of constraints with ASPE. Similar or higher increasing cost rates are obtained with the other encrypted schemes (e.g., [Nabeel et al., 2012, Ion et al., 2012]).

Subscription containment (also referred as coverage) is a characteristic of subscriptions workload that is largely used to speed up the filtering operation [Jafarpour et al., 2009, Carzaniga et al., 2001, Jacobsen et al., 2010]. We remind that it is said that a subscription s_i *contains* another subscription s_j if any event that matches s_j also matches s_i, in other words if s_i is more general than s_j. For instance, subscription $\{x > 0\}$ contains both subscriptions $\{x = 1\}$ and $\{x = 1 \wedge y = 0\}$. Figure 3.2 displays the containment relationships established between five subscriptions. When a subscription contains another subscription, any publication known not to match the former will never match the latter. Conversely,

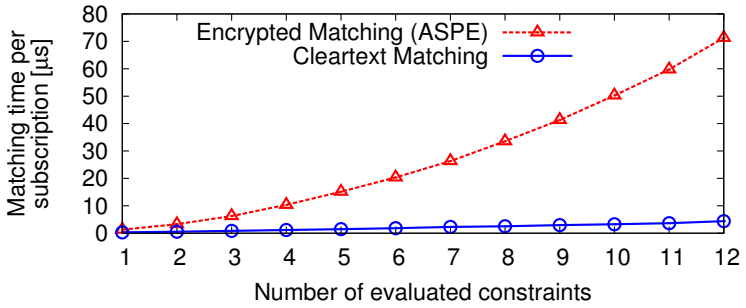

Figure 3.1: Performance overhead of encrypted matching: example of the ASPE mechanism.

any publication known to match the latter will always match the former. Note that the containment relationship creates a partial order on subscriptions. By organizing subscriptions according to that order, it is possible to quickly discard or accept many subscriptions before they are tested against a publication.

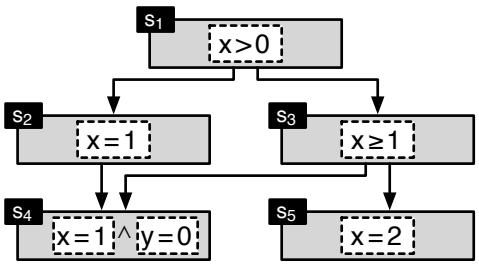

Figure 3.2: Five sample subscriptions and their containment relationships (represented by arrows).

Despite the practical functional advantages, the ability to determine containment between subscriptions can pose a threat to confidentiality, even when these subscriptions are encrypted. It indeed allows determining groups of similar subscriptions. These groups may later be used in conjunction with domain-specific knowledge in order to infer information about the subscriptions. For instance, in the example of Figure 3.2, three subscriptions for the same field name would be grouped together in the same containment sub-tree. In a stock exchange scenario, using domain-specific information on relative stock popularity, an attacker may be able to derive information about the symbol such a sub-tree is associated to, based on the relative sizes of these sub-trees. Imagine for instance the situation of a highly transactioned symbol. It is likely to assume that this can be found in the root of a dense populated encrypted subscriptions sub-tree. Even without such domain-specific knowledge, containment information can be leveraged by an attacker to obtain the information that two particular subscribers have common interests (yet not knowing what these common interests are). Similar observations were made in [Raiciu and Rosenblum, 2006]. As described in Section 2.1 some schemes do not offer by design the possibility to determine containment [Shikfa et al., 2009, Li et al., 2004]. Some other schemes [Choi et al., 2010, Raiciu and Rosenblum, 2006, Nabeel et al., 2012] have optional support for containment. For example, with ASPE [Choi et al., 2010], the ability to determine containment is determined by the availability of additional reference vectors attached to each subscription. It is the responsibility of the application to

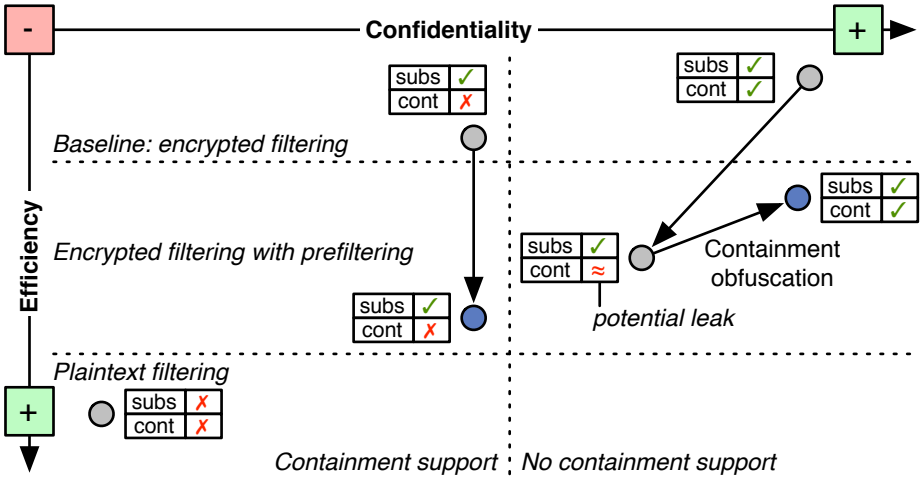

Figure 3.3: High-level picture of the performance and confidentiality tradeoffs addressed in this chapter (`subs` stands for subscriptions confidentiality and `cont` for containment confidentiality).

include or not this additional information. We name the ability of a encrypted matching scheme to prevent containment determination as *containment confidentiality*.

The main drawback of encrypted matching techniques is their high computational cost. The objective in the work presented in this chapter is to reduce these costs while preserving the confidentiality properties. The main contributions are represented by Figure 3.3 and summarized below. We do not propose novel encrypted matching techniques but build upon existing ones. We consider both the case where containment determination is permitted by the scheme (left-hand side of Figure 3.3) and the case where the scheme provides containment confidentiality (right-hand side of Figure 3.3).

3.1.1 Prefiltering using Bloom filters

The first part of the work covered in this chapter defines an efficient *prefiltering* technique to significantly reduce the space of subscriptions that must be tested by the encrypted filtering engine during routing. In a nutshell, the principle is to embed Bloom filters [Bloom, 1970] inside publications and subscriptions. These Bloom filters encode the values carried by the publications and the equality constraints of the subscriptions. By testing the Bloom filters of subscriptions for inclusion in those of publications, one can efficiently determine the possibility for a message to match a subscription: if the test is negative, the message is guaranteed *not* to match; otherwise, it *might* match. An encrypted publication is matched against an encrypted subscription only in the latter case. The prefiltering operator can also leverage the possibility to determine containment information by the encrypted filtering operator. The prefiltering operator significantly improves the performance of privacy-preserving encrypted matching, with or without the support for containment determination. The work on prefiltering we present in this chapter follows closely the solution we published in [Barazzutti et al., 2012b].

3.1.2 Containment obfuscation

The second part of this chapter considers explicitly the case where the base scheme provides containment confidentiality. Since prefiltering requires additional information (the Bloom filters) to be attached to subscriptions, we analyze the power this additional information gives to an attacker. Indeed, despite their hash-based construction and their probabilistic nature, Bloom filters may still convey sensitive information. In particular, identical subscriptions yield equal Bloom filters, and contained subscriptions produce Bloom filters that are included in one another. As illustrated on the right-hand side of Figure 3.3, the use of prefiltering atop a scheme providing containment confidentiality may result in potential leaks of containment information–albeit strictly less so than with the ability to determine containment directly from encrypted subscriptions.

We propose a method for containment obfuscation, and determine through a comprehensive mathematical analysis the impact of prefiltering on containment confidentiality. Containment obfuscation introduces randomness in the prefiltering process by randomly choosing a small number of hash functions from a larger set for each subscription equality constraint encoded in the Bloom filters. The main difficulty is the analysis of the dependencies introduced by hash functions collisions, which are cumbersome to model mathematically. We show that there is a fundamental tradeoff between the information that can be leaked to an attacker from the Bloom filters and the performance of the prefiltering operation. We provide evidence, both mathematically and using numerical simulations, that efficient prefiltering preserving containment confidentiality can be achieved nonetheless. The results on containment obfuscation presented in this chapter follow the analysis we published in [Mercier et al., 2013].

3.2 The prefiltering algorithm

In this section, we present a prefiltering operator based on Bloom filters embedded with subscriptions and publications, and that can take advantage of containment relationships when available. The two mechanisms: Bloom based prefiltering and containment, can be used in isolation but provide the best performance when combined. Both aim at reducing the number of calls to the actual encrypted matching operation, which we denote as ENC-Match. They do so by pre-discarding subscriptions that are known not to match (*negative matching*) and pre-selecting subscriptions that are known to match (*positive matching*), before calling ENC-Match for these subscriptions. The contributions we bring are oblivious to the nature of the ENC-Match encrypted matching operation used, the only difference being made by the possibility or restrictions of leveraging subscriptions containment.

3.2.1 Exploiting containment relations

We start by discussing how we can leverage *containment relationships* at the level of entire subscriptions. This approach only works when the encrypted matching scheme does not preserve containment confidentiality and therefore it allows determining containment relations from encrypted subscriptions.

We follow a classical approach to exploit containment relationships as part of the filtering process [Carzaniga et al., 2001, Jacobsen et al., 2010]. We organize subscriptions in a *partially ordered set* (or *poset* for short). Precedence relations in the poset correspond to

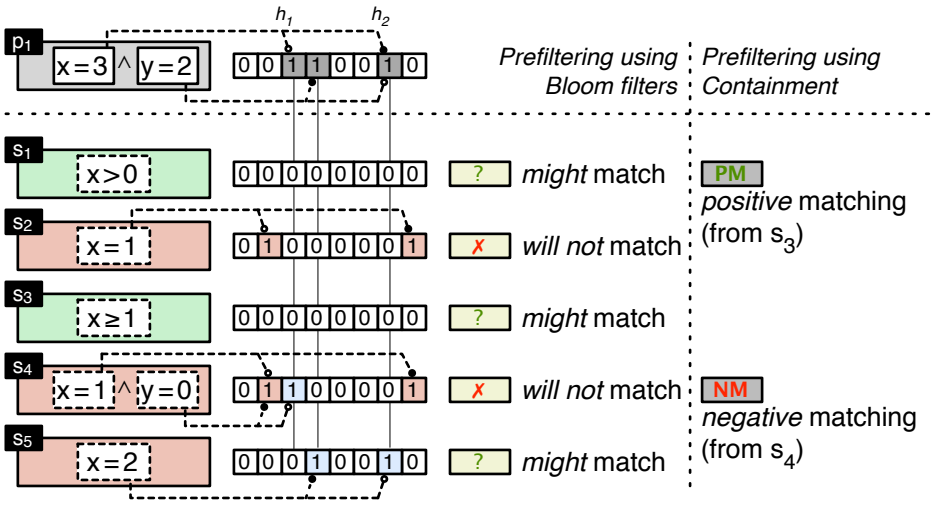

Figure 3.4: Information used for prefiltering: Bloom filters embedded with subscriptions and publications, and containment-based matching.

containment relations between subscriptions. Note that a poset can be formed of several, non connected sub-graphs. Upon filtering, when encountering a subscription s that matches a publication, we know that all its *containers* (ancestors in the containment poset) also match the publication. This triggers the positive matching of its containers up to the root of the corresponding sub-graph of the poset. Conversely, if s does not match the publication, all its *containees* (descendants in the poset) do not match either, and are applied the negative matching decision.

We leverage both positive and negative containment-based matching. As we describe in Section 3.2.3, the algorithm we propose works by evaluating the complete set of subscriptions, split in several lists. The order in which subscriptions are tested is of great importance for the ability to take matching decisions based on containment. A random evaluation order may lead, for instance, to containers being evaluated with positive matching before their containees, resulting in two calls to the costly ENC-Match function instead of a single one with the opposite evaluation order. Similarly, it is better to evaluate containers for negative matching before their containees. Determining positive matching requires calls to ENC-Match while determining negative matching is based both on calls to ENC-Match or, in a more cost-efficient manner, on the Bloom filters attached to subscriptions and publications, as we describe in the next section. We evaluate subscriptions in an order that respects the precedence relation in the poset, to take full advantage of these early negative matching decisions. We further describe the data structure used to store subscriptions in Section 3.2.3.

3.2.2 Negative matching using Bloom filters

One of the key principles of prefiltering is to quickly identify subscriptions that are known *not* to match an incoming publication (negative matching), without calling the costly ENC-Match function. To that end, we embed Bloom filters [Bloom, 1970] in publications and subscriptions, when applicable, and use simple bit-wise operations to discard non-matching subscriptions.

Bloom filters are probabilistic data structures that provide efficient testing of whether or not an item belongs to a set. A Bloom filter is essentially a bit array. When adding an

item, one or several hash functions $h_1, \ldots, h_k$ are used to identify bit(s) of the Bloom filter that must be set to 1. To test whether an item belongs to the set, it is hashed again and, if all corresponding bits are set, it is likely to be part of the set; otherwise, it is guaranteed to not belong to the set. Therefore, Bloom filters can yield false positives but no false negatives. The accuracy of the Bloom filter can be tuned by properly choosing its size and the number of hash functions.

When injecting a publication p in the system, besides encrypting it, we additionally hash the values of *all* its fields and insert them in a non-encrypted Bloom filter $B(p)$. On the other side, every subscription s also embeds a Bloom filter $B(s)$, that contains the hashed values of its constraints with *equality* constraints (or a subset of those). For instance, given a constraint *x=2*, the value 2 will be added to the Bloom filter. Figure 3.4 shows the same subscriptions as Figure 3.2, along with the information that can be used for prefiltering for a sample publication p_1. We assume for the illustration a size of 8 bits for the filters and 2 hash functions h_1, h_2.[1] Upon matching, if $B(s) \not\subseteq B(p)$,[2] we know that s does not match p, irrespectively of the other (non-equality) constraints, and we can discard it. This is the case, for instance, for s_2 where two bits in $B(s_2)$ are set that are not set in $B(p_1)$. Otherwise, s must be checked using the ENC-MATCH function (s_1, s_3 and s_5).

The Bloom filters do not produce false negatives: no matching subscription can be discarded by the Bloom filter comparison. In respect to false positives, non-matching subscriptions may be evaluated by the ENC-MATCH function for three reasons. First, a Bloom filter $B(s)$ only encodes values associated to the equality constraints of s. A subscription may not match due to other, non-encoded constraints such as inequalities (e.g., $s = \{x = 3 \wedge y < 0\}$). Second, $B(s)$ includes values irrespective of the attribute they are associated with. This can yield false positives, e.g., with s_5 the value 2 is associated to a constraint on x while it is associated to a value for attribute y in p_1. Finally, Bloom filters themselves produce false positives because distinct values can be associated to the same bits. The likelihood of such collisions can be controlled by carefully choosing the size and number of hash functions used by the Bloom filters. We will evaluate the false positive rate for different configurations and workloads in Section 3.5.

For security reasons, the hash function configuration used in the Bloom encoding is kept private from the brokers. The hashes can be typically parameterized by a secret key, and use a cryptographically secure scheme such as SHA-1 in a HMAC [Bellare et al., 1996] construction.[3] This hash key is known only to subscribers and publishers. The exchange of this secret key between the peers can be carried on by the preliminary key agreement protocol required by the encrypted matching scheme itself. We elaborate on the key management issue in Chapter 5.

Despite the above summarized security measures, the Bloom filters attached to subscriptions may however permit determining a subset of containment relationships. We analyze this risk and provide a mitigation in Section 3.4. Beside these potential containment relations, by design the Bloom filters do not allow untrusted parties to obtain further information about the non-encrypted subscriptions.

[1] Note that two values may hash to the same bit, hence the number of bits set in the Bloom filter is not necessarily equal to the number of hash functions multiplied by the number of distinct values associated with equality constraints, as for s_4 in the figure.

[2] Operator $\subseteq$ denotes here bitwise inclusion, i.e., all the bits in $B(s)$ are also set in $B(p)$.

[3] Even if SHA-1 (or other possible cryptographic hashes) output values over 160 bits, we can consider without loss of generality that a summary of these is used to set bits in the filter.

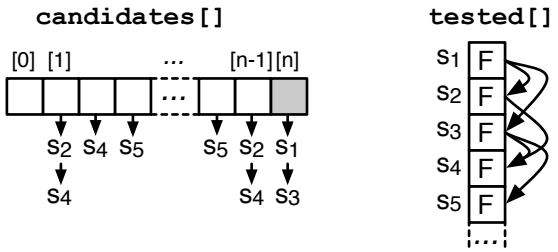

Figure 3.5: Data structures used for prefiltering.

3.2.3 Algorithm description

The prefiltering algorithm takes advantage of both types of information available about subscriptions that we described previously and represented in Figure 3.4. Bloom filters allow to determine negative matching, and containment relationships (when available) allow determining negative and positive matching. Both allow reducing the number of calls to the costly ENC-Match encrypted matching function.

The prefiltering algorithm uses the data structures shown in Figure 3.5. We maintain an array (`candidates[]`) of $n+1$ lists of candidate subscriptions, where n is the number of bits in Bloom filters. The i^{th} list for $i \in \{0, 1, \ldots, n-1\}$ contains subscriptions that have the i^{th} bit of their Bloom filter set. It follows that each subscription may belong to several lists. The last list contains subscriptions that have no equality constraint represented in the Bloom filter, i.e., an empty Bloom filter $B(p)$. A subscription part of the default list is not part of the other candidate lists.

When containment relationships can be determined between encrypted subscriptions, we sort subscriptions in the candidate lists in such a way that containers appear before their containees. To that end, when inserting a new subscription, we place it just before its first containee appearing in the list, or at the end if no containee is found.

We additionally maintain an array (`tested[]`) to keep track of which subscriptions have already been tested,[4] as well as any known containment relationships between subscriptions (denoted by arrows in the figure but actually stored as a graph as in Figure 3.2).

We will consider several variants of the basic algorithm. For instance, if containment relationships are known, we may discard additional subscriptions when one is found that does not match, or conversely accept several subscriptions at once upon successful match, respectively denoted by `CB-NEGATIVE-MATCH` and `CB-POSITIVE-MATCH` in the code (`CB` stands for *containment-based*).

The prefiltering process works as shown in Algorithm 1. Initially, the `tested[]` array is reset to indicate that no matching has taken place yet (line 2). We then traverse the list `candidates[i]` associated with each bit i set in the publication's Bloom filter $B(p)$, as well as the default list (lines 3–5). If the current subscription s has not yet been tested, as indicated by `tested[s]`, we check if it is a possible match (lines 6–8). This can happen in two cases: s is part of the default list, i.e., $B(s)$ is empty; or $B(s) \subseteq B(p)$. If so, we must use the costly but accurate ENC-Match encrypted matching function that tells us with certainty if the publication must be forwarded to the corresponding subscriber (line 9).

In case of a match, we forward the publication to the interested subscriber (line 11) and we optionally use containment-based positive matching to accept any yet-untested subscription (lines 12–16). On the other end, if the publication does not match s, we may

[4]The reason for using an array is that we can efficiently reset it (e.g., using `bzero`).

Algorithm 1: Prefiltering algorithm.

```
1  PF-MATCH(p);
2  tested[] ← {false, ..., false};                          // No subscription tested yet
3  for i ← 0...n do                                         // Check every list (including default)
4      if i = n∨B(p)[i] = 1 then                            // Default list or bit set in BF?
5          foreach s ∈ candidates[i] do                     // Traverse list
6              if ¬tested[s] then                           // Not yet tested?
7                  m ← false;                               // Initially not a match
8                  if i = n∨B(s) ⊆ B(p) then                // Possible match?
9                      m ← ENC-MATCH(p, s);                 // Must do costly test
10                 if m then                                // Match?
11                     FORWARD(p, s);                       // Forward to subscriber
12                     if CB-POSITIVE-MATCH then            // Do positive match?
13                         foreach s' ∈ containers(s) do
14                             if ¬tested[s'] then          // Not yet tested?
15                                 tested[s'] ← true;
16                                 FORWARD(p, s');          // Forward to subscriber
17                 else                                     // No match
18                     if CB-NEGATIVE-MATCH then            // Do negative match?
19                         foreach s' ∈ containees(s) do
20                             tested[s'] ← true;
21                 tested[s] ← true;
```

use containment-based negative matching to mark all the subscription's containees as having been tested, effectively discarding them for the rest of the filtering process (lines 18–20).

3.2.4 Discussion

Let us first consider prefiltering when containment is not available or restricted in the encrypted matching scheme. Bloom filters are still attached to messages, and this might reveal some degree of containment. However, as we will present in Section 3.3 *containment obfuscation* can still be obtained by changing the Bloom filter encoding. In fact, the very role of the Bloom filter is to perform aggressive negative matching without knowing containment relationships between subscriptions. During prefiltering, we only need to traverse the candidate lists associated with the bits set in the publication's Bloom filter, as well as the default list. All other subscriptions are implicitly discarded. This negative matching step is expected to account for a vast majority of subscriptions that can be filtered out, as we will study during the evaluation in Section 3.5. The number of lists to traverse is therefore function of the number of fields (i.e., attributes) of the publication, which is expected to be much smaller than the size of the Bloom filters. Further, no subscription is tested more than once thanks to the bookkeeping of the `tested[]` array.

Let us now consider *containment-based* prefiltering. If containment relationships are known, one can perform additional negative and positive matching. When testing a subscription that does not match, one can immediately discard all of its containees. Remember that we place subscriptions in candidate lists such that containers appear before containees, therefore in all likelihood one can significantly reduce the number of tests (and hence calls to the costly ENC-MATCH encrypted matching function) by applying this negative matching optimization.

However, it might happen that a subscription is tested before some of its containers, because the latter may not belong to the same candidate lists as the former. In such cases,

when successfully matching a subscription, one can apply the positive matching optimization to immediately accept any yet-untested container.

The likelihood for any of these scenarios to occur depends on the nature and distribution of the publications and subscriptions. We experimentally evaluate the efficiency of the prefiltering variants in Section 3.5.

3.3 Prefiltering truncation: handling containment confidentiality losses

We now focus on the case where it is not possible to determine containment relations between subscriptions. This can result either from a design limitation of the used encrypted matching scheme, or can be the consequence of security restrictions that explicitly require disabling the containment support, even if the encrypted matching scheme could support it. As depicted in Figure 3.3, we are interested in preserving containment confidentiality while still benefiting from the gain of performance obtained from prefiltering.

As mentioned in the introduction, we consider that in a non trusted environment such as a public cloud, there is a risk that an attacker collects a snapshot of a broker's state, and in particular its set of stored subscriptions. We are interested in knowing whether the addition of the prefiltering Bloom filters to subscriptions allows an attacker to determine *more information* about subscribers than otherwise possible (i.e., when observing encrypted subscriptions only). We mention again that the prefiltering solution described in this chapter can be applied independent of the encrypted matching scheme used, and therefore we are not interested here in evaluating any encrypted matching scheme security. We assume the encrypted matching preserves the messages confidentiality, and we consider this as the security level baseline for the additions we make. More precisely, we consider that the attacker is only allowed to use the available operations on the encrypted matching scheme (ENC-Match and `contains`) as well as any computation on the prefiltering Bloom filters. The normal prefiltering described in the previous section introduces a problem when the `contains` operation is restricted.

A property of prefiltering using Bloom filters is that contained subscriptions produce Bloom filters that are included in one another. This may allow an attacker to derive some of the containment relationships between subscriptions. Indeed, if subscription s contains s' ($s \sqsupseteq s'$), there are two possibilities. If $B(s) = B(s') = 0$ (no bit set), s and s' do not contain any equality constraint and the link cannot be inferred from the filters. If $B(s) \subseteq B(s')$, then $s \sqsupseteq s'$ can be inferred but with less certainty than using containment evaluation: it is not possible to distinguish between (1) real inclusions; (2) inclusions only for the equality constraints but not for the range constraints, e.g., if $s = \{x = 1 \wedge y < 3\}$ and $s' = \{x = 1 \wedge y > 12\}$; (3) false positives due to collisions in the filter or constraints setting values for different attributes (e.g., $s = \{x = 1\}$ and $s' = \{y = 1 \wedge z = 2\}$). Conversely, when $s \not\sqsupseteq s'$, if $B(s) \not\subseteq B(s')$ then $s \not\sqsupseteq s'$ is a correct guess.

