Trust and Privacy in Development of Publish/Subscribe Systems

Emanuel Onica
Alexandru Ioan Cuza University of Iaşi
Iaşi, Romania *eonica@info.uaic.ro*

Hugues Mercier
Université de Neuchâtel
Neuchâtel, Switzerland *hugues.mercier@unine.ch*

Etienne Rivière
ICTEAM, UCLouvain
Louvain-la-Neuve, Belgium *etienne.riviere@uclouvain.be*

Abstract

Publish/subscribe (pub/sub) is a widely deployed paradigm for information dissemination in a variety of distributed applications such as financial platforms, e-health frameworks and the Internet-of-Things. In essence, the pub/sub model considers one or more publishers generating feeds of information and a set of subscribers, the clients of the system. A pub/sub service is in charge of delivering the published information to interested clients. With the advent of cloud computing, we observe a growing tendency to externalize applications using pub/sub services to public clouds. This trend, despite its advantages, opens up multiple important data privacy and trust issues. Although multiple solutions for data protection have been proposed by the academic community, there is no unified view or framework describing how to deploy secure pub/sub systems on public clouds. To remediate this, we advocate towards a trust model which we believe can serve as basis for such deployments.

Keywords: publish/subscribe, data privacy, security, trust, distributed systems.

1. Introduction

Many online applications are provided as distributed services, either because they are directed towards geographically-spread end users or because they connect a dispersed enterprise environment. Providing timely, efficient and scalable information dissemination is a requirement in all distributed applications. Publish/subscribe (pub/sub) is a paradigm for efficient information exchange in complex distributed systems. This communication model follows the producer/consumer design pattern, considering one or more *publishers* as source of feeds of information, and multiple *subscribers* as clients of this information. Subscribers register *subscriptions* detailing their interest in the published data. A variety of applications rely on this model, ranging from electronic commerce [8] to the management of medical records [24]. A canonical example used in the literature [5, 14, 35] is a financial application where publishers represent stock exchanges emitting information on new stock quotes, and where subscribers are investors monitoring market changes. A middleware layer situated between the publishers and the subscribers stores the subscriptions, matches these subscriptions with the incoming publication stream, and routes the relevant information to interested subscribers. Various designs exist for implementing such a middleware layer starting from a loosely defined architecture in the late 1980s [6] up to very recent blockchain-based frameworks [38]. The most common design is to distribute the matching and routing operations across an overlay of *broker* nodes. An overview of this model is given by Figure 1.

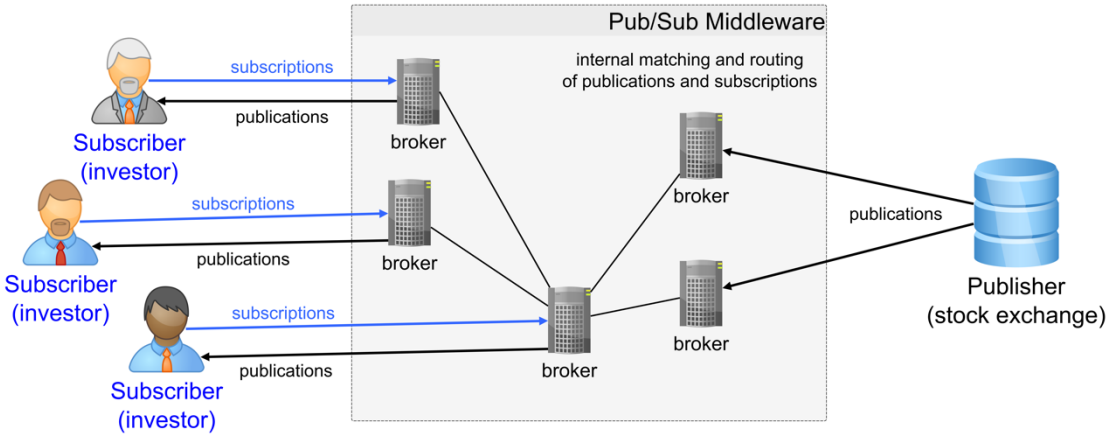

Figure 1. Overview of a publish/subscribe architecture applied to a stock exchange use case

Both the structure of the information available in publications and subscriptions and the matching model that the brokers allow have a strong impact on the design of a pub/sub system. An overview of design variants is presented in a survey by Eugster *et al.* [14]. There are two dominant variants of pub/sub: *topic-based* and *content-based*. In topic-based pub/sub subscriptions and publications are assigned to topics. Subscriptions match publications for the same topic, forming channels of information. For instance, in the stock exchange use case, messages could be assigned to a topic given by the stock symbol: subscribers registering for the symbol "`IBM`" will receive all publications tagged "`IBM`". In content-based pub/sub matching is based on criteria on publications' content. This content is typically condensed in a publication header, structured in key-value fields according to a *publication schema*. Subscriptions in the content-based model are conjunction of constraints over the values for these publication fields. Following our stock exchange example, a publication schema could include fields like:

```
"symbol" (type:string),
"value" (type:float),
"date" (type:date),
"variation" (type:float)
```

