

Privacy-preserving content-based publish/subscribe with encrypted matching and data splitting

Nathanaël Denis, Pierre Chaffardon, Denis Conan, Maryline Laurent, Sophie Chabridon, Jean Leneutre

▶ **To cite this version:**

Nathanaël Denis, Pierre Chaffardon, Denis Conan, Maryline Laurent, Sophie Chabridon, et al.. Privacy-preserving content-based publish/subscribe with encrypted matching and data splitting. SECRYPT 2020: 17th International Conference on Security and Cryptography, Jul 2020, Lieusaint - Paris, France. pp.405-414, 10.5220/0009833204050414 . hal-02910407

HAL Id: hal-02910407
https://hal.science/hal-02910407

Submitted on 17 Mar 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Privacy-Preserving Content-Based Publish/Subscribe with Encrypted Matching and Data Splitting

Nathanaël Denis[1], Pierre Chaffardon[1], Denis Conan[1], Maryline Laurent[1], Sophie Chabridon[1], and Jean Leneutre[2]

[1] *SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, France*

[2] *LTCI, Télécom Paris, Institut Polytechnique de Paris, France*

firstname.surname@telecom-sudparis.eu, jean.leneutre@telecom-paris.fr

Keywords:
publish-subscribe, privacy, attribute-based encryption, data splitting, encrypted matching

Abstract:
The content-based publish/subscribe paradigm enables a loosely-coupled and expressive form of communication. However, privacy preservation remains a challenge for distributed event-based middleware especially since encrypted matching incurs significant computing overhead. This paper adapts an existing attribute-based encryption scheme and combines it with *data splitting*, a non-cryptographic method called for alleviating the cost of encrypted matching. Data splitting enables to form groups of attributes that are sent apart over several independent broker networks so that it prevents the identification of an end-user; and, only identifying attributes are encrypted to prevent data leakage. The goal is to achieve an acceptable privacy level at an affordable computing price by encrypting only the necessary attributes, whose selection is determined through a *Privacy Impact Assessment*.

1 Introduction

Publish/subscribe (pub/sub) or Distributed Event-Based System (DEBS) is a communication paradigm that allows flexible and dynamic communication between a large number of entities, like in the Internet of Things (IoT). Publish/subscribe communicating parties are loosely-coupled—i.e. time, space, and synchronisation decoupling of subscribers and publishers [Eugster *et al.*, 2003]. Subscribers express their interest in a set of publications through constraints specified in the subscription. In broker-based DEBS middleware, the publications are routed to the interested consumers by a third-party entity composed of brokers.

The most popular models to determine which publications consumers are interested in are topic-based and content-based filtering [Eugster *et al.*, 2003]. In topic-based filtering, e.g. in AMQP [AMQP Consortium, 2008] and in MQTT [OASIS, 2019], filtering is achieved by complementing publication data with metadata tags named topics, and subscription filters are routing keys expressed as regular expressions on topics. Topics are shared by consumers and publishers, and the content of the publication is left opaque since it is not used for routing. In content-based filtering, filters are conjunctions of elementary filters that parse parts of the content data, including attribute values. Consequently, content-based filtering is expensive, but more expressive. While content-based filtering prevents the use of common encryption schemes requiring to disclose the whole publication content for routing, it is much more suitable for IoT because of its expressiveness.

Since the European General Data Protection Regulation (GDPR) became enforceable in 2018, privacy has been turning into a burning issue. Besides, the European regulation is the most widely adopted standard worldwide [Sullivan, 2019]. When privacy is at risk, companies have to conduct a *Privacy Impact Assessment* (PIA). The PIA is as "*a methodology for assessing the impacts on privacy of a project, policy, programme, service, product or other initiative and, in consultation with stakeholders, to iden-*

tify solutions" [Wright, 2012]. Among the main purposes of PIA is the identification of privacy controls to mitigate unacceptable risks. One of the main security concerns in pub/sub systems is confidentiality. This is particularly true under the semi-trusted broker assumption where brokers are considered honest-but-curious, which means that they will route the publications to the interested consumers, but can make use of the data for their own interest [Onica, E. *et al.*, 2016]. More precisely, "*confidentiality is the property that an information is not made available or revealed to unauthorised persons, entities or processes*" [ISO, 1989]. In the context of pub/sub middleware, confidentiality concerns encompass (1) part or all of the constraints of the subscriptions, (2) part or all the information in the publication that is used for routing against subscriptions, and (3) the payload of the publications [Onica, E. *et al.*, 2016].

Through encryption, a full degree of confidentiality can be achieved but with a significant overhead. The will of the user as well as the desired degree of confidentiality must be considered. Depending on the nature of the data sent by the users of a pub/sub system, parts of the publications might be sent in clear text under some assumptions. Particularly, the main concern of the users may be to prevent the brokers from identifying them. Following the work of [Domingo-Ferrer, J. *et al.*, 2019], we distinguish three categories of attributes: (1) identifying attributes that individually disclose the identity of a subject, (2) quasi-identifying attributes that do not identify subjects when considered separately, but their combination may, and (3) confidential attributes that convey sensitive features of an individual (income, religion, health condition, etc.) and may be sent in clear text as long as they can not be associated with an identity. Any attribute that does not fit any of these categories is considered as *non-confidential* and be outsourced as it is.