To counter the possibility to derive containment relationships from Bloom filters, we provide *containment obfuscation* mechanisms. Their principle is to introduce additional randomness in the Bloom filters by randomly removing set bits from the subscriptions Bloom filters (*truncation*) and randomly setting bits in the publications Bloom filters (*pollution*). Pollution and truncation are illustrated by Figure 3.6. As can be seen from the figure, s_1

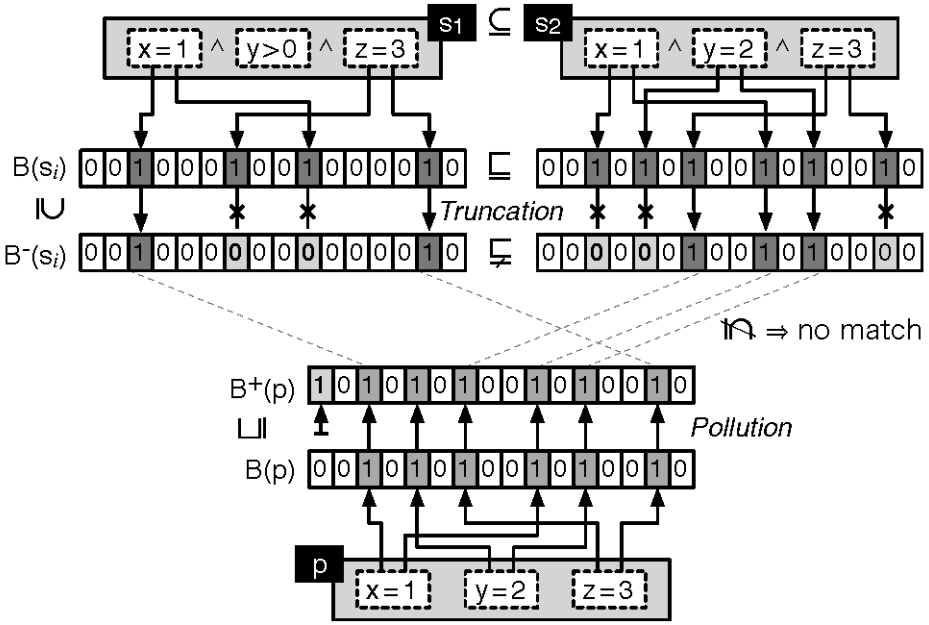

Figure 3.6: Polluting publications Bloom filters and truncating subscriptions Bloom filters.

contains s_2, but after truncation their Bloom filters do not contain each other anymore. This noise makes it much harder for an attacker to derive accurate containment graphs.

It should be clear that polluting publications and truncating subscriptions increases the probability of false positive and do not introduce false negatives. Polluting subscriptions and/or truncating publications would introduce false negatives, and prefiltering would not work.

3.4 Security analysis

In the following study, we target the situations where the used encryted matching scheme does not support subscription containment, or has it disabled for security reasons (e.g., in case of ASPE removing some necessary additional structures added in the encryption process). Therefore, we focus in the following on the analysis of truncation. More precisely, we study the tradeoff between prefiltering efficiency and containment information leaked to an attacker. We show that efficient prefiltering is possible while making containment information so noisy that it essentially becomes useless for potential attackers.

3.4.1 Notation and assumptions

Consider a subscription s_1 with $c_1 \geq 0$ different equality constraints encoded in a Bloom filter $B_\alpha(s_1)$ of size n. By truncating, in the encoding we use only α out of total k hash functions available per constraint. Furthermore, let us consider a second subscription s_2 with $c_2 \geq c_1$ different equality constraints also encoded in a Bloom filter $B_\alpha(s_2)$ of size n with α out of k hash functions per constraint. We also consider a publication p with $c_p \geq c_1$ different values. The publication is encoded in a Bloom filter $B_k(p)$, that is, all k hash functions are used. We write $B_\alpha(s_1) \subseteq B_\alpha(s_2)$ if and only if all the nonzero bits in the Bloom filter of s_1 are also

s_i	Subscription i
c_i	Number of different equality constraints in s_i
p	Publication
c_p	Number of different values in p
k	Cardinality of the set of hash functions used to encode values in Bloom filters
α	Number of hashes randomly chosen out of k to encode a value in a subscription Bloom filter
$B_\alpha(s_i)$	Bloom filter representation of Subscription s_i using α hash functions per value
$B_k(p)$	Bloom filter representation of Publication p using all k hash functions per value
n	Bloom filter size
a	Number of attributes in the domain
v	Number of values per attribute in the domain
γ	Number of common values between two subscriptions (or between a subscription and a publication)
P_α	The probability that $s_1 \sqsubseteq s_2$ when $B_\alpha(s_1) \subseteq B_\alpha(s_2)$
$P_{\text{f\_pos}}$	The false positive probability that $B_\alpha(s_1) \subseteq B_k(p)$ when $s_1 \not\sqsubseteq p$

Table 3.1: Notation and definitions for prefiltering.

nonzero in the Bloom filter of s_2.[5] Likewise, we write $B_\alpha(s_1) \subseteq B_k(p)$ if the Bloom filter of a subscription s_1 is included in the Bloom filter of a publication p. The notation is summarized in Table 3.1.

We only consider the number of distinguishable values for subscriptions and publications since it is assumed that the same value appearing for multiple attributes is only hashed once. We only consider the values for equality constraints, which are the only ones encoded in the Bloom filters. Thus, s_1, s_2 and p can formally be defined as sets of cardinality c_1, c_2 and c_p, respectively, and we use $A \sqsubseteq B$ to denote that A is a subset of B. The reason behind this nonstandard notation is to keep in mind that when $s_1 \sqsubseteq s_2$, the set s_1 is included in the set s_2, but the subscription s_1 is more general and is a container of s_2.

It is assumed that the Bloom filters use a set of k independent and perfectly random hash functions. The hashing strategy we use: $\mathcal{H}_\alpha$ consists, for a fixed $\alpha \in \{1, 2, \ldots, k\}$, of randomly selecting α among the k hash functions for each subscription constraint value to be encoded. We emphasize again that the hashing strategy is only applied to subscriptions; all the k hash functions must be used for publications, otherwise false negatives can occur.

3.4.2 Subscription/publication domain

We assume that there are a possible attributes $A_1, A_2, \ldots, A_a$ with equality constraints that can be used for each subscription and in publications, and that the set of v possible values for attribute A_i is $V_i = \{v_1^i, v_2^i, \ldots, v_v^i\}$. It is assumed that when a subscriber or publisher selects an attribute to put in a subscription or publication, it chooses any of the a attributes with probability $\frac{1}{a}$. It is also assumed that the attribute takes each of its possible values with probability $\frac{1}{v}$. The domain we consider assumes that $V_i \cap V_j = \emptyset$ for $i \neq j$. Non-disjoint value

[5] Bloom filters can be viewed as sets.

sets can always be made disjoint by encoding each value with a small prefix representing its attribute.

We assume that an attacker knows the domain and its probability distribution and also k and α. When this attacker examines a subscription Bloom filter, it knows the number of different equality constraints that were encoded in it. This increases its power, but only slightly because the number of nonzero bits in a Bloom filter is highly correlated with the number of encoded values. Furthermore, this allows us to do the analysis without any assumption on the distribution of the number of equality attributes per subscription.

The domain uniformity is not a realistic assumption for most content-based systems, although two remarks must be made. Firstly, the domain uniformity allows us to analyze rigorously how an attacker can infer subscription containment from the Bloom filters as well as the performance of the prefiltering scheme. Secondly, the analysis can be carried out numerically for any content-based system for which an estimate of the statistics at the subscriber, publisher, or system level is available. One such example is presented in Section 3.5.

3.4.3 Problem statement

The objective we have is to study the tradeoff between the amount of information about subscriptions containment that can be inferred by an attacker and the performance of the prefiltering process. We formalize these two notions below.

Containment leaks

We want to prove that randomly selecting a small number of hash functions for each coded attribute of the Bloom filters restricts the attacker capacity to derive containment between the two subscriptions based on their attached Bloom filters. The expression of interest is

$$P_\alpha \triangleq \Pr[s_1 \sqsubseteq s_2 \mid B_\alpha(s_1) \subseteq B_\alpha(s_2)]. \tag{3.1}$$

The higher this probability is, the easier it is for an attacker to build an accurate containment graph when having access to a large number of subscription Bloom filters included in other subscription Bloom filters. Using Bayes' law, we can write

$$P_\alpha = \Pr[s_1 \sqsubseteq s_2] \cdot \frac{\Pr[B_\alpha(s_1) \subseteq B_\alpha(s_2) \mid s_1 \sqsubseteq s_2]}{\Pr[B_\alpha(s_1) \subseteq B_\alpha(s_2)]}. \tag{3.2}$$

When all the k hash functions are chosen, $\Pr[B_k(s_1) \subseteq B_k(s_2) \mid s_1 \sqsubseteq s_2] = 1$.

Prefiltering efficiency

Decreasing the amount of information to potential attackers is useless if it results in a prefiltering operation that fails to discard a significant fraction of subscriptions that do not match an incoming publication. The expression of interest is the probability of false positive

$$P_{\text{f\_pos}} \triangleq \Pr[B_\alpha(s_1) \subseteq B_k(p) \mid s_1 \not\sqsubseteq p]. \tag{3.3}$$

It is the probability, when the equality constraints of a subscription are not included in the values set of a publication, that the Bloom filter of the subscription is included in the Bloom filter of the publication.

3.4.4 Exact analysis with collisions

We first present preliminary results followed by the exact expressions for P_α and $P_{\text{f\_pos}}$. We start by analyzing the number of common values between a pair of subscriptions. Let $\Pr_{\text{common}}[\gamma]$ be the probability that there are γ common values between subscriptions s_1 and s_2 for $0 \leq \gamma \leq c_1$.

Lemma 1

$$\Pr_{common}[\gamma] = \frac{\binom{c_2}{\gamma} \cdot \sum_{\delta=0}^{c_1-\gamma} \left[\binom{c_2-\gamma}{\delta} \binom{a-c_2}{c_1-\gamma-\delta} (v-1)^\delta v^{c_1-\gamma-\delta} \right]}{\binom{a}{c_1} v^{c_1}}.$$

Proof. There are $\binom{a}{c_1}$ ways to choose c_1 attributes and v possible values for each attribute, thus the number of ways to choose the values for subscription s_1 is given by

$$\binom{a}{c_1} v^{c_1}. \tag{3.4}$$

We now fix subscription s_2 and count, among the possible ways to assign the values for s_1, how many of them have exactly γ common values with s_2. First, the number of ways to select the attributes containing the values in s_1 which are also common to s_2 is given by

$$\binom{c_2}{\gamma}. \tag{3.5}$$

The $c_1-\gamma$ values of s_1 that must not be in s_2 fall in one of the following two categories: either they are associated with an attribute that is also present in s_2 (but must take a different value) or they are from an attribute not present in s_2. Let δ be the number of values of s_1 that fall in the first category. Since there are $c_2-\gamma$ attributes to choose from, each with $v-1$ allowed values, the number of ways to assign δ values of s_1 in the first category is

$$\binom{c_2-\gamma}{\delta} (v-1)^\delta. \tag{3.6}$$

With γ common values and δ different values in the first category, there are $c_1-\delta-\gamma$ values of s_1 that must be put in the second category. Since there are $a-c_2$ attributes to choose from, and since all v values per attribute are allowed, the number of ways to assign $c_1-\gamma-\delta$ values in the second category is

$$\binom{a-c_2}{c_1-\gamma-\delta} v^{c_1-\gamma-\delta}. \tag{3.7}$$

From (3.5)-(3.7) and the fact that δ can vary between 0 and $c_1-\gamma$, the number of ways to assign values to s_1 with γ common values with s_2 is

$$\binom{c_2}{\gamma} \cdot \sum_{\delta=0}^{c_1-\gamma} \left[\binom{c_2-\gamma}{\delta} \binom{a-c_2}{c_1-\gamma-\delta} (v-1)^\delta v^{c_1-\gamma-\delta} \right]. \tag{3.8}$$

The probability that s_1 and s_2 have exactly γ common values follows from (3.8) and (3.4). ■

Corollary 2

$$\Pr[s_1 \sqsubseteq s_2] = \Pr_{common}[c_1] = \frac{\binom{c_2}{c_1}}{\binom{a}{c_1} v^{c_1}}.$$

Proof. The probability that $s_1 \sqsubseteq s_2$ is the probability that all c_1 values of s_1 also appear in s_2. ■

We now focus on $\Pr[B_\alpha(s_1) \subseteq B_\alpha(s_2) \mid s_1 \sqsubseteq s_2]$ and $\Pr[B_\alpha(s_1) \subseteq B_\alpha(s_2)]$. The main insight is to split the hashes into two categories. First, consider an attribute value common to subscriptions s_1 and s_2. If, for this value, there is a hash function h, which is randomly chosen for both Bloom filters $B_\alpha(s_1)$ and $B_\alpha(s_2)$, we say there is a *good hash* between s_1 and s_2. Second, it is also possible that unrelated hash functions selected by both s_1 and s_2 randomly hash at the same position in the Bloom filters; this is called a *lucky hash*. Since we want $B_\alpha(s_1) \subseteq B_\alpha(s_2)$, all the hashes of s_1 in $B_\alpha(s_1)$ must be covered by the hashes of s_2 in $B_\alpha(s_2)$ by *good* and/or *lucky hashes*.

Let $\Pr_{\text{good}}[g, \gamma]$ be the probability that there are g *good hashes* between s_1 and s_2, where γ is the number of common values between both subscriptions. $\Pr_{\text{good}}[g, \gamma]$ is difficult to calculate for $\alpha > 1$ because, for instance, the probability that there are two good hashes for one common value between s_1 and s_2 is different from the probability that there are two common values with one good hash each. Thus, to evaluate $\Pr_{\text{good}}[g, \gamma]$ we must consider the integer partitions of g [Andrews, 1998].

Let $\text{IntPart}(g, \gamma, \alpha_{min}, \alpha)$ be the set of all integer partitions of g into exactly γ parts with the additional constraint that the size of each part is at least α_{min} and at most α. We write $\lambda \vdash \text{IntPart}(g, \gamma, \alpha_{\min}, \alpha)$ to denote that λ is an integer partition of g with the desired properties. The frequency representation of a partition $\lambda \vdash \text{IntPart}(g, \gamma, \alpha_{\min}, \alpha)$ is $(\alpha_{\min}{}^{p_{\alpha_{\min}}} \alpha_{\min+1}{}^{p_{\alpha_{\min+1}}} \dots \alpha^{p_\alpha})$, meaning that the value α_i appears p_{α_i} times in the partition.

Lemma 3

$$\Pr_{good}[g, \gamma] = \sum_{\lambda \vdash IntPart(g,\gamma,\alpha_{min},\alpha)} \left[\binom{\gamma}{p_{\alpha_{min}}, p_{\alpha_{min+1}}, \dots, p_\alpha} \cdot \prod_{\beta=\alpha_{min}}^{\alpha} \left[\frac{\binom{\alpha}{\beta}\binom{k-\alpha}{\alpha-\beta}}{\binom{k}{\alpha}} \right]^{p_\beta} \right]$$

where $\alpha_{min} = \max(0, 2\alpha - k)$.

Proof. We first explain why we must consider all the partitions $\lambda \vdash \text{IntPart}(g, \gamma, \alpha_{\min}, \alpha)$ (we recall that $\text{IntPart}(g, \gamma, \alpha_{\min}, \alpha)$ is the set of all integer partitions of g into exactly γ parts with the additional constraint that the size of each part is at least $\alpha_{\min}$ and at most α). The g *good hashes* must be chosen among the hash functions of the γ common values between s_1 and s_2, thus we consider the partitions of g into exactly γ parts. Since there are α hashes per attribute value, each part of the partitions is at most α. Furthermore, when $\alpha > \frac{k}{2}$, it is not possible that the hashes chosen by s_1 and s_2 for a common value are mutually exclusive. More precisely, the number of good hashes per common value is at least $\alpha_{\min} = \max(0, 2\alpha - k)$, which explains why the size of each part in the partitions is at least $\alpha_{\min}$. The reason to consider all such partitions is that they have different probabilities of occurrence when $\alpha > 1$.

As mentioned before, we consider a fixed partition $\lambda \vdash \text{IntPart}(g, \gamma, \alpha_{\min}, \alpha)$ using the frequency representation

$$(\alpha_{\min}{}^{p_{\alpha_{\min}}} \alpha_{\min+1}{}^{p_{\alpha_{\min+1}}} \dots \alpha^{p_\alpha}),$$

meaning that the value α_i appears p_{α_i} times in λ. The number of ways to assign the number of good hashes for each common value for the fixed λ corresponds to the number of different permutations of γ objects, where there are p_{α_i} indistinguishable objects of type α_i for $\alpha_{min+1} \leq \alpha_i \leq \alpha$. This is given by the multinomial coefficient

$$\binom{\gamma}{p_{\alpha_{\min}}, p_{\alpha_{min+1}}, \ldots, p_{\alpha}} \triangleq \frac{\gamma!}{p_{\alpha_{\min}}! \cdot p_{\alpha_{min+1}}! \cdot \ldots \cdot p_{\alpha}!}. \tag{3.9}$$

We now fix the number of *good hashes* for each attribute and calculate the probability of occurrence of this assignment. The probability of having β good hashes (and $\alpha - \beta$ lucky hashes) for a common value between both subscriptions is given by

$$\frac{\binom{\alpha}{\beta}\binom{k-\alpha}{\alpha-\beta}}{\binom{k}{\alpha}} \tag{3.10}$$

and it follows that the probability of having g *good hashes* for a given number of good hashes per common value is given by

$$\prod_{\beta=\alpha_{\min}}^{\alpha} \left[\frac{\binom{\alpha}{\beta}\binom{k-\alpha}{\alpha-\beta}}{\binom{k}{\alpha}} \right]^{p_\beta}. \tag{3.11}$$

From (3.10) and (3.11), the probability of having g *good hashes* for a fixed partition λ is

$$\binom{\gamma}{p_{\alpha_{\min}}, p_{\alpha_{min+1}}, \ldots, p_{\alpha}} \prod_{\beta=\alpha_{\min}}^{\alpha} \left[\frac{\binom{\alpha}{\beta}\binom{k-\alpha}{\alpha-\beta}}{\binom{k}{\alpha}} \right]^{p_\beta} \tag{3.12}$$

and by considering all partitions $\lambda \vdash \text{IntPart}(g, \gamma, \alpha_{\min}, \alpha)$ we can conclude that

$$\Pr_{\text{good}}[g, \gamma] = \sum_{\lambda \vdash \text{IntPart}(g,\gamma,\alpha_{\min},\alpha)} \binom{\gamma}{p_{\alpha_{\min}}, p_{\alpha_{min+1}}, \ldots, p_{\alpha}} \prod_{\beta=\alpha_{\min}}^{\alpha} \left[\frac{\binom{\alpha}{\beta}\binom{k-\alpha}{\alpha-\beta}}{\binom{k}{\alpha}} \right]^{p_\beta}. \tag{3.13}$$

■

The formula is much simpler when $\alpha = 1$, since there is only one integer partition of g into γ parts when each part is either 0 or 1. This is described by the following corollary.

Corollary 4 *When $\alpha = 1$,*

$$\Pr_{good}[g, \gamma] = \binom{\gamma}{g} \left(\frac{1}{k}\right)^g \left(\frac{k-1}{k}\right)^{\gamma-g}.$$

Proof. When $\alpha = 1$, the problem is a direct application of the binomial distribution with g successes out of γ events and probability of success $\frac{1}{k}$. ■

If $B_\alpha(s_1) \subseteq B_\alpha(s_2)$, all the Bloom filter bits derived from the hash functions randomly chosen for s_1 must somehow be covered by some of the hash functions randomly chosen for the Bloom filter of s_2. If we have g *good hashes* between s_1 and s_2, that leaves $c_1\alpha - g$ *lucky hashes* of s_1 that must be covered by some of the hash functions of s_2 randomly hashing at the same positions in the Bloom filter. Let $\Pr_{\text{lucky}}[h_1, h_2]$ be the probability that h_1 hashes of a subscription or publication are covered by some of the h_2 hashes of a second subscription or publication randomly hashing at the same positions in the Bloom filter.

Lemma 5

$$\Pr_{lucky}[h_1, h_2] = \frac{1}{n^{h_1+h_2}} \sum_{i=0}^{h_2} \left[\binom{n}{i} i! \left\{ {h_2 \atop i} \right\} \sum_{j=0}^{i} \binom{i}{j} j! \left\{ {h_1 \atop j} \right\} \right].$$

where $\left\{ {k_1 \atop k_2} \right\}$ *represent the Stirling numbers of the second kind [Graham et al., 1994].*

Proof. The probability that a Bloom filter of size n with c constraints and α hash functions per constraint has exactly i nonzero bits is

$$\frac{\binom{n}{i} \cdot i! \cdot \left\{ {c\alpha \atop i} \right\}}{n^{c\alpha}} \tag{3.14}$$

More precisely, there are $\binom{n}{i}$ distinct Bloom filters with i nonzero bits, each occurring with probability

$$\frac{i! \cdot \left\{ {c\alpha \atop i} \right\}}{n^{c\alpha}}. \tag{3.15}$$

We prove the lemma with two subscriptions s_1 and s_2. We remove the bits of $B_\alpha(s_1)$ obtained from the *good hashes* between s_1 and s_2 and only consider the h_1 *lucky hashes* that need to be randomly covered by the h_2 hashes of s_2. Let $B'_\alpha(s_1)$ be the Bloom filter of s_1 with only the h_1 lucky hash functions.

$$\Pr_{\text{lucky}}[h_1, h_2] = \sum_{i=1}^{h_2} \sum_{B'_\alpha(s_2) \text{ with } i \text{ nonzero bits}} \Pr[B'_\alpha(s_2)] \cdot \Pr[B'_\alpha(s_1) \subseteq B'_\alpha(s_2)] \tag{3.16}$$

and from (3.14) and (3.15) it follows that

$$\begin{aligned}
\Pr_{\text{lucky}}[h_1, h_2] &= \sum_{i=1}^{h_2} \sum_{\delta=1}^{\binom{n}{i}} \left[\frac{i! \cdot \left\{ {h_2 \atop i} \right\}}{n^{h_2}} \sum_{j=1}^{i} \frac{\binom{n}{j} \cdot j! \cdot \left\{ {h_1 \atop j} \right\}}{n^{h_1}} \cdot \frac{\binom{i}{j}}{\binom{n}{j}} \right] \\
&= \sum_{i=1}^{h_2} \left[\frac{\binom{n}{i} \cdot i! \cdot \left\{ {h_2 \atop i} \right\}}{n^{h_2}} \sum_{j=1}^{i} \frac{\binom{n}{j} \cdot j! \cdot \left\{ {h_1 \atop j} \right\}}{n^{h_1}} \cdot \frac{\binom{i}{j}}{\binom{n}{j}} \right] \\
&= \frac{1}{n^{h_1+h_2}} \sum_{i=1}^{h_2} \left[\binom{n}{i} \cdot i! \cdot \left\{ {h_2 \atop i} \right\} \sum_{j=1}^{i} \binom{i}{j} \cdot j! \cdot \left\{ {h_1 \atop j} \right\} \right].
\end{aligned} \tag{3.17}$$

■

Lemma 6

$$Pr[B_\alpha(s_1) \subseteq B_\alpha(s_2) \mid s_1 \sqsubseteq s_2] = \sum_{g=0}^{c_1\alpha} \left(\Pr_{good}[g, c_1] \Pr_{lucky}[c_1\alpha - g, c_2\alpha] \right).$$

Proof. Since $s_1 \sqsubseteq s_2$, there are c_1 common attribute values between s_1 and s_2. The probability that $B_\alpha(s_1) \subseteq B_\alpha(s_2)$ can be obtained by summing, for all values of g between 0 and $c_1\alpha$, the probability of having g *good hashes* times the probability that the $c_1\alpha - g$ *lucky*

hashes of s_1 can be covered by the $c_2\alpha$ hash functions of s_2 randomly hashing at the same place in the Bloom filter of s_2. ■

Lemma 7

$$\Pr[B_\alpha(s_1) \subseteq B_\alpha(s_2)] = \sum_{\gamma=0}^{c_1} \left(\Pr_{common}[\gamma] \sum_{g=0}^{\gamma\alpha} \left(\Pr_{good}[g,\gamma] \Pr_{lucky}[c_1\alpha{-}g, c_2\alpha] \right) \right).$$

Proof. Suppose that there are γ common attribute values between s_1 and s_2. The probability that $B_\alpha(s_1) \subseteq B_\alpha(s_2)$ given γ can be obtained by summing, for all values of g between 0 and $\gamma\alpha$, the probability of having g *good hashes* times the probability that the $c_1\alpha{-}g$ *lucky hashes* of s_1 can be covered by the $c_2\alpha$ hash functions of s_2 randomly hashing at the same place in the Bloom filter of s_2. The probability that $B_\alpha(s_1) \subseteq B_\alpha(s_2)$ can be obtained by summing, for all values of γ between 0 and c_1, the probability that there are γ common attribute values between s_1 and s_2 times the probability that $B_\alpha(s_1) \subseteq B_\alpha(s_2)$ given γ. ■

We are now ready to prove the main result.

Theorem 1

$$P_\alpha = \frac{\left(\binom{a}{c_1} v^{c_1}\right)^{-1} \binom{c_2}{c_1} \sum_{g=0}^{c_1\alpha} \left(\Pr_{good}[g,c_1] \Pr_{lucky}[c_1\alpha{-}g, c_2\alpha] \right)}{\sum_{\gamma=0}^{c_1} \left(\Pr_{common}[\gamma] \sum_{g=0}^{\gamma\alpha} \left(\Pr_{good}[g,\gamma] \Pr_{lucky}[c_1\alpha{-}g, c_2\alpha] \right) \right)}.$$

Proof. The theorem can be derived by merging (3.2), Lemmas 1, 3, 5, 6, 7, and Corollaries 2 and 4. ■

Theorem 2

$$P_{f\_pos} = \sum_{\gamma=0}^{c_1-1} \left(\Pr_{common}[\gamma] \Pr_{lucky}[\alpha(c_1{-}\gamma), c_p k] \right).$$

Proof. From (3.3), $P_{\text{f\_pos}} \triangleq \Pr[B_\alpha(s_1) \subseteq B_k(p) \mid s_1 \not\sqsubseteq p]$. Suppose that there are $\gamma < c_1$ common attribute values between s_1 and p. Since the publication uses all the k hash functions for each value in its Bloom filter, it follows that there are $\alpha\gamma$ *good hashes* for s_1, leaving $\alpha(c_1{-}\gamma)$ *lucky hashes* that must be covered by the $c_p k$ hashes of p randomly hashing at the same positions in the Bloom filters. The probability of this occurring is given by $\Pr_{\text{lucky}}[\alpha(c_1{-}\gamma), c_p k]$. The result can be obtained by summing, for all values of γ between 0 and c_1 ($s_1 \sqsubseteq p$ if $\gamma = c_1$), the probability that there are γ common attribute values between s_1 and p times the probability of $\alpha(c_1{-}\gamma)$ *lucky hashes*. ■

3.4.5 Other attacks

An attacker can also try to derive containment relationships based on mismatched Bloom filters, i.e., Bloom filters $B_\alpha(s_1)$ and $B_\alpha(s_2)$ such that $B_\alpha(s_1) \not\subseteq B_\alpha(s_2)$ and $B_\alpha(s_2) \not\subseteq B_\alpha(s_1)$.