A subscriber could then register a subscription such as:

```
[("symbol"="IBM") and ("value">140) and ("date"="3.2.2019")]
```

The content-based model allows clients to select at a finer grain the information they wish to receive. However, this model is also more challenging to implement, both from a performance point of view (since the matching and routing operations are more complex) but also, and perhaps more importantly, from the perspectives of *privacy* and *trust*. Subscriptions include sensitive information revealing subscribers' interests. This information must be protected from prying eyes, yet allowing the matching operation to be performed by the routing middleware. For instance, investors interested in price variations of stocks do not want to divulge their financial strategies to other investors. In an e-health scenario supported by pub/sub, publications of medical records are highly sensitive. An e-commerce application does not want the untrusted owner of the middleware infrastructure to sell information about buyers' interests to competitor merchants.

Data privacy and trust issues became more acute with the advent of cloud computing. Multiple services including pub/sub middleware are externalized to public and shared infrastructures. Cloud computing brings an economic advantage to service owners who are able to reduce their deployment and operation costs. However, middleware services, deployed on machines situated in a public domain are more prone to attacks [16, 29] and, therefore, to privacy and trust breaches.

Multiple commercial frameworks and communication protocols implement the pub/sub paradigm in their design. Examples include the Microsoft .NET microservices framework [12], the MQTT protocol [39] for IoT integration, the instant messaging and VoIP protocol XMPP [21], and Pusher Channels [40]. However, none of these frameworks

considers by design data privacy and trust aspects. On the other hand, there exists an extensive amount of academic research that addresses the protection of data in subscriptions and publications, as well as associated performance/security tradeoffs [3]. These solutions generally focus on very specific technical aspects of a particular mechanism, do not consider the wider context of trust in a practical scenario, and are not used in practice. This state of fact is not an accident: *designing secure and efficient pub/sub systems using untrusted middleware is very hard.*

The contribution of the present study is twofold. First, we provide in Section 2 more specific details about trust and data privacy issues in current pub/sub deployments, and we present an overview of current approaches for preserving data privacy in Section 3, grouped according to their subject of application. Second, in Section 4, we propose a trust model, which we believe can serve as basis for the successful integration of the mechanisms described in Section 3. We conclude our study in Section 5.

2. Trust and Data Privacy

In general, preserving data privacy refers to the capability of using the data of an entity while protecting the non-disclosure preferences of that entity. The enforcement of such preferences must be based on trust relations between entities and service providers. Privacy preferences can refer to either hiding the identity of the entity - *anonymity*, or to hiding data that might reveal some interests or other facts about the entity - *confidentiality*. We discuss below these two data privacy and trust perspectives in the context of pub/sub services.

2.1. The Anonymity Perspective

Anonymity in pub/sub is naturally preserved, to some extent, by the system design, which decouples the clients of the service, the subscribers, from the publishers. A publisher does not know to which of the service subscribers the published data is delivered, and a subscriber does not know from which publisher it received a given publication. The decoupled nature of the communication is enforced by the middleware providing the pub/sub service. In most scenarios the provider of the pub/sub service is entitled to know the identity of its clients or that of the data providers, and consequently trusted not to divulge these identities.

The basic level of anonymity provided by decoupling is typically acceptable, given that subscribers emit a request and expect receiving information only from the pub/sub service. This is fundamentally different from other service models (i.e., electronic voting), where the only purpose of the service is to process information originating from the clients. Of course, situations could exist when stronger anonymity guarantees are necessary (e.g., a free service for monitoring some patients' state and notifying them about some developing condition, where patients would not want to expose their identities). In Section 3 we overview several techniques that can be used to provide such guarantees. However, in the vast majority of use cases clients pay for the pub/sub service, which already implies revealing their identity to the service. Therefore, the focus on data privacy typically lies less with anonymity and more with confidentiality.

2.2. The Confidentiality Perspective

Data privacy, from a confidentiality perspective, refers to the protection against the access to sensitive information in queries emitted by the system clients – in subscriptions, or in sensitive information disseminated using the pub/sub service – in publications. *Trust*, in respect to a node (machine) where the pub/sub system is deployed, refers to a degree of confidence in the entities able to access the specific node and the sensitive data at that node. Multiple degrees of trust can be defined, depending on the sensitivity of some specific data. Similarly, multiple degrees of trust can be defined with respect to the different entities that can access a specific node. For simplicity of the discussion, we consider, for now, a single

node trust level. We say that a node where a distributed pub/sub system is deployed is *trusted* if we have full confidence in all the entities able to access that node and read sensitive data it holds. Similarly, we consider a node *untrusted* if we do not have confidence in any entity able to access the node and read its sensitive data. In case of a trusted node there is no need for data privacy. In case of an untrusted node, mechanisms for enforcing data privacy are required.[1]

As introduced in Section 1, trust and data privacy issues in pub/sub services become more stringent with the externalization to public cloud infrastructures. Essentially clients can trust the provider of the pub/sub middleware to properly operate the service, without willingly leaking any sensitive data. However, when this pub/sub service is deployed on virtual machines on a public cloud, multiple other entities, potentially with malicious intent, can gain access to the respective cloud nodes, rendering these untrusted.