The proposition of this paper is to use a masking method, namely *data splitting*, with a cryptographic scheme to balance performance and security. This allows to avoid the use of encrypted matching when possible, regarding security requirements. In order to assess the feasibility of our proposal, we implemented our solution before proceeding to performance tests.

The paper is structured as follows. In Section 2, we describe then illustrate the security concerns through a motivating scenario. Afterwards, we discuss related works in Section 3. In Section 4, we detail our contribution. In Sections 5 and 6, we analyse the security of our system and provide the results of some performance tests. Finally, we conclude the paper in Section 7.

2 Motivation

In Section 2.1, we illustrate the security needs through a security-oriented lifeguard scenario. Then, in Section 2.2, we highlight the security concerns caused by data splitting.

2.1 Scenario

In our motivating scenario, we consider bathers on beaches and lifeguards whose mission is to protect bathers from drowning. All bathers and lifeguards are equipped with RFID wristbands that include geolocation sensors. The lifeguards need to collect the geolocation information of bathers and personal data to fulfill their role. To comply with the GDPR regulation, the collected data type must be determined in advance and exposed clearly to the bathers to get their consent.

Moreover, different physical or logical overlays of brokers are present around the beach. They collect information from the bathers and relay it to the lifeguards. To avoid the automatic use of encryption that would result in performance issues, the data are rather split into non-sensitive chunks sent to different overlays. For instance, let us consider the case where the bather has to publish the following attributes: $\{name, location, age, gender, occupation\}$. The *name* is an identifying attribute. As a consequence, it has to be encrypted to prevent an immediate identification. The *location*, without being combined with any identifying information, is not considered as sensitive information. While *gender*, *age*, and *occupation* are not confidential or identifying attributes when left alone, they might identify someone when grouped together. They are quasi-identifiers and need specific processing. Therefore, we split them into two groups: $\{age, gender\}, \{occupation\}$. Note that many combinations are possible. For example, we could split them as $\{age\}, \{gender\}$, and $\{occupation\}$, which is the most basic way to split the data, but also the one that needs the highest number of distinct overlays of brokers. So, these two groups of quasi-identifiers are sent to two different overlays of brokers in order to avoid re-identification attacks. An additional overlay of

brokers may be used to publish the encrypted attributes and the non-confidential ones together, which is not troublesome anymore.

This scenario stresses the need to process the data properly, firstly to avoid sending groups of attributes that would allow re-identification, but to limit the number of overlays as well. However, the second issue is more a matter of performance rather than security and we do not address it in this paper.

2.2 Security threats and requirements

We consider the semi-trusted model where the confidentiality of subscriptions, publications, and payloads (see Section 1) is at risk, as well as their privacy, if any of these three items contains identifying or confidential attributes. Under that model, our threat model includes the two following attackers:
(I) *The standalone broker*: One broker belonging to one overlay network gets knowledge only from the flows it has directly access to;
(II) *The colluding brokers*: $k-1$ brokers belonging to different overlays (see Section 2.1) collude to gather all their knowledge, i.e. quasi-identifiers that were originally split apart thanks to the data splitting method. Note that k corresponds to the number of overlays used by a publisher to send one of their quasi-identifiying attributes as part of their publication.

The objective of both attackers is to:
1. Reidentify the publisher or the subscriber thanks to the identity disclosure attack well known in the statistical disclosure control domain. [Domingo-Ferrer, J. *et al.*, 2015];
2. Get confidential information about the publisher or the subscriber thanks to the following possible attribute disclosure attacks:
• Inference attack: "*This category covers attacks where the attacker has used existing knowledge to aid the attack*" [Henriksen-Bulmer, J. *et al.*, 2016], e.g. deducing the working place of a person by knowing their movements with recurrent GPS positioning;
• Homogeneity attack: "*Re-identification by homogeneity consists of showing a subject's belonging to a homogeneous group in order to deduce all, or part of his/her identity*" [Aïmeur, E. *et al.*, 2012]. Let us consider a correlation attack in the lifeguard scenario. A malicious user may use the auxiliary Table 1(a) to learn from the pseudonymised table 1(b) that Alice is either a student or a lifeguard, as only two people in the lifeguard pseudonymised table are aged 18.

Names	Age	Occupation
Alice	18	Student

(a) Auxiliary table

Names	Age	Occupation
Ae25ade	18	Student
Jfhs1s3	18	Lifeguard
Pdkfd23	24	Policeman

(b) Pseudonymised table used by lifeguards

Table 1: Lifeguards table

3 Related work

Current systems for privacy-preserving pub/sub are mostly based on encrypted matching for preserving the subscription and publication privacy. [Ion, M. *et al.*, 2012] designed an encrypted matching scheme based upon attribute-based encryption (ABE) and multi-user searchable data encryption (SDE). The scheme applies to any numerical operator as well as string equalities. Attribute-based encryption is used to enforce the confidentiality of publication payload. Subscription constraints are encoded into an access tree. The non-leaf nodes of the tree are threshold gates that specify the number of sub-trees to be satisfied. In order to handle inequality constraints, the tree uses "bag of bits" representation [Bethencourt, J. *et al.*, 2007]. The encrypted matching scheme is based on a premapped equality comparison. It means that publication attributes and subscription constraints are premapped to a set of values. This mapping enables a matching strictly based on equality comparisons. The preservation of the publication payload privacy is achieved by extending the ElGamal scheme [El Gamal, 1985] to a proxy re-encryption context, the proxy being the access broker of the publisher.