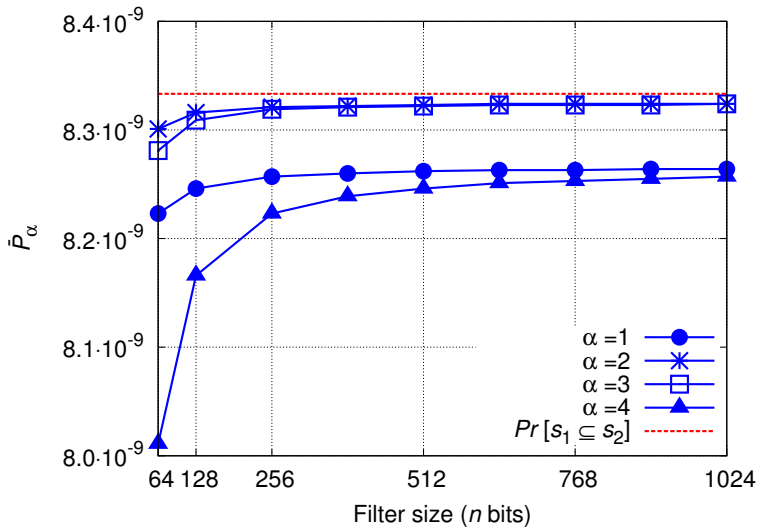

Figure 3.7: $\overline{P}_\alpha$ for Bloom filters of varying size ($c_1 = c_2 = 3$).

Let us consider the probability $\overline{P}_\alpha \triangleq \Pr\left[s_1 \sqsubseteq s_2 \mid B_\alpha(s_1) \not\subseteq B_\alpha(s_2)\right]$, which from Bayes' law can be rewritten as

$$\begin{aligned}\overline{P}_\alpha &= \frac{\Pr[s_1 \sqsubseteq s_2]\cdot\Pr[B_\alpha(s_1) \not\subseteq B_\alpha(s_2) \mid s_1 \sqsubseteq s_2]}{\Pr[B_\alpha(s_1) \not\subseteq B_\alpha(s_2)]} \\ &= \frac{\Pr[s_1 \sqsubseteq s_2]\cdot(1-\Pr[B_\alpha(s_1) \subseteq B_\alpha(s_2) \mid s_1 \sqsubseteq s_2])}{1-\Pr[B_\alpha(s_1) \subseteq B_\alpha(s_2)]}.\end{aligned} \tag{3.18}$$

$\overline{P}_\alpha$ can be evaluated using the formulas derived earlier in this section. It should be clear that $\overline{P}_k = 0$ and that $\overline{P}_\alpha > 0$ for $\alpha < k$, so at first sight randomly choosing some of the hash functions appears to provide more information to a potential attacker than selecting all the hash functions. This is misleading, since $\overline{P}_\alpha$ essentially increases from zero to $\Pr[s_1 \sqsubseteq s_2]$, providing no more information to an attacker than from random subscriptions. This is a welcome tradeoff considering how the hashing strategy we use decreases the amount of information leaked from contained subscriptions. An example comparing $\overline{P}_\alpha$ and $\Pr[s_1 \sqsubseteq s_2]$ is displayed in Figure 3.7.

Finally, although it is unclear what could be done with such information, an attacker could try to derive "noncontainment" attributes about pairs of encrypted subscriptions. In that case, the probability of interest is $\Pr\left[s_1 \not\sqsubseteq s_2 \mid B_\alpha(s_1) \not\subseteq B_\alpha(s_2)\right]$, which with Bayes' law and a little work can be rewritten as

$$\Pr[s_1 \not\sqsubseteq s_2 \mid B_\alpha(s_1) \not\subseteq B_\alpha(s_2)] = 1-\frac{\Pr[s_1 \sqsubseteq s_2]\cdot\Pr[B_\alpha(s_1) \not\subseteq B_\alpha(s_2) \mid s_1 \sqsubseteq s_2]}{\Pr[B_\alpha(s_1) \not\subseteq B_\alpha(s_2)]}. \tag{3.19}$$

Interestingly, $\Pr\left[s_1 \not\sqsubseteq s_2 \mid B_\alpha(s_1) \not\subseteq B_\alpha(s_2)\right] = 1$ with $\alpha = k$ and smaller than 1 when $\alpha < k$. This means that when $\alpha < k$, the attacker cannot even conclude with certainty that two subscriptions are not included into each other.

3.5 Evaluation and discussion

In this section, we evaluate the performance of the prefiltering strategy we proposed under a wide range of scenarios. The evaluation when containment information is available is presented in Section 3.5.1, whereas the performance-confidentiality tradeoffs with containment obfuscation are covered in Section 3.5.2.

3.5.1 Prefiltering performance

First, we experimentally study the efficiency of prefiltering. We focus on the reduction in the number of calls to the ENC-MATCH encrypted matching function with the different variants of prefiltering. Afterwards, we study the effect of varying the size of the Bloom filters, and the number of hash functions on the effectiveness of all prefiltering variants. Finally, we evaluate the actual performance gains on a complete implementation of the matching engine using ASPE [Choi et al., 2010] as the encrypted matching operation.

Workloads

We constructed an experimental workload inspired by the one used for the evaluation of the Meghdoot publish/subscribe system [Gupta et al., 2004]. We gathered five years of quotes for 200 randomly selected stocks from the Yahoo! finance website,[6] which corresponds to a set of over 250,000 publications with between 8 and 11 attributes. We built synthetic subscriptions based on the same categories as in [Gupta et al., 2004]. These subscriptions contain a variety of equality and range constraints on the attributes of stock quotes, namely the symbol, date, exchanged volume, and daily statistics on their price (open, close, high, low). By default, equality constraints are expressed on the symbol, while ranges apply to the volume and the various daily statistics. For workloads with additional equality constraints, we used extra attributes from the Yahoo! finance data: the stock exchange on which the stock is available; the trend indicating if the stock is higher or lower than the previous day; and the type of product (among 8 possible categories).

We also experimented with workloads containing more attributes than available in the original Yahoo! finance data. To that end, we simply created additional copies of the volume and daily statistics attributes by merging data from multiple quotes. We produced workloads with twice and four times more attributes on which subscriptions can express range constraints.

We finally experimented with different subscription distributions, by choosing constraint values either uniformly at random, or according to a Zipf's law with exponent $s = 1$.[7] Recall that publications are real data from Yahoo! finance and we did not modify their distribution in any way.

Table 3.2 describes the nine workloads used in the evaluation. For every type, we experimented with populations of up to 100,000 subscriptions.

Table 3.3 presents characteristics of the workloads that are instrumental in the interpretation of experimental results. For each workload of 100,000 subscriptions, we created a containment graph. The second column indicates the number of nodes in the graph, with identical subscriptions being collapsed in a single node. Columns 3–5 respectively show the

[6] `http://finance.yahoo.com/`.

[7] Zipf's law and other power laws appear in a wide range of disciplines like finance, demography, biology, networks, ... [Newman, 2005].

Workload name	Number of equality constraints	Number of attributes	Distribution of values
e100	100% : 1 eq. pred.	8–11 (default)	Uniform
e80	20% : 0 eq. pred. 80% : 1 eq. pred.	8–11 (default)	
e80-2x		2× more	
e80-4x		4× more	
e+-2x	15% : 0 eq. pred. 60% : 1 eq. pred.	2× more	
e+-4x	15% : 2 eq. pred. 10% : 3 eq. pred.	4× more	
e80-z	20% : 0 eq. pred.	8–11 (default)	Zipf on symbol
e80-z+	80% : 1 eq. pred.		Zipf on all attributes
e100-z+	100% : 1 eq. pred.		

Table 3.2: Description of the workloads.

Workload	Nodes	Roots	Leaves	R ∩ L	Depth			Children		Parents		Identical		constraints			Equality			Matches
					μ^-	μ^+	max	μ	max	μ	max	μ	max	μ	min	max	μ	min	max	μ (%)
e100	89102	1688	3839	27	53.04	53.80	192	1.47	13	1.44	10	1.12	5	2.20	2	3	1.00	1	1	0.19
e80	79786	45	3504	0	78.41	298.29	2056	3.02	75	2.89	17	1.25	28	2.00	1	3	0.80	0	1	7.48
e80-2x	99944	1705	21812	320	3.90	27.73	106	10.64	336	8.46	49	1.00	2	3.20	2	5	0.80	0	1	2.78
e80-4x	100000	41315	78624	32229	1.61	2.09	9	16.52	307	6.02	63	1.00	1	5.60	4	9	0.80	0	1	0.40
e+-2x	99980	1956	36416	621	3.76	23.68	88	12.25	573	7.94	43	1.00	2	3.60	2	7	1.21	0	3	2.14
e+-4x	100000	46635	83596	38793	1.55	1.93	9	17.60	344	5.41	59	1.00	1	5.99	4	11	1.20	0	3	0.30
e80-z	63671	31	2821	0	118.70	309.75	2019	3.27	76	3.13	18	1.57	29	2.00	1	3	0.80	0	1	7.49
e80-z+	43059	23	2453	1	59.91	205.96	483	2.92	201	2.75	19	2.32	1015	2.00	1	3	0.80	0	1	8.93
e100-z+	46125	1148	2648	32	45.30	54.70	477	1.57	11	1.52	11	2.17	898	2.20	2	3	1.00	1	1	0.21

Table 3.3: Workload characteristics for 100,000 subscriptions (matches evaluated on 1,000 publications).

number of nodes that are root, leaf, or both (i.e., isolated nodes) in the graph. A node may be reachable from several roots and by multiple paths. Column 6 presents the average shortest (μ^-) and longest (μ^+) paths from some root to each node, as well as the maximum path length. Column 7 gives the average and maximum number of children at each node (excluding leaves), while Column 8 provides the same statistics for parents of each node (excluding roots). Columns 9, 10, and 11 respectively list the number of identical subscriptions in the workloads, the number of constraints in the subscriptions, and the number of equality constraints. Finally, the last column specifies the percentage of matching subscriptions for each of the considered workloads, averaged over a set of 1,000 publications.

Reducing the number of tests

We first study the number of calls to the ENC-Match encrypted matching function. We express this value as a "tests ratio" over the number of subscriptions. A value of 1 indicates that a call to ENC-Match must be performed for each subscription, as would be the case with a naive approach consisting in traversing the whole collection of subscriptions to test each one individually.

We evaluate the following algorithms:

- `CB`: Containment-based strategy (baseline).
- `PF`: Prefiltering with complete Bloom filters.
- `PF-TB`: Prefiltering with truncated Bloom filters, using $\alpha = 1$ out of $k = 3$ hash functions.

- `PF+NM`: PF combined with containment-based negative matching.
- `PF+NM+PM`: PF combined with containment-based negative and positive matching.

We use Bloom filters of 128 bits. The reason for using Bloom filters of this dimension is that, as results show, this size is sufficient to provide good results for the considered workloads.

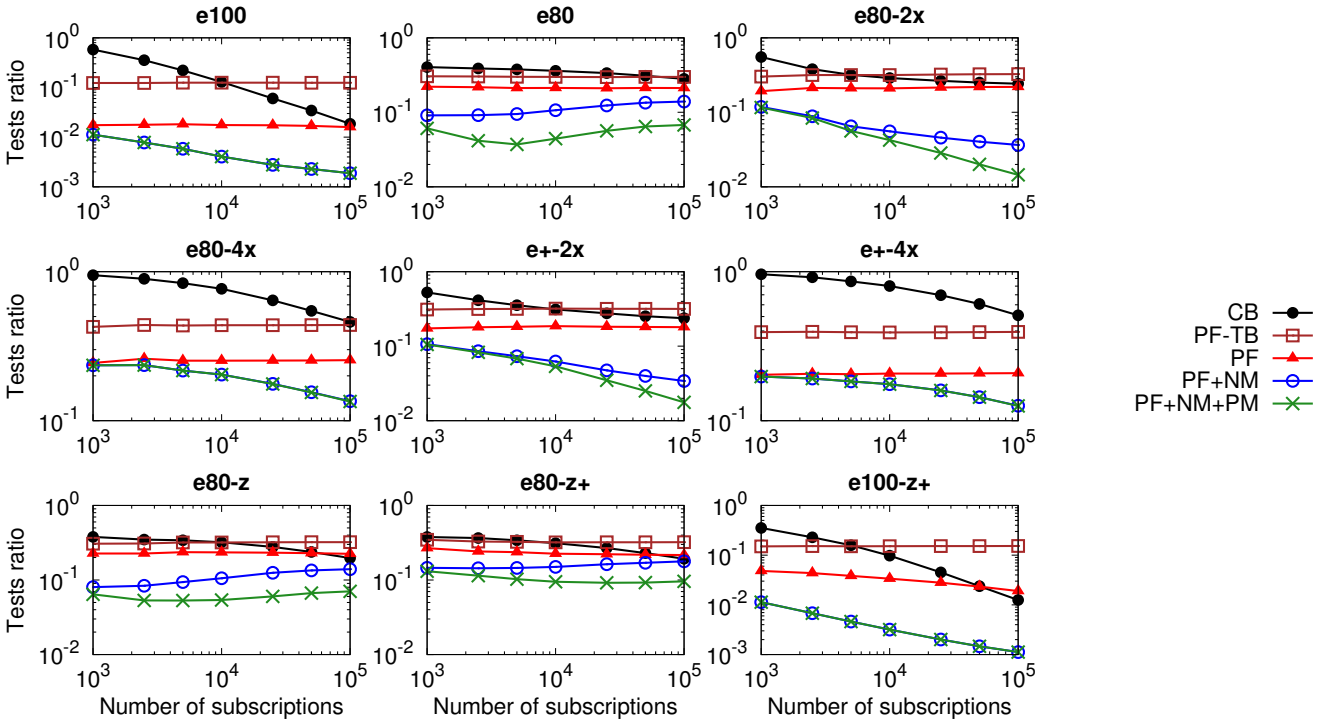

Figure 3.8: Evolution of the number of tests, i.e., calls to the encrypted matching function, with different populations of subscribers (128-bit Bloom filters and 3 hash functions).

Results are shown in Figure 3.8 (note the logarithmic scales). Let us first consider the baseline `CB` strategy. One can observe that the most important savings can be obtained with workloads that have few matches (`e100`, `e80-4x`, `e+-4x`, and `e100-z+`, all of which report less than 1% of matches in Table 3.3) because many subscriptions can be discarded early while traversing the containment graph. With other workloads, the performance of `CB` still improves with the number of subscriptions, albeit at a lower rate.

Both `PF` and `PF-TB` exhibit almost constant performance independent of the number of subscriptions. The reason is that savings depend in these cases only on negative matching obtained from subscriptions Bloom filters, which do not have an impact on the matching of other subscriptions as containment is not exploited. The only noticeable exception is with `e80-z+` and `e100-z+` where `PF` slightly improves. This is due to the fact that `PF` collapses identical subscriptions (which reach up to 1,000 in these workloads) into a single node in the lists, while `PF-TB` does not (recall that `PF-TB` is designed for containment obfuscation, thus it does not allow to determine subscriptions equality).

`PF` is significantly more efficient than `PF-TB` because truncation adds false positives. The difference between both strategies is more important for workloads that report few matches. It is also worth pointing out that `PF-TB` it more efficient that `CB` on most of the workloads up to large populations of subscriptions. This is notably the case of workloads with shallow containment graphs (`e80-4x` and `e+-4x`, with average node depth lower than 2).

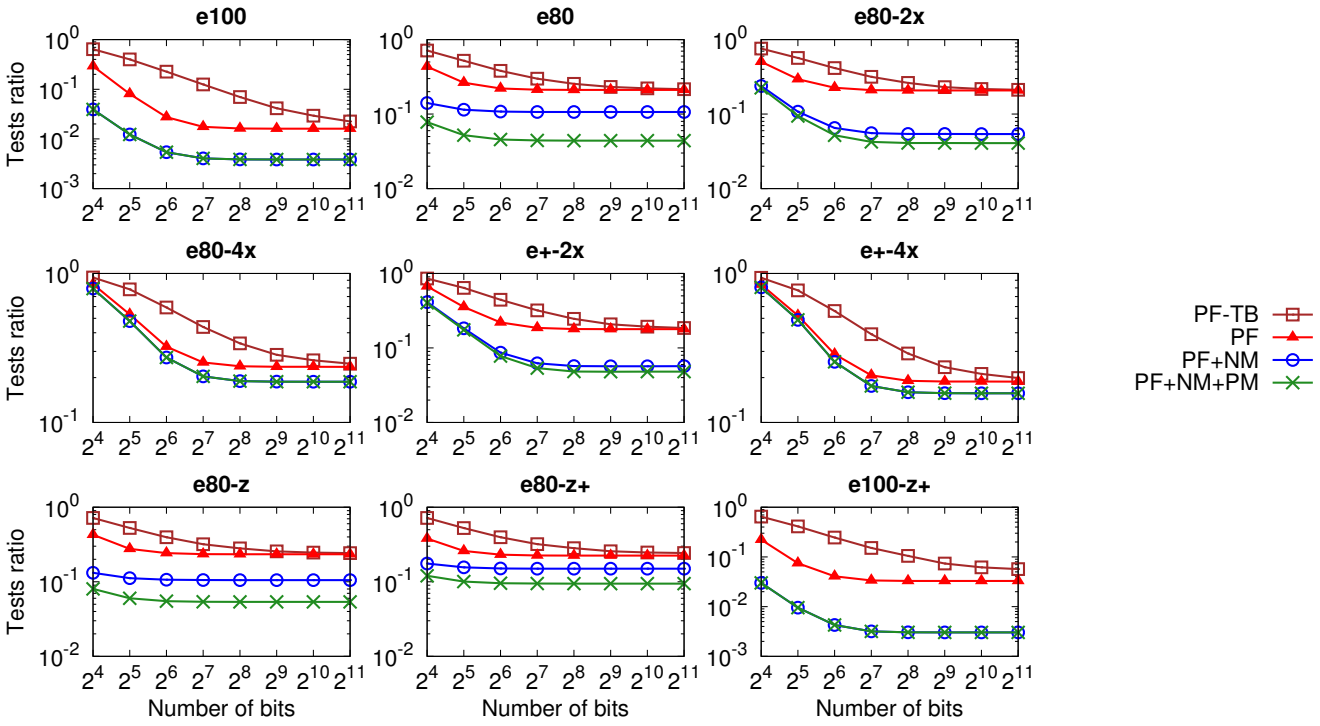

Figure 3.9: Evolution of the number of tests when varying the size of the Bloom filters, using k =3 hash functions.

Finally, the `PF+NM` and `PF+NM+PM` strategies perform best in all the experiments. They combine the benefits of `PF` and `CB`: like the former, they perform already well with few subscriptions; like the latter, they can take advantage of the large number of containment relationships available with many subscriptions. Unsurprisingly, positive matching is helpful with workloads that exhibit sufficient matches (`e80`, `e80-2x`, `e+-2x`, `e80-z`, and `e80-z+`, which report between 2% and 9% of matches). Workloads with high depth and a good proportion of subscriptions with no equality constraint (`e80`, `e80-z`, and `e80-z+`) exhibit an interesting trend, with performance initially improving and then slightly degrading as the number of subscription grows. The reason why `PF+NM` and `PF+NM+PM` do not follow the same slope as `CB` is that traversal is performed in containment order within the lists, but not globally in the whole graph. Consider two subscriptions s and s' with $s \sqsupseteq s'$ with s containing s'. The containee s' may have additional equality constraints and, therefore, belong to more lists in the prefilter structure than s. If s' is first encountered as part of a list to which s does not belong, the containee may be tested before its container against a publication p. If neither matches p, `PF+NM` would still test both subscriptions while `CB` would only test s as it strictly follows the global containment graphs. Note that despite the difference in slopes, `CB` does not catch up `PF+NM` nor `PF+NM+PM` in any of the workloads up for to 100,000 subscriptions, and the gains of the latter are in most cases close to one order of magnitude.

Tuning Bloom filters

We further evaluate the impact of varying the Bloom filter size and the number of hash functions on the number of tests that have to be performed using the costly ENC-Match operation.

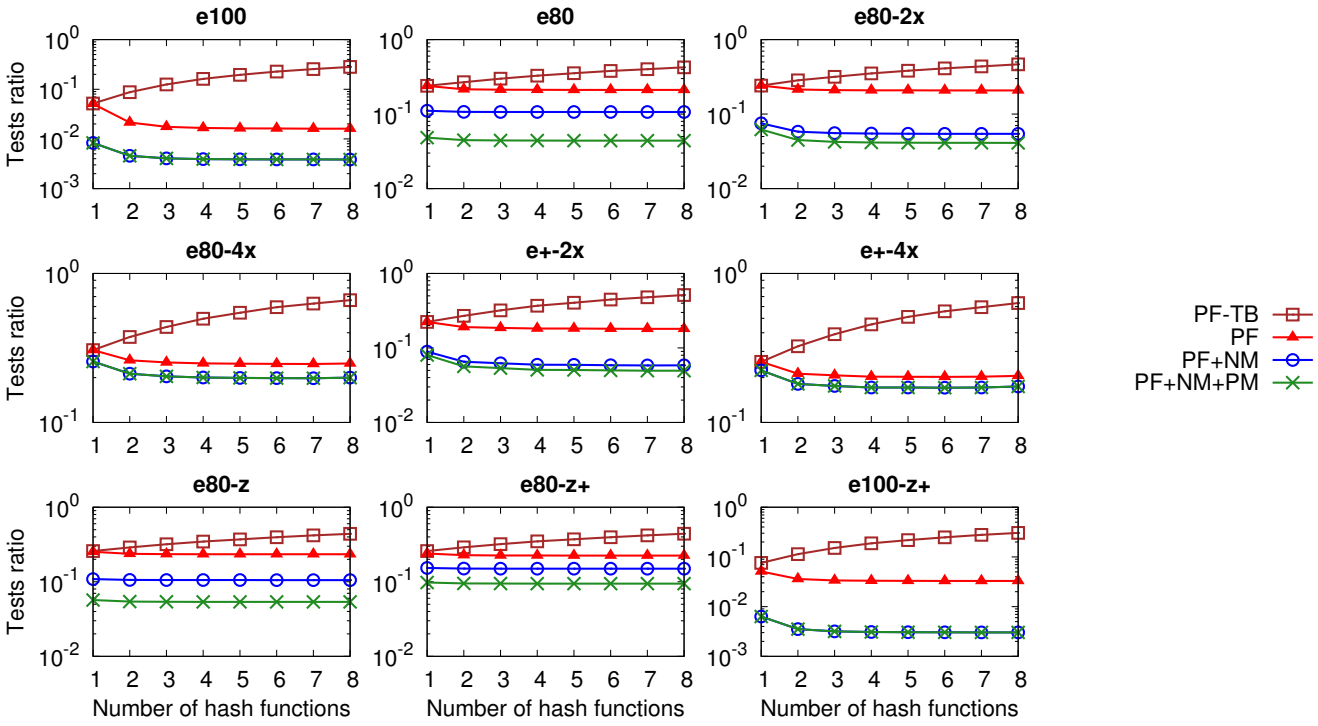

Figure 3.10: Evolution of the number of tests when varying the number of hash functions, using n =128-bit Bloom filters.

Figure 3.9 shows the tests ratios as function of the size of the Bloom filters. In Figure 3.10 we count the same test ratio when we vary the number of hash functions.

A first observation is that the default values we have chosen (128-bit Bloom filters and 3 hash functions) perform well with all workloads and most algorithm variants. The benefits one can expect from larger Bloom filters and more hash functions are not significant, although for workloads with more attributes and equality constraints one should likely increase the size of the Bloom filters.

`PF-TB` is the exception: its efficiency continues to increase with larger Bloom filters, reaching almost the performance of `PF` with 2,048 bits. This shows that the higher rate of false positives resulting from truncation can be compensated by properly dimensioning the Bloom filters. Conversely, with more hash functions, the performance of `PF-TB` degrades faster than the performance of `PF` because the number of bits in the Bloom filters of the subscriptions does not change (only one hash function is used), while the Bloom filters of the publications become more densely populated, yielding additional false positives.

Improving filtering time

To complete the performance evaluation, we present the impact of prefiltering on the time complexity of confidentiality-aware pub/sub. We use for encrypted matching engine our implementation of ASPE [Choi et al., 2010]. The goal is to verify how the reduction in calls to the ENC-MATCH function affects the actual filtering time. To that end, we have executed the same experiment as in Figure 3.8 on a 2.4 GHz Xeon with 64 GB RAM. Figure 3.11 reports the average time necessary to filter a publication with all considered workloads and algorithm variants, with the addition of a "naive" strategy that simply iterates through the set of subscriptions and tests them one by one using ASPE.