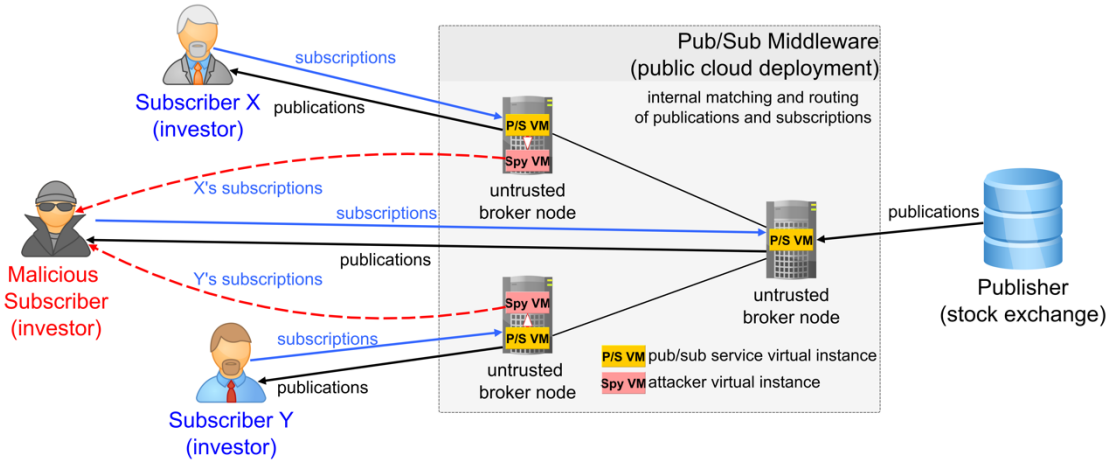

Figure 2. High-level overview of an attack scenario via VM co-location in the stock market use case

A first category of threats is represented by attackers who either have an interest in finding subscribers' intentions or in accessing a publications stream without permission. If a virtual machine of an attacker is placed in the public cloud on the same untrusted node as for the victim pub/sub service, a variety of side-channel attacks can be attempted for leaking sensitive information [29, 36]. Figure 2 presents an overview of the threat scenario adapted to our stock market example, where a malicious investor attempts to gain information on competitors from co-located virtual machine. An empirical study performed in 2015 [33] on the placement of virtual machines on U.S. regions of Amazon EC2, Google Compute Engine and Microsoft Azure cloud infrastructures has shown that an attacker using 10 to 30 virtual machine instances had a probability of co-residency on the same node as a victim's virtual machine that ranges from 0.3 to 1, depending on the number of virtual machines run by the victim. A more recent study from 2017 [1], conducted with the participation of the U.S. Army Research Laboratory, devised a mathematical model for computing the probability of co-residency for an attacker's virtual machine with a victim's machine, confirming this threat.

A second category of threats, and subsequent interest in providing data privacy guarantees, gained momentum following the "Snowden effect" [17] of global surveillance disclosures. A subscriber might not be comfortable with knowing that his interests might be spied upon by a governmental agency. Even though the provider of a pub/sub service might be located in a country that is theoretically regarded as well-regulated from this perspective (i.e., the European space with the adoption of the GDPR), if the provider is using a public cloud infrastructure for his service, the machines where the service effectively runs might be geographically located elsewhere, and under the legal jurisdiction of another country. This may expose the infrastructure where the service resides to hidden

[1] In a more general case, if multiple degrees of trust are defined at the level of one node, then privacy of *some* data should be enforced against *some* entities able to access that node. We reference the generalization in our trust model approach proposed in Section 4.

surveillance from that country. This exposure is accentuated if the cloud provider periodically migrates virtual instances across hosts located in different jurisdictions.

Both types of threats have a similar target: gaining access to sensitive data on untrusted nodes. In practical deployments, a part of the pub/sub service can be externalized to a public cloud for cost effectiveness, whereas the other part can be deployed on a private infrastructure that belongs to the service operator, which is normally considered secure. This, as well as different sensitivity levels of data, requires defining a more complex trust model for the nodes in the system, which we propose in Section 4. Nevertheless, threats can be addressed by implementing data privacy mechanisms on the untrusted broker nodes, more precisely techniques for preserving confidentiality of subscriptions and publications. The natural approach would be encrypting this data. The difficulty lies in simultaneously maintaining the functionality of the pub/sub service. If the data cannot be accessed at the level of the untrusted broker node, then the pub/sub matching cannot be performed. Therefore, specific mechanisms are required. We provide an overview of the existing approaches for such mechanisms in Section 3. Several general aspects should be considered in relation with the practical use case when choosing a solution:

a) *Expressivity of the pub/sub model.* In topic-based pub/sub the matching operation is simpler, implying typically only evaluation of the equality on topic. Therefore, it is normally enough to provide support for confidential equality matching. Matching in content/based pub/sub involves more complex operations over subscriptions' constraints, such as numerical range evaluations, substring comparisons, etc.

b) *Specific sensitivity of messages.* Depending on the use case, data privacy might be important for subscriptions, publications, or both. Most developed mechanisms take in account both types of messages. However, depending on the trust assumptions, selectively protecting only one type of messages might be acceptable and could increase the cost-effectiveness of the solution.

c) *Payload of publications.* A publication always contains a payload in a topic-based pub/sub scenario, including the effective publication content. In content-based pub/sub this payload might be absent (e.g., in the stock market example, all information on a quote can be included in the header fields used in matching). This payload, when present, requires similar protection as the header. Since the payload is not subject to the matching operation, standard encryption techniques can be used.