[Duan, L. *et al.*, 2019] proposed a comprehensive access control framework (CACN) by providing: (1) a privacy-preserving bi-directional policy matching scheme, and (2) a fully homomorphic encryption scheme for encrypting the publication payload. The bi-directional policy matching allows subscribers and publishers to control the publication data to balance security requirements and capabilities of the service. Service privacy is provided through attribute encoding and anonymous-set-based principle. Fur-

thermore, Bloom filters are used as proposed by [Barazzutti, R. *et al.*, 2017] and sent as part of access credentials with the publications. Bloom filters enable to pre-filter publications—i.e. to filter out some publications before using the expensive encrypted matching function.

To avoid the significant overhead of the encrypted matching function, there is another direction related to the statistical disclosure control domain, not yet applied to pub/sub systems, but cited as a solution in a survey about the privacy-aware outsourcing technologies in Cloud computing [Domingo-Ferrer, J. *et al.*, 2019]. The authors review masking methods based on data splitting and anonymization. These different methods are compared in terms of overhead, accuracy preservation, and impact on data management. Homomorphic encryption includes an overhead that is linear with the data size; it involves expensive primitives; and it provides high-level privacy of data. Conversely, in data splitting, the overhead is constant for all operations that are transparent for the Cloud service providers, but the data privacy might be broken due to collusions or compromised metadata during the process.

To support the confidentiality of the publication payload, the Attribute-Based Encryption designed for encrypted data sharing can be applied. The Key-Policy Attribute-Based Encryption (KP-ABE) crypto system, designed in [Goyal, V. *et al.*, 2006] to enforce fine-grained access control, relies on labelling cipher-texts with sets of attributes while private keys are associated with access structures to determine which cipher-texts a user might decrypt. Ciphertext-Policy ABE (CP-ABE) is another form of attribute-based encryption system in which the private keys are labelled with attributes, while the cipher-text is embedded with access policy [Bethencourt, J. *et al.*, 2007]. In CP-ABE, the publisher can decide which architectural entity is allowed to decrypt, while in KP-ABE, this role belongs to the authority that generated the keys.

Since the encryption scheme proposed by [Ion, M. *et al.*, 2012] fits our data model and the content-based pub/sub requirements, our work extends the data splitting principles to the pub/sub paradigm, and combines data splitting to [Ion, M. *et al.*, 2012] encryption scheme in order to enforce privacy and confidentiality in pub/sub systems.

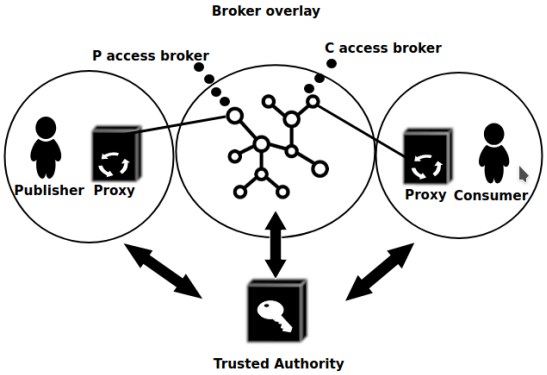


Figure 1: Architectural entities of our DEBS system

4 Our contribution

In Section 4.1, we provide an overview of our proposal. We introduce the architectural entities in Section 4.2, and then the data model of the publications and the filter model of the subscriptions in Section 4.3. Next, we explain how we perform data splitting over different overlays of brokers in Section 4.4. Finally, in Sections 4.5, 4.6, and 4.7, we present the algorithms of the cryptographic system, respectively for handling subscriptions and publications.

4.1 Overview

Our main contribution is the implementation of a system using both data splitting and cryptographic techniques to achieve the following properties:

- Publication payload confidentiality: our solution fully relies on [Ion, M. *et al.*, 2012], which is based on KP-ABE encryption;
- Publication privacy: our solution is based on both the encrypted matching solution of [Ion, M. *et al.*, 2012] and data splitting principles, according to how much confidential are the attributes (see Section 2.1). That is, for confidential and identifying attributes, an encrypted matching is performed by the brokers, and for quasi-identifying attributes, data splitting is performed prior to sending the quasi-identifying attributes onto several separated overlays of brokers;
- Subscription filter privacy: thanks to the proxies of both producer and consumer, the consumer and the producer can benefit from the same level of privacy.

4.2 Architectural entities

Figure 1 depicts the architectural entities of our DEBS system:
• Producer P: the entity sending a publication p to the DEBS system;
• Consumer C: the entity willing to receive publications satisfying subscription filter F;
• Trusted authority (TA): the entity that is responsible for generating and distributing the keys to the different parties of the system;
• Access broker B_P of producer P: the broker which P is connected to, and that first receives the publications from P;
• Access Broker B_C of consumer C: the broker which C is connected to, and that first receives the filter from C;
• Brokers B: Brokers are organised into several overlays (as displayed in Figure 2) in order to split the data into the required amount and preserve privacy;
• Px_P and Px_C are respectively the producer-side proxy and the consumer-side proxy. Px_P is responsible for splitting the data according to the PIA. Publication messages are self-describing so that Px_C can reassemble the original publication from the different parts.