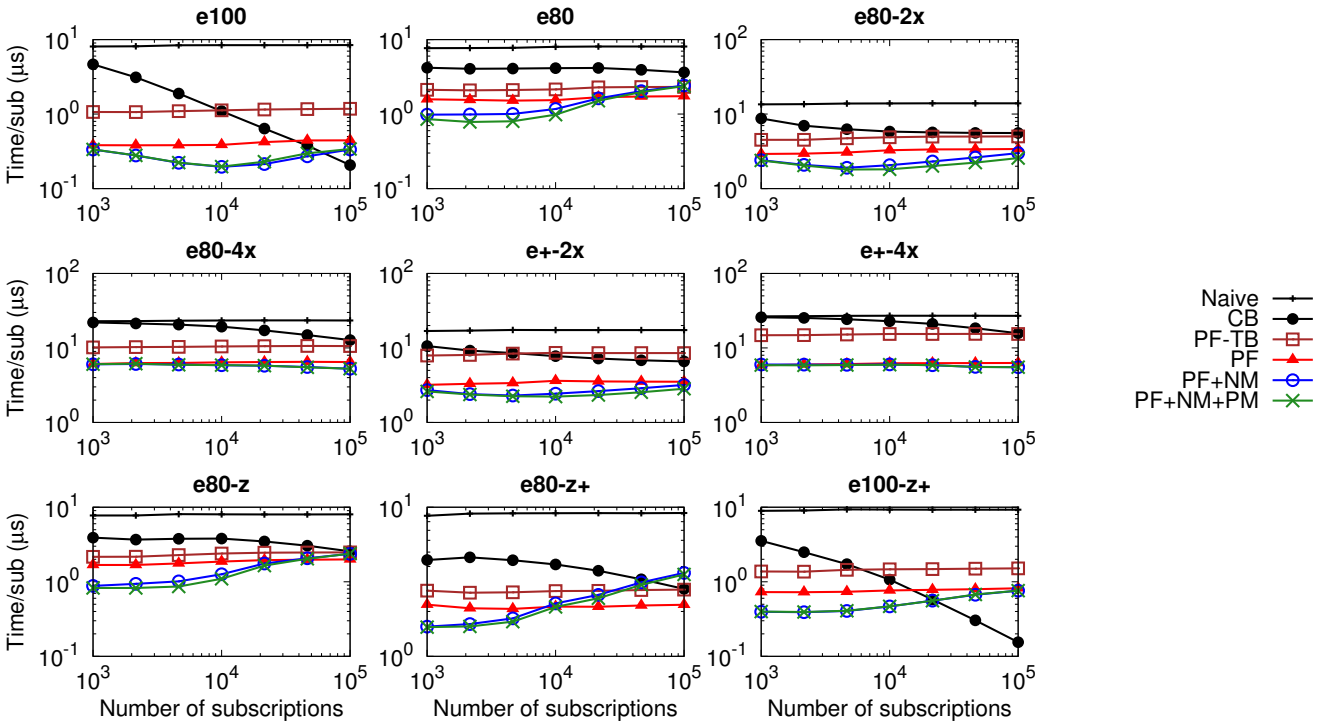

Figure 3.11: Filtering performance using ASPE with different populations of subscribers (n =128-bit Bloom filters and k =3 hash functions).

The gains expected from the evaluation of the tests ratio are mirrored in most of the graphs. While the behavior of the algorithms is consistent with results of Figure 3.8, `PF+NM` and `PF+NM+PM` strategies do not scale as well as anticipated on some workloads, notably when the average depth of nodes is high and many subscriptions have no equality constraint (`e80`, `e80-z`, and `e80-z+`). This can be explained by memory management and cache effects on the test hardware, as well as the prohibitive size of the last list in the prefilter structure. More precisely, even though the ENC-Match function may not be actually called, the whole list must still be traversed; in contrast, exploration of the containment graph can be interrupted early. Despite these implementation artifacts, performance results clearly show that prefiltering provides a significant time reduction of the matching operation.

3.5.2 Performance/confidentiality tradeoffs

We now use Theorems 1 and 2 to study the fundamental tradeoffs between the efficiency and the containment confidentiality of prefiltering using truncation for containment obfuscation. We first consider as an example the uniform domain with $a = 10$ attributes and $v = 100$ values per attribute. We further assume that all the subscriptions contain $c_1 = c_2 = 3$ equality constraints, that all the publications contain $c_p = 5$ values, and that a set with $k = 5$ hash functions is used. Figure 3.12 shows how P_α and $P_{\text{f\_pos}}$ vary with the Bloom filter size. For both P_α and $P_{\text{f\_pos}}$, the five curves correspond to $\alpha \in \{1, 2, 3, 4, 5\}$, where α is the number of hash functions randomly selected out of k. It can be observed that P_α converges to 1 as the Bloom filter size increases. This can be explained by the fact that when Bloom filters are large, the probability that a pair of subscriptions that do not cover each other generates two Bloom filters such that one is included in the other approaches zero. We also notice that P_α converges much slower to 1 with small values of α. This indicates that using truncation

used in conjunction with Bloom filters of reasonable size is an efficient strategy to decrease containment information leakage.

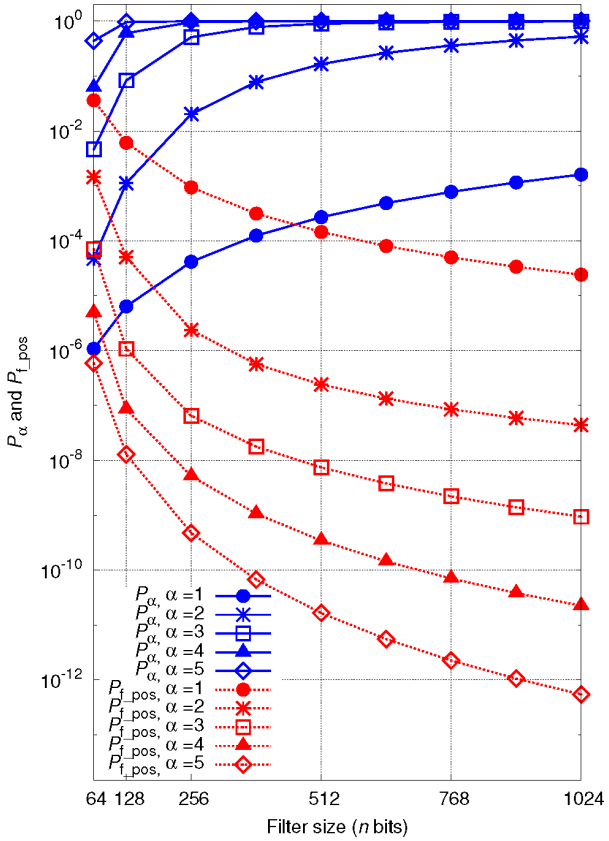

Figure 3.12: $P_\alpha/P_{\text{f\_pos}}$ tradeoff as n increases. Fixed parameters: $a = 10$, $v = 100$, $c_1 = c_2 = 3$, $c_p = 5$ and $k = 5$.

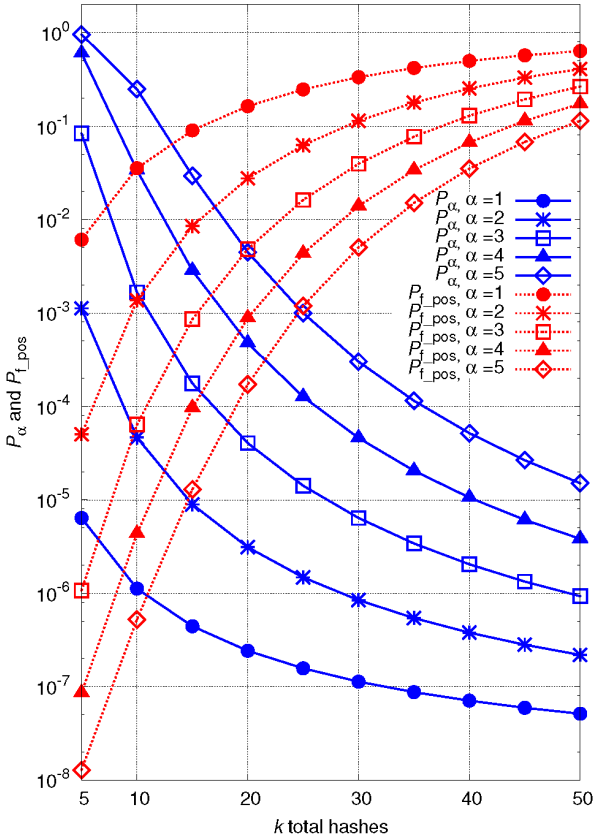

Figure 3.13: $P_\alpha/P_{\text{f\_pos}}$ tradeoff as k increases. Fixed parameters: $a = 10$, $v = 100$, $c_1 = c_2 = 3$, $c_p = 5$ and $n = 128$.

Figure 3.12 also shows that $P_{\text{f\_pos}}$ decreases as n and k increase. Figure 3.13 uses the same domain as Figure 3.12, except that it shows how P_α and $P_{\text{f\_pos}}$ vary with k with Bloom filters of 128 bits. It should be clear from both figures that in order to minimize the amount of information leaked to an attacker, we can choose a small Bloom filter and select one hash function randomly out of a very large set. In fact, we can show from (3.2) and Theorem 1 that

$$\begin{aligned}
&\lim_{k\to\infty} P_\alpha \\
&= \lim_{k\to\infty} \left(\Pr[s_1 \sqsubseteq s_2] \cdot \frac{\Pr[B_\alpha(s_1) \subseteq B_\alpha(s_2) \mid s_1 \sqsubseteq s_2]}{\Pr[B_\alpha(s_1) \subseteq B_\alpha(s_2)]} \right) \\
&= \Pr[s_1 \sqsubseteq s_2] \cdot \lim_{k\to\infty} \frac{\Pr[B_\alpha(s_1) \subseteq B_\alpha(s_2) \mid s_1 \sqsubseteq s_2]}{\Pr[B_\alpha(s_1) \subseteq B_\alpha(s_2)]} \\
&= \Pr[s_1 \sqsubseteq s_2].
\end{aligned} \tag{3.20}$$

With a very large value of k, there is no containment information leaked to an attacker who knows the domain probability space. However in this case, since all the hash functions must be encoded in the Bloom filters for publications, the publications Bloom filters have most

of their bits set to one. The probability of false positive then approaches 1 and prefiltering becomes useless.

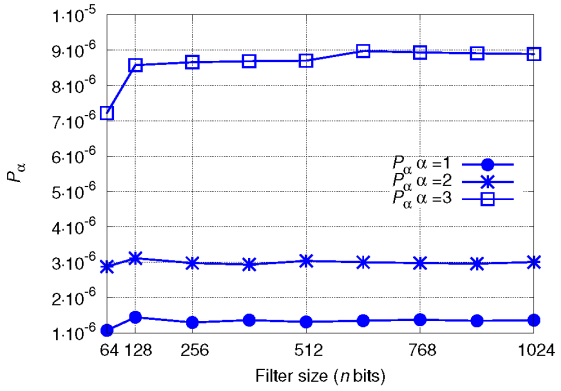

Figure 3.14: Evaluation of P_α when $P_{\text{f\_pos}} \approx 0.03$. Other fixed parameters: $a = 10$, $v = 100$, $c_1 = c_2 = 3$ and $c_p = 5$.

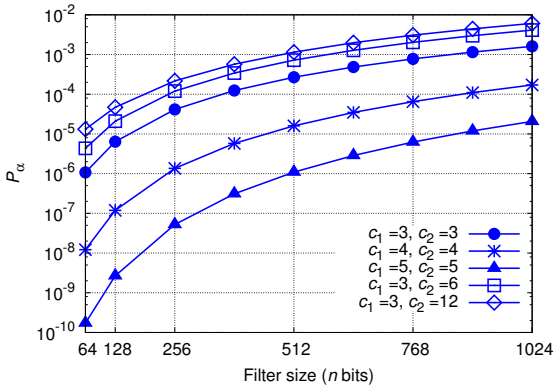

Figure 3.15: Evaluation of P_α for varying n and different c_1 and c_2. Fixed parameters: $a = 10$, $v = 100$, $k = 5$ and $\alpha = 1$.

Figure 3.14 further illustrates the tradeoffs. In Figure 3.14, we choose k such that the false positive rate is always 3% ($P_{\text{f\_pos}} \approx 0.03$). We study how it affects P_α as α and the Bloom filter size vary. The observation is that for a fixed false positive rate, the containment confidentiality of the prefiltering is essentially fixed no matter how we vary the size of the Bloom filters and the cardinality of the hash functions set. One observation that can be drawn from this is that for complexity reasons we should set the Bloom filter size as small as acceptable (for very small Bloom filters it is not possible to set the probability of false positive low enough). Typically, Bloom filters of size 64 or 128 are more than sufficient. Figure 3.14 also shows that the best tradeoff is reached when randomly selecting only $\alpha = 1$ hash function from the set. We mention that for some values of c_1 and c_2 we obtained a slightly better tradeoff for $\alpha = 2$ than for $\alpha = 1$, but we believe that the small gain, the increased complexity (much larger k) and infrequent occurrence do not really justify considering $\alpha > 1$. Despite the observed tradeoffs, the truncating strategy is a valuable mechanism. For instance, it can be observed from Figure 3.14 that for a 3% false positive rate, which is excellent, P_α is approximately $1{\cdot}10^{-6}$. This makes it challenging for an attacker to build an accurate containment graph and run statistical attacks using a set of encrypted subscriptions.

Figure 3.15 shows that the confidentiality of the prefiltering scheme increases dramatically when the number of equality constraints per subscription increases, but decreases slightly if an attacker tries to gain information by comparing a subscription with a small number of constraints with a second subscription with a large number of equality constraints. In Figure 3.16 we again choose k such that the false positive rate is always 3% ($P_{\text{f\_pos}} \approx 0.03$). It shows that the performance/confidentiality tradeoff is better when the number of subscription equality constraints approaches the number of publication values. Of course, in practical systems, subscribers do not necessarily generate subscriptions with a fixed number of equality attributes nor do publishers generate publications with a fixed number of values. When this occurs, the performance/confidentiality tradeoff must be approximated based on an estimation of the system workload.

Finally, since real-world publication and subscription domains are rarely uniform, we also test the prefiltering scheme on a nonuniform domain. We use Zipf's law with exponential parameter 0, i.e., $\Pr[X = x] = \frac{\Pr[X=1]}{x}$ for $x = \{1, 2, 3, \dots\}$. We consider that the domain attributes follow Zipf's law; likewise, we assume that the possible values for each attribute

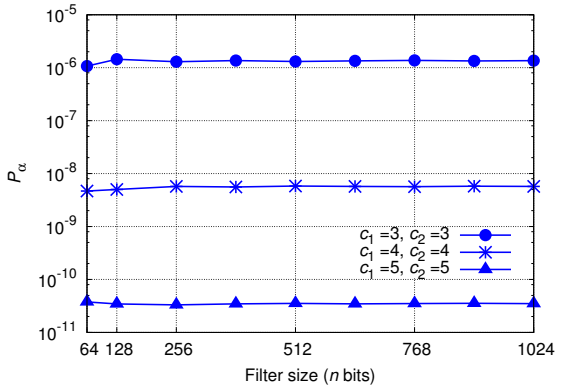

Figure 3.16: Evaluation of P_α when $P_{\text{f\_pos}} \approx 0.03$. Experiment using fixed parameters: $a = 10$, $v = 100$, $c_p = 5$ and $\alpha = 1$.

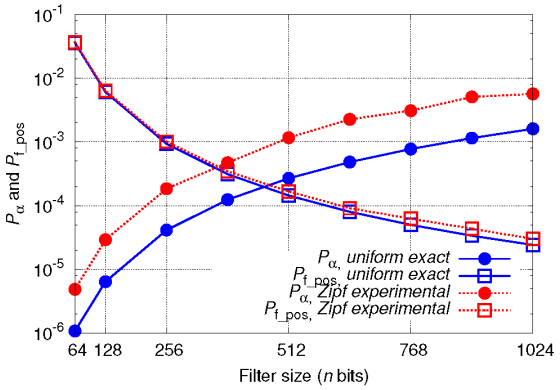

Figure 3.17: Comparison of P_α and $P_{\text{f\_pos}}$ for uniform and non-uniform domains. Numerical simulations using $a = 10$, $v = 100$, $c_1 = c_2 = 3$, $c_p = 5$, $k = 5$ and $\alpha = 1$.

also follow Zipf's law. We use $a = 10$, $v = 100$, 5 values per publication, 3 equality constraints per subscription, $k = 5$ and $\alpha = 1$. We compute P_α and $P_{\text{f\_pos}}$ using numerical simulations. The results are shown in Figure 3.17. It can be observed that P_α is slightly higher when the domain follows Zipf's law, which is not surprising since the domain entropy is lower and the probability that the subscriptions share common values increases. $P_{\text{f\_pos}}$ is almost equal for small Bloom filters. This example provides evidence that the prefiltering scheme we proposed can be made secure and efficient with nonuniform real-life practical workloads.

3.6 Prefiltering extensions

The prefiltering mechanism we presented brings obvious advantages for the performance exhibited by confidentiality-aware pub/sub systems, as evaluated in the previous section. However, one of the major limitations comes from the application restriction to equality constraints. In this section we present an extension that is a first step towards also prefiltering inequality constraints.

Our approach has similarities to a dictionary search technique using reference points. Raiciu *et al.* [Raiciu and Rosenblum, 2006] already used such a technique for ranges filtering in their encrypted matching scheme proposed for pub/sub. We have overviewed their work in Section 2.1.2. The idea follows the generic *pre-mapped equality comparison design* in encrypted matching schemes described in Section 2.1.1. More precisely, the pub/sub data domains are first partitioned in smaller intervals. We use the boundaries of these intervals by encoding them in "inequality Bloom filters" to approximate inequality constraints. An example is given by Figure 3.18. In this example, the domain $V = \{1, 2, \ldots, 100\}$ is partitioned into five intervals, each containing 20 values. If a subscription contains the constraint $v > 25$, the closest interval boundary covering the inequality, in this case ≥ 20, will be encoded in its inequality Bloom filter. Each incoming publication must encode all the interval boundaries covering its values. In the figure, the publication is $v = 30$. The interval bounds ≥ 1, ≥ 20, ≤ 100, ≤ 80, ≤ 60 and ≤ 40 thus must all be encoded in its inequality Bloom filter. The inequality Bloom filters of subscriptions and publications can then be compared exactly like the Bloom filters for equality constraints. This can be performed in a second prefiltering stage executed after the prefiltering on equality constraints. It should be noted that failure to encode

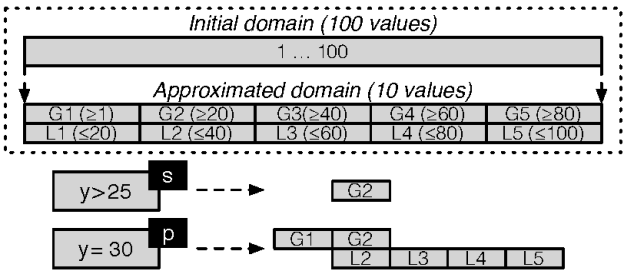

Figure 3.18: Inequality prefiltering pre-mapping mechanism.

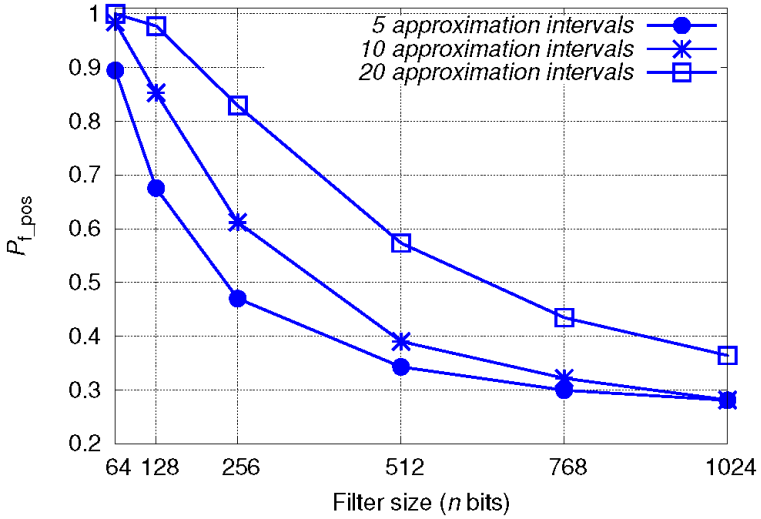

Figure 3.19: Probability of false positives for inequality prefiltering as the size of the Bloom filters and the number of domain intervals vary, using $a = 10$, $v = 100$, $c_1 = 3$, $c_p = 5$, $k = 5$ and $\alpha = 1$.

all the covering interval bounds in publications may lead to false negatives. Interestingly, the pollution and truncation containment obfuscation techniques discussed in Section 3.3 can also be applied to the inequality Bloom filters.

One drawback of prefiltering inequalities is the amount of information leaked by the Bloom filters. The analysis of containment information leaked from the subscriptions inequality filters is similar to the analysis done for equality constraints, with two differences: on the one hand, the size of the subscription domain is reduced to twice the number of interval boundaries, but on the other hand those boundaries do not correspond to the exact inequalities. Unfortunately, the size of the publication domain is also reduced to the number of interval boundaries, which can be very small.

We simulated inequality prefiltering with uniform publication and subscription domains of $a = 10$ attributes and $v = 100$ values per attribute, $c_1 = 3$ inequality constraints per subscription, $c_p = 5$ attributes per publication, and one hash function randomly chosen out of 5 for the subscriptions Bloom filters. The results are in Figure 3.19 for $\{5, 10, 20\}$ equal-size domain partitions. For a Bloom filter of 256 bits and 5 domain intervals, we can prefilter more than 50% of the non-matching publications. Unsurprisingly, the performance is significantly worse than for equality constraints. This is due to the difference between the real constraints and the interval boundaries, but more importantly to the large number of values that must be encoded in the publications Bloom filters.

3.7 Summary

Prefiltering, and when available containment relationships between subscriptions, allow us to greatly reduce the performance gap between non-encrypted and encrypted filtering. By adding Bloom filters that encode publication values and subscription equality constraints, we can discard a large fraction of subscriptions before reaching the costly encrypted filtering operation. Evaluations confirmed that the mechanisms we proposed reduce the number of such costly operations required to filter an incoming publication by approximately one order of magnitude. Furthermore, when selecting a subset of the hash functions for encoding subscription equality constraints in the Bloom filters, we showed that it is still possible to prefilter a large fraction of the subscriptions while leaking very little containment information to potential attackers.

Several extensions of this work are possible. In Section 3.6, we discussed how Bloom filters could be used to prefilter inequality constraints. We also believe that the results and techniques we obtained, can be applied to other problems in distributed systems where set inclusions are represented using a compact structure such as Bloom filters, and where confidentiality requirements are stringent. One such application is encrypted distributed databases [Popa et al., 2011, Kurpicz, 2013]. Also, a rigorous mathematical analysis following the same steps as ours could be applied to other case that were presented empirically, for instance the filter obfuscation technique in Perl *et al.* [Perl et al., 2012].

Chapter 4

Advances in ASPE functionality

4.1 Multidimensional pub/sub matching with ASPE

We introduced the basics of the ASPE (Asymmetric Scalar-product Preserving Encryption) applied in pub/sub [Choi et al., 2010] in Section 2.1.2. In their discussion, the authors mainly consider the unidimensional case. We support multidimensional subscriptions, case which is just briefly suggested by the original work. In the following, we present our extensions that cover this situation.

Let n be the number of attributes in a publication. A subscription also contains up to n constraints on these attributes. Each value v_i in a constraint is first encapsulated in an intermediate multidimensional subscription point $S_i = (s_1, s_2, \ldots, v_i, \ldots, s_n)$, where all the other dimensions: $s_1, \ldots, s_n$ are randomly chosen. Two reference subscription points $S_{i1} = (q_1, q_2, \ldots, v_i - r_i, \ldots, q_n)$ and $S_{i2} = (q_1, q_2, \ldots, v_i + r_i, \ldots, q_n)$ are chosen symmetrically at random and at equal distance r_i from S_i, as for the unidimensional case. The r_i values are private parameters of the scheme. The need for identical q_k random values for S_{i1} and S_{i2} is not specifically stated in [Choi et al., 2010], but our analysis proves this requirement. We formulate the following theorem.