3. Mechanisms for Data Privacy in Pub/Sub Systems

We now overview mechanisms that can be implemented for preserving data privacy in pub/sub systems. We follow the two perspectives – anonymity and confidentiality – discussed in the previous section. We also group these mechanisms according to their specific subject of application. This allows us to establish a set of data privacy *properties,* a notion which we further use in defining a trust model in Section 4.

3.1. Anonymity Mechanisms

Mechanisms for preserving anonymity in pub/sub have the purpose of protecting the identity of subscribers and/or publishers by respectively hiding the source of subscriptions and/or publications from an untrusted node. We can, therefore, define two data privacy properties, according the subject protected by the mechanism: *subscriber anonymity* and *publisher anonymity.* As discussed in Section 2, anonymity has less importance in pub/sub systems due to the natural decoupling between publishers and subscribers, and also because in many use cases subscribers are paying customers, whose identity should be known at

the broker node level for delivering the service[2]. Most techniques that can be applied in a pub/sub scenario are derived from more general techniques in distributed systems. We overview the main approaches in the following.

a) *Proxy forwarding and re-encryption.* This mechanism relies on one or more trusted third party proxy nodes that forward (and potentially encrypt [37]) the messages, and send them on behalf of the original sources after removing their identity. Typically, choosing from multiple proxy nodes can harden the system against brute force attempts to identify the source via traffic monitoring. Due to the fact that the identity of the sender is discarded, this technique is adequate mainly for publishers anonymity (i.e., using the proxy for sending the publication to an untrusted broker node).
Subscribers anonymity would require maintaining a mapping history of discarded source identities at the level of the proxy, for tracing back the route towards subscribers. Since keeping such a mapping with the exact identifier of a subscriber on a third-party proxy node could still be a vulnerability (and make the proxy itself a target), a more cautious approach is to map the source domain of the subscription. This domain can be represented by a private trusted infrastructure, which is under full control of the service provider, and where a final mapping is kept for reaching the client of the service. The objective is to disallow an attacker from tracing the identity of the subscription beyond this trusted domain, up to the effective subscriber. A disadvantage is the high cost of maintaining such a setup.

b) *Onion routing.* This is an alternative, more secure variation, to simple proxy forwarding for ensuring publishers anonymity. It is based on the concept of mix nodes [7], for which the most significant implementation is the TOR network [13]. Instead of a single proxy, an *onion path* of mix nodes can be used for sending an encrypted publication to a pub/sub broker. Each mix in the path can only decipher the next mix hop. Since the path must be fully known for the encryption at the source of the message, this mechanism cannot be easily used to send the publication from the untrusted broker up to the final subscriber without maintaining expensive persistent channels.

3.2. Confidentiality Mechanisms

We now discuss confidentiality mechanisms of content-based pub/sub. Confidentiality in the less expressive topic-based pub/sub can be provided by a subset of the content-based mechanisms. As before we focus our discussion on protecting untrusted broker nodes located in a public cloud where the pub/sub service middleware is deployed. We can define a set of properties according to the subject of application of the confidentiality mechanisms: *subscription confidentiality*, *publication confidentiality* and *payload confidentiality*. The first two refer to the protection of the messages' headers used for matching by brokers, while the latter refers to the confidentiality of any extra payload that a publication might contain.

Subscription and Publication Confidentiality

Most mechanisms are designed to provide both subscriptions and publications confidentiality, therefore we overview these together. Typically, a form of encryption is used allowing the untrusted broker node to determine the matching result without accessing the sensitive content of the subscription or the publication header. Decryption is not always provided since the final purpose is the matching result (only the publication payload is

[2] We refer here to *identity* strictly from the perspective of the source of a subscription required as destination for delivering the matching publications, i.e., the source IP address of a subscriber. The individual identity of a subscriber client can of course be subject of anonymization at the level of a broker node, depending on the agreement between the client and the service provider, using various techniques (i.e., pseudonyms). However, this falls outside the scope of the effective operation of the pub/sub service, since it does not interfere with the routing of the publications towards the subscriber.

delivered to the subscriber after a match). However, when publications do not include a payload, the information of interest for subscribers resides in the publication header fields, in which case decryption must be supported. Since this case is similar to the payload confidentiality we discuss it later. In the following we summarize the main categories of mechanisms that provide confidential matching.