4.3 Publication data model and subscription filter model

We base our model on content-based DEBS with structured records:
• A publication p is a non-empty set of attributes $\{a_1, ..., a_n\}$;
• An attribute a_i is a $(name_i, val_i)$ pair;
• Attribute names are unique: $i \neq j \Rightarrow name_i \neq name_j$.

The corresponding filter model is :
• A filter F is a conjunction of attribute filters: $F = A_1 \wedge ... \wedge A_k$. A publication p matches filter F if and only if it satisfies all the attributes filters of F;
• An attribute filter is a triple $A_i = (name_i, op_i, valueOfRef_i)$, with $name_i$ being the attribute name, op_i being the test operator, and, $valueOfRef_i$ being the reference value for the test.

Attributes can be optional in the publication, and new attributes added without affecting existing filters.

4.4 Data splitting over different overlays of brokers

Our DEBS system combines data splitting with cryptography to achieve required security requirements. To the best of our knowledge, after masking methods were suggested by [Domingo-Ferrer, J. *et al.*, 2019], no solutions using both methods at the same time were developed for the pub/sub paradigm. Our solution discriminates the attributes in the publication and split the attributes according to their type. The identifying attributes are systematically encrypted, while quasi-identifiers are sent apart to avoid re-identification.

The purpose of our solution is to provide an alternative to fully-encrypted publications and subscriptions when confidentiality requirements are not utmost. To illustrate our approach, let us consider a publication p. The PIA has sorted the publication as follows: $p = id_{Attr} + qi_{Attr} + rm_{Attr} + m$, where id_{Attr} is the set of identifying attributes, qi_{Attr} is the set of quasi-identifiers, rm_{Attr} the remaining attributes, and m may be unstructured content to be delivered to subscribers. Afterwards, the proxy has to split the quasi-identifiers into subgroups. Each subgroup has a limited number of quasi-identifiers that together are not able to identify a user. Consequently, we can further detail the composition of our publication: $p = id_{Attr} + \{qi_{Attr_j}\}_{j \in \mathcal{J}} + rm_{Attr}$, where qi_{Attr_j} is a subgroup of quasi-identifiers attributes and $\mathcal{J} = \{1, ..., N\}$ is the set of subgroups of cardinality N.

Once the splitting is performed at proxy Px_P, the outcome is given to publisher P, which encrypts the identifiers and gives them back to Px_P. Then, Px_P distributes the publication to the N overlays of brokers. Each overlay of brokers is responsible for one subgroup of quasi-identifiers; the encrypted attributes, and the remaining attributes can be distributed on any of these overlays. The overlays of brokers route the different parts of the publication. Proxy Px_C of consumer C reassembles the different parts to reconstruct the publication, which is provided to consumer C.

4.5 Algorithms of the cryptographic system

The cryptographic system of our DEBS system puts into action the following algorithms:
• $Init(1^\kappa)$: at the beginning, the Trusted

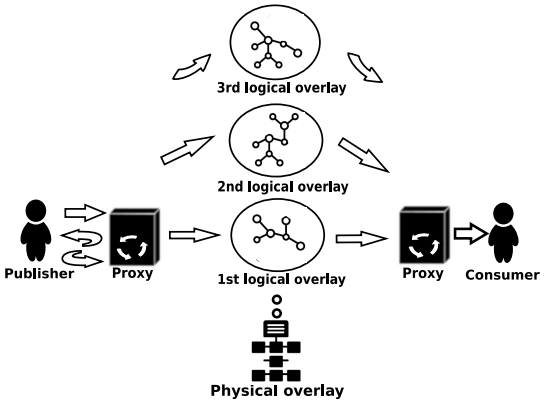


Figure 2: Data splitting workflow for privacy-preserving pub/sub

Authority initiates the crypto system with $PublicParameters$ and a master key mk by running $Init(1^\kappa)$, where κ is a security parameter;
• $KeyGen(mk, E)$: the Trusted Authority uses its master key mk to compute a key for entity E, e.g. the user-side key ku_P for publisher P or ku_C for consumer C, or the server-side keys—i.e broker-side keys—for brokers B_P or B_C;
• $\mathsf{RequestDecryptionKey}(C, \gamma_F)$: the Trusted Authority returns the decryption key D_F to a requesting consumer C owning the set of attributes γ_F;
• $Trapdoors(a)$: publisher P computes a trapdoor for an attribute as in multi-user SDE [Dong, C. *et al.*, 2008]. By extension, $Trapdoors(\gamma_a)$ computes a trapdoor for each attribute in the set of attributes γ_a and outputs $T(\gamma_a)$;
• $Encr\text{-}Trapdrs(ks_P, T(\gamma_a))$: broker B_P encrypts a trapdoor $T(\gamma_a)$ using its key ks_P and outputs c_t;
• $Encr(ku_C, F, \gamma_F)$: consumer C encrypts a filter F using the user-side key ku_C and the set of attributes appearing in F, γ_F, to output c_F;
• $Re\text{-}Enc\text{-}Filt(ks_C, c_F)$: Broker B_C re-encrypts an encrypted filter c_F using the server-side key ks_C and outputs c_f;
• $Match(c_t, c_f)$: a broker checks whether each encrypted attribute of c_f matches any of the attributes encrypted into the trapdoors c_t;
• $KP\text{-}ABE\text{-}Encr(p, \gamma_p)$: publisher P encrypts a publication p using KP-ABE under the set of attributes γ_p, and outputs c_n;
• $Pre\text{-}Decr(ks_C, c_n)$: broker B_C pre-decrypts the content c_n using its server-side key ks_C, for next the consumer C to be able to decrypt the publication. B_C outputs c_n';
• $KP\text{-}ABE\text{-}Decr(D_F, c_n')$: consumer C finalises the decryption of the publication using its decryption key D_F over the pre-decrypted content c_n' sent by its access broker B_C.