Theorem 3 ***Matching theorem.*** *Let $S_{i1} = (q_1, q_2, \ldots, c, \ldots, q_n)$, $S_{i2} = (q_1, q_2, \ldots, d, \ldots, q_n)$ and $P(x_1, x_2, \ldots, m, \ldots, x_n)$ be three points in a n-dimensional space, such that $c = v_i - r_i$ and $d = v_i + r_i$ ($v_i = (c+d)/2$). Then, $dist(S_{i1}, P) \diamond dist(S_{i2}, P) \Leftrightarrow m \diamond v_i$, where $\diamond \in \{>, <, =\}$.*

Proof. Without loss of generality we consider $\diamond$ as $>$. We have

$$\begin{aligned} dist(S_{i1}, P) &> dist(S_{i2}, P) \Leftrightarrow \\ (q_1 - x_1)^2 + \cdots + (c - m)^2 + \cdots + (q_d - x_n)^2 &> (q_1 - x_1)^2 + \cdots + (d - m)^2 + \cdots + (q_d - x_n)^2 \Leftrightarrow \\ c^2 - 2cm + m^2 &> d^2 - 2dm + m^2 \Leftrightarrow \\ 2m &> (d^2 - c^2)/(d - c) \Leftrightarrow \\ 2m &> d + c \Leftrightarrow \\ m &> v_i. \end{aligned} \tag{4.1}$$

■

We observe that matching each constraint value v_i with publication P can be obtained from $D_i = dist(S_{i1}, P) - dist(S_{i2}, P)$. The effective encryption of the points resembles the

unidimensional case for the scheme discussed in Section 2.1.2, and proceeds as follows. The distance difference D_i can be expressed as a sum of scalar products

$$D_i = dist(S_{i1}, P) - dist(S_{i2}, P) = \| S_{i1} \|^2 - \| S_{i2} \|^2 + 2(S_{i2} - S_{i1})P. \tag{4.2}$$

ASPE encrypts points S_{i1}, S_{i2} and P such that the scalar product is preserved and the distance difference can still be obtained from the encrypted data. The key K used on the subscriber's side is M^T, the transpose of an invertible matrix. On the publisher's side the key is the matrix inverse M^{-1}. The resulting encrypted values are:

$$\begin{aligned} S'_{Ki1} &= E_K(S_{i1}) = M^T(S_{i1}, -0.5 \| S_{i1} \|)^T; \\ S'_{Ki2} &= E_K(S_{i2}) = M^T(S_{i2}, -0.5 \| S_{i2} \|)^T; \\ P'_K &= E_K(P) = M^{-1}q(P, 1)^T. \end{aligned}$$

where q is a positive random scaling factor added for security purposes. An untrusted broker that receives S'_{Ki1}, S'_{Ki2} and P'_K can compare S with P by evaluating

$$(S'_{Ki2} - S'_{Ki1})P'_K = 0.5qD_i. \tag{4.3}$$

We can observe that for the multidimensional case determining, matching is basically reduced to a sequence of unidimensional matches. The solution is similar to projecting point P on the axis determined by S_{i1} and S_{i2}, and solving the problem for the dimension of interest i. A similar reduction applies also in extending the ASPE containment support for n dimensions, and proving its correctness, as we detail in Section 4.2.

4.2 Pub/sub containment with ASPE

As discussed in the existing research [Raiciu and Rosenblum, 2006, Barazzutti et al., 2012b], and previously in Chapter 3, containment support diminishes the achievable level of confidentiality in secure pub/sub scenarios. Nevertheless, as overviewed in Chapter 2 there are encrypted matching schemes like [Choi et al., 2010, Raiciu and Rosenblum, 2006, Nabeel et al., 2012], which offer valuable optional containment in applications with less stringent security constraints. Original work on ASPE proposes a containment extension to the original scheme [Wong et al., 2009], but only considers the unidimensional case, and in a rather informal manner. We develop a proof of correctness, and make the necessary adjustments to the ASPE unidimensional containment case presented in [Choi et al., 2010]. We continue with a description of the multidimensional case. This can be solved by reduction to a unidimensional case following an approach similar to the generic multidimensional matching we discussed in Section 4.1.

The unidimensional containment case presented in [Choi et al., 2010] is based on the following example, which we further denote as the *generic setting*. Two subscriptions on values a and b must be evaluated for containment by a broker. The subscriptions are constructed and encrypted as in the normal ASPE case by using two reference subscription points for each. For a the reference points are $c = a-s$ and $d = a+s$. For b the reference points are $e = b-t$ and $f = b+t$. In both cases s and t are positive integers. In addition, to obtain a containment decision, two points $y >> d$ and $z >> f$ are selected for the two subscriptions, respectively. Both y and z are encrypted following the publication encryption algorithm and are attached

to their corresponding encrypted subscription. The containment objective is to obtain a relation $(>, <, =)$ between the two subscription points a and b using the ASPE encrypted values of c,d,e,f,y and z. In [Choi et al., 2010] the authors try to detemine containment by evaluating distance relations computed between points in plaintext form. The ASPE property to preserve the distance difference for encrypted values, ensures that this containment result is valid also after encryption. In the following, we offer a correctness proof on obtaining the containment relation between a and b, starting from the generic setting baseline. This extends the description in the original scheme, eliminating several inaccuracies.

Lemma 8 *If $y > b$, then:*

$$dist(c, y)+dist(d, y) > dist(e, y)+dist(f, y) \Rightarrow dist(a, y) > dist(b, y).$$

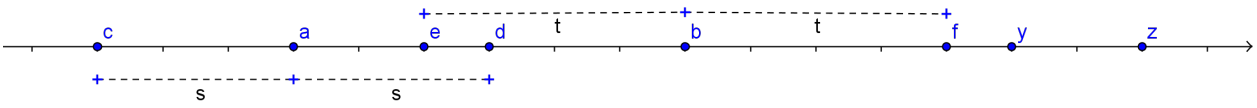

Figure 4.1: Subscription points when $y > b$ and $y \geq f$.

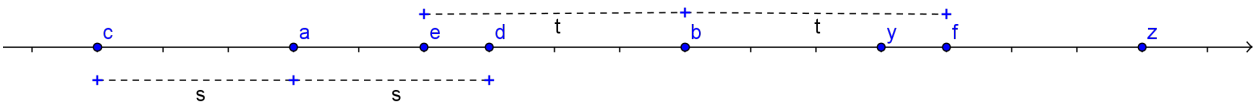

Figure 4.2: Subscription points when $y > b$ and $y < f$.

Proof From $y > b$ we have that $dist(e, y) = t+dist(b, y)$. It follows that

$$2s+2dist(d, y) > t+dist(b, y)+dist(f, y) \Leftrightarrow 2dist(a, y) > t+dist(b, y)+dist(f, y). \quad (4.4)$$

We further distinguish two cases: $y \geq f$, shown in Figure 4.1 and $y < f$, shown in Figure 4.2:

- If $y \geq f$, we have $t+dist(f, y) = dist(b, y)$, and it results from (4.4) that $dist(a, y) > dist(b, y)$.
- If $y < f$, from $y > b$ we have $dist(b, y)+dist(f, y) = dist(b, f) = t$ and $dist(b, y) < t$. From this and (4.4) it follows that $dist(a, y) > t \Rightarrow dist(a, y) > dist(b, y)$.

■

Lemma 9 *If $z > a$, $y > b$, and $z > d$, then:*

$$dist(c, z)+dist(d, z) > dist(e, z)+dist(f, z) \Rightarrow dist(a, y) > dist(b, y).$$

Proof Figures 4.1 and 4.2 show valid points configurations for the lemma assumptions. From $z > a$ we have that $dist(c, z) = s{+}dist(a, z)$. It follows that:

$$s{+}dist(a, z){+}dist(d, z) > 2t{+}2dist(f, z) \Leftrightarrow s{+}dist(a, z){+}dist(d, z) > 2dist(b, z). \quad (4.5)$$

From $z > d$ we have $dist(a, z) = dist(a, d){+}dist(d, z) = s{+}dist(d, z)$. From this and (4.5) it follows that $dist(a, z) > dist(b, z)$.

Then, from $y > b$ it follows that $dist(a, y) > dist(b, y)$. ■

Lemma 10 *If $y > b$, then:*

$$dist(a, y) > dist(b, y) \Rightarrow dist(c, z){+}dist(d, z) > dist(e, z){+}dist(f, z).$$

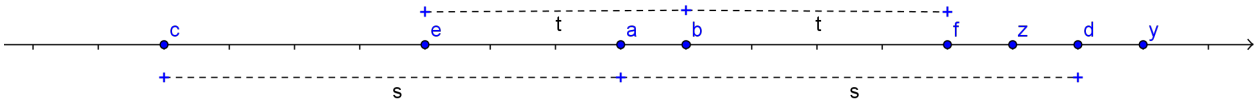

Figure 4.3: Subscription points when $y > b$ and $z \leq d$.

Proof Figures 4.1, 4.2 and 4.3 show valid point configurations for the lemma assumptions. From $dist(a, y) > dist(b, y)$, $y > b$ and $z > f$ (from the initial generic setting), we have

$$dist(a, z) > dist(b, z). \quad (4.6)$$

We prove the lemma by contradiction. Suppose that

$$dist(c, z){+}dist(d, z) \leq dist(e, z){+}dist(f, z). \quad (4.7)$$

From (4.7), $dist(a, c) = s$ and $dist(e, b) = t$ we have

$$s{+}dist(a, z){+}dist(d, z) \leq t{+}dist(b, z){+}dist(f, z) \Leftrightarrow s{+}dist(a, z){+}dist(d, z) \leq 2dist(b, z). \quad (4.8)$$

From (4.6) and (4.8) it follows that

$$s{+}dist(d, z) < dist(b, z). \quad (4.9)$$

We now have to consider the possible cases for the position of z relatively to d on the axis:

- If $z \leq d$ (Figure 4.3), from $dist(a, z) \leq dist(a, d) \Leftrightarrow dist(a, z) \leq s$ it follows that (4.9) contradicts (4.6).
- If $z > d$ (Figure 4.1 or 4.2), then from $s{+}dist(d, z) = dist(a, d){+}dist(d, z) = dist(a, z)$ it also results that (4.9) contradicts (4.6).

It follows that the assumption of (4.7) cannot be true, which proves the lemma. ■

Lemma 11 *If $y > b$ and $y > f$, then*

$$dist(a, y) > dist(b, y) \Rightarrow dist(c, y){+}dist(d, y) > dist(e, y){+}dist(f, y).$$

Proof Figures 4.1 and 4.3 show two valid point configurations for the lemma assumptions. We prove the lemma by contradiction. Suppose that

$$dist(c,y)+dist(d,y) \leq dist(e,y)+dist(f,y). \tag{4.10}$$

It follows that

$$s+dist(a,y)+dist(d,y) \leq t+dist(b,y)+dist(f,y) \Leftrightarrow 2dist(a,y) \leq t+dist(b,y)+dist(f,y). \tag{4.11}$$

From (4.11) and the initial premise of $dist(a,y) > dist(b,y)$ it follows that

$$dist(a,y) < t+dist(f,y). \tag{4.12}$$

If $y > f$, $t+dist(f,y) = dist(b,y)$. It follows that (4.12) contradicts the premise of $dist(a,y) > dist(b,y)$. ■

Lemma 12 *If $y < f$, then $z > d$.*

Proof We know that $d < y$ and $f < z$. If $y < f$ it trivially results that $z > d$. ■

We now have all the preliminary results required to formulate a containment theorem that can be used to rigorously determine when a constraint in a subscription contains (or covers) a constraint in another subscription by comparing distances preserved by the ASPE encryption.

Theorem 4 ***Containment theorem.*** *Let a, b, c, d, e, f, y, z be a set of points on an axis such that $a-s = c$, $a+s = d$, $b-t = e$, $b+t = f$, $y > d$, $z > f$, where s and t are random positive integers. Consider also the reference points x_1, x_2 such that $y-u = x_1$ and $y+u = x_2$ where u is a random positive integer (for generality, in a similar manner, two values q_1, q_2 can be considered in respect to z). The relation between points a and b can be determined based on the following cases.*

Case 1:

$$dist(e,y)-dist(f,y) \leq 0 \Rightarrow a < b \tag{4.13}$$

Case 2:

$$dist(c,z)-dist(d,z) \leq 0 \Rightarrow b > a \tag{4.14}$$

Case 3:

$$\begin{aligned}((dist(c,y)+dist(d,y) > dist(e,y)+dist(f,y) \wedge dist(x_1,z)-dist(x_2,z) \leq 0) \vee \\ (dist(c,z)+dist(d,z) > dist(e,z)+dist(f,z) \wedge dist(x_1,z)-dist(x_2,z) > 0)) \\ \Leftrightarrow b > a\end{aligned} \tag{4.15}$$

Proof

Case 1:

If $dist(e,y)-dist(f,y) \leq 0$, then $y \leq b$. Since $a < y$ it results that $a < b$.

Case 2:

If $dist(c,z)-dist(d,z) \leq 0$, then $z \leq a$. Since $b < z$ it results that $b < a$.

Case 3:

The first two cases cover the situation when $y \leq b \vee z \leq a$. If $(dist(e,y) - dist(f,y) > 0) \wedge (dist(c,z) - dist(d,z) > 0)$ (contradicting premises of both preceding cases) it follows that $y > b \wedge z > a$. This is complementary with the preceding cases, covering all space of posibilities. We prove the expression in (4.15), having $y > b \wedge z > a$, and the following possible situations:

- If $y > f$, then from Lemma 8, Lemma 11 and $y > b$, we have that

$$dist(c,y) + dist(d,y) > dist(e,y) + dist(f,y) \Leftrightarrow dist(a,y) > dist(b,y) \Leftrightarrow b > a. \quad (4.16)$$

- If $z > d$, then from Lemma 9, Lemma 10 and $y > b$, we have that

$$dist(c,z) + dist(d,z) > dist(e,z) + dist(f,z) \Leftrightarrow dist(a,y) > dist(b,y) \Leftrightarrow b > a. \quad (4.17)$$

We don't know which of the assumptions of $y > f$ or $z > d$ is valid. Therefore, we don't know which of (4.16) or (4.17) to test for determining the relation between a and b. Nevertheless, from Lemma 12 we know that $y > f$ and $z > d$ are mutually exclusive. We have to determine which case is valid. For this, it is enough to compute

$$dist(x_1,z) - dist(x_2,z) \leq 0 \Leftrightarrow z \leq y. \quad (4.18)$$

The two possible valid cases are:

- If $z \leq y$, from $z > f$, it follows that $y > f$, and we can determine whether $b > a$ from (4.16).
- If $z > y$, from $y > d$, it follows that $z > d$, and we can determine whether $b > a$ from (4.17).

■

About false positives and negatives

The reference points x_1 and x_2 introduced in the Containment theorem are a mandatory addition to the scheme presented in [Choi et al., 2010], in order to support subscription containment. These points should be encrypted like c and d in the first subscription. Without x_1 and x_2, the scheme is not able to determine precisely the containment result. We explain the necessity to add these points in the following.

The idea is that we cannot determine the position of y relative to f, and of z relative to d, without these additional points. Let us try, lacking these points, to determine if $b > a$.

If $dist(c,y) + dist(d,y) > dist(e,y) + dist(f,y)$ we can determine that $b > a$. However, if $dist(c,y) + dist(d,y) \leq dist(e,y) + dist(f,y)$ (left side of 4.16 is false), then we are not certain whether $b > a$ is false. We have the following possibilities:

- If $y > f$, from the if-and-only-if in (4.16) it results that $b \leq a$. This situation is depicted in Figure 4.4.
- If $y < f$ and $dist(a,y) \leq t$ it might happen that $b > a$. An example is shown in Figure 4.5.

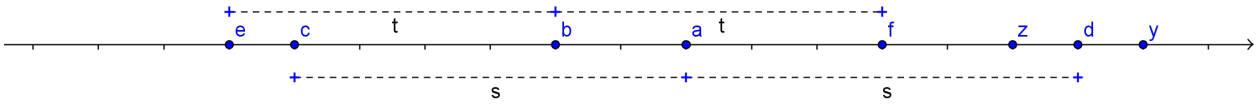

Figure 4.4: Subscription points when $dist(c, y)+dist(d, y) \leq dist(e, y)+dist(f, y)$ and $y > f$.

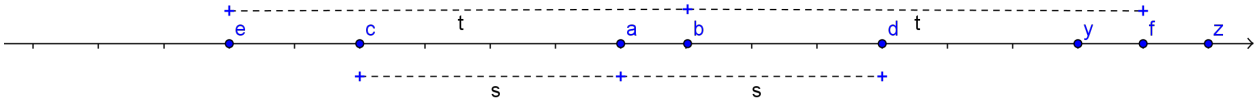

Figure 4.5: Subscription points when $dist(c, y)+dist(d, y) \leq dist(e, y)+dist(f, y)$ and $y < f$.

- If $y < f$ it also might happen that $b \leq a$. A situation of this case is presented in Figure 4.6.

To determine which case is valid, and continue exploring the space of solutions, we require the positioning of y relative to f. Without the points x_1 and x_2 we do not have enough elements in the scheme to determine the relation between y and f when $dist(c, y)+dist(d, y) \leq dist(e, y)+dist(f, y)$ (left side of 4.16 is false). It is impossible therefore to obtain an accurate containment decision.

The description in [Choi et al., 2010] suggests in this case to evaluate containment similarly using z and (4.17). However, following a similar path, we can obtain an accurate answer for the evaluation of $b > a$ only in the case where we know if $z > d$. Since we do not know if $y > f$, we also cannot know if $z > d$.

This lack of precision in the process of determining the containment leads both to false positives and false negatives. Let us consider a containment function $contains(S_1, S_2)$ that determines if subscription S_1 contains S_2. Let us consider if the left side of (4.16) is false we give an answer that $b > a$ not true (with a high probability of being correct, the other case being quite restrictive). Depending on the operators of constraints in the evaluated subscriptions we can reach the situations in the following example:

- Let $S_1 : [field < 5]$ and $S_2 : [field < 7]$ $(a = 5, b = 7)$. $contains(S_1, S_2)$ should return true. If we evaluate $b > a$ $(7 > 5)$ erroneously as false, we obtain that S_1 does not contain S_2, which is a false negative.
- Let $S_1 : [field > 5]$ and $S_2 : [field > 7]$ $(a = 5, b = 7)$. $contains(S_1, S_2)$ should return false. If we evaluate $b > a(7 > 5)$ erroneously as not true, we obtain that S_1 does contain S_2, which is a false positive.

The additional reference points we introduced: x_1 and x_2, prevent such inaccurate results.

Multidimensional Containment

We can evaluate containment for two multidimensional subscriptions by splitting per constraint, using the formulas and following the same reasoning as for a single dimension. Let us consider two subscriptions S_1 and S_2, having a and respectively b as constraint values for a dimension of interest d_i.

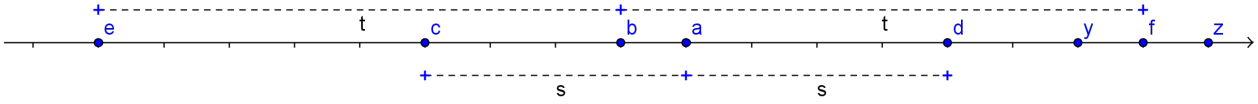

Figure 4.6: Subscription points when $dist(c,y)+dist(d,y) \leq dist(e,y)+dist(f,y)$ and $y < f$.

The prerequisites for multidimensional encrypted matching described in Section 4.1 require setting C and D, two intermediary subscription points associated with the value a for d_i in S_1. Similarly, let E and F be the intermediary subscription points associated with value b for d_i in S_2. C and D have as structure $(r_1, r_2, \ldots, a-r_i, \ldots, r_n)$ and respectively $(r_1, r_2, \ldots, a+r_i, \ldots, r_n)$, where the r values are randoms. E and F present a similar construction in respect to b. Points Y and Z are chosen such that the value of the dimension of interest d_i in Y is $>>$ than the value of d_i in D, and respectively the value of d_i in Z is $>>$ than the value of d_i in F. Finally, two multidimensional points X_1, X_2 are chosen as for the unidimensional case, symmetrically to the dimension of interest d_i in point Y. We can rewrite now for the multidimensional case the expressions (4.13), (4.14) and (4.15) in the Containment theorem:

$$dist(E,Y)-dist(F,Y) \leq 0 \Rightarrow a < b; \tag{4.19}$$

$$dist(C,Z)-dist(D,Z) \leq 0 \Rightarrow b > a; \tag{4.20}$$

and respectively:

$$\begin{gathered}(dist(C,Y)+dist(D,Y) > dist(E,Y)+dist(F,Y) \wedge dist(X_1,Z)-dist(X_2,Z) \leq 0) \vee \\ (dist(C,Z)+dist(D,Z) > dist(E,Z)+dist(F,Z) \wedge dist(X_1,Z)-dist(X_2,Z) > 0) \\ \Leftrightarrow b > a.\end{gathered} \tag{4.21}$$

Multidimensional case containment requires an additional restriction. The c, d, e, f values must be situated on the same axis in order to preserve the correctness in relation 4.21 (as proven in the unidimensional case). In this example, $c = a-r_i$, $d = a+r_i$, and e and f are the similar values constructed in vectors E and F. To enable this restriction every random r value in vectors C, D, E, F has to be the same per corresponding dimension, with the exception of the dimension of interest (the values subtracted and added to a, respectively b). Even more, the same restriction on the dimension of interest has to be valid also for points X_1 and X_2.

From the Matching theorem in Section 4.1, we observe that elements in vectors Y and Z do not need however to be set as points on the same axis as the corresponding values in C, D, E, F, X_1, X_2. Y and Z can be constructed uniquely per subscription. More precisely, there is no need to have an Y and Z vector per dimension, but each dimension's value in the vectors is chosen to correspond to a dimension of interest in the subscription points. The Y and Z vectors basically resemble the case of the publication in the multidimensional *Matching theorem* in Section 4.1.

4.3 From confidential kNN queries to confidential pub-/sub

We discuss in this section the rationale of the ASPE scheme adaptation from its initial use for confidential kNN queries in databases [Wong et al., 2009] to an encrypted matching scheme for pub/sub [Choi et al., 2010]. We believe that the approach taken by the authors can be applied for other existing schemes and thus we offer a generalization of the procedure.

A kNN query on a database searches for the nearest k entries (records) $A_1, ..., A_k$ that are closest to a query point Q. The database entries and the query can be modelled as points in a n-dimensional space, where each dimension corresponds to a field in the database schema. Being closer to the query is determined by comparing the Euclidean distances between the query point and each of the record points (e.g., $dist(Q, A_1)$ compared with $dist(Q, A_2)$).

Subscriptions and publications in a pub/sub system can be modelled as points in a n-dimensional space. However, the matching in pub/sub is fundamentaly different from the evaluation of kNN queries. The objective for pub/sub is not to evaluate how close a point is to another one. The target for pub/sub matching is *the exact relation* $(>, <, =)$ between an individual dimension of a point (the subscription) and the same dimension in another one (the publication). A kNN query does not specifically take into account the relations between individual dimensions, being resolved by just comparing distances. For instance, in Figure 4.7, points $A_1 = (1, 2)$ and $A_2 = (3, 0)$ are equally close to query point $Q = (2, 1)$. Therefore, a kNN query for point Q, will not distinguish between points A_1 and A_2. The kNN query result does not help us to determine the exact relation between the corresponding individual dimensions (the x and y coordinates) in Q and in either A_1 or A_2.

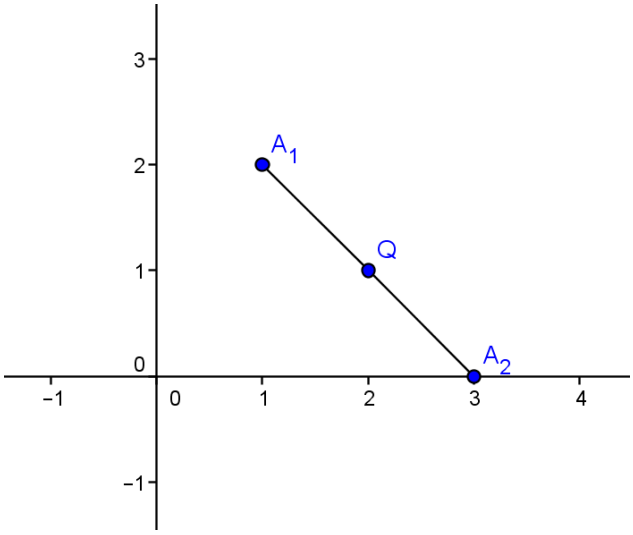

Figure 4.7: Distance comparison between points in a kNN query.

For pub/sub matching we need the exact relation per dimension, and we proceed as in the following. Let us consider the particular case where all corresponding dimensions in each point are equal, except the dimension of interest d_i. Furthermore, let us consider that the relation $(<, >, =)$ between the value of d_i in A_1 and A_2 is known. We can obtain the target result: the relation between the value of d_i in Q and in points A_1 and A_2, through a kNN query comparing $dist(Q, A_1)$ and $dist(Q, A_2)$. Q, A_1 and A_2 are all on an axis determined by the other dimensions except d_i, which is the only coordinate that varies. Since d_i is the only

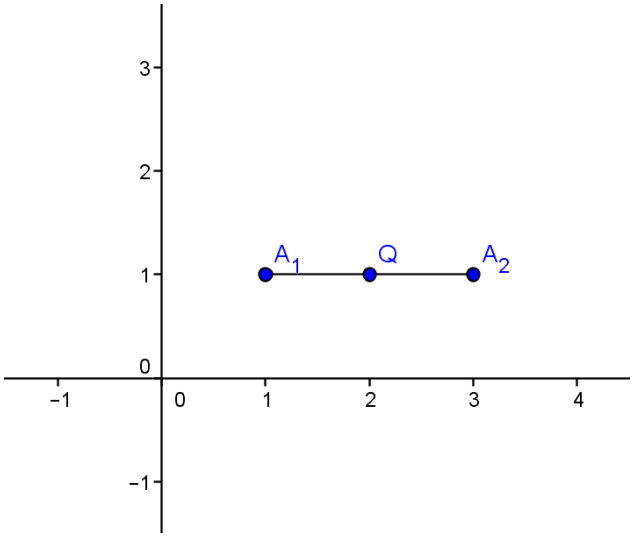

Figure 4.8: Distance comparison between points in the pub/sub case.

dimension that influences the comparison between $dist(Q, A_1)$ and $dist(Q, A_2)$, the result also excludes false positives.