a) *Specific cryptographic schemes.* A series of cryptographic schemes were developed for allowing an untrusted broker to match an encrypted subscription with encrypted publications. Some of these schemes [9, 23] have roots in homomorphic encryption, preserving an isomorphism between the encrypted comparison of subscriptions and publications and the equivalent plaintext comparison. Other schemes [18, 20, 28, 32] rely on domain mappings that permit translating any range or string comparison to an equality comparison in the encrypted form. We surveyed in our previous work [25] such *encrypted matching* schemes. The impediment of adopting such schemes in a practical pub/sub scenario typically lies in the difficulty of implementing a necessary key management solution. We discuss this issue in the final part of this section.

b) *Standard cryptographic schemes and access control.* Some of the proposed mechanisms [2, 19, 34] rely on standard cryptographic protocols for protecting sensitive fields in publications and constraints in subscriptions, which do not permit matching these fields on untrusted broker nodes. The solutions typically define various access control models, allowing the brokers to perform matching only over certain non-sensitive fields according to the access control policies. Depending on the solution, various levels of sensitivity for message fields (and respectively trust for brokers) can be defined. A disadvantage of this approach is that matching over sensitive fields is not allowed.

c) *Trusted hardware support.* Some recent CPUs offer the possibility to perform the matching in hardware-protected memory zones such that the access can be restricted even for the owner or operator of the physical machine, i.e., using the Intel SGX or AMD SEV technologies [22]. This is a novel approach towards confidentiality in pub/sub, with yet few implementations [27]. A limitation of the solution is its dependence on this hardware support.

Payload Confidentiality and the Key Exchange Issue

As discussed in Section 2.2, payload confidentiality is typically ensured by using standard encryption schemes. The publication payload is not subject of a matching operation on the untrusted broker nodes. Therefore, no access is needed to the payload, so in theory any end-to-end encryption is sufficient.

Symmetric encryption standards (i.e., AES [11]) are typically recommended due to cost effectiveness. However, in this case the encryption key is secret and should be exchanged by the two communicating parties. The pub/sub context decouples the communication between the publisher and the subscriber. Normally it is impossible to know prior to matching which subscriber will receive a certain publication. This creates a significant difficulty in solving the required key exchange.

Using asymmetric encryption (e.g., RSA [30]), where the publication payload could be encrypted with the public key of a subscriber who then decrypts with an associated private key, poses a similar problem despite the absence of key exchange. The publisher would not know who the receiving subscriber is, and consequently whose public key to use in the encryption. Besides this, public key encryption is typically costly and is applied only on messages of small dimension.

Currently, establishing a key correspondence between publishers and subscribers for payload confidentiality is one of the major open issues in privacy preserving pub/sub systems. The issue is even more severe considering that it also applies to publication confidentiality when the publication is only formed of the header used in matching, which should be decrypted when reaching the subscriber.

The easiest way to approach the key exchange is by abandoning the complete

decoupled nature of the system in favor of a partial coupling where a publisher would know at least the group of subscribers who might receive emitted publications, and pre-exchange a group encryption key with these. However, new issues arise:

- *Confidentiality breaches.* The size of the group of clients should remain limited. Pre-exchanging a key with a larger group increases the chance of confidentiality breaches: a malicious subscriber could try to intercept payloads sent to a different subscriber and encrypted with the same key. Specific cryptographic techniques exist for efficient dissemination of keys in groups preventing some breaches, such as broadcast encryption [15] with tracing traitors [10] who would leak the key outside a trusted group. So far, these techniques have been applied to the protection of secure pay-per-view channels in media transmissions and for implementing digital rights management (DRM) policies, where the encrypted data sent to the group does not differ from one client to the other and where, therefore, a client of the trusted group would not be interested in spying another client. Adapting such techniques to pub/sub is an open problem.
- *Limited scalability.* One of the advantages of the decoupled nature of pub/sub systems is the potential for scaling to large numbers of subscribers [31]. This is achieved through various methods, some implying dynamic changes in the system such as work migration from one node to another for load balancing [4] or the addition of new nodes (i.e., new publishers). An initial mapping between publishers and subscribers due to a key pre-exchange would limit the flexibility of any potential changes and could consequently impair scalability.
- *Potential anonymity breaches.* A pre-exchange of encryption keys between known sets of publishers and subscribers could expose their identity to one another. Although, as discussed, this might not be an issue in some situations, in other use cases this would breach anonymity properties as defined in Section 3.1. However, this problem is typically easy to resolve by coordinating the key exchange through a trusted third party, which guarantees anonymizing the identity of the key receiving nodes.

The key management in pub/sub systems also presents another difficulty, this time related to subscription confidentiality. In a general context, a key used in the encryption of an information stream must be periodically refreshed to resist brute force attacks. If encrypted matching is the chosen mechanism for subscriptions confidentiality, the new key encryption should also be applied to subscriptions previously stored by untrusted broker nodes. Otherwise, the matching will not be possible after a key refresh since publications will be encrypted with different keys. The naïve solution is that subscribers re-submit all subscriptions encrypted with the new key. This can inflict an unwanted load on the system which might cause communication and latency issues. Only few solutions have been proposed for solving this problem [26], but these are highly dependent on the chosen encryption scheme.