4.6 Subscription handling

We now describe subscription handling in this section and publication handling in the next section. Figure 3 displays the corresponding interaction diagram, where message numbers of the figure correspond to item numbers in the sections.

A consumer C that subscribes to filter $F = (A_1 \wedge ...)$ performs the following actions:
1. C builds the set of attribute names $\gamma_F = \{name_{A_1}, ...\}$;
2. C sends a request to the Trusted Authority to get its user-side key ku_C;
3. C sends a request with γ_F to the Trusted Authority for the decryption key D_F ($\mathsf{RequestDecryptionKey}(C, \gamma_F)$);
4. C encrypts the filter F by using algorithm $Encr(ku_C, F, \gamma_F) = \{Encr(ku_C, (name_{A_1}, op_{A_1}, valueOfRef_{A_1}), \{name_{A_1}\}), ...\}$;
5. C sends to its access broker B_C a sub call with the content of $Encr(ku_C, F, \gamma_F)$, which we note $c_F = \{c_{F_{A_1}}, ...\}$ in the following.

When an access broker B_C receives the subscription from C, B_C performs the following actions:
6. B_C sends a request with the identity of C to the Trusted Authority to get C's server-side key ks_C;
7. B_C re-encrypts the encrypted subscription filter by executing algorithm $Re\text{-}Enc\text{-}Filt(ks_C, c_F) = \{Re\text{-}Enc\text{-}Filt(ks_C, c_{F_{A_1}}), ...\}$;
8. B_C broadcasts the encrypted filter to all its neighbouring brokers, which forward it to all their neighbouring brokers, etc., so that the encrypted filter is installed on all the brokers of the overlay.

4.7 Publication handling

A publication $p = \{a_1, ...\}$ is published by the producer P by performing the following actions:
9. P builds the set of attribute names $\gamma_p = \{name_{a_1}, ...\}$;
10. P sends a request with its identity to the Trusted Authority to get its user-side key ku_P;
11. P sends a request with γ_p to the Trusted Authority to get the public parameters to compute the Lagrange coefficients as in [Bethencourt, J. *et al.*, 2007] $L(\gamma_p) = \{L(name_{a_1}), ...\}$;
12. P encrypts the publication by executing al-

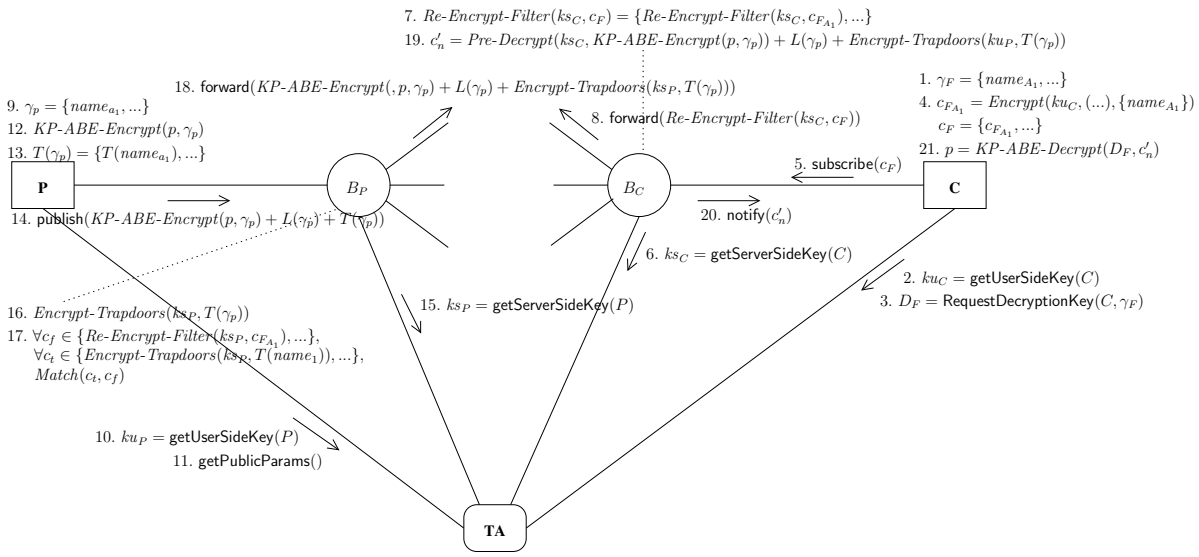


Figure 3: Interaction diagram of subscription handling and publication handling with encryption

gorithm *KP-ABE-Encr*(p, γ_p);
13. P builds the set of trapdoors $Trapdrs(\gamma_p) = T(\gamma_p) = \{T(name_{a_1}), ...\}$;
14. P sends to its access broker B_P a pub call with the content *KP-ABE-Encr*$(p, \gamma_p)+L(\gamma_p)+T(\gamma_p)$.