In Figure 4.8 we depict an example, that follows the above reasoning for the bidimensional points $A_1 = (1, 1)$, $A_2 = (3, 1)$ and $Q = (2, 1)$. Let us consider the case where pub/sub matching needs to evaluate the exact relation between the value corresponding to the first dimension d_1 in point Q and the one in the other two points A_1 and A_2. As assumed above, we consider a known fact that the value for d_1 in A_1 is smaller than the value for d_1 in A_2 (and the second coordinate is equal). We know the kNN query result for the query point Q, which shows that $dist(Q, A_1) = dist(Q, A_2)$. It naturally results that the value for d_1 in Q is bigger than the value for d_1 in A_1 and smaller than the value for d_1 in A_2. In particular, from the distances equality, we also observe that the d_1 dimension in Q coincides with the middle of the segment $[A_1A_2]$. This additional observed relation is fundamental for generalizing the way to adapt kNN queries to pub/sub matching.

From the matching theorem in Section 4.1 we can observe that the distance comparison between: $dist(Q, A_1) \sim dist(Q, A_2)$ is preserved when Q is projected on the axis determined by A_1 and A_2. Therefore, it is enough that only points A_1 and A_2 (and not necessarilly also Q) have all corresponding dimensions equal, except d_i.

We can define a generic approach, similar with the one used for ASPE, that facilitates incrementally constructing a confidentiality-preserving pub/sub solution. The base requirement is having any encryption scheme that preserves the possibility to compare distances between pairs of encrypted points. Then, for matching a subscription S with a publication P on dimension d_i, we consider two points S_{i1} and S_{i2}, symmetrically placed in respect to d_i. This is similar to choosing points A_1 and A_2 in respect to the middle of segment $[A_1A_2]$ in the previously discussed example. From the comparison result given by the encryption scheme: $dist(S_{i1}, P) \sim dist(S_{i2}, P)$ (where $\sim \in \{<, >, =\}$), we obtain the relation between the value for dimension d_i in P and dimension d_i in S. Following the above example, this is similar to the relation obtained between dimension d_i in Q and the middle of segment $[A_1A_2]$.

The security of the obtained confidential pub/sub solution relies (as in the ASPE case) on the fact that the original distance comparison-preserving encryption scheme is secure. Note that *preserving distances comparison* is a weaker property than *preserving distances*. In particular, the latter should not be possible to derive from the former, contrary resulting in

confidentiality leaks as discussed in [Wong et al., 2009]. Secure kNN schemes are a particular case that might preserve distances comparison, due to their functionality.

4.4 Summary

In this chapter, we have discussed the work on the multidimensional matching and containment support for the ASPE scheme. This extends the functionalities from [Choi et al., 2010], and covers gaps in the original scheme adaptation that can lead to both false positives and negatives.

We have also discussed how to adapt a secure kNN query to an encrypted matching scheme used in pub/sub systems. This solution was adopted by the authors of [Choi et al., 2010] for the ASPE scheme. We believe that a similar approach could be adopted for other secure kNN queries, and in that purpose we have generalized the design. It remains as future work to investigate which secure kNN schemes could undergo such adaptations.

Chapter 5

Key management for confidential pub/sub

5.1 Motivation, problem description and contributions

As introduced in Chapter 1, the typical assumption in confidential pub/sub is to consider brokers as *honest-but-curious*. For routing flexibility purposes, the most interesting solution to encrypt sensitive data passed through the broker overlay is to use an encrypted matching scheme. Encrypted matching schemes typically require a prior key exchange between communicating parties. We already summarized the key management problem in confidential publish/subscribe systems in Section 1.3.4 of Chapter 1. In this section we further detail its implications, and introduce our solution. A difficulty lies in the fact that publishers have no a-priori knowledge of the subscribers which will receive a given publication. As we have seen in the overview of Chapter 2, schemes based on secret key (symmetric) encryption algorithms [Choi et al., 2010, Raiciu and Rosenblum, 2006, Li et al., 2004] expicitly require a key exchange, but consider this an orthogonal issue. The corresponding papers do not develop a solution. Schemes that adapt public key algorithms still require either common private information used in encryption to be sent at both the publisher and subscriber [Ion et al., 2012] or an initial communication link between these two entities [Nabeel et al., 2012].

Besides the initial key exchange, updating a key introduces even more serious challenges. A key update is needed in cases of changes in the client trust (e.g., a host using the current key is corrupted and has to be evicted), or simply as a periodic key refresh in order to increase resilience to brute-force attacks. Following, all subscriptions stored by the brokers and encrypted with the old key can no longer be matched with publications encrypted with the new key. A naive solution requires that the clients resubmit all their subscriptions, encrypted with the new key. This presents several major drawbacks. It forces subscribers to keep track and store their set of previous subscriptions. This might be a problem in case of failures. The key update process becomes prohibitively long (depending on the constraints of the network layer) and resource consuming, as a large amount of incoming subscriptions and un-subscriptions have to be handled by the brokers. Consequently, the impact on the quality of service, e.g., in terms of throughput and delays, can become significant.

We present in this chapter DynamiK, a key management architecture maintaining a partial decoupling between the individual publisher and subscriber nodes. We eliminate the need for subscription resubmission upon a key update by introducing a particular requirement for the encrypted matching scheme: the ability to perform *in-broker subscriptions re-encryption*. Untrusted brokers storing subscriptions can be given a specific token K_R directly related to both the previous and the new encryption key that allows them to update

encrypted subscriptions *in place*. Obviously, this token should not permit obtaining the original or the new encryption key.

We present an implementation of the DynamiK architecture for an existing encrypted matching scheme, Asymmetric Scalar-product Preserving Encryption (ASPE) [Choi et al., 2010]. In this purpose we modify the scheme introduced in Section 2.1.2, for which we have already covered the multidimensional case functionality gaps in Chapter 4. In this chapter we present a novel set of substantial additions to the original version. We introduce the support of *in-broker subscriptions re-encryption* to ASPE and prove that this addition does not allow deriving the actual encryption keys. This requires changes that also increase the security of the scheme in its basic form. Note that these changes and related analysis apply on the multidimensional case of the scheme, but without the optional containment support, and therefore without the containment additions presented in Chapter 4. Our implementation integrates with the StreamHub [Barazzutti et al., 2013] content-based pub/sub engine. It takes advantage of the ZooKeeper coordination service [Hunt et al., 2010] for the key distribution and synchronization, inheriting its dependability. Finally, we present evaluation results for our implementation of DynamiK that demonstrate that the key update can be conducted with a negligible impact on the quality of service.

5.2 Architecture requirements

Key management for pub/sub is considered a difficult challenge to overcome due to the decoupled nature of communication [Ion et al., 2012, Srivatsa and Liu, 2007]. However, this disregards a practical side of the problem. Indeed, following the stock exchange example in Section 1, a host belonging to an investment company that registers a subscription does not know the exact host that emits publications. However, the investment company does know from which stock exchange it wants to obtain the publication flow (e.g., New York, London, etc). Likewise, the stock exchange might charge fees for providing the publication flow to investment companies. It is often the case that while no direct communication between the publishers and subscribers themselves exists, there are relations between their respective domains. This assumption gives the first requirement for our architecture:

Requirement 1: *grouping hosts in established security domains* for individual subscribers, publishers and brokers.

Our key dissemination architecture follows a simple and generic model. Each security domain is administered by a domain manager. A key distribution coordinator handles the generation and transmission of key information to each domain manager, which distributes the key information to the individual hosts of the domain. This distribution covers the initial key dissemination and subsequent key updates in the system (e.g., when a host joins or is evicted).

Security domains have different levels of trust. Brokers are typically part of an untrusted security domain. The key is not to be distributed to this domain. However, the key management architecture supports the disseminating of *key update tokens*. These tokens allow brokers to update the encrypted subscriptions they store in place, if the underlying encryption scheme supports it.

Our architecture model makes available *common* or *correlated* key information to hosts in different domains. A partial decoupling of these hosts is preserved. The model keeps only a loose coupling at the domain level.

Our proposal resembles a generic hierarchical secure group communication key management architecture. In fact, we consider viable to adapt well established secure group communication key management protocols (e.g., [Wong et al., 2000, Sherman and McGrew, 2003, Perrig et al., 2001]) for key dissemination in pub/sub systems. As mentioned in our overview in Chapter 2 the idea of adapting such a protocol for pub/sub systems was proposed in [Bacon et al., 2008] with OFT [Sherman and McGrew, 2003] for key management, but without support for encrypted matching. In [Li et al., 2004] the authors suggest ELK [Perrig et al., 2001] as another example but do not develop the idea. We establish the second requirement of our architecture:

Requirement 2: *a secure group communication key management protocol* that can be adapted to the trust assumptions of the pub/sub domain grouping.

A specificity of pub/sub key management is that untrusted domains (containing the brokers) persistently store encrypted information (the subscriptions). These need to be updated when there is a key update. We already detailed in Section 5.1 why resubmitting all subscriptions encrypted with the new key will create an unacceptable burden for the pub/sub system. We favor a novel approach where the transition from subscriptions encrypted with a previous key to subscriptions encrypted with the new key is performed directly by the broker and does not require the subscribers to re-encrypt or re-send the subscriptions themselves. This requires that the used encrypted matching scheme supports in-broker subscriptions re-encryption. We detail this operation below. Let us consider $E_K(S)$, the ciphertext obtained by encrypting subscription S with key K using an encrypted matching algorithm E. The support for in-broker subscriptions re-encryption implies that for two keys K_1 and K_2, a function $K_R = f(K_1, K_2)$ can be defined, such that for an encrypted subscription $E_{K_1}(S)$ there exists a transformation $\tau(E_{K_1}(S), K_R) = E_{K_2}(S)$. K_R is then the token that is submitted to the brokers to perform the in-place re-encryption. Obviously, making K_R available in an untrusted domain must not disclose any information about the encryption keys K_1 and K_2 or the subscription plaintext S. The third requirement is thus:

Requirement 3: support for *in-broker subscriptions re-encryption* by the encrypted matching scheme.

5.3 The DynamiK key management architecture

In this section we detail DynamiK, our key management architecture for privacy-preserving pub/sub using encrypted matching. The description corresponds to an actual implementation that we evaluate in Section 5.5. Each component follows from one of the requirements we set in the previous section. The architecture is illustrated by Figure 5.1. When applicable, we also list alternative choices for these components that would also match the requirements.

5.3.1 Grouping the individual hosts in security domains

We consider three security domains: one for subscribers, one for publishers and one for brokers. The subscribers and publishers domains share the same trust level. They receive the full key information required to encrypt subscriptions and publications. Brokers do not have access to the plain messages content and receive just the restricted key update token when an encryption key is updated in the system, allowing them to re-encrypt the corresponding

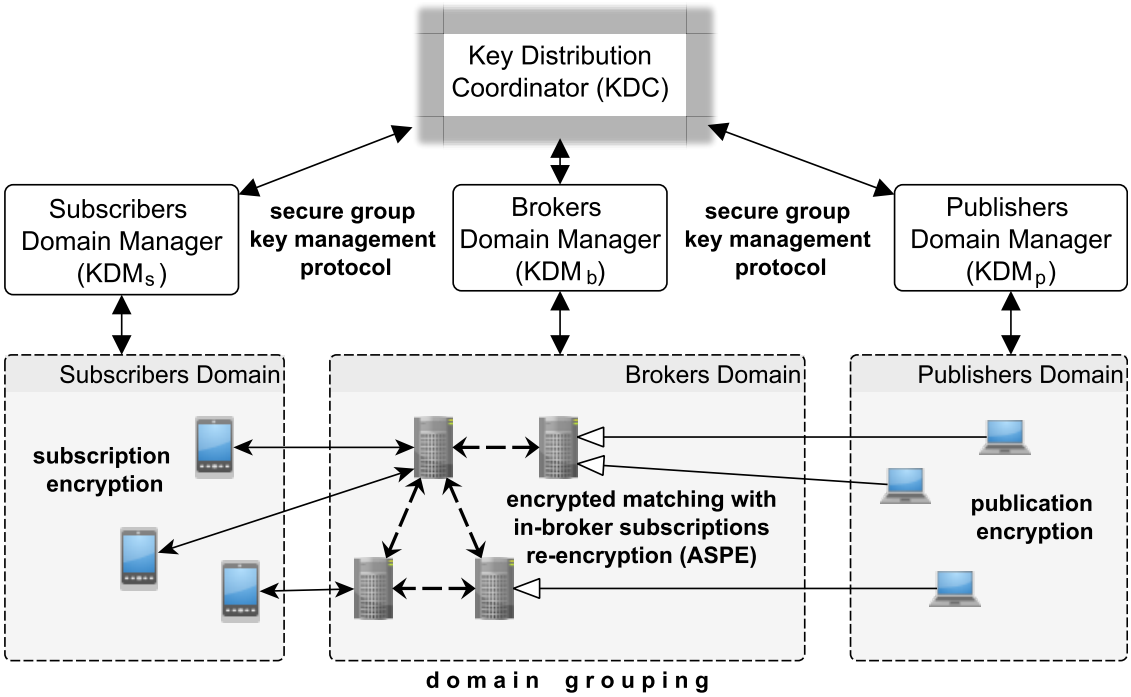

Figure 5.1: DynamiK architecture displaying three distinct domains (note that the generic model considers the possibility of multiple domains of one type).

stored subscriptions. We emphasize that our key management model allows splitting the participating hosts in multiple domains using different keys. For simplicity and evaluation purposes we however limit our implementation and deployment to three domains.

5.3.2 The secure group communication key management protocol

We develop a simple protocol for key dissemination that fits our architecture requirements. The main purpose of this protocol is to offer an example instance of our model that provides support for *in-broker subscriptions re-encryption*. Our implementation leverages the dependable ZooKeeper [Hunt et al., 2010] coordination service.

Preliminaries

We define the entities participating in our protocol:

- KDC - a central *key distribution coordinator*;
- KDM_s, KDM_p, KDM_b - three *key domain managers* for subscribers, publishers, and brokers, respectively;
- the *individual hosts* in each domain.

The KDC maintains a set of *cryptographic contexts*, which represent the information attached to a key, and in particular the various forms that a key can take for different types of users: publication encryption key for publishers, subscription encryption key for subscribers and the key update token for brokers.

We define a *secure pub/sub chain* as a set of publisher-broker-subscriber security domains passed by pub/sub data encrypted using the same cryptographic context. For

example, when an investment company registers encrypted subscriptions to a pub/sub broker service for encrypted publications emitted by a stock exchange, the three domains form a secure pub/sub chain. The KDC keeps the mapping from cryptographic contexts to secure pub/sub chains where these are used. A domain can be part of multiple secure pub/sub chains, and therefore use multiple cryptographic contexts.

Service reliability

Our implementation relies on the ZooKeeper [Hunt et al., 2010] coordination service. ZooKeeper simplifies the synchronization of distributed applications. It offers to the nodes of a distributed system access to a shared namespace organized hierarchically, similarly to a file system. ZooKeeper uses an ACL system to control the access to the shared configuration data. One of the features of ZooKeeper is the support for *watches*, that allow a client to be notified when there is an update (including creation and deletion) to a particular datum in the namespace. The KDC and the KDMs use watches to disseminate information to the other entities in the hierarchy, both for sharing keys and for notifications and synchronization of the key update protocol.

The first reason for using ZooKeeper is making the key management system highly-available. ZooKeeper provides dependability and consistency guarantees by replicating the state over multiple servers and ensuring update linearizability. All the state used for the protocol is stored in ZooKeeper. If the KDC or one of the KDMs crash while executing the protocol for key distribution, they can be restarted, read their last configuration from ZooKeeper and resume the key update. The second reason is the ease and facilities offered by the synchronization API in coordinating the action of the entities involved in the protocol.

Key distribution

The KDC distributes the key information in a cryptographic context to the KDMs in the secure pub/sub chain using that context. The particular key given to each KDM depends on the domain: a specific *pub/sub encryption key* to KDM_p and KDM_s and the *key update token* to KDM_b. The KDC encrypts the distributed key with the KDM's public key. In a practical scenario, the number of domains in a secure pub/sub chain is limited. This makes feasible to use individual public key encryption in the KDC to KDM distribution.

Further, the KDMs distribute the retrieved key information to the hosts in their security domain. In our current implementation, we simply consider intra-domain communication secure (e.g., the hosts are isolated in a network that is private to that domain). The intra-domain distribution could take different forms depending on the use case, optimization needs and security requirements (e.g., by adapting other secure protocols [Sherman and McGrew, 2003, Perrig et al., 2001] for key dissemination).

We describe next the key distribution steps when there is an update for an existing key. The process for the initial key distribution is similar. A key update can be triggered by a KDM joining or leaving a secure pub/sub chain, by an explicit request from a KDM (e.g., when a individual host in the KDM's domain gets corrupted and has to be evicted), or simply as a periodical key refresh.

During a key update, the brokers maintain a *transitional state* and keep the subscriptions encrypted with the old key along with the ones encrypted with the new key. When in-broker subscriptions re-encryption is supported by the encrypted matching scheme, the brokers generate the second set using the key update token (including for subscriptions received during the key update phase). Otherwise, the protocol requires all subscribers to

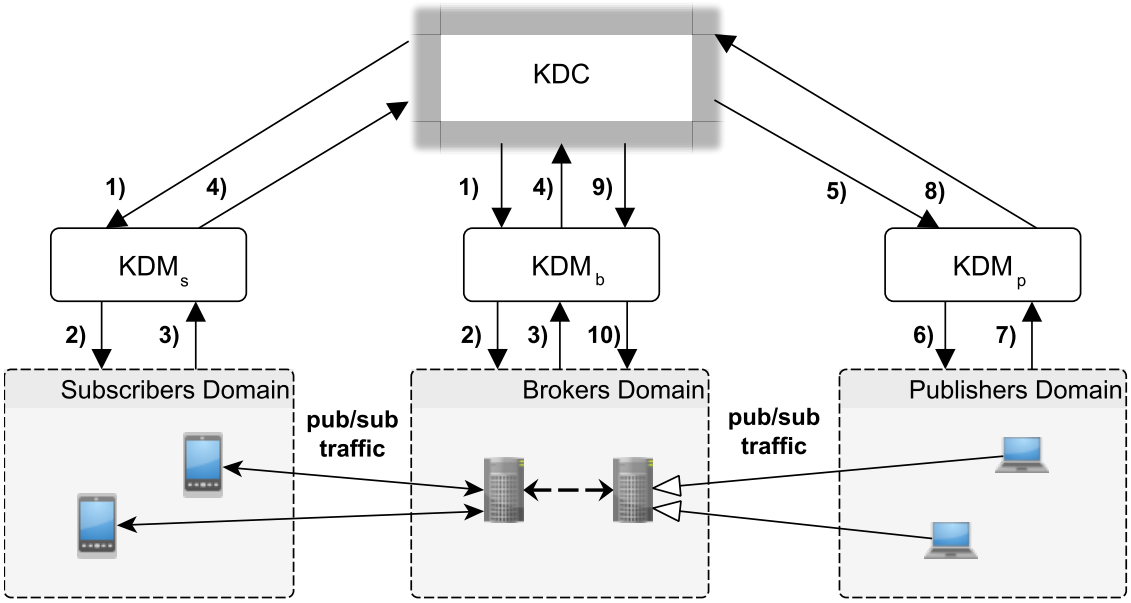

Figure 5.2: Illustration the DynamiK key update protocol.

re-encrypt and re-send their subscriptions. The protocol guarantees the consistency between the version of encryption keys in use at the publisher's side and the stored encrypted subscriptions. The old set of subscriptions is removed when the key update phase ends and the publishers have been notified of the new key. The *transitional state* ensures that the pub/sub service is continuously functional, which is essential for critical applications. As evaluated in Section 5.5, the time to maintain the transitional state is consistently reduced by the third component of our system: the in-broker subscriptions re-encryption.

We describe below the complete key update protocol phase. The sequence of steps is provided below and illustrated in Figure 5.2. Each step is executed in parallel by the involved entities. The implementation of the protocol is solely based on interactions through the ZooKeeper coordination service. Reaction to new data (e.g., when a new key is available at step 2) or joining mechanisms (e.g., when the KDC waits upon KDMs to complete a distribution) are implemented by the use of ZooKeeper watches on the corresponding datum in the shared namespace.

- **Step 1)** The KDC sets an *updated* flag. Through previously set watches, all KDM_s and KDM_b are notified of a new key availability. The key information (the new key itself or the key update token) is stored in the ZooKeeper namespace. It is encrypted with the corresponding KDMs's public keys.
- **Step 2)** All KDM_s and KDM_b retrieve and decrypt the new key information. They disseminate the key information in their domain.
- **Step 3)** Subscribers obtain the new key. They start using it in encryption, and notify the KDM_s. Brokers obtain the key update token. They start re-encrypting the stored subscriptions. They keep both the new and the old subscriptions sets in a *transitional state*. When the re-encryption is finished, they notify the KDM_s.
- **Step 4)** When they have been notified by all hosts in their domains, KDM_s and KDM_b notify the KDC through creating a *retrieved* flag.
- **Step 5)** The KDC waits for retrieval of retrieved flags from all KDM_s and KDM_b. Then, it sets an *updated* flag to notify all KDM_p about the new key.

- **Step 6)** Each KDM_p retrieves and decrypts the new key information. It notifies the publishers in its domain.
- **Step 7)** Publishers obtain the new key. They start using it in encryption, and notify their KDM_p.
- **Step 8)** Each KDM_p waits for notification from all publishers in its domain. It then sets a *retrieved* flag.
- **Step 9)** The KDC waits for the retrieved flag notifications from all KDM_p. In turn, it notifies the KDM_bs through a *validated* flag.
- **Step 10)** The KDM_bs are notified of the validated flag. They notify the brokers in their domains. This triggers the end of the *transitional state* and the deletion of old subscriptions sets.

Since step 3 is executed in parallel at both subscriber and broker hosts, it might happen that a broker receives a subscription encrypted with an old key after the stored subscriptions have already been re-encrypted. We handle this case by allowing the broker to re-encrypt *on-the-fly* such subscriptions and storing them along the others.

The protocol guarantees that publications encrypted with the new key are sent only after all stored subscriptions are in the new re-encrypted form. Also, subscriptions encrypted with the old key are removed only after the publisher hosts start using the new key. However, due to network asynchrony, it might happen that some publications encrypted with an older key are delayed and arrive at the broker after the subscriptions encrypted with the old key have been removed. To avoid this problem, we make an assumption on a bound of such asynchrony and maintain the transitional state for a short additional time (typically, less than a second).

5.3.3 In-broker subscriptions re-encryption

As defined in Section 5.2, the third requirement of our architecture model is that the encrypted matching scheme supports *in-broker subscriptions re-encryption*. We require a scheme that permits obtaining a key update token K_R that can be used in a transformation τ allowing the untrusted brokers to securely re-encrypt the stored subscriptions. The scheme we selected is Asymmetric Scalar-product Preserving Encryption (ASPE) [Choi et al., 2010]. We first overviewed ASPE in Chapter 2. The original published scheme does not support in-broker subscriptions re-encryption. We have however been able to extend ASPE to support it. In Section 5.4, we prove that this modification to ASPE does not only keep the security strength of the original scheme, but that we were also able to improve its resilience against the attacks on which it was first analyzed.

The original pub/sub adapted ASPE published version [Choi et al., 2010] is focused on the restrictive unidimensional case. We have already thoroughly covered ASPE multidimensional case in Chapter 4 filling gaps in the published version. Therefore, in our work of extending ASPE for in-broker subscription re-encryption we refer to the multidimensional ASPE, which complies better with a practical scenario. For continuity purposes we summarize in the following the description of this case, detailed in Section 4.1 of Chapter 4.

The number of attributes in a publication, and the maximum number of subscription constraints on these attributes is d. Each value v_i of a constraint is encapsulated in an intermediary subscription point $S_i = (s_1, s_2, \ldots, v_i, \ldots, s_d)$. The all other dimensions except v_i are randomly set. $S_{i1} = (q_1, q_2, \ldots, v_i - r_i, \ldots, q_d)$ and $S_{i2} = (q_1, q_2, \ldots, v_i + r_i, \ldots, q_d)$ are

two reference subscription points chosen symmetrically at random and at equal distance r_i from S_i. The r_i values are private parameters in the scheme. P designates a multidimensional publication and q a positive random scaling factor added for security purposes. ASPE uses as key K on the subscriber's side the transpose of an invertible matrix M^T, and on the publisher's side the matrix inverse M^{-1}. As detailed in Section 4.1, points S_{i1}, S_{i2} and P are encrypted as:

$$\begin{aligned} S'_{Ki1} &= E_K(S_{i1}) = M^T(S_{i1}, -0.5 \parallel S_{i1} \parallel)^T \\ S'_{Ki2} &= E_K(S_{i2}) = M^T(S_{i2}, -0.5 \parallel S_{i2} \parallel)^T \\ P'_K &= E_K(P) = M^{-1}q(P, 1)^T \end{aligned}$$

The target is to obtain an expression on $D_i = dist(S_{i1}, P) - dist(S_{i2}, P)$, which can be used to check the matching between a constraint value v_i and the publication P. In that purpose, an untrusted broker receives S'_{Ki1}, S'_{Ki2} and P'_K, and evaluates:

$$(S'_{Ki2} - S'_{Ki1})P'_K = 0.5qD_i. \tag{5.1}$$

Hardened ASPE

Our first contribution is to increase the scheme resilience to the attacks considered in [Wong et al., 2009, Choi et al., 2010]. We enforce the rule that the subscriber computes on its own the difference $S'_{Ki} = S'_{Ki2} - S'_{Ki1}$, and submits S'_{Ki} to the brokers, instead of S'_{Ki2} and S'_{Ki1} as proposed in the original scheme. We discuss the security and practical implications of hardened ASPE in Section 5.4.