4. Towards a Trust Model for Privacy Preserving Publish/Subscribe

We believe that besides the issues identified in the previous section, a general reason for the lack of adoption of mechanisms for data privacy in pub/sub systems is that only few solutions consider the necessity of a trust model. Indeed, most solutions only focus on the technical details of the proposed mechanism. In this section we lay the ground for a trust model, which we believe can help in integrating mechanisms referred in Section 3 in a practical deployment. The following elements compose this model:

- *Trust domain.* In Section 2.2 we referred for the clarity of the discussion to a single degree of trust. However, in practice, multiple degrees can be defined depending on the sensitivity of data and the privacy needs of the use case. We consider one given set of nodes grouped in the same *trust domain*, if the expectations in terms of privacy preservation for these machines are exactly the same. A trust domain does not only correspond to the machines it contains but also includes network links. If the machines in a trust domain are trusted for not leaking private information, their interconnection links are also assumed to guarantee a similar level of privacy.

- *Strength of properties.* The privacy expectations can be ensured via data privacy mechanisms and modelled through the *properties* defined in Section 3: subscribers and publishers anonymity, and subscriptions, publications and payload confidentiality. Until now we mainly referred to a single attacker with a purpose of spying on sensitive information on one untrusted node. However, if the attacker is able to implement a colluding attack, i.e., accessing data on multiple nodes, then the attacker can potentially gain a significant advantage against some specific mechanisms for data privacy, which are not designed to resist to collusion (e.g., gaining the capability to derive encryption keys). Therefore, it is useful to specify in addition to a required property in a trust domain, the needed *strength* for the respective property, where strength specifies the colluding power under which the property should hold. We formalize this in our model as tuples having the structure: <*property*, *strength*>. We consider specifically the following variants:

    a) <*property*, *0*> identifies the case of a trust domain, where nodes are trusted and the property must not be enforced (the attacker has no access);

    b) <*property*, *1*> identifies the case of a trust domain, where no collusion can occur (the attacker's access is strictly limited to at most one node);

    c) <*property*, *k*> identifies the case of a trust domain, where collusion might occur, with the attacker's access limited to at most $k > 1$ colluding nodes.

  Ordering the properties permits establishing the general *trust level* of a trust domain in comparison to other domains. More precisely, we can distinguish the following cases for two domains A and B:

    1) *Equally trusted*: if all trust properties and their strength associated to the trust domain A are also associated to the trust domain B, and conversely.

    2) *Less (more) trusted: domain* A is less (more) trusted than domain B if for properties associated with domain B, domain A has at least one corresponding property with higher (lower) strength, with the other properties being of equal strength For instance a domain A with (<*subscription confidentiality*, *1*>, <*publisher anonymity*, *1*>) is less trusted than a domain B with (<*subscription confidentiality*, *1*>, <*publisher anonymity*, *0*>).

    3) *Direct comparison not possible:* this is the case if different properties have opposite associated strength, for instance if domain A has (<*subscriber anonymity*, *1*> <*publisher anonymity*, *0*>) and domain B has (<*subscriber anonymity*, *0*> <*publisher anonymity*, *1*>). In such instances *most powerful common properties* can be defined: (<*subscriber anonymity*, *1*> <*publisher anonymity*, *0*>). This set, as the rest of the ordering, is useful for deploying privacy mechanisms across domains as discussed below.

- *Trust boundaries.* Nodes running the pub/sub service might reside on different trust domains, where different properties with different strength are required. For preserving the appropriate strength of these properties, it might be necessary to enforce privacy mechanisms, or respectively to remove their effect, when publications and subscriptions cross the trust domains. For the purpose of modelling these changes we define such domain crossing as a *trust boundary*. We consider two types of such boundaries:

    a) *Hard boundary.* This is the boundary that appears when crossing from a more trusted domain to a less trusted domain. The properties required in the less trusted domain must be enforced before reaching this domain;

    b) *Soft boundary.* This is the boundary that appears when crossing from a less trusted domain to a more trusted domain, requiring properties with lower strength. Any stronger data privacy property can be disabled any time after crossing the boundary.

When two domains require some properties that cannot be compared, the boundary is associated with the set of most powerful common properties, as defined above. These should be enforced when crossing into both domains for preserving trust.

In Figure 3 we present an example of our trust model elements following the course of a publication on a simple deployment over five trusted domains and a non-trusted one.