When an access broker B_P receives the publication from P, B_P performs the following actions:
15. B_P sends a request with the identity of P to the Trusted Authority to get P's server-side key ks_P;
16. B_P encrypts the trapdoors by executing algorithm *Encr-Trapdrs*$(ks_P, T(\gamma_p))$, which outputs the set $\{Encr\text{-}Trapdrs(ks_P, T(name_1)), ...\}$;
17. B_P performs the encryption matching as follows: $\forall c_f \in \{Re\text{-}Enc\text{-}Filt(ks_P, c_{F_{A_1}}), ...\}$, $\forall c_t \in \{Encr\text{-}Trapdrs(ks_P, T(name_1)), ...\}$, $Match(c_t, c_f)$;
18. When the publication matches a filter, that is the previous formula is satisfied, B_p forwards the publication to the brokers along the paths to the access brokers of the corresponding consumer, the content being *KP-ABE-Encr*$(p, \gamma_p)+L(\gamma_p)+$ *Encr-Trapdrs*$(ks_P, T(\gamma_p))$.

When an access broker B_C of consumer C receives a publication, B_C performs the following actions:
19. B_C predecrypts the publication by executing *Pre-Decr*$(ks_C$, *KP-ABE-Encr*$(p, \gamma_p))$.
20. B_C notifies consumer C with the content c'_n that is equal to *Pre-Decr*$(ks_C$, *KP-ABE-Encr*$(p, \gamma_p))$ $+L(\gamma_p)$ $+$*Encr-Trapdrs*$(ks_P, T(\gamma_p))$.

When a consumer C receives a publication, C performs the following actions:
21. C decrypts the publication by executing algorithm *KP-ABE-Decr*(D_F, c'_n), which outputs the publication $p = \{a_1, ...\}$.

5 Security analysis

Since our encrypted matching solution is based upon the scheme of [Ion, M. *et al.*, 2012], our security analysis focuses on how data splitting might alter this solution. Note that the encrypted matching scheme from [Ion, M. *et al.*, 2012] supports both publication and subscription confidentiality, but also payload confidentiality. Consequently, when the security requirements require strong confidentiality, our model can provide it by fully relying on this scheme. Brokers are then able to match the filters against the publications and to deliver the publications to the interested consumers with no ability to learn the content of the publication.

Let us now consider the broker attackers (I) and (II) described in Section 2.2, and the type of information that goes through different overlays (see Section 4.4):

- ***(a)*** *Encrypted and non confidential attributes—over one unique overlay*: This is a special case in which attributes are sent over one specific overlay;
- ***(b)*** *Each subgroup of quasi-identifier attributes—over N overlays*: If a quasi-identifier is made of N subgroups of at-

tributes, then these sets of quasi-attributes are sent over N overlays;

(c) *Remaining attributes—over one unique overlay*: This is a special case in which the remaining attributes are sent over one specific overlay;

(d.) *Encrypted, non confidential, and remaining attributes—over N overlays*: The encrypted, non confidential and remaining attributes are sent over the N overlays corresponding each to a set of quasi-identifiers.

According to the attacker type and the kind of attributes *(a)*, *(b)*, or *(c)* that are made accessible to brokers, the following security analysis can be conducted.

Let us start with the attacker targeting the publication and thus the publisher's privacy:
(I) with (a): The standalone broker has no further knowledge than in [Ion, M. *et al.*, 2012], and benefits directly from the security level of this latter scheme because our encrypted matching approach is based upon this scheme. Note that this scheme is proved to be indistinguishable under the chosen plain-text attack (IND-CPA): in case the attacker has the possibility to get the cipher-texts corresponding to their message choices several times, they cannot successfully solve a game with a non-negligible probability, where the game asks the attacker to decide among two cleartexts, of which one corresponds to a given cipher-text. Since our semi-trusted model considers passive attackers, the IND-CPA assumption is appropriate;
(I) with (b): The standalone broker gets access to only one subgroup of quasi-identifiers, which by definition does not permit to reidentify the publisher;
(I) with (c): Because the remaining attributes bring non-sensitive information, the attacker cannot deduce any identifying or any relevant information about the publisher;
(I) with (d): This case is a sub-case of the next considered attack—i.e.,(II) with (a), (b), (c), and (d)—in which $N = 1$: the standalone broker gets access to one overlay: encrypted, non confidential, one subgroup of quasi-identifiers, and the remaining attributes;
(II) with (a), (b), (c) and (d): The colluding brokers might aggregate several attributes including encrypted, non confidential, and remaining attributes, with $N - 1$ subgroups of attributes belonging to one quasi-identifier of the publisher. In that case, $N - 1$ subgroups of attributes do not leak any identifying information. Thus, the attackers are not able to reidentify the publisher. However, they can leak information about some attributes for an attacker to infer information about the publisher. Actually, the publisher decides how to implement data splitting to meet the privacy requirements.