ASPE with in-broker subscriptions re-encryption

Our second contribution is the support for in-broker subscriptions re-encryption. The subscription encryption key in ASPE is a matrix M^T. Let us consider a key $K_1 = M^T$ and a key $K_2 = N^T$. We want a key update token K_R and a transformation τ that using K_R re-encrypts a subscription encrypted with K_1 into the same subscription encrypted with K_2.

A constraint value v_i in a subscription S encrypted with K_1 has the form

$$S'_{K_1 i} = S'_{K_1 i2} - S'_{K_1 i1} = M^T((S_{i2}, -0.5 \parallel S_{i2} \parallel) - (S_{i1}, -0.5 \parallel S_{i1} \parallel))^T. \tag{5.2}$$

The same encryption using K_2 results in

$$S'_{K_2 i} = S'_{K_2 i2} - S'_{K_2 i1} = N^T((S_{i2}, -0.5 \parallel S_{i2} \parallel) - (S_{i1}, -0.5 \parallel S_{i1} \parallel))^T. \tag{5.3}$$

From (5.2) and (5.3), we observe that only M^T and N^T change when the key is updated, while the rest of the expression remains the same. Thus, the key update token is $K_R \triangleq N^T M^{T^{-1}}$, and the transformation τ is

$$S'_{K_2 i} = K_R S'_{K_1 i}. \tag{5.4}$$

For extra obfuscation and for rescaling the matrix elements, the new key matrix N^T can be multiplied with a random value r' before distributing K_R to the brokers. This results in $K_R \triangleq (N^T r')M^{T^{-1}}$. The scaling does not impact the matching correctness, and subsequent key updates are completely independent of the previous random factors. We prove the security of the obtained ASPE in-broker subscriptions re-encryption in Section 5.4.

5.4 Security analysis

As we have already discussed in Section 2.1.1 encrypted matching schemes used in pub/sub differ significantly from typical encryption algorithms. Encrypted matching schemes do not necessarily feature a decryption algorithm, and there is no specific model in the literature for evaluating the security of such schemes. The authors of [Raiciu and Rosenblum, 2006] argue that the indistinguishability that can be obtained between encrypted messages is limited by the matching possibility itself. This directly restricts the usage of typical indistinguishability game proofs considered for attacks where the attacker has the power to actively choose either the plaintext or the ciphertext. The security level achieved is typically lower than active attacks, and it is limited to passive models, where the matching result between encrypted messages is an acceptable leak.

The security analysis on the original ASPE scheme, initially proposed in [Wong et al., 2009] for secure kNN queries, covers three attack levels:

- Level 1: the attacker can access only the ciphertexts set C (ciphertext only attack — COA);
- Level 2: the attacker knows C and a subset of plaintext messages P, but not the exact correspondence between P and the corresponding subset in C;
- Level 3: the attacker knows the exact correspondence between the plaintext set P and a subset I of the ciphertexts set C (known plaintext attack — KPA).

The analysis in [Wong et al., 2009] concludes that the scheme is not secure against a Level 3 attack, the attacker being able to form and solve a system of equations that allows obtaining the key. The ASPE adaptation for pub/sub in [Choi et al., 2010] mostly relies on the previous evaluation, without conducting any extra analysis.

5.4.1 Hardened ASPE security

We analyze the security of the hardened ASPE scheme introduced in Section 5.3.3. We follow the same methodology presented in [Wong et al., 2009]. We consider as encryption key the matrix M^T with elements $m_{i,j}$. r_i is the random distance used in the encryption of each constraint as described in Section 5.3.3, and d is the number of attributes in the messages. Let us assume that the attacker has Level 3 knowledge of the mapping between each plaintext value v_i in a constraint and its ciphertext $S'_{Ki} = (e_{1i}, e_{2i}, \ldots, e_{(d+1)i})$. From this correspondence and from $S'_{Ki} = S'_{Ki2} - S'_{Ki1}$, for each v_i the attacker can form the system

$$\begin{gathered} 2r_i(m_{1,i} - v_i m_{1,d+1}) = e_{1i}; \\ \ldots \\ 2r_i(m_{d+1,i} - v_i m_{d+1,d+1}) = e_{(d+1)i}. \end{gathered} \tag{5.5}$$

By solving for r_i in each equation, the system can be rewritten as

$$\frac{e_{1i}}{m_{1,i} - v_i m_{1,d+1}} = \frac{e_{2i}}{m_{2,i} - v_i m_{2,d+1}} = \cdots = \frac{e_{(d+1)i}}{m_{d+1,i} - v_i m_{d+1,d+1}}. \tag{5.6}$$

The terms in (5.6) can be paired in individual equations of the form

$$e_{ji} m_{k,i} - e_{ji} v_i m_{k,d+1} - e_{ki} m_{j,i} + e_{ki} v_i m_{j,d+1} = 0. \tag{5.7}$$

The total number of such equations per constraint is $\binom{d+1}{2}$, which corresponds to all possible combinations of the matrix rows k and j. The unknowns are the elements of the i-th and d+1-th columns of the matrix key M^T.

An attacker can form an extended system with the complete sets of equations obtained in this manner for each constraint value v_i in a subscription. Given that a complete subscription covering all the attributes has d constraints, the total number of equations in this system is $d\binom{d+1}{2}$. The unknowns are all the $(d+1)^2$ elements of the matrix key. For the same key target, the attacker can further augment the system by adding equations from multiple subscriptions for which the mapping between the plaintext and ciphertext is known.

No matter how many subscriptions are in possession of the attacker, we can observe the following. The system is composed of equations having the form of (5.7), which has no free term. Therefore, the system is linear and homogenous. Consequently, the system has at least one solution for the unknown key, i.e., the all-zero matrix. We also know that this is not the only solution, since the key matrix cannot be null. It follows that the solution for the unknown key is the nullspace of the system's coefficient matrix. A nullspace determines an infinity of individual solutions (although having a specific structure) [Strang, 2009]. Therefore, an attacker cannot determine the key with the information given by the system.

The authors of [Wong et al., 2009] follow the same reasoning for the original ASPE scheme by forming a system of equations that can be solved by a Level 3 attack (KPA); the attack is also valid for the pub/sub adapted ASPE version presented in [Choi et al., 2010]. In both cases, the system of equations can be solved because the brokers are given two separate vectors S'_{Ki1} and S'_{Ki2} instead of only $S'_{Ki} = S'_{Ki1} - S'_{Ki2}$ as we propose. It should be noted that if a random value used in encryption is leaked (r_i in (5.5)), then a Level 3 attacker can solve our system like it can solve the scheme from [Choi et al., 2010]. However, guessing a random value used in encryption can be critical also in other schemes (e.g., in [Nabeel et al., 2012] the encryption mechanism uses two random values that must be kept private). Our hardened ASPE thus provides a greater level of security than the original scheme, adding resilience against Level 3 attacks, while the original scheme does not. A drawback of the increased security is that the broker cannot determine containment between encrypted subscriptions anymore (e.g., if a subscription like $value > 100$ is more general than $value > 300$). While it can improve the matching performance, as we have seen in Chapter 3 containment support is however not desirable from a security point-of-view as it may allow inferring the nature of subscriptions based on knowledge of their definition domain.

5.4.2 In-broker subscriptions re-encryption security

We now analyze the security impact of the extension to ASPE for supporting in-broker subscriptions re-encryption. We recall that a key update implies sending to the brokers the matrix product $K_R = (N^T r')M^{T^{-1}}$. To simplify the discussion, and without loss of generality, we consider $r' = 1$ in the following. Each element of $M^{T^{-1}}$ can be expressed using the terms in the original key: $M^{T^{-1}} = \frac{adj(M^T)}{|M^T|}$, where $adj(M^T)$ is the adjugate, and $|M^T|$ the determinant of M^T. Let $N^T(i)$ and $K_R^T(i)$ be the ith column of the matrices N^T and K_R^T, respectively. We also write $[0]_{d+1}$ to denote the null square matrix of size d+1. From the definition of K_R^T, it

follows that

$$\begin{bmatrix} adj(M^T)^T & [0]_{d+1} & \dots & [0]_{d+1} \\ [0]_{d+1} & adj(M^T)^T & \dots & [0]_{d+1} \\ \dots & & & \\ [0]_{d+1} & [0]_{d+1} & \dots & adj(M^T)^T \end{bmatrix} \begin{bmatrix} N^T(1) \\ N^T(2) \\ \dots \\ N^T(d{+}1) \end{bmatrix} = \begin{bmatrix} K_R^T(1)|M^T| \\ K_R^T(2)|M^T| \\ \dots \\ K_R^T(d{+}1)|M^T| \end{bmatrix}. \tag{5.8}$$

This expression corresponds to the system of equations that can be formed by an attacker. If $a_{j,k}$ and $t_{j,k}$ are respectively the elements of $adj(M^T)$ and K_R, the system of equations can be written as

$$\begin{aligned} a_{1,1}n_{1,1}{+}a_{2,1}n_{1,2}{+}\ \dots\ {+}a_{d+1,1}n_{1,d+1} &= t_{1,1}|M^T|; \\ a_{1,2}n_{1,1}{+}a_{2,2}n_{1,2}{+}\ \dots\ {+}a_{d+1,2}n_{1,d+1} &= t_{1,2}|M^T|; \\ &\dots \\ a_{1,d+1}n_{d+1,1}{+}a_{2,d+1}n_{d+1,2}{+}\ \dots\ {+}a_{d+1,d+1}n_{d+1,d+1} &= t_{d+1,d+1}|M^T|. \end{aligned} \tag{5.9}$$

For a given instance of the old key M^T and considering the elements of the new key N^T as unknowns, we can observe that (5.9) is a linear system. This illustrates that for each solution that an attacker might obtain for M^T, he can obtain a specific dependent solution for N^T. As proved before, the set of solutions for M^T in case of a Level 3 attack is expressed through a nullspace form and is of infinite size. Hence, without a solution for M^T, an attacker cannot determine the dependent solution for N^T (and vice versa).

The security of in-broker subscriptions re-encryption support for ASPE thus depends on the security of the original ASPE scheme and does not impact it. An attacker might try to prune the search space for M^T by eliminating cases where (5.9) has no solution for N^T, but this is not possible since we can prove that for every matrix M^T, (5.9) has a nonzero solution for N^T. The coefficient matrix of system (5.9) is the first term in expression (5.8), which is a diagonal block matrix. Its determinant is the product of the determinants of the diagonal blocks $adj(M^T)^T$. Since M^T is invertible, we know that the determinant of $adj(M^T)^T$ is nonzero. It follows that the determinant of the coefficient matrix in system (5.9) is nonzero, which proves that a nonzero solution exists for N^T.

Our solution maintains the same level of security as the original scheme against a direct attempt to determine a key. However, we are aware that the proposed ASPE in-broker subscriptions re-encryption presents issues in regard to forward secrecy. More precisely, if one key M^T is leaked, then an attacker having access to an untrusted broker can try to obtain the key N^T from M^T and $K_R = (N^T r')M^{T^{-1}}$. The r' random scaling factor obfuscates the exact value of the new key, adding some resilience against obtaining it. Optionally, for increased security, a key update can be periodically performed by distributing the key only to trusted parties without using the in-broker subscriptions re-encryption. This will require subscription resubmission, but since this is a background operation, it can be done at a slower pace and without strict constraints on the key update duration. The transitional state design described in Section 5.3 applies also to this case, thus if the re-subscriptions flow do not overwhelm the brokers, the delays of processing publications shall remain unaffected.

In case of a Level 2 attack, the attacker knows a subset of plaintext messages, but not the correspondence between these and the ciphertexts. The original ASPE scheme is analyzed in such conditions by [Wong et al., 2009] against a brute-force attack. This relies on solving equation systems, formed similarly to the case discussed for Level 3, but on a trial-and-error basis. Due to the search space, the attack is considered unfeasible by [Wong et al., 2009] even if the systems can be solved. In our hardened ASPE, the corresponding

systems of equations do not provide an exact solution for the key. Therefore, the resilience against a Level 2 attack is increased.

5.5 Evaluation

We present in this section the evaluation of the DynamiK key management architecture. Our implementation targets the StreamHub [Barazzutti et al., 2013, Heinze et al., 2014] content-based pub/sub engine. StreamHub aims at filtering at high-throughput and with low delays. It was designed for deployment on public clouds and for using encrypted matching. StreamHub splits the pub/sub service in three successive operations: partitioning, matching and notification. It maps these operations to a chain of operators deployed on a stream processing engine. Each operator is deployed over several machines and its load is partitioned. The central operation, matching, maps to the *Matcher* operator. The Matcher stores subscriptions and matches publications against them using ASPE with in-broker subscriptions re-encryption. We also equip the Matcher operator instance with the ZooKeeper-based key management system presented in Section 5.3. Our implementation is done in C++ and for the key exchange protocol uses the Crypto++ and ZooKeeper libraries.[1] We deploy the system on 12 nodes each with 8 Xeon cores at 2.4 GHz and 8 GB of memory. Six nodes are dedicated to the key management infrastructure: KDC, KDMs, and ZooKeeper server replicas. Six nodes support the StreamHub operators, among which 3 where we deploy 15 instances of the Matcher that perform the filtering in parallel.[2] Each of these Matcher instances corresponds to a broker in the DynamiK model terminology. The remaining nodes are used for workload injection. We emulate the set of subscribers and one publisher on a single host. The main purpose of our evaluation is to exemplify the critical advantage brought by the in-broker subscriptions re-encryption in DynamiK, and the impact on the service quality of the pub/sub engine itself.

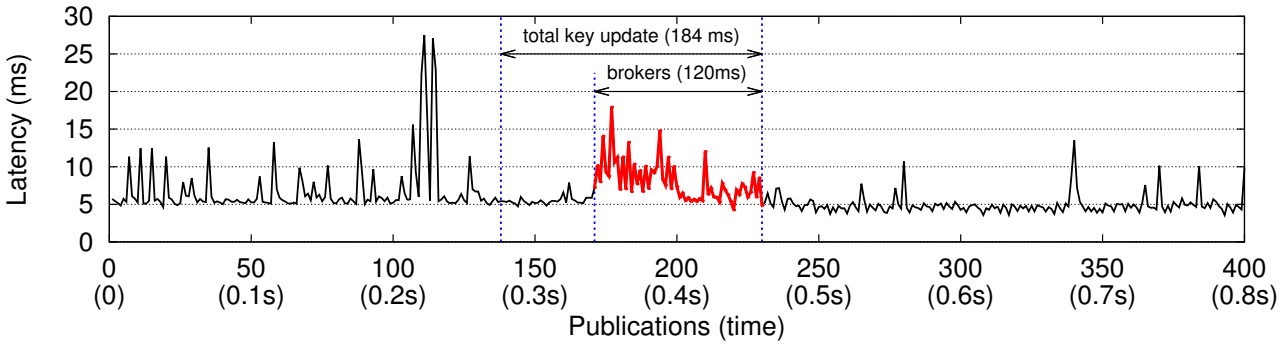

Figure 5.3: Impact of a key update on the notification delay.

We start by evaluating the impact of a key update on the pub/sub engine performance, when in-broker subscriptions re-encryption is enabled. We use a constant flow of 500 publications per second. The system stores 5,000 subscriptions replicated on the 15 Matcher instances. We use a synthetic workload of subscriptions and publications, each with 4

[1] `http://www.cryptopp.com/` and `http://zookeeper.apache.org/`

[2] The subscriptions are replicated on all Matchers and we split the incoming flow of publications. Such approach is possible, as considered in [Barazzutti et al., 2012a]. It is also possible to do the opposite, as detailed in [Barazzutti et al., 2013].

constraints or attributes. Each incoming publication matches 0.2% of the stored subscriptions, triggering 10 notifications. We trigger a key update when sending the 138th publication. Figure 5.3 presents the delay between the emission of a publication and the reception of the first notification. The total key update time represents the time measured at the central KDC for the duration of the entire key update phase across all domains. The brokers key update time is the time measured at the broker domain's KDM for the protocol phases that involve this domain, starting with a key update notification trigger, up to the final confirmation of deleting the previous-key encrypted subscriptions. In this experiment the total time for the key update is 184 ms, with 120 ms specifically for the protocol steps involving the broker domain.

The impact of the in-broker subscriptions re-encryption on the experienced delay in normal conditions is present, but minimal. The 60 publications processed during the key update phase involving the brokers have a median notification delay of 7.13 ms (average 8.05 ms, dev. 2.6 ms) compared to the median notification delay of 4.7 ms (average 4.7 ms, dev. 1.24 ms) for the 160 publications that follow the key update.

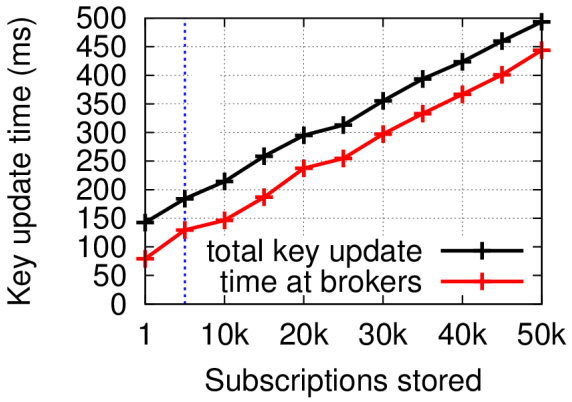

Figure 5.4: Impact of subscriptions set size on key update time.

In Figure 5.4, we present the impact of the number of subscriptions stored at the brokers on the key update time, from 1 subscription to 50,000 subscriptions. The in-broker subscriptions re-encryption requires a linear time as expected, but is able to re-encrypt a massive number of subscriptions in less than half a second. This is orders of magnitude less than the time it would take to all subscribers to re-encrypt and re-submit the same set. We actually emulate such a situation with 5,000 subscriptions when in-broker subscriptions re-encryption is disabled. In this case, the subscribers need to re-encrypt and re-send. We emulate the limitation at the transport layer by capping the bandwidth at a maximal number of subscriptions sent per second. When the key update is triggered, the workload injection machine emulates the resubmission by re-encrypting and sending 100, 250 and 500 subscriptions per second. We present the measured notification delays and the total key update time in Figure 5.5. The key update time is now completely dominated by the time required for subscribers to re-send their subscriptions. If the key update is triggered to revoke a previously authorized subscriber, the revocation may only be effective after several seconds. This leaves a much longer window of opportunity to an attacker than when using in-broker subscriptions re-encryption. Another important aspect is the impact on the measured notification delays. This depends on the rate of received re-subscriptions. Due to the large number of publications processed during the key update, we present the median of the notification delays rather than all actual times as we did for Figure 5.3. The increase in

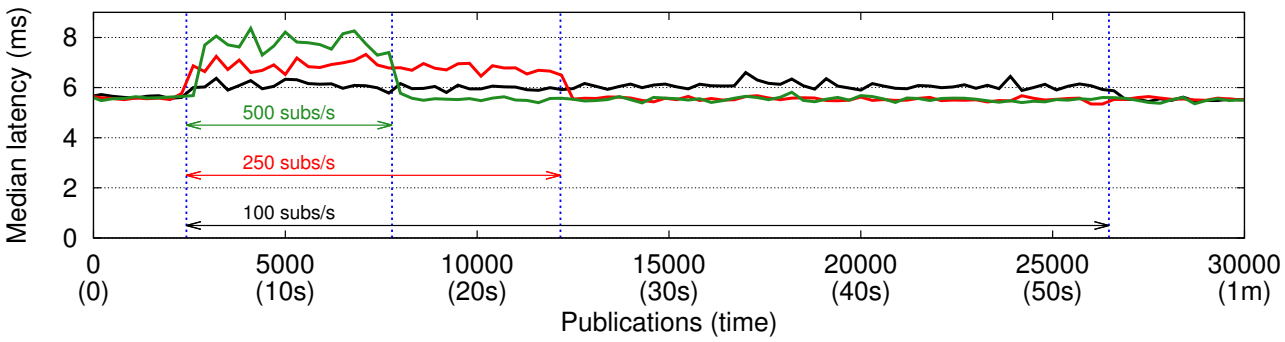

Figure 5.5: Key update impact without in-broker subscriptions re-encryption.

notification delay results from the extra load imposed by the re-subscriptions, which have to be processed by the Matchers. It illustrates the tradeoff between the total key update time and its impact on the service performance. Trying to minimize the key update time by increasing the re-subscription rate adds a clear negative impact on notification delivery time. Using in-broker subscriptions re-encryption is clearly a better approach in this respect.

The above evaluation targets the impact of the in-broker subscriptions re-encryption in the distribution protocol, which is the main innovative contribution of our model and implementation. However, the simplification of emulating the set of subscribers and the publisher on a single host impacts the overall system key dissemination time by abstracting the group dissemination time for the subscribers set. In a real large-scale deployment, this time will depend on the number of subscribers and their transport links characteristics.[3] We present the evaluation of the time taken by the key dissemination protocol when using multiple hosts, including multiple subscribers, in Figure 5.6.

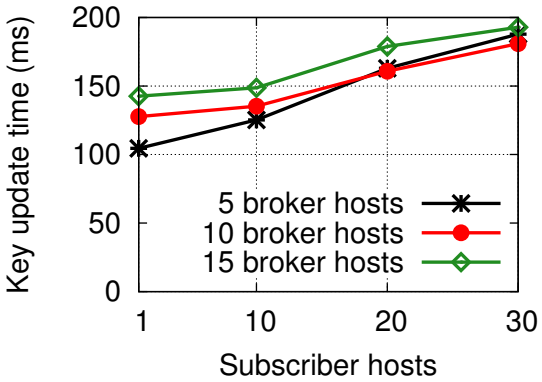

Figure 5.6: Key distribution time.

To highlight the costs of the entire protocol itself each broker stores only 1 subscription. In-broker subscriptions re-encryption takes negligible time. The increase in time for a key update resulting from an increase in the number of hosts is limited: only up to 6 ms per added host. We observe in our evaluation that the key distribution time is dominated by the largest domain group. This is the reason why the variations obtained for 30 subscribers with 5, 10, and 15 brokers are not significant (in the range of 20 ms). We believe that splitting the

[3]External protocols [Sherman and McGrew, 2003, Perrig et al., 2001] can be adapted to optimize the distribution.

domains into smaller groups might provide an interesting tradeoff in terms of performance but leave this investigation for future work.

5.6 Summary

In this chapter we presented DynamiK, a novel key management architecture for privacy-preserving pub/sub systems. Our solution addresses two important constraints associated with pub/sub key management. In terms of design, we preserve a partial decoupling between publishers and subscribers. In terms of performance and practicality, we are the first to take in account the problem of subscription resubmission when the key is updated. We propose the *in-broker subscriptions re-encryption*, a solution that eliminates the need for subscribers to re-encrypt and re-send their subscriptions upon such a key update. We add a novel mechanism for the ASPE encrypted matching scheme [Choi et al., 2010] to support this feature, as well as additional security strengthening. Our implementation of the key update protocol leverages the support of the ZooKeeper [Hunt et al., 2010] coordination service and is instantiated for the StreamHub [Barazzutti et al., 2013] high-performance pub/sub engine.

Our model is generic enough to fit other schemes that could be extended to support in-broker subscriptions re-encryption, such as RASP [Xu et al., 2014]. We will investigate this possibility. Our future work also includes the integration of encrypted matching and the DynamiK key management architecture with secure group communication protocols that fit our model requirements [Sherman and McGrew, 2003, Perrig et al., 2001]. This can lead to performance improvements, especially when domains grow large.

We finally consider that our work could be a first step in a larger-scale agenda of research for key management protocols adapted to scenarios where key-dependent encrypted queries are stored in untrusted domains and need to be updated in-place upon a key update. Pub/sub is not the only context where using encrypted and persistent queries in untrusted domains makes sense. Operators of Complex Event Processing (CEP) engines and active databases in general could also benefit from similar techniques, using other encrypted processing techniques (e.g., searchable symmetric encryption [Curtmola et al., 2006]).

Chapter 6

Conclusion and open directions

This thesis is focused on techniques for supporting confidential publish/subscribe. In Chapter 1, we identified two main research directions for providing pub/sub confidentiality. The first direction is to allow an untrusted party (the broker) to perform the matching operation over encrypted publications and encrypted subscriptions. For this, the schemes used follow a specific paradigm, which we named encrypted matching. The second direction is towards security models, where providing confidentiality is facilitated by access control mechanisms establishing functions and rights for the system entities. We reviewed existing work in Chapter 2. The recent amount of research and advances proves that confidentiality in pub/sub systems is a topic of high interest. Nevertheless, we can also observe gaps, especially related to functionality support or performance drawbacks. We attempt to cover these in our work and, as discussed in the introduction, we specifically target solving issues in solutions based on encrypted matching. Our work is not completely separate, however, from the design of security models. Combining existing solutions that target different issues is just one of the open directions in pub/sub confidentiality research, which we summarize in the following.

Key management

No existing confidentiality-preserving pub/sub solution comes with a key management procedure that is practically viable. Each presents unsolved key management challenges.