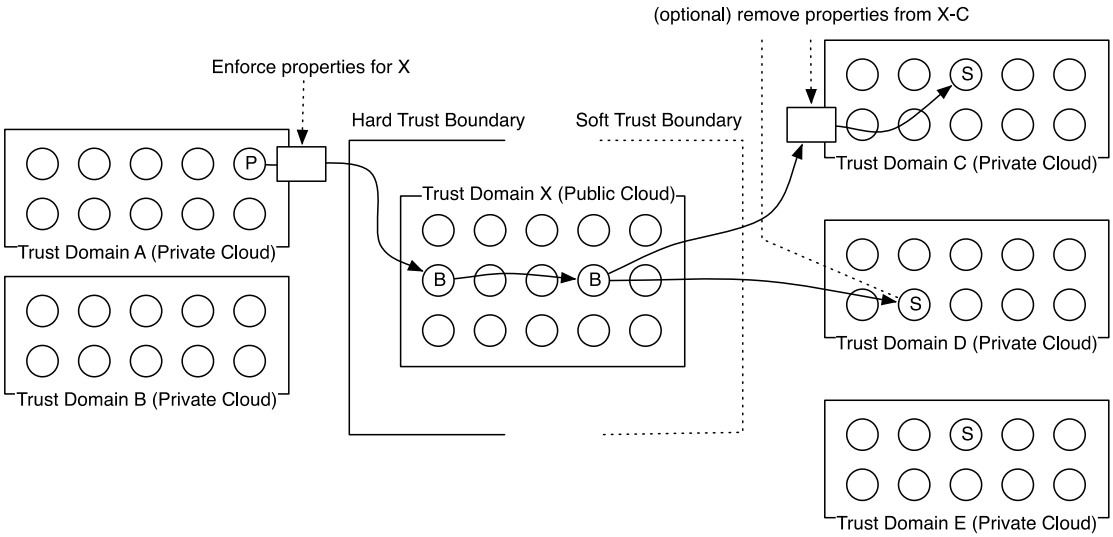

Figure 3. Trust model over a pub/sub service deployment with five trusted domains and an untrusted domain. It shows a publisher (P) generating a publication, which is then matched by an overlay of brokers (B) and routed to matching subscribers (S)

The implementation of the model in practice would also require the presence of an external trusted component called the *Trust Relationship Authority*, TRA for short. The responsibility of the TRA is to match the trust domain properties to the privacy preserving mechanisms and apply these mechanisms at trust boundaries. A more precise description of the operation of this TRA requires a discussion about key exchange support between the domains, which as presented in Section 3.2 is still an open problem and dependent on the actual mechanisms used. We note that while the TRA can be simply considered as a centralized component, the possibility of integrating it as a decentralized set of cooperating operators that would only run on the most trusted domains of the infrastructure is also an interesting perspective, open for future investigation.

5. Conclusion

We have presented a study about the trust and data privacy issues and necessities in publish/subscribe services, an important area of distributed systems designed for information dissemination used in multiple use cases and environments. We have discussed aspects related to data privacy from two perspectives: anonymity and confidentiality, and we have overviewed the most important categories of mechanisms used for enforcing specific privacy properties. Finally, we have defined the basic ground for a trust model, open to further development, but which we believe to be an important step for integration of privacy mechanisms in pub/sub service deployments.

References

1. Atya, A.O.F., Qian, Z., Krishnamurthy, S.V., Porta, T.L., McDaniel, P., Marvel, L.: Malicious co-residency on the cloud: Attacks and defense. In: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications. (2017)
2. Bacon, J., Eyers, D.M., Singh, J., Pietzuch, P.R.: Access control in publish/subscribe systems. In: 2nd Intl. Conference on Distributed Event-Based Systems. (2008)

3. Barazzutti, R., Felber, P., Mercier, H., Onica, E., Riviere, E.: Efficient and Confidentiality-Preserving Content-Based Publish/Subscribe with Prefiltering. In: IEEE Transactions on Dependable and Secure Computing 14(3), pp. 308-325. (2015)
4. Barazzutti, R., Heinze, T., Martin, A., Onica, E., Felber, P., Fetzer, C., Jerzak, Z., Pasin, M., Riviere, E.: Elastic Scaling of a High-Throughput Content-Based Publish/Subscribe Engine. In: IEEE 34th Intl. Conference on Distributed Computing Systems - ICDCS '14. (2014)
5. Bernstein, P.A., Newcomer, E.: Principles of transaction processing. Elsevier/Morgan Kaufmann Publishers. (2009)
6. Birman, K., Joseph, T.: Exploiting virtual synchrony in distributed systems. In: 11th ACM Symposium on Operating Systems Principles - SOSP '87. (1987)
7. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. In: Communications of the ACM 24 (2), pp. 84–90. (1981)
8. Cheng, E.C.: A Publish/Subscribe Framework: Push Technology in Electronic Commerce. In: Internet Applications. pp. 486–494. Springer Berlin Heidelberg. (1999)
9. Choi, S., Ghinita, G., Bertino, E.: A Privacy-Enhancing Content-Based Publish/Subscribe System Using Scalar Product Preserving Transformations. In: Database and Expert Systems Applications - DEXA '10. (2010)
10. Chor, B., Fiat, A., Naor, M.: Tracing Traitors. In: Advances in Cryptology — CRYPTO '94. pp. 257–270. Springer Berlin Heidelberg. (1994)
11. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer Verlag. (2002)
12. De la Torre, C., Wagner, B., Rousos, M.: Implementing event-based communication between microservices. In: .NET Microservices: Architecture for Containerized .NET Applications. Microsoft Developer Division. (2018)
13. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: The 13th USENIX Security Symposium. (2004)
14. Eugster, P.T., Felber, P., Guerraoui, R., Kermarrec, A.-M.: The many faces of publish/subscribe. In: ACM Computing Surveys 35 (2), pp. 114–131. (2003)
15. Fiat, A., Naor, M.: Broadcast Encryption. In: Advances in Cryptology — CRYPTO' 93. pp. 480–491. Springer Berlin Heidelberg. (1994)
16. Han, Y., Chan, J., Alpcan, T., Leckie, C.: Using Virtual Machine Allocation Policies to Defend against Co-resident Attacks in Cloud Computing. In: IEEE Transactions on Dependable and Secure Computing 14(1), pp. 95-108. (2015)
17. Hautala, L.: The Snowden effect: Privacy is good for business, Online: https://www.cnet.com/news/the-snowden-effect-privacy-is-good-for-business-nsa-data-collection/, (2016)
18. Ion, M., Russello, G., Crispo, B.: Design and implementation of a confidentiality and access control solution for publish/subscribe systems. In: Computer Networks 56 (7), pp. 2014–2037. (2012)
19. Khurana, H.: Scalable security and accounting services for content-based publish/subscribe systems. In: ACM Symposium on Applied Computing - SAC '05. (2005)
20. Li, J., Lu, C., Shi, W.: An efficient scheme for preserving confidentiality in content-based publish/subscribe systems. Georgia Institute of Technology, Technical Report (2004)
21. Milard, P., Saint-Andre, P., Meijer, R.: XEP-0060: Publish-Subscribe, (2018)
22. Mofrad, S., Zhang, F., Lu, S., Shi, W.: A comparison study of Intel SGX and AMD memory encryption technology. In: 7th International Workshop on Hardware and Architectural Support for Security and Privacy - HASP '18. (2018)
23. Nabeel, M., Shang, N., Bertino, E.: Privacy-Preserving Filtering and Covering in Content-Based Publish Subscribe Systems. Purdue University, Technical Report. (2009)