Let us continue with the attacker targeting the subscriber's filter, and thus the subscriber's privacy. Since the attacker implements the matching operation between the publication attributes and the filter attributes, they can only deduce information of interest in case of matching, as follows:
(I) with (a): The attackers are not able to infer information of interest from encrypted attributes. However, they can infer some information from non-confidential attributes. This is up to the publisher to decide which attributes are considered as non-confidential, i.e. not critical for the privacy of the publisher and the subscriber;
(I) with (b): The attacker has no further information than what is available through the subgroup of attributes. That is why, it is of importance to split attributes into adequate subgroups of quasi-identifiers, so that the privacy of the subscriber is preserved in case of matching;
(I) with (c): The same remark applies as for attacker (I) with (a) with non-confidential attributes;
(II) with (a), (b), and (c): By combining several data, the attacker can extend their knowledge, but without being be able to infer information about the subscriber or the publisher. As such, the carefulness with which data splitting is performed over attributes is crucial for preserving privacy of both the publisher and the subscriber.

6 Performance evaluation

In order to assess the performance of our solution, we implemented the prototype PP-DEBS, for *Privacy-Preserving DEBS*[1], with encryption matching and data splitting in the Java language (version 8). The prototype was deployed on a machine with an Intel Xeon E5 v3 3.1GHz and 8GB of RAM. Our main interest is to illustrate the potential benefit in performance brought by the use of data splitting. To provide a complete set of results, we perform four different experiments on a simple scenario. The scenario consists in publications that contain an increasing number of attributes (from 1 to 20), by a single publisher, and

[1] `https://fusionforge.int-evry.fr/scm/?group_id=175`

delivered to a single subscriber. We only consider a single broker to perform the matching operation on the three types of attributes: identifiers, quasi-identifiers and non-confidential attributes. Consequently, the time needed to route all the packets to the corresponding overlays of brokers is not considered. The aforementioned tests target the following phases: the initialisation performed by the Trusted Authority at the beginning, the encryption of both attributes and filters by the clients as well as the re-encryption by the access brokers, and the matching on encrypted and non-encrypted data handled by the brokers.

First of all, we conduct initialisation tests with the following key lengths in bits: 64, 128, 256, 512, and 1024. Given the format of the encryption scheme, it would be illusory to use the three first lengths, but it remains relevant to measure their impact on the initialisation phase to determine more precisely their evolution following the key lengths. The tests reveal an exponential increase of time as well as an important rise of the standard deviation. We observe indeed an average of 6.50ms for 64 bits length, with 8.8% of standard deviation on 10.000 executions, while we note an average of 23.20s for 1.024 bits length, with 72.6% of standard deviation on 200 executions. The method used to generate the first prime numbers on which the encryption scheme is based explains these results. We use one iterative function provided by the BigInteger class for generating a prime number q. This function considers two tests, one for testing a first try that q is prime with a probability of $1 - 2^{100}$ and one for testing that $2q + 1$ is prime.

Secondly, we execute performance tests for measuring the matching time on encrypted and non-encrypted attributes. We only consider attributes of type Integer with an arbitrary value of 1000, and filters with the equal operator. We make the distinction between split and unsplit data for non-encrypted attributes. We test these operations with 1024 bits key length, on 500 000 executions, and for a number of attributes varying between 1 and 20. The graph in Figure 4 displays matching time for non-encrypted data, with confidence level of 95%.

The first noticeable fact lies in the impossibility to discern the confidence intervals from the curves, which demonstrates the stability of the matching process. The first experiment reveals that matching on encrypted data follows a linear evolution, consuming 0.158ms per attribute on average. Regarding this result, it is obvious

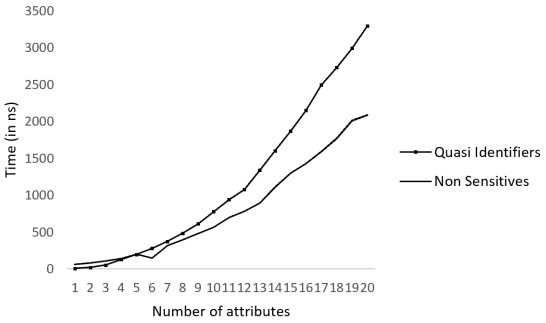


Figure 4: Matching time for non-encrypted data

that matching on encrypted data seriously alters performances: we can observe two to three orders of magnitude comparing to matching on non-encrypted data, for a high to a low number of attributes. It is although not surprising to notice that the curve for split non-encrypted data tends to be similar to a square function, which can be explained by the implementation. We chose to split the group of quasi-identifiers in the maximum possible packets, e.g. with one attribute per packet. To perform matching, we consequently had to check for each attribute all the filters corresponding to quasi-identifier attributes.

Encryption and re-encryption of attributes and filters are still the most time-consuming phases. We conducted tests for these operations in the same conditions as previously, for 1500 executions. The graphs in Figure 5 displays encryption time on publisher's side, this time including also the re-encryption by the access broker, with a confidence interval for 95% of certainty.