The first challenge lies in the fundamental decoupling of publish/subscribe: publishers do not know the destination of the publications, and subscribers do not know their origin. This represents a major issue for the necessary exchange of common needed key information between parties. We saw that solutions based on attribute-based encryption (ABE), such as [Ion et al., 2012, Tariq et al., 2010], could partially overcome this problem by associating keys with messages instead of associating them with system nodes. However, even with this strategy, a trusted authority is required to provide some common information to the participants. As we did in the solution presented in Chapter 5, one way to address this problem is to consider weaker decoupling assumptions, which is reasonable for several applications, like the stock exchange scenario we presented. The idea is that the host-to-host pub/sub is still decoupled, but the association between publisher and subscriber domains is known. In this loosely decoupled setting it might be possible to adapt standard key management solutions for secure group communication. We proposed as an example a simple protocol based on public key encryption, and using ZooKeeper [Hunt et al., 2010] to ensure availability. As referenced in Chapter 5 existing secure group management protocols such as OFT (One-way Function Trees) [Sherman and McGrew, 2003] or ELK [Perrig et al., 2001] might be adapted for pub/sub. A reason for this is to address further issues like scalability in case of large client numbers. Secure group key management is a complex research area spanning a variety

of techniques and protocol architectures [Rafaeli and Hutchison, 2003, Zou et al., 2005]. Adapting secure group communication for pub/sub systems seems nevertheless promising and should be studied further.

Another key management challenge, that we considered in Chapter 5 is the key update problem in the context of encrypted matching schemes. The untrusted brokers store encrypted subscriptions on which they must perform matching operations, making the key update problematic. When publishers use the new key to encrypt the publications, matching over subscriptions stored by the broker but still encrypted with an older key does not work. The naive solution is that at each key update, the subscribers resubmit their subscriptions re-encrypted with the new key, which is prohibitive for large systems. This prevents the successful deployment of confidentiality preserving pub/sub systems in critical environments. We have introduced modifications to the ASPE [Choi et al., 2010] encrypted matching scheme in Chapters 4 and 5, through which this problem is solved by a support technique we named *in-broker subscription re-encryption*. We believe a similar approach can be taken in case of other encrypted matching schemes. In particular we consider the adaptation of RASP [Xu et al., 2014] for pub/sub, for supporting this functionality. Searching ways to adapt other schemes, possibly more advantageous in terms of performance or functionality is an attactive direction of further development.

Confidentiality, performance and functional limitations

Some schemes, like ABE-based used for payload encryption, have intrinsic confidentiality limitations under certain conditions. These can leak information since keys, associated with message content, can be assigned to individual constraints. Consider a subscriber who is authorized to receive publications about the two subscriptions *(patient = "John" ∧ injury cause = "car accident")* and *(patient name = "Smith" ∧ injury cause = "fall")*. If the keys are associated to each attribute (a typical ABE based design), then the subscriber can also decrypt the publication *(patient = "John", injury cause = "fall")* for which it is not authorized. This situation was first observed by [Tariq et al., 2010], and to our knowledge they are the only by now who took it into account. To avoid such cases, the encryption scheme should be capable of binding multiple attribute keys into one single key per subscription. The presence of this issue still remains to be investigated for other schemes that use ABE.

Other schemes also face performance/confidentiality tradeoffs. For instance, encrypted matching is currently significantly slower than plaintext matching. Optimization tools exploiting subscription containment can be developed to speed up the process. Unfortunately, if containment over encrypted subscriptions is not permitted for confidentiality reasons, then these tools cannot be used. One way to mitigate this problem is to use a security model allowing to keep some of the data in plaintext, however, this can only be used for non-sensitive data. We have presented our take on the problem with the *prefiltering* solution developed in Chapter 3. Our solution, based on an additional Bloom filter encoding, quickly rejects a fraction of the non-matching publications before using the costly encrypted matching. The mechanism presents an attractive tradeoff between performance increases and confidentiality leaks. While efficient encrypted matching is a straightforward goal in the ongoing research, there exists also a side of the problem that relates with the scheme design. More precisely it is desirable that an encrypted matching scheme uses a modular architecture, where containment is optional (most of the schemes actually do this: [Choi et al., 2010, Raiciu and Rosenblum, 2006, Nabeel et al., 2012], but there are some which do not, e.g., [Tariq et al., 2010]). This

way, containment information can be leveraged to improve matching performance in selective cases, when confidentiality requirements are less stringent.

Besides the mentioned confidentiality and performance issues, encrypted matching schemes often present also limitations in the expressiveness of the supported encrypted subscriptions. For instance, string constraints and numerical inequalities are currently not supported in many of the schemes overviewed in Section 2.1.2. We believe that a promising path to overcome this issue is to investigate the possibility to adapt techniques developed for encrypted queries on databases [Kamara, 2013].

Attack model limitations

Most of the techniques using encrypted matching consider confidentiality threats under the passive *honest-but-curious* model. In practice, malicious entities might actively attack the pub/sub system. For instance, a malicious subscriber can try to corrupt several brokers on the public cloud and mount attacks using the capabilities of the colluded set of entities under his control. Although some of the schemes we overviewed in Chapter 2 discuss how to mitigate some active attacks [Ion et al., 2012, Choi et al., 2010], the overall resilience against these threats is currently poorly understood and should be studied further. In this regard, the security models overviewed in Section 2.2 are more flexible, since they are not typically constrained to a specific cryptographic algorithm and can use classic schemes proven secure for a wider range of scenarios.

Besides *confidentiality* provisioning, it is essential to consider the *integrity* of the pub/sub messages and the overall *availability* of the system, which are often targeted by active attacks. There is very little work done about these important topics in the context of secure pub/sub, and we think that they could be studied further.

Toward security models with encrypted matching

To conclude our work, we have to admit that providing confidentiality in a pub/sub architecture without encrypted matching seems more easier than with encrypted matching, since well-established security models can be used that rely on thoroughly tested classic encryption algorithms. However, without encrypted matching, the routing capabilities which are the very core of content-based pub/sub systems are significantly decreased. In our humble opinion, this current state of affair is only temporary.

The goal of homomorphic encryption is to do useful processing on encrypted data without leaking sensitive information. This is a problem much more general than its application in pub/sub, and a very active field of research more relevant than ever with the advent of cloud computing. Unless most interesting encrypted matching applications for pub/sub can be proven to be intrinsically non confidential (we think this is highly improbable), it is clear for us that the numerous benefits of encrypted matching will encourage researchers to keep working on this problem. As new algorithms are developed and potential attacks are studied, encrypted matching will be integrated in good security models and pub/sub will become an attractive solution for a large number of real-life applications.

Bibliography

Adi, A., Botzer, D., Nechushtai, G., and Sharon, G. (2006). Complex event processing for financial services. In *Proceedings of IEEE Services Computing Workshops*, SCW, pages 7–12, Chicago, USA. IEEE Computer Society.

Andrews, G. E. (1998). *The Theory of Partitions*. Cambridge Mathematical Library.

Astley, M., Auerbach, J., Bhola, S., Buttner, G., Kaplan, M., Miller, K., Robert Saccone, J., Strom, R., Sturman, D. C., Ward, M. J., and Zhao, Y. (2004). Achieving scalability and throughput in a publish/subscribe system. Research Report RC23103, IBM.

Bacon, J., Eyers, D., Moody, K., and Pesonen, L. I. W. (2005). Securing publish/subscribe for multi-domain systems. In *Proceedings of the ACM/IFIP/USENIX International Conference on Middleware*, Middleware, pages 1–20, Grenoble, France. Springer-Verlag.

Bacon, J., Eyers, D., Singh, J., Shand, B., Migliavacca, M., and Pietzuch, P. (2010). Security in multi-domain event-based systems. *it - Information Technology*, 51(5):277–284.

Bacon, J., Eyers, D. M., Singh, J., and Pietzuch, P. R. (2008). Access control in publish/subscribe systems. In *Proceedings of the 2nd international conference on Distributed event-based systems*, DEBS, pages 23–34, Rome, Italy. ACM.

Bacon, J., Moody, K., and Yao, W. (2002). A model of oasis role-based access control and its support for active security. *ACM Transactions on Information and System Security*, 5(4):492–540.

Barazzutti, R., Felber, P., Mercier, H., Onica, E., Pineau, J.-F., Rivière, E., and Fetzer, C. (2012a). Infrastructure provisioning for scalable content-based routing: Framework and analysis. In *Network Computing and Applications (NCA), 2012 11th IEEE International Symposium on*, NCA.

Barazzutti, R., Felber, P., Mercier, H., Onica, E., and Rivière, E. (2012b). Thrifty privacy: Efficient support for privacy-preserving publish/subscribe. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, DEBS.

Barazzutti, R. P., Felber, P., Fetzer, C., Onica, E., Pasin, M., Pineau, J.-F., Rivière, E., and Weigert, S. (2013). Streamhub: A massively parallel architecture for high-performance content-based publish/subscribe. In *Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems*, DEBS'13, Arlington, TX, USA.

Bellare, M., Canetti, R., and Krawczyk, H. (1996). Keying hash functions for message authentication. In *16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO.

Bellare, M., Rogaway, P., and Wagner, D. (2003). Eax: A conventional authenticated-encryption mode. Cryptology ePrint Archive, Report 2003/069. `http://eprint.iacr.org/`.

Belokosztolszki, A., Eyers, D. M., Pietzuch, P. R., Bacon, J., and Moody, K. (2003). Role-based access control for publish/subscribe middleware architectures. In *Proceedings of the 2nd international workshop on Distributed event-based systems*, DEBS, pages 1–8, San Diego, USA. ACM.

Bertino, E. and Ferrari, E. (2002). Secure and selective dissemination of xml documents. *ACM Transactions on Information and System Security*, 5(3):290–331.

Bethencourt, J., Sahai, A., and Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, SP, pages 321–334, Berkeley, USA. IEEE Computer Society.

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Comm. of the ACM*, 13(7).

Boneh, D. and Waters, B. (2007). Conjunctive, subset, and range queries on encrypted data. In *Proceedings of the 4th conference on Theory of cryptography*, TCC, pages 535–554, Amsterdam, The Netherlands. Springer-Verlag.

Carzaniga, A., Rosenblum, D. S., and Wolf, A. L. (2001). Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383.

Chand, R. and Felber, P. (2004). Xnet: a reliable content based publish subscribe system. In *Proceedings of the 23rd Symposium on Reliable Distributed Systems*, SRDS, pages 264–273, Florianopolis, Brazil. IEEE Computer Society.

Choi, S., Ghinita, G., and Bertino, E. (2010). A privacy-enhancing content-based publish/subscribe system using scalar product preserving transformations. In *Database and Expert Systems Applications*, volume 6261 of *Lecture Notes in Computer Science*, pages 368–384. Springer Berlin / Heidelberg.

Cugola, G., Nitto, E. D., and Fuggetta, A. (2001). The jedi event-based infrastructure and its application to the development of the opss wfms. *IEEE Transactions on Software Engineering*, 27(9):827–850.

Curtmola, R., Garay, J., Kamara, S., and Ostrovsky, R. (2006). Searchable symmetric encryption: Improved definitions and efficient constructions. In *CCS 2006: 13th ACM conference on Computer and communications security*.

Daemen, J. and Rijmen, V. (2002). *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer Verlag.

Dong, C., Russello, G., and Dulay, N. (2008). Shared and searchable encrypted data for untrusted servers. In *Data and Applications Security XXII*, volume 5094 of *Lecture Notes in Computer Science*, pages 127–143. Springer-Verlag.

El Gamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer Berlin / Heidelberg, New York, NY, USA.

Eugster, P. T., Felber, P., Guerraoui, R., and Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM Computing Survey*, 35(2):114–131.

Eze, B., Kuziemsky, C., Peyton, L., Middleton, G., and Mouttham, A. (2010). Policy-based data integration for e-health monitoring processes in a b2b environment: experiences from canada. *Journal of theoretical and applied electronic commerce research*, 5(1):56–70.

Farrell, S. and Housley, R. (2002). An internet attribute certificate profile for authorization.

Fiege, L., Zeidler, A., Buchmann, A., Kilian-Kehr, R., Mühl, G., and Darmstadt, T. (2004). Security aspects in publish/subscribe systems. In *Proceedings of the Third International Workshop on Distributed Event-Based Systems*, DEBS, Edinburgh, Scotland, UK. IEEE Computer Society.

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

Graham, R. L., Knuth, D. E., and Patashnik, O. (1994). *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing, second edition.

Gupta, A., Sahin, O. D., Agrawal, D., and Abbadi, A. E. (2004). Meghdoot: content-based publish/subscribe over p2p networks. In *5th ACM/IFIP/USENIX Intl. conference on Middleware*.

Heinze, T., Martin, A., Pasin, M., Barazzutti, R., Felber, P., Jerzak, Z., Onica, E., and Riviere, E. (2014). Elastic scaling of a high-throughput content-based publish/subscribe engine. ICDCS'14.

Hunt, P., Konar, M., Junqueira, F. P., and Reed, B. (2010). Zookeeper: Wait-free coordination for internet-scale systems. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'10, pages 11–11, Berkeley, CA, USA. USENIX Association.

Ion, M., Russello, G., and Crispo, B. (2010a). An implementation of event and filter confidentiality in pub/sub systems and its application to e-health. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS, pages 696–698, Chicago, USA. ACM.

Ion, M., Russello, G., and Crispo, B. (2010b). Supporting publication and subscription confidentiality in pub/sub networks. In *Security and Privacy in Communication Networks*, volume 50 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer Berlin Heidelberg, Singapore.

Ion, M., Russello, G., and Crispo, B. (2012). Design and implementation of a confidentiality and access control solution for publish/subscribe systems. *Computer Networks*, 56(7):2014–2037.

Jacobsen, H.-A., Cheung, A., Lia, G., Maniymaran, B., Muthusamy, V., and Kazemzadeh, R. S. (2010). *The PADRES Publish/Subscribe System*. Hershey: IGI Global.

Jafarpour, H., Mehrotra, S., Venkatasubramanian, N., and Montanari, M. (2009). MICS: an efficient content space representation model for publish/subscribe systems. In *3rd ACM International Conference on Distributed Event-Based Systems*, DEBS.

Jakobsson, M. (1999). On quorum controlled asymmetric proxy re-encryption. In *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 632–632. Springer-Verlag, London, UK, UK.

Kamara, S. (2013). How to search on encrypted data. Presentation Slides.

Khurana, H. (2005). Scalable security and accounting services for content-based publish/subscribe systems. In *Proceedings of the ACM symposium on Applied computing*, SAC, pages 801–807, Santa Fe, USA. ACM.

Kurpicz, M. (2013). Towards efficient privacy-preserving sql processing in untrusted clouds. Master's thesis, University of Neuchâtel.

Li, G., Hou, S., and Jacobsen, H.-A. (2005). A unified approach to routing, covering and merging in publish/subscribe systems based on modified binary decision diagrams. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, ICDCS, pages 447–457, Columbus, USA. IEEE Computer Society.

Li, J., Lu, C., and Shi, W. (2004). An efficient scheme for preserving confidentiality in content-based publish/subscribe systems. Technical Report GIT-CC-04-01, Georgia Institute of Technology.

Machanavajjhala, A., Vee, E., Garofalakis, M., and Shanmugasundaram, J. (2008). Scalable ranked publish/subscribe. In *Proceedings of the Very Large Data Bases Endowment*, volume 1 of *VLDB*, pages 451–462. VLDB Endowment.

Martin-Flatin, J.-P., Znaty, S., and Hubaux, J.-P. (1999). A survey of distributed enterprise network and systems management paradigms. *Journal of Network and Systems Management*, 7:9–26.

Mercier, H., Onica, E., Rivire, E., and Felber, P. (2013). Performance/security tradeoffs for content-based routing supported by Bloom filters. In *Proceedings of the 20th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 8179 of Lecture Notes in Computer Science, pages 129–140, Ischia, Italy.

Mühl, G. (2001). Generic constraints for content-based publish/subscribe. In Batini, C., Giunchiglia, F., Giorgini, P., and Mecella, M., editors, *Cooperative Information Systems*, volume 2172 of *Lecture Notes in Computer Science*, pages 211–225. Springer Berlin / Heidelberg.

Mühl, G. (2002). *Large-Scale Content-Based Publish-Subscribe Systems*. PhD thesis, TU Darmstadt, Darmstadt.

Nabeel, M., Shang, N., and Bertino, E. (2009). Privacy-preserving filtering and covering in content-based publish subscribe systems. CERIAS Tech. Report 15, Purdue University, West Lafayette, IN.

Nabeel, M., Shang, N., and Bertino, E. (2012). Efficient privacy preserving content based publish subscribe systems. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, SACMAT '12, pages 133–144, New York, NY, USA. ACM.

Nanavati, M., Colp, P., Aiello, B., and Warfield, A. (2014). Cloud security: A gathering storm. *Commun. ACM*, 57(5):70–79.

Newman, M. E. J. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46:323–351.

Oki, B., Pfluegl, M., Siegel, A., and Skeen, D. (1993). The information bus: an architecture for extensible distributed systems. In *Proceedings of the 14th ACM symposium on Operating systems principles*, SOSP, pages 58–68, Asheville, USA. ACM.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag.

Pal, P., Lauer, G., Khoury, J., Hoff, N., and Loyall, J. (2012). P3s: a privacy preserving publish-subscribe middleware. In *Proceedings of the 13th International Middleware Conference*, Middleware '12, pages 476–495, New York, NY, USA. Springer-Verlag New York, Inc.

Parzyjegla, H., Graff, D., Schröter, A., Richling, J., and Mühl, G. (2010). From active data management to event-based systems and more. chapter Design and implementation of the Rebeca publish/subscribe middleware, pages 124–140. Springer-Verlag, Berlin, Heidelberg.

Perera, S. and Gannon, D. (2009). A scalable and robust coordination architecture for distributed management. Technical report, Indiana University.

Perl, H., Mohammed, Y., Brenner, M., and Smith, M. (2012). Fast confidential search for bio-medical data using Bloom filters and homomorphic cryptography. In *E-Science (e-Science), 2012 IEEE 8th International Conference on*, pages 1–8.

Perrig, A., Song, D., and Tygar, J. D. (2001). Elk, a new protocol for efficient large-group key distribution. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, SP '01, pages 247–, Washington, DC, USA. IEEE Computer Society.

Pesonen, L. I. W. and Bacon, J. (2005). Secure event types in content-based, multi-domain publish/subscribe systems. In *Proceedings of the 5th international workshop on Software engineering and middleware*, SEM, pages 98–105, Lisbon, Portugal. ACM.

Pesonen, L. I. W., Eyers, D. M., and Bacon, J. (2007a). Access control in decentralised publish/subscribe systems. *Journal of Networks*, 2(2):57–67.

Pesonen, L. I. W., Eyers, D. M., and Bacon, J. (2007b). Encryption-enforced access control in dynamic multi-domain publish/subscribe networks. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, DEBS, pages 104–115, Toronto, Canada. ACM.

Pietzuch, P. R. and Bacon, J. (2002). Hermes: A distributed event-based middleware architecture. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, ICDCS, pages 611–618, Vienna, Austria. IEEE Computer Society.

Pietzuch, P. R., Shand, B., and Bacon, J. (2004). Composite event detection as a generic middleware extension. *IEEE Network*, 18(1):44–55.

Pohlig, S. and Hellman, M. (1978). An improved algorithm for computing logarithms over gf(p) and its cryptographic significance (corresp.). *IEEE Transactions on Information Theory*, 24(1):106 – 110.

Popa, R. A., Redfield, C. M. S., Zeldovich, N., and Balakrishnan, H. (2011). CryptDB: protecting confidentiality with encrypted query processing. In *23rd ACM Symposium on Operating Systems Principles*, SOSP.

Rafaeli, S. and Hutchison, D. (2003). A survey of key management for secure group communication. *ACM Computing Surveys*, 35:309–329.

Raiciu, C. and Rosenblum, D. S. (2006). Enabling confidentiality in content-based publish/subscribe infrastructures. In *Proceedings of the Second IEEE/CreatNet International Conference on Security and Privacy in Communication Networks*, Securecomm, pages 1–11, Baltimore, USA. IEEE Computer Society.

Ristenpart, T., Tromer, E., Shacham, H., and Savage, S. (2009). Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS, pages 199–212, Chicago, USA. ACM.

Rose, I., Murty, R., Pietzuch, P., Ledlie, J., Roussopoulos, M., and Welsh, M. (2007). Cobra: Content-based filtering and aggregation of blogs and rss feeds. In *Proceedings of the 4th USENIX Symposium on Networked Systems Design & Implementation*, NSDI, pages 29–42, Cambridge, USA. USENIX Association.

Rowstron, A., Kermarrec, A.-M., Castro, M., and Druschel, P. (2001). Scribe: The design of a large-scale event notification infrastructure. In Crowcroft, J. and Hofmann, M., editors, *Networked Group Communication*, volume 2233 of *Lecture Notes in Computer Science*, pages 30–43. Springer Berlin / Heidelberg.

Russell, J. (2002). Oracle 9i, application developer's guide - fundamentals. Developer Manual.

Segall, B., Arnold, D., Boot, J., Henderson, M., and Phelps, T. (2000). Content based routing with elvin4. In *Proceedings of the Australian UNIX Users Group*, AUUG, Canberra, Australia.

Segall, W. and Arnold, D. (1997). Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings of the Australian UNIX Users Group*, AUUG, Brisbane, Australia.

Sherman, A. T. and McGrew, D. A. (2003). Key establishment in large dynamic groups using one-way function trees. *IEEE Transactions on Software Engineering*, 29(5):444–458.

Shi, E., Bethencourt, J., Chan, T.-H. H., Song, D., and Perrig, A. (2007). Multi-dimensional range query over encrypted data. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, SP, pages 350–364, Berkeley, USA. IEEE Computer Society.

Shikfa, A., Ãnen, M., and Molva, R. (2008). Privacy-preserving content-based publish/Subscribe networks. Technical Report EURECOM+2633, Eurecom.

Shikfa, A., Önen, M., and Molva, R. (2009). Privacy-preserving content-based publish/subscribe networks. In *Emerging Challenges for Security, Privacy and Trust*, volume 297 of *IFIP Advances in Information and Communication Technology*. Springer Boston.

Singh, J., Eyers, D. M., and Bacon, J. (2011). Disclosure control in multi-domain publish/subscribe systems. In *Proceedings of the 5th ACM international conference on Distributed event-based system*, DEBS, pages 159–170, New York, USA. ACM.

Somorovsky, J., Heiderich, M., Jensen, M., Schwenk, J., Gruschka, N., and Lo Iacono, L. (2011). All your clouds are belong to us: security analysis of cloud management interfaces. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, CCSW, pages 3–14, Chicago, USA. ACM.

Song, D. X., Wagner, D., and Perrig, A. (2000). Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, SP, pages 44–55, Berkeley, USA. IEEE Computer Society.

Srivatsa, M. and Liu, L. (2005). Securing publish-subscribe overlay services with eventguard. In *Proceedings of the 12th ACM conference on Computer and communications security*, CCS, pages 289–298, Alexandria, USA. ACM.

Srivatsa, M. and Liu, L. (2007). Secure event dissemination in publish-subscribe networks. In *Proceedings of the IEEE International Conference on Distributed Computing Systems*, ICDCS, page 22, Toronto, Canada. IEEE Computer Society.

Srivatsa, M., Liu, L., and Iyengar, A. (2011). Eventguard: A system architecture for securing publish-subscribe networks. *ACM Transactions on Computer Systems*, 29(4):10:1–10:40.

Strang, G. (2009). *Introduction to Linear Algebra, 4th edition*. Wellesley-Cambridge Press and SIAM.

Strom, R., Banavar, G., Chandra, T., Kaplan, M., Miller, K., Mukherjee, B., Sturman, D., and Ward, M. (1998). Gryphon: An information flow based approach to message brokering. *ArXiv Computer Science e-prints*.

Tariq, M. A., Koldehofe, B., Altaweel, A., and Rothermel, K. (2010). Providing basic security mechanisms in broker-less publish/subscribe systems. In *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, DEBS '10, pages 38–49, New York, USA. ACM.

Wang, C., Carzaniga, A., Evans, D., and Wolf, A. (2002). Security issues and requirements for internet-scale publish-subscribe systems. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, volume 9 of *HICSS*, page 303, Hawaii, USA. IEEE ComSoc.

Wong, C. K., Gouda, M., and Lam, S. S. (2000). Secure group communications using key graphs. *IEEE/ACM Trans. Netw.*, 8(1):16–30.

Wong, W. K., Cheung, D. W.-l., Kao, B., and Mamoulis, N. (2009). Secure knn computation on encrypted databases. In *Proceedings of the 35th ACM SIGMOD international conference on Management of data*, SIGMOD, pages 139–152, Providence, USA. ACM.

Wun, A. and Jacobsen, H.-A. (2007). A policy management framework for content-based publish/subscribe middleware. In Cerqueira, R. and Campbell, R., editors, *Middleware 2007*, volume 4834 of *Lecture Notes in Computer Science*, pages 368–388. Springer Berlin / Heidelberg.

Xu, H., Guo, S., and Chen, K. (2014). Building confidential and efficient query services in the cloud with RASP data perturbation. *Knowledge and Data Engineering, IEEE Transactions on*, 26(2):322–335.

Zhao, Y. and Sturman, D. (2006). Dynamic access control in a content-based publish/subscribe system with delivery guarantees. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, ICDCS, page 60, Lisboa, Portugal. IEEE Computer Society.

Zhuang, S. Q., Zhao, B. Y., Joseph, A. D., Katz, R. H., and Kubiatowicz, J. D. (2001). Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, NOSSDAV, pages 11–20, Port Jefferson, USA. ACM Press.

Zou, X., Ramamurthy, B., and Magliveras, S. S. (2005). *Secure Group Communications Over Data Networks*. Springer-Verlag New York, Inc.