24. Narus, S.P., Rahman, N., Mann, D.K., He, S., Haug, P.J.: Enhancing a Commercial EMR with an Open, Standards-Based Publish-Subscribe Infrastructure. In: AMIA Annual Symposium. (2018)

25. Onica, E., Felber, P., Mercier, H., Rivière, E.: Confidentiality-Preserving Publish/Subscribe: A Survey. In: ACM Computing. Surveys 49 (2), pp. 1–43. (2016)

26. Onica, E., Felber, P., Mercier, H., Rivière, E.: Efficient Key Updates through Subscription Re-encryption for Privacy-Preserving Publish/Subscribe. In: 16th ACM/IFIP/USENIX Intl. Middleware Conference - Middleware '15. (2015)

27. Pires, R., Pasin, M., Felber, P., Fetzer, C.: Secure Content-Based Routing Using Intel Software Guard Extensions. In: 17th ACM/IFIP/USENIX Intl. Middleware Conference - Middleware '16. (2016)

28. Raiciu, C., Rosenblum, D.S.: Enabling confidentiality in content-based publish/subscribe infrastructures. In: 2nd IEEE/CreatNet International Conference on Security and Privacy in Communication Networks. (2006)

29. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: 16th ACM Conference on Computer and Communications Security - CCS '09. (2009)

30. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. In: Communications of the ACM. 21 (2), pp. 120–126. (1978)

31. Rotaru, M., Olariu, F., Onica, E., Rivière, E.: Reliable messaging to millions of users with MigratoryData. In: 18th ACM/IFIP/USENIX Intl. Middleware Conference - Middleware '17. (2017)

32. Tariq, M.A., Koldehofe, B., Rothermel, K.: Securing Broker-Less Publish/Subscribe Systems Using Identity-Based Encryption. In: IEEE Transactions on Parallel and Distributed Systems 25 (2), pp. 518–528. (2014)

33. Varadarajan, V., Zhang, Y., Ristenpart, T., Swift, M.: A placement vulnerability study in multi-tenant public clouds. In: 24th USENIX Security Symposium. (2015)

34. Wun, A., Jacobsen, H.-A.: A Policy Management Framework for Content-Based Publish/Subscribe Middleware. In: 8th ACM/IFIP/USENIX Intl. Middleware Conference - Middleware '07. (2007)

35. Yang, L.T.: Research in Mobile Intelligence: Mobile Computing and Computational Intelligence. John Wiley & Sons, Hoboken. (2010)

36. Zhang, Y., Juels, A., Reiter, M.K., Ristenpart, T.: Cross-Tenant Side-Channel Attacks in PaaS Clouds. In: 21st ACM Conference on Computer and Communications Security - CCS '14. (2014)

37. Zheng, Q., Zhu, W., Zhu, J., Zhang, X.: Improved anonymous proxy re-encryption with CCA security. In: 9th ACM Symposium on Information, Computer and Communications Security - ASIA CCS '14. (2014)

38. Zupan, N., Zhang, K., Jacobsen, H.-A.: Hyperpubsub: a decentralized, permissioned, publish/subscribe service using blockchains: demo. In: 18th ACM/IFIP/USENIX Intl. Middleware Conference - Middleware '17. (2017)

39. Message Queuing Telemetry Transport (MQTT) Standard v3.1.1, Online: https://www.iso.org/standard/69466.html, (2016)

40. Pusher Channels. Pusher Ltd., Online: https://pusher.com/channels, (2019)