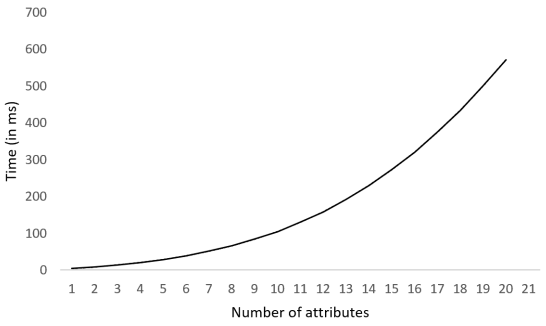


Figure 5: Encryption on publisher's side

The experiments demonstrate that the encryption process is very stable. The encryption on the subscriber's side follows a linear evolution, consuming 3ms per attribute, while the encryption on the publisher's side consumes almost 600ms for 20 attributes. It is crucial to remember

that encryption at the subscriber's side is only performed once for a given filter, whilst it has to be computed for each publication on the publisher's side.

As a summary, the experiments illustrate that data-splitting is highly recommended in pub/sub systems, given the amount of time spent by every broker for each publication in matching over encrypted-data. In addition, the encryption on the publisher's side, followed by a re-encryption by the access broker, represents a considerable overhead on performance, given it has to be computed for each publication. Finally, the encryption process done at the subscriber's side is followed by a re-encryption process which stands for a more limited impact, yet not negligible on the global performances of the system.

7 Conclusion

In this paper, we presented a pub/sub system combining data encryption and data splitting to reduce the number of encrypted matching function calls. Our scheme preserves the privacy of the users by preventing their identification, and lets the users choose the degree of privacy they need. Privacy can then be enforced by relying on encrypted matching.

To demonstrate the relevance of our solution, we conducted a security analysis based on an original attacker model, introducing new threats with the data splitting procedure, such as the collusion between overlays for gathering quasi-identifiers. Besides, we implemented our solution in the Java language and evaluated the performance of this implementation. This highlighted the cost of the encrypted matching function and the efficiency of our solution in terms of performance.

Acknowledgements

This work was partially funded by the SAMOVAR and LTCI labs. The authors would like to thank authors of [Ion, M. *et al.*, 2012], especially Giovanni Russello, for their support.

REFERENCES

[AMQP Consortium, 2008] AMQP Consortium (2008). Advanced Message Queuing Protocol, v. 0-9-1.

[Aïmeur, E. *et al.*, 2012] Aïmeur, E. *et al.* (2012). Reconstructing Profiles from Information Disseminated on the Internet. In *ASE/IEEE SOCIALCOM-PASSAT*.

[Barazzutti, R. *et al.*, 2017] Barazzutti, R. *et al.* (2017). Efficient and Confidentiality-Preserving Content-Based Publish/Subscribe with Prefiltering. *IEEE TDSC*, 14(3).

[Bethencourt, J. *et al.*, 2007] Bethencourt, J. *et al.* (2007). Ciphertext-Policy Attribute-Based Encryption. In *IEEE SP*.

[Domingo-Ferrer, J. *et al.*, 2015] Domingo-Ferrer, J. *et al.* (2015). Database Privacy. In *Privacy in a Digital, Networked World*.

[Domingo-Ferrer, J. *et al.*, 2019] Domingo-Ferrer, J. *et al.* (2019). Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges. *Computer Communications*, 140.

[Dong, C. *et al.*, 2008] Dong, C. *et al.* (2008). Shared and Searchable Encrypted Data for Untrusted Servers. In *Data and Applications Security XXII*.

[Duan, L. *et al.*, 2019] Duan, L. *et al.* (2019). A Comprehensive Security Framework for Publish/Subscribe-Based IoT Services Communication. *IEEE Access*.

[El Gamal, 1985] El Gamal, T. (1985). A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *CRYPTO*.

[Eugster *et al.*, 2003] Eugster *et al.* (2003). The Many Faces of Publish/Subscribe. *CACM*, 35(2).

[Goyal, V. *et al.*, 2006] Goyal, V. *et al.* (2006). Attribute-based Encryption for Fine-grained Access Control of Encrypted Data. In *ACM CCS*.

[Henriksen-Bulmer, J. *et al.*, 2016] Henriksen-Bulmer, J. *et al.* (2016). Re-identification attacks—A systematic literature review. *IJIM*, 36(6, Part B).

[Ion, M. *et al.*, 2012] Ion, M. *et al.* (2012). Design and implementation of a confidentiality and access control solution for publish/subscribe systems. *Computer Networks*, 56(7).

[ISO, 1989] ISO (1989). Information processing systems, Open Systems Interconnection, Basic Reference Model, Part 2: Security Architecture. Technical report.

[OASIS, 2019] OASIS (2019). MQTT v. 5.0.

[Onica, E. *et al.*, 2016] Onica, E. *et al.* (2016). Confidentiality-Preserving Publish/Subscribe: A Survey. *CACM*, 49(2).

[Sullivan, 2019] Sullivan, C. (2019). EU GDPR or APEC CBPR? A comparative analysis of the approach of the EU and APEC to cross border data transfers and protection of personal data in the IoT era. *Computer Law & Security Review*, 35(4).

[Wright, 2012] Wright, D. (2012). The state of the art in privacy impact assessment. *Computer Law & Security Review*, 28(